

RANDAL W. BEARD

VISION-BASED ESTIMATION AND CONTROL OF MULTIROTOR AND WINGE VTOL SYSTEMS

BRIGHAM YOUNG UNIVERSITY

Copyright © 2020 Randal W. Beard

PUBLISHED BY BRIGHAM YOUNG UNIVERSITY

TYPESET USING TUFTE-LATEX.GITHUB.IO/TUFTE-LATEX/

Current version: April 2023

Dedicated to ...

Contents

1	<i>Introduction</i>	15
<i>I Models & Control & Planning</i>		17
2	<i>Preliminaries</i>	19
2.1	<i>Notation</i>	19
2.2	<i>Vectors</i>	20
2.2.1	<i>A first look at rotation matrices.</i>	21
2.2.2	<i>Vector composition.</i>	21
2.2.3	<i>Cross product and skew symmetric matrices.</i>	22
2.2.4	<i>Useful matrix identities</i>	23
2.3	<i>The Rodrigues Formula</i>	24
2.4	<i>Attitude Representation</i>	29
2.4.1	<i>Euler Angles</i>	29
2.4.2	<i>Rotation Matrices</i>	30
2.4.3	<i>Passive Versus Active Rotations</i>	32
2.4.4	<i>Quaternions</i>	33
2.4.5	<i>Angle-axis Representation</i>	35
2.4.6	<i>Transforms between representations</i>	36
2.5	<i>Rotational Kinematics</i>	41
2.5.1	<i>Equation of Coriolis</i>	41
2.5.2	<i>Rotational Kinematics</i>	42
2.5.3	<i>Integrating the Rotational Kinematics</i>	44

<i>2.5.4</i>	<i>Numerical Differentiation of the Rotation Matrix</i>	46
<i>2.5.5</i>	<i>Quaternion Kinematics</i>	46
<i>2.5.6</i>	<i>Euler Angle Kinematics</i>	46
<i>2.5.7</i>	<i>Rotation Vector Kinematics</i>	47
<i>2.6</i>	<i>Homogeneous Transformations</i>	47
<i>2.6.1</i>	<i>Homogeneous Transformation Matrices</i>	47
<i>2.6.2</i>	<i>Passive and Active Transformations</i>	48
<i>2.6.3</i>	<i>Rigid body kinematics</i>	49
<i>2.7</i>	<i>Lie Groups</i>	52
<i>2.7.1</i>	<i>Group Theory</i>	53
<i>2.7.2</i>	<i>$SO(2)$: A gentle introduction</i>	54
<i>2.7.3</i>	<i>The Lie group $SO(3)$</i>	56
<i>2.7.4</i>	<i>The Lie group $SE(3)$</i>	57
<i>2.7.5</i>	<i>The Lie group S^3</i>	57
<i>2.8</i>	<i>Adjoints and Jacobians</i>	58
<i>2.8.1</i>	<i>The Adjoint Operator and Adjoint Matrix</i>	60
<i>2.9</i>	<i>Right and Left Jacobian of the Exponential</i>	63
<i>2.9.1</i>		64
<i>2.9.2</i>	<i>Jacobian of the Matrix Exponential</i>	65
<i>2.10</i>	<i>Useful Tables</i>	66
<i>2.11</i>	<i>Micro Lie Theory</i>	68
<i>3</i>	<i>Multirotor Equations of Motion</i>	85
<i>3.1</i>	<i>Equations of Motion</i>	85
<i>3.1.1</i>	<i>Rotor Forces and Torques</i>	86
<i>3.2</i>	<i>Multirotor Mass Properties</i>	89
<i>3.3</i>	<i>Multirotor Aerial Vehicles, Mahony, Kumar, Corke, IEEE RA Mag, Sept, 2020.</i>	90
<i>4</i>	<i>Trajectory Following</i>	105
<i>4.1</i>	<i>A Primer on Lyapunov Stability Theory</i>	105
<i>4.2</i>	<i>Angular Rate Regulation</i>	107

4.3	<i>Attitude Stabilization Using Euler Angles</i>	108
4.4	<i>Attitude Stabilization Using Rotation Matrices</i>	110
4.5	<i>Attitude Stabilization Using Quaternions</i>	115
4.6	<i>Velocity controller</i>	115
4.7	<i>Trajectory Tracking Control</i>	115
4.7.1	<i>Cascaded Control</i>	115
4.8	<i>Geometric Tracking Control of a Quadrotor UAV on $SE(3)$.</i>	118
4.9	<i>Trajectory following using LQR</i>	127
5	<i>Trajectory Generation and Local Planning</i>	129
5.1	<i>Differential Flatness</i>	129
5.1.1	<i>Analytic calculation of desired angular velocity</i>	132
5.2	<i>An Introduction to b-Spline</i>	133
5.2.1	<i>B-Spline Basis Functions</i>	134
5.2.2	<i>Uniform B-Splines</i>	139
5.2.3	<i>Derivatives of B-splines</i>	144
5.2.4	<i>B-Spline Planning for Chains of Integrators</i>	149
5.2.5	<i>B-Splines on Lie Groups</i>	149
5.2.6	<i>B-Splines Surfaces</i>	149
5.3	<i>Minimum Snap Trajectories</i>	150
5.3.1	<i>Minimum snap trajectories without obstacles</i>	152
5.3.2	<i>Minimum snap trajectories with obstacles</i>	154
5.4	<i>Ensuring dynamic feasibility using time scaling</i>	154
5.4.1	<i>Curvature constraints</i>	156
5.5	<i>Path Planning using a Digital Elevation Map</i>	156
II	<i>Vision Based Flight</i>	157
6	<i>Camera Models and Feature Detection</i>	159
6.1	<i>Pinhole model</i>	159
6.2	<i>Camera Field of View</i>	162

6.3	<i>Some notes on Points and lines in images</i>	164
6.4	<i>Feature Detection</i>	166
6.4.1	<i>Harris Corner Detector</i>	168
6.4.2	<i>Shi-Tomasi Corner Detector</i>	168
7	<i>Optical Flow and Feature Tracking</i>	169
7.1	<i>Levenberg-Marquardt (LM) Optimization</i>	169
7.2	<i>Optical Flow</i>	170
7.3	<i>KLT Feature Tracking</i>	172
7.4	<i>KLT Feature Tracking with IMU</i>	173
7.5	<i>Multirotor guidance algorithms using optical flow</i>	183
7.5.1	<i>Simple relationships between moving camera and optical flow</i>	183
7.5.2	<i>Landing strategy</i>	186
7.5.3	<i>Collision avoidance strategy</i>	186
7.5.4	<i>Terrain following strategy</i>	186
7.5.5	<i>Wall following strategy</i>	187
7.5.6	<i>Canyon following strategy</i>	188
7.6	<i>Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft, TRO, 2006</i>	188
8	<i>Scene Reconstruction</i>	199
8.1	<i>IMU Integration</i>	199
8.2	<i>Essential Matrix</i>	202
8.2.1	<i>Point Triangulation</i>	205
8.2.2	<i>Naive Point Triangulation</i>	205
8.2.3	<i>Point Triangulation Using Maximum Likelihood Estimation</i>	208
8.3	<i>Simple point reconstruction methods</i>	209
8.4	<i>Visual Inertial Odometry</i>	209
8.4.1	<i>Error State EKF</i>	210
8.4.2	<i>Almost-Invariant EKF for VIO</i>	211
8.5	<i>Inverse Depth Parametrization for Monocular SLAM, TRO, 2008</i>	215
8.6	<i>Visual Inertial Odometry: Robust Visual Inertial Odometry Using a Direct EKF-Based Approach, IROS, 2015</i>	230
8.7	<i>Building a Voxel Map</i>	238

9	<i>Flight Control using Visual Servoing</i>	241
9.1	<i>Feature motion related to vehicle motion</i>	241
9.2	<i>The Body-Level Frame</i>	245
9.3	<i>Landing and target following using visual servoing</i>	248
9.4	<i>Multiple View Geometry-Transformations of a plane</i>	251
9.5	<i>Visual Servo Control Part I: Basic Approaches, RAM, 2006</i>	265
9.6	<i>Visual Servo Control Part II: Advanced Approaches, RAM, 2007</i>	275
9.7	<i>Autonomous Landing of a VTOL UAV on a Moving Platform Using Image-based Visual Servoing, ICRA, 2012</i>	286
9.8	<i>Using visual servoing to point the gimbal at a target</i>	293
10	<i>Visual Tracking</i>	309
10.1	<i>The Euclidian homography matrix</i>	310
10.1.1	<i>Computation of the homography matrix</i>	311
10.1.2	<i>Group Velocity</i>	312
10.1.3	<i>Homography Filter</i>	314
10.2	<i>Relative pose estimation</i>	315
10.3	<i>Moving feature segmentation</i>	328
10.4	<i>Feature and track propagation</i>	328
10.5	<i>Recursive-RANSAC</i>	331
10.6	<i>Track Selection</i>	338
10.6.1	<i>Target closest to the image center</i>	338
10.6.2	<i>Template matching</i>	338
10.6.3	<i>Machine learning</i>	338
10.6.4	<i>User input</i>	338
10.7	<i>Target following</i>	339
10.8	<i>Target following 2.0</i>	340
10.9	<i>Autonomous Flight for Detection, Localization, and Tracking of Moving Targets with Small Quadrotor, RA-L, 2017 (in review)</i>	343

<i>III Vision Based Estimation</i>	353
11 <i>Attitude Estimation</i>	355
11.1 <i>Inertial Measurement Unit (IMU)</i>	355
11.1.1 <i>Accelerometers</i>	355
11.1.2 <i>Rate Gyros</i>	358
11.1.3 <i>Magnetometer</i>	359
11.2 <i>Wahba's Problem</i>	361
11.2.1 <i>Single vector measurement</i>	361
11.2.2 <i>Multiple vector measurements</i>	362
11.3 <i>Complementary Filter</i>	362
11.3.1 <i>Model Free Complementary Filter</i>	362
11.4 <i>Old Stuff that might be useful</i>	369
11.5 <i>Sensors</i>	369
11.5.1 <i>Camera</i>	369
11.6 <i>State Estimation</i>	371
11.6.1 <i>Low Pass Filters</i>	371
11.6.2 <i>Angular Rates p, q, and r.</i>	372
11.6.3 <i>Dynamic Observer Theory</i>	372
11.6.4 <i>Essentials from Probability Theory</i>	374
11.6.5 <i>Derivation of the Kalman Filter</i>	375
11.6.6 <i>Application to the quadrotor</i>	378
12 <i>Trajectory Estimation</i>	383
13 <i>Visual Odometry</i>	385
13.1 <i>Visual Odometry Tutorial PI, RAM, 2011</i>	385
13.2 <i>Visual Odometry Tutorial PII, RAM, 2012</i>	399
13.3 <i>SVO: Fast Semi-Direct Monocular Visual Odometry, ICRA, 2014</i>	413
13.4 <i>Robust Visual Inertial Odometry Using a Direct EKF-Based Approach, IROS, 2015</i>	
	422

14 *Vision Based SLAM* 431

Bibliography 433

Index 437

Contributors

The following individuals have contributed section to this book. Their assistance and insight have been invaluable, and their contributions are gratefully acknowledged.



James Jackson was a PhD student at BYU from 2014–2018. He is currently at Aurora Innovation.



Tim McLain is a professor in the Department of Mechanical Engineering at Brigham Young University.



Jacob Willis is currently a graduate student in the Electrical and Computer Engineering Department at Brigham Young University.

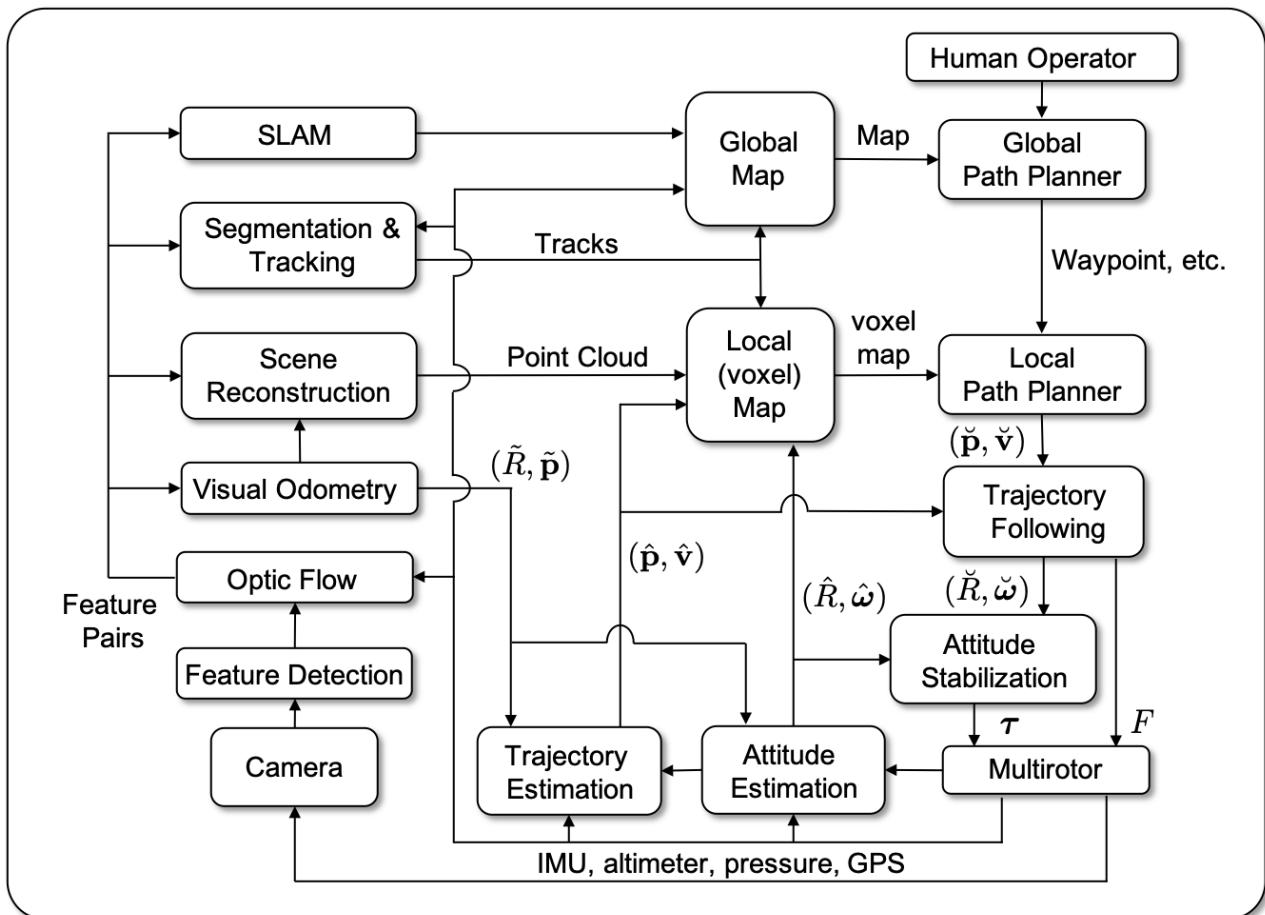


Jake Johnson is currently a graduate student in the Electrical and Computer Engineering Department at Brigham Young University.

1

Introduction

The book will be organized around the architecture shown in Figure ??.



In the bottom right corner is the *Multirotor* block representing both the physical flying vehicle and the onboard sensors, including

Figure 1.1: Software architecture for multirotor estimation and control.

IMU, altimeter, pressure sensors, and possibly a GNSS receiver. We also assume that the platform has a camera, which we have drawn in its own block. The equations of motion for the multirotor and the mathematical models for the sensors will be described in Chapter 3. The commanded inputs to the multirotor will be assumed to be the force in the body z -axis F , and the torque applied to the body τ . The *Attitude Stabilization* block is discussed in Chapter ???. In the literature, three different representations for attitude are commonly used, namely Euler angles, quaternions, and rotation matrices. We will describe attitude stabilization schemes for each representation. The output of the attitude stabilization block is the applied torque τ . The input is the desired attitude \check{R} and the desired angular rate $\check{\omega}$. The *Trajectory Following* block is discussed in Chapter 4. The input to the trajectory follower is the desired position \check{p} and the desired velocity \check{v} .

The second part of the book is concerned with cameras, computer vision, and state estimation. The *Camera* and *Feature Detection* blocks shown in Figure ?? are described in Chapter 6. *Optical Flow* is discussed in Chapter 7, both how optic flow is computed, as well as how it can be used to navigate a multirotor through urban canyons. *Visual Odometry* is discussed in Chapter 13. We then return to the important task of state estimation. *Attitude Estimation* is described in Chapter 11, where we include estimation schemes that use feature matching, optic flow, and visual odometry. In a similar way, *Trajectory Estimation* is described in Chapter 12.

The final part of the book focuses on mapping and planning. *Scene Reconstruction* using a *Local Voxel Map* is detailed in Chapter 8. *Segmentation and Tracking* is discussed in Chapter 10. The *Local Path Planner* is then described in Chapter ?? where we describe differential flatness, spline-based planning, and visual servoing. One particular implementation of Simultaneous Localization and Mapping (*SLAM*) is then described in Chapter ??, and global path planning using the global map created by SLAM is described in Chapter ??.

Part I

Models & Control & Planning

2

Preliminaries

Author: RWB, James Jackson

This chapter motivates and derives many commonly-used formulae used when performing coordinate transformations in robotics and computer vision. There are a many great resources on this topic ^{1,2,3,4}.

One difference with this document when compared with others is the use of extra notation throughout. This notation is much more verbose than a lot of other literature on robotics or computer vision, but the extra syntax clarifies what is going on, and makes it easier to visualize and understand what is going on.

2.1 Notation

Throughout the book we will use lower case symbols to represent scalars, bold lower case to represent vectors, and upper case to represent matrices. Hence

$$\begin{aligned} A &= \begin{pmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{pmatrix} \\ &= \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}. \end{aligned}$$

A coordinate frame is denoted by $\mathcal{F} = \{\mathbf{i}, \mathbf{j}, \mathbf{k}\}$ where the unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are mutually orthogonal and form a right handed coordinate system, i.e., $\mathbf{i} \times \mathbf{j} = \mathbf{k}$. Coordinate frames are identified using a superscript notation. For example $\mathcal{F}^i = \{\mathbf{i}_i, \mathbf{j}_i, \mathbf{k}_i\}$ will denote an inertially defined frame, $\mathcal{F}^b = \{\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b\}$ will denote the multirotor body frame, $\mathcal{F}^c = \{\mathbf{i}_c, \mathbf{j}_c, \mathbf{k}_c\}$ will denote the camera frame. When we say that a vector \mathbf{r} is expressed with respect to frame b , we mean that

¹ Timothy Barfoot. *State Estimation For Robotics*. Cambridge University Press, 2019

² Tom Drummond. Lie groups, lie algebras, projective geometry and optimization for 3d geometry, engineering and computer vision, 2014. Available at <https://www.dropbox.com/s/5y3tvypzps59s29/3DGeometry.pdf?dl=0>

³ Ethan Eade. Lie groups for 2d and 3d transformations, 2017. Available at <http://ethaneade.com/lie.pdf>,

⁴ Dinesh Atchuthan Joan Sola, Jeremie Deray. A micro lie theory for state estimation in robotics. 2019. Available at <https://arxiv.org/pdf/1812.01537.pdf>

$\mathbf{r} = \alpha \mathbf{i}^b + \beta \mathbf{j}^b + \gamma \mathbf{k}^b$, and we typically write

$$\mathbf{r}^b = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$

When coordinate vectors are expressed with respect to their own frames we have $\mathbf{i}_a^a = \mathbf{e}_1$, $\mathbf{j}_a^a = \mathbf{e}_2$, $\mathbf{k}_a^a = \mathbf{e}_3$, where

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

We use the notation $\mathbf{r}_{a/b}^c$ to represents a vector of the physical quantity \mathbf{r} (e.g. position, velocity, etc) of point a with respect to point b , expressed in frame c . R_a^b is a transformation or rotation that executes a change of basis from frame a to frame b . Given a vector $\mathbf{a} \in \mathbb{R}^3$, the skew symmetric matrix associated with \mathbf{a} is defined using the “wedge” operator

$$[\mathbf{a}]_{\times} \doteq \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}. \quad (2.1)$$

It is straight forward to show that for any $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$,

$$[\mathbf{a}]_{\times} \mathbf{b} = \mathbf{a} \times \mathbf{b}.$$

Therefore $[\mathbf{a}]_{\times}$ is a matrix implementation of the cross product operation. We will also use the “wedge” operator to mean the same thing, i.e., $\mathbf{a} \in \mathbb{R}^3$, $\mathbf{a}^{\wedge} \triangleq [\mathbf{a}]_{\times}$. The “vee” operator goes in the other direction:

$$\begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}^{\vee} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad (2.2)$$

2.2 Vectors

Consider the illustration in Figure 2.1. We have some frame \mathcal{F}^a and another frame \mathcal{F}^b . Let $\mathbf{r}_{a/b}$ be the vector which describes the position of center of frame b with respect to the center of frame a . Note that we could flip the vector around by negating the quantity, which means that for any arbitrary frame of reference

$$\mathbf{r}_{b/a} = -\mathbf{r}_{a/b}.$$

Although it may be sometimes hard to actually draw some quantities clearly (like the angular rate of two coordinate frames), in engineering, we typically define vectors as some quantity with respect to

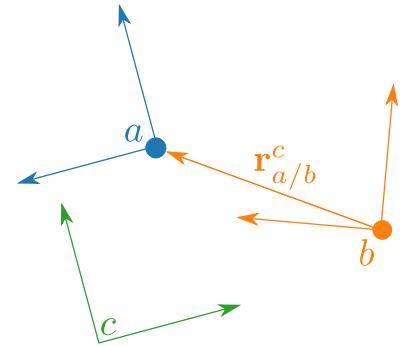


Figure 2.1: Illustration of a TOC vector with notation

some other reference. For example, the angular velocity of a rotating body with respect to the earth might be written as $\omega_{b/E}$. While it may be difficult to visualize, we could also consider the opposite case, what the angular velocity of the earth was with respect to the rotating body, $\omega_{E/b}$. It turns out that even in this case, the “earth-centric” representation is just the negative of the “body-centric” representation.

$$\omega_{E/b} = -\omega_{b/E}.$$

The next piece of information that we need is the frame of reference it is described in (its basis). For example, in Figure 2.1, we could describe \mathbf{r} in any coordinate frame we want. We could use frame a , b or c , it doesn’t really matter, so long as we always do all operations between vectors represented in the same frame.

We will use the superscript notation to describe the frame of reference, as in $\mathbf{r}_{a/b}^c$.⁵ All vectors have this concept of frame of reference, even if they are hard to visualize, like angular rate, acceleration, etc...

2.2.1 A first look at rotation matrices.

We can transform vector coordinates from one frame to another using a rotation matrix. Let $R_a^b \in \mathbb{R}^{3 \times 3}$ be the matrix that transforms the coordinates of physical vector expressed in frame a to the coordinates of the same physical vector expressed in frame b . Note that R_a^b will transform the coordinate axes of frame a expressed in a to the coordinate axes of frame a expressed in b . Therefore if we write R_a^b in terms of its columns $R_a^b = (\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3)$, then we have

$$R_a^b \mathbf{i}_a^a = (\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \mathbf{r}_1 = \mathbf{i}_a^b.$$

Using similar logic for the other columns we have

$$R_a^b = (\mathbf{i}_a^b \quad \mathbf{j}_a^b \quad \mathbf{k}_a^b).$$

Suppose that a vector $\mathbf{r}_{a/b}$ is expressed with respect to frame c and we would like to express it with respect to frame d , then we can use the rotation matrix R_c^d as the transformation

$$\mathbf{r}_{a/b}^d = R_c^d \mathbf{r}_{a/b}^c$$

2.2.2 Vector composition.

Recall that vector space is a collection of vectors and a scalar field with operations for vector addition and scalar multiplication. The

⁵ In words $r_{a/b}^c$ means “the position of frame a with respect to frame b , expressed in frame c .”

The columns of the rotation matrix R_a^b are the coordinate axes of frame a expressed in frame b .

The notation is such that you can “cancel out” the frames, as in

$$\mathbf{r}_{a/b}^b = R_f^b \mathbf{r}_{a/b}^f.$$

Rotations can also be compounded as in

$$\mathbf{r}_{a/b}^e = R_e^e R_f^b \mathbf{r}_{a/b}^f.$$

following table provides the definition for these operations.

Axiom	Meaning
Associativity	$\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c}$
Commutativity	$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$
Identity vector	There exists a vector $\mathbf{0}$ such that $\mathbf{a} + \mathbf{0} = \mathbf{a}$ for every vector in the space
Inverse vector	for every vector \mathbf{a} , there is one and only one vector $-\mathbf{a}$ such that $\mathbf{a} + (-\mathbf{a}) = \mathbf{0}$.
distributivity of scalar multiplication	$(v + u)\mathbf{a} = v\mathbf{a} + u\mathbf{a}$ $v(\mathbf{a} + \mathbf{b}) = v\mathbf{a} + v\mathbf{b}$
Identity scalar	$1\mathbf{a} = \mathbf{a}$

Physical vectors are part of the 3D Euclidian vector space \mathbb{E}^3 the scalar field being given by the real numbers \mathbb{R} . This will be in contrast to rotation matrices that are not elements of a vector space. Later in this chapter we will define the notion of a group, which will be much more useful in dealing with rotation matrices.

We like vector spaces because computers are well-suited for solving linear algebra problems. There are high-performance BLAS libraries (basic linear algebra subprograms) and methods for performing matrix decompositions (SVD, LU, QR, Cholesky etc...) that allow us to solve complicated problems quickly and accurately.

It turns out that rotations and transformations do *not* lie in a vector space. This means that we end up performing manipulations to get the problems into a vector space where we can use powerful linear algebra techniques, and the distinction between vector spaces and non-vector spaces will become very important.

Figure 2.2 shows three coordinate frames a , b , and c . We would like to add the orange vectors $\mathbf{r}_{b/c}^c$ and $\mathbf{r}_{a/b}^b$. Vector addition applies, but only if the vectors are expressed in the same coordinate frame. Therefore, to add the vectors we must first transform $\mathbf{r}_{a/b}^b$ into frame c to get

$$\mathbf{r}_{a/c}^c = \mathbf{r}_{b/c}^c + R_{b/c}^c \mathbf{r}_{a/b}^b.$$

2.2.3 Cross product and skew symmetric matrices.

The cross product of two vectors in \mathbb{R}^3 is defined as

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}.$$

Table 2.1: Table of the rules of a Vector Space

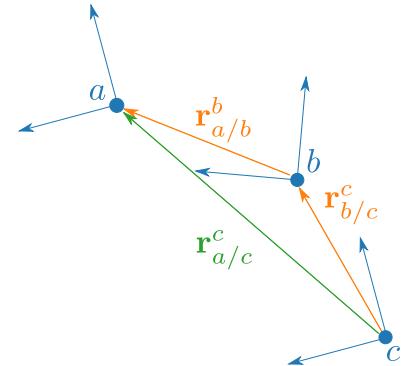


Figure 2.2: Illustration of a vector triangle

Direct manipulation shows that cross product satisfies the following identities:

$$\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a} \quad (2.3)$$

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = \mathbf{a}^\top \mathbf{c} \mathbf{b} - \mathbf{a}^\top \mathbf{b} \mathbf{c} \quad (2.4)$$

$$\mathbf{a}^\top (\mathbf{b} \times \mathbf{c}) = \mathbf{c}^\top (\mathbf{a} \times \mathbf{b}) = \mathbf{b}^\top (\mathbf{c} \times \mathbf{a}) \quad (2.5)$$

$$(\mathbf{a} \times \mathbf{b})^\top (\mathbf{c} \times \mathbf{d}) = (\mathbf{a}^\top \mathbf{c})(\mathbf{b}^\top \mathbf{d}) - (\mathbf{a}^\top \mathbf{d})(\mathbf{b}^\top \mathbf{c}) \quad (2.6)$$

$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\| \|\mathbf{b}\| \sin \phi, \quad \phi = \text{angle between } \mathbf{a} \text{ and } \mathbf{b} \quad (2.7)$$

$$\mathbf{a}^\top (\mathbf{b} \times \mathbf{c}) = \det \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix}. \quad (2.8)$$

If R is a rotation matrix, then

$$R(\mathbf{a} \times \mathbf{b}) = (R\mathbf{a}) \times (R\mathbf{b}). \quad (2.9)$$

It is straight forward to show that for any $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$,

The “wedge” operator is defined in Equation (2.1).

$$[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b}.$$

Therefore $[\mathbf{a}]_\times$ is a matrix implementation of the cross product operation. From the properties of the cross product, we can derive the following properties for skew symmetric matrices:

$$[\mathbf{a}]_\times = -[\mathbf{a}]_\times^\top \quad (2.10)$$

$$[\mathbf{a}]_\times [\mathbf{b}]_\times = \mathbf{b} \mathbf{a}^\top - (\mathbf{a}^\top \mathbf{b}) I \quad (2.11)$$

$$\mathbf{a}^\top [\mathbf{b}]_\times = ([\mathbf{a}]_\times \mathbf{b})^\top = -\mathbf{b}^\top [\mathbf{a}]_\times \quad (2.12)$$

$$([\mathbf{a}]_\times \mathbf{b})^\top [\mathbf{c}]_\times = \mathbf{a}^\top (\mathbf{c} \mathbf{b}^\top - \mathbf{b}^\top \mathbf{c} I). \quad (2.13)$$

From property (2.11), we have that if \mathbf{n} is a unit vector, then

$$[\mathbf{n}]_\times^2 = \mathbf{n} \mathbf{n}^\top - I. \quad (2.14)$$

From property (2.9) we have

$$[R\mathbf{a}]_\times = R [\mathbf{a}]_\times R^\top. \quad (2.15)$$

Define the symmetric and skew-symmetric operators as

$$\mathbb{P}_s(A) \triangleq \frac{1}{2}(A + A^\top) \quad (2.16)$$

$$\mathbb{P}_a(A) \triangleq \frac{1}{2}(A - A^\top), \quad (2.17)$$

$$\begin{aligned} R(\mathbf{a} \times \mathbf{b}) &= (R\mathbf{a}) \times (R\mathbf{b}) \\ \iff R[\mathbf{a}]_\times \mathbf{b} &= [R\mathbf{a}]_\times R\mathbf{b} \\ \iff R[\mathbf{a}]_\times &= [R\mathbf{a}]_\times R \\ \iff R[\mathbf{a}]_\times R^\top &= [R\mathbf{a}]_\times \end{aligned}$$

and note that $A = \mathbb{P}_s(A) + \mathbb{P}_a(A)$. Therefore, any square matrix A can be decomposed into a symmetric part, and a skew-symmetric part.

2.2.4 Useful matrix identities

In this section, we collect some useful matrix identities.

The trace of a matrix $A = \{a_{ij}\} \in \mathbb{R}^{m \times n}$ is defined as the sum of the the diagonal terms, i.e.,

$$\text{tr}(A) = \sum_{i=1}^{\min(m,n)} a_{ii}.$$

The following properties of the trace of a matrix well known, where it is assumed that matrices have compatible dimensions:

$$\text{tr}(A^\top) = \text{tr}(A), \quad (2.18)$$

$$\text{tr}(AB) = \text{tr}(BA), \quad (2.19)$$

$$\text{tr}(\alpha A + \beta B) = \alpha \text{tr}(A) + \beta \text{tr}(B), \quad (2.20)$$

$$A - \text{symmetric}, B - \text{skew symmetric} \implies \text{tr}(AB) = 0, \quad (2.21)$$

$$\text{tr}([\mathbf{a}]_\times [\mathbf{b}]_\times) = -2\mathbf{a}^\top \mathbf{b}. \quad (2.22)$$

The Frobenius norm of matrix $A = \{a_{ij}\} \in \mathbb{R}^{n \times n}$ is defined as

$$\|A\|_F \triangleq \sqrt{\text{tr}(A^\top A)} = \sum_{ij} |a_{ij}|^2.$$

Lemma 2.2.1 *If $R \in SO(3)$, then*

$$\|I - R\|_F^2 = 2 \text{tr}(I - R). \quad (2.23)$$

Proof:

$$\begin{aligned} \|I - R\|_F^2 &= \text{tr}((I - R)^\top (I - R)) \\ &= \text{tr}(I - R - R^\top + R^\top R). \end{aligned}$$

Since $R \in SO(3)$ we have that $R^\top R = I$. Therefore from properties (2.18) and (2.20) we get Equation (2.23). \blacksquare

2.3 The Rodrigues Formula

In this section we derive the so-called *Rodrigues Formula* that describes a right handed rotation of angle θ about a vector \mathbf{n} . We will use this formula throughout the book. Our derivation follows that given in *Stevens & Lewis*⁶. Consider the geometry shown in Figure 2.3, where the vector \mathbf{q} is obtained by a right-handed rotatation of the vector \mathbf{p} about the unit vector \mathbf{n} by an angle of θ .

From the geometry, note that

$$\mathbf{q} = \vec{ON} + \vec{NW} + \vec{WQ}. \quad (2.24)$$

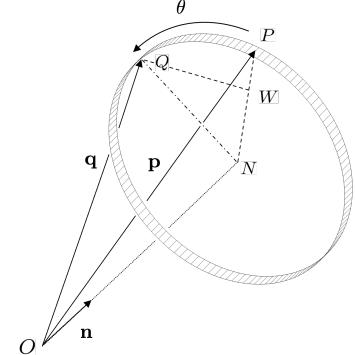


Figure 2.3: Right-handed rotation of a vector \mathbf{p} about the unit vector \mathbf{n} by an angle of θ to obtain the vector \mathbf{q} .

⁶ Brian L Stevens and Frank L Lewis. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2003

The vector \vec{ON} can be found by taking the projection of \mathbf{p} on the unit vector \mathbf{n} in the direction of \mathbf{n} , giving

$$\vec{ON} = (\mathbf{p} \cdot \mathbf{n}) \mathbf{n} = (\mathbf{n}\mathbf{n}^\top) \mathbf{p}.$$

The vector \vec{NW} is in the direction of $\mathbf{p} - \vec{ON}$ with a length of $\|\vec{NQ}\| \cos \theta$. Noting that the length NQ equals the length NP which is equal to $\|\mathbf{p} - \vec{ON}\|$ we get that

$$\begin{aligned} \vec{NW} &= \frac{\mathbf{p} - (\mathbf{p} \cdot \mathbf{n}) \mathbf{n}}{\|\mathbf{p} - (\mathbf{p} \cdot \mathbf{n}) \mathbf{n}\|} \|\vec{NQ}\| \cos \theta \\ &= \cos \theta (I - \mathbf{n}\mathbf{n}^\top) \mathbf{p}. \end{aligned}$$

The vector \vec{WQ} is perpendicular to both \mathbf{p} and \mathbf{n} and has length $\|\vec{NQ}\| \sin \theta$. Noting that $\|\vec{NQ}\| = \|\mathbf{p}\| \sin \phi$ and that $\|\mathbf{n} \times \mathbf{p}\| = \|\mathbf{p}\| \sin \phi$, where ϕ is the angle between \mathbf{n} and \mathbf{p} , we get

$$\begin{aligned} \vec{WQ} &= \frac{\mathbf{n} \times \mathbf{p}}{\|\mathbf{n} \times \mathbf{p}\|} \|\vec{NQ}\| \sin \theta \\ &= \mathbf{n} \times \mathbf{p} \sin \theta \\ &= \sin \theta [\mathbf{n}]_\times \mathbf{p}. \end{aligned}$$

Therefore Equation (2.24) becomes

$$\mathbf{q} = \left(\mathbf{n}\mathbf{n}^\top + \cos \theta (I - \mathbf{n}\mathbf{n}^\top) + \sin \theta [\mathbf{n}]_\times \right) \mathbf{p}.$$

Let

$$R(\mathbf{n}, \theta) = \mathbf{n}\mathbf{n}^\top + \cos \theta (I - \mathbf{n}\mathbf{n}^\top) + \sin \theta [\mathbf{n}]_\times,$$

and note that using Equation (2.14) we can write

$$R(\mathbf{n}, \theta) = I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2, \quad (2.25)$$

Equation (2.14) is

$$([\mathbf{n}]_\times)^2 = \mathbf{n}\mathbf{n}^\top - I.$$

which is called the Rodrigues formula. As described in the next section, we will often use angle-axis notation to define rotations. In angle-axis notation, the rotation vector is given by $\mathbf{a} = \theta \mathbf{n}$. Therefore, we will typically write the Rodrigues formula as

$$R(\theta \mathbf{n}) = I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2. \quad (2.26)$$

Note that for any vector \mathbf{a} we have $\mathbf{a} = \|\mathbf{a}\| \frac{\mathbf{a}}{\|\mathbf{a}\|}$. Equating θ with $\|\mathbf{a}\|$ and \mathbf{n} with the unit vector $\mathbf{a}/\|\mathbf{a}\|$, the Rodrigues formula becomes

$$\begin{aligned} R(\mathbf{a}) &= I + \left(\frac{\sin \|\mathbf{a}\|}{\|\mathbf{a}\|} \right) [\mathbf{a}]_\times + \left(\frac{1 - \cos \|\mathbf{a}\|}{\|\mathbf{a}\|^2} \right) [\mathbf{a}]_\times^2 \\ &= I + \text{sinc}(\|\mathbf{a}\|) [\mathbf{a}]_\times + \frac{1}{2} \text{sinc}^2 \left(\frac{\|\mathbf{a}\|}{2} \right) [\mathbf{a}]_\times^2. \end{aligned} \quad (2.27)$$

We have used the identity

$1 - \cos x = 2 \sin^2(x/2)$ to get $\frac{1 - \cos x}{x^2} = \frac{1}{2} \text{sinc}^2(x/2)$, where $\text{sinc}(x) = \frac{\sin(x)}{x}$.

As an illustration of the Rodrigues formula, we get that a rotation of ϕ about the vector $\mathbf{e}_1 = (1, 0, 0)^\top$ is

$$\begin{aligned} R(\mathbf{e}_1, \phi) &= I + \sin \phi [\mathbf{e}]_\times + (1 - \cos \phi) [\mathbf{e}_1]_\times^2 \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \sin \phi \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} + (1 - \cos \phi) \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{pmatrix}. \end{aligned}$$

Similarly, the Rodrigues formula gives that rotations by θ about \mathbf{e}_2 and ψ about \mathbf{e}_3 are

$$\begin{aligned} R(\mathbf{e}_2, \theta) &= \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \\ R(\mathbf{e}_3, \psi) &= \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

One may wonder if all rotation can be described by the Rodrigues formula. It turns out that Leonhard Euler proved in 1775 the following theorem.⁷

Theorem 2.3.1 (Euler's Rotation Theorem) *In three dimensional space, any sequence of rotations is equivalent to a single rotation about a fixed vector.*

Therefore, any rotation and any sequence of rotations, can be represented by a single rotation matrix with some angle θ and some unit vector \mathbf{n} , where the corresponding rotation is $R(\mathbf{n}, \theta)$. It turns out that the eigenvalues and eigenvectors of any rotation $R(\mathbf{n}, \theta)$ can be derived analytically.

Lemma 2.3.2 *Given a rotation matrix $R(\mathbf{n}, \theta)$ the eigenvalues are $\lambda = 1, \exp(j\theta), \exp(-j\theta)$, with corresponding eigenvectors \mathbf{n} , $\mathbf{n}_2 + j\mathbf{n}_3$, and $\mathbf{n}_2 - j\mathbf{n}_3$, where $\{\mathbf{n}, \mathbf{n}_2, \mathbf{n}_3\}$ form a right handed coordinate frame.*

Proof: From the Rodrigues formula we get

$$R(\mathbf{n}, \theta)\mathbf{n} = \mathbf{n} + \sin \theta [\mathbf{n}]_\times \mathbf{n} + (1 - \cos \theta) [\mathbf{n}]_\times [\mathbf{n}]_\times \mathbf{n} = 1 \cdot \mathbf{n}.$$

Therefore $\lambda = 1$ is an eigenvector of $R(\mathbf{n}, \theta)$ with eigenvector \mathbf{n} .

Pick any \mathbf{n}_2 that is orthogonal to \mathbf{n} , and let $\mathbf{n}_3 = [\mathbf{n}]_\times \mathbf{n}_2$ so that

⁷ https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem.

We have used the fact that $\mathbf{n} \times \mathbf{n} = 0$ implies that $[\mathbf{n}]_\times \mathbf{n} = 0$.

$\{\mathbf{n}, \mathbf{n}_2, \mathbf{n}_3\}$ form a right-handed coordinate system. Then

$$\begin{aligned} R(\mathbf{n}, \theta)\mathbf{n}_2 &= \mathbf{n}_2 + \sin \theta [\mathbf{n}]_{\times} \mathbf{n}_2 + (1 - \cos \theta) [\mathbf{n}]_{\times} [\mathbf{n}]_{\times} \mathbf{n}_2 \\ &= \mathbf{n}_2 + \sin \theta \mathbf{n}_3 - (1 - \cos \theta) \mathbf{n}_2 \\ &= \cos \theta \mathbf{n}_2 + \sin \theta \mathbf{n}_3. \end{aligned}$$

Similarly, it is straightforward to show that $R(\mathbf{n}, \theta)\mathbf{n}_3 = \cos \theta \mathbf{n}_3 - \sin \theta \mathbf{n}_2$. Therefore

$$\begin{aligned} R(\mathbf{n}, \theta)(\mathbf{n}_2 \pm j\mathbf{n}_3) &= (\cos \theta \mathbf{n}_2 + \sin \theta \mathbf{n}_3) \pm j(\cos \theta \mathbf{n}_3 - j \sin \theta \mathbf{n}_2) \\ &= (\cos \theta \pm j \sin \theta) \mathbf{n}_2 \mp j(\cos \theta \pm j \sin \theta) \mathbf{n}_3 \\ &= e^{\pm j\theta}(\mathbf{n}_2 \mp j\mathbf{n}_3). \end{aligned}$$

Therefore, the other two eigenvectors are $\mathbf{n}_2 \pm j\mathbf{n}_3$ with associated eigenvalues of $e^{\mp j\theta}$. Conversely, it can be shown that all rotation matrices have a similar eigen-structure. Therefore, every rotation matrix $R \in SO(3)$ will have eigenvalues of 1 and $e^{\mp j\theta}$, and R can be visualized as rotating a vector \mathbf{p} by a left-handed rotation of angle θ about the eigenvector \mathbf{n} associated with eigenvalue 1. ■

It is standard convention to call \mathbf{n} the eigen-axis of rotation.

The Rodrigues formula also gives the following result for the trace of R .

Lemma 2.3.3 *For any $R(\mathbf{n}, \theta) \in SO(3)$ we have*

$$\text{tr}(R) = 1 + 2 \cos \theta.$$

Proof: Since $[\mathbf{n}]_{\times}^2 = \mathbf{n}\mathbf{n}^T - I$, we get that

$$\begin{aligned} \text{tr}(R) &= \text{tr}(I) + \sin \theta \text{tr}([\mathbf{n}]_{\times}) + (1 - \cos \theta) \text{tr}(\mathbf{n}\mathbf{n}^T - I) \\ &= 3 + \sin \theta \cdot 0 + (1 - \cos \theta)(n_1^2 + n_2^2 + n_3^2 - 3) \\ &= 1 + 2 \cos \theta, \end{aligned}$$

where $\mathbf{n} = (n_1, n_2, n_3)^T$. ■

We conclude this section by deriving an extremely useful expression for the Rodrigues formula. Recall from linear algebra, that the exponential of a square matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$e^A = \exp(A) = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \frac{1}{4!}A^4 + \dots \quad (2.28)$$

Lemma 2.3.4 *For any unit vector \mathbf{n} and scalar θ ,*

$$R(\mathbf{n}, \theta) = \exp(\theta [\mathbf{n}]_{\times}).$$

Proof: From Equation (2.14) we have that

$$\begin{aligned} [\mathbf{n}]_{\times}^3 &= [\mathbf{n}]_{\times} (\mathbf{n}\mathbf{n}^{\top} - I) \\ &= -[\mathbf{n}]_{\times}, \end{aligned}$$

implying that for $m = 0, 1, 2, \dots$, $[\mathbf{n}]_{\times}^{(2m+1)} = (-1)^m [\mathbf{n}]_{\times}$. Therefore $[\mathbf{n}]_{\times}^4 = -[\mathbf{n}]_{\times}^2$, implying that for $m = 0, 1, 2, \dots$, $[\mathbf{n}]_{\times}^{(2m+2)} = (-1)^m [\mathbf{n}]_{\times}^2$. Using Equation (2.28) we get

$$\begin{aligned} \exp(\theta [\mathbf{n}]_{\times}) &= I + \theta [\mathbf{n}]_{\times} + \frac{\theta^2}{2!} [\mathbf{n}]_{\times}^2 + \frac{\theta^3}{3!} [\mathbf{n}]_{\times}^3 + \frac{\theta^4}{4!} [\mathbf{n}]_{\times}^4 + \dots \\ &= I + \theta [\mathbf{n}]_{\times} + \frac{\theta^2}{2!} [\mathbf{n}]_{\times}^2 - \frac{\theta^3}{3!} [\mathbf{n}]_{\times} - \frac{\theta^4}{4!} [\mathbf{n}]_{\times}^2 + \dots \\ &= I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots \right) [\mathbf{n}]_{\times} + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \dots \right) [\mathbf{n}]_{\times}^2 \\ &= I + \sin \theta [\mathbf{n}]_{\times} - (\cos \theta - 1) [\mathbf{n}]_{\times}^2 \\ &= R(\mathbf{n}, \theta). \end{aligned}$$

■

We can also define the log of a rotation matrix as the inverse operation, where we require that $\log(\exp(R)) = R$.

Lemma 2.3.5 *Given any $R \in SO(3)$, the logarithm of R is given by*

$$\log(R) = \frac{1}{2} \frac{R - R^{\top}}{\text{sinc } \theta}$$

where

$$\theta = \cos^{-1} \left(\frac{\text{tr}(R) - 1}{2} \right).$$

Proof: Euler's rotation theorem implies that there exist \mathbf{n} and θ such that $R = \exp(\theta [\mathbf{n}]_{\times})$. Therefore

$$\log(R) = \log(\exp(\theta [\mathbf{n}]_{\times})) = \theta [\mathbf{n}]_{\times}.$$

From Lemma 2.3.3 we have that

$$\theta = \cos^{-1} \left(\frac{\text{tr}(R) - 1}{2} \right).$$

Using the Rodrigues formula we have⁸

$$\begin{aligned} R - R^{\top} &= I + \sin \theta [\mathbf{n}]_{\times} + (1 - \cos \theta) [\mathbf{n}]_{\times}^2 \\ &\quad - I^{\top} - \sin \theta [\mathbf{n}]_{\times}^{\top} - (1 - \cos \theta) [\mathbf{n}]_{\times}^{\top 2} \\ &= 2 \sin \theta [\mathbf{n}]_{\times}, \end{aligned}$$

which implies that⁹

$$[\mathbf{n}]_{\times} = \frac{R - R^{\top}}{2 \sin \theta}.$$

Therefore

$$\log R = \theta [\mathbf{n}]_{\times} = \theta \frac{R - R^{\top}}{2 \sin \theta} = \frac{1}{2} \frac{R - R^{\top}}{\text{sinc } \theta}.$$

⁸ We have used the fact that $[\mathbf{n}]_{\times}^{\top} = -[\mathbf{n}]_{\times}$ and $[\mathbf{n}]_{\times}^{\top 2} = [\mathbf{n}]_{\times}^2$.

⁹ Note that this formula shows that an equivalent expression for the logarithm of R is $\log(R) = \frac{1}{2} \frac{R - R^{\top}}{\text{sinc } \theta}$ where $\text{sinc } \theta = \frac{\sin \theta}{\theta}$.

■

2.4 Attitude Representation

In this section we describe four different attitude representations that will be used in the book. The attitude or orientation of a multirotor can be described using Euler angles, a rotation matrix, a unit quaternion, or using a rotation vector, which is often called angle-axis representation. The following subsection will describe each of these conventions and how they are used to describe the attitude of a multirotor, and the attitude of an attached camera. We will find that each of these representations are convenient for different aspects of vision-based estimation and control, and so we also describe how to transform between the different representations.

2.4.1 Euler Angles

The most common representation for attitude is the 3-2-1 Euler angles, often called the aircraft yaw, pitch, and roll angles. The body frame of a multirotor is defined as $\mathcal{F}_b = \{\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b\}$. The body \mathbf{i}_b -axis is in a plane parallel to the rotors, and points in a direction that is designated as the forward-direction for the multirotor. The body \mathbf{k}_b -axis is perpendicular to the rotor plane, and points opposite the direction of thrust, designated as the body down direction. The body \mathbf{j}_b -axis is so that $\mathbf{j} = \mathbf{k} \times \mathbf{i}$ and points to the right in the body.

The yaw angle of a multirotor is defined as a right-handed rotation about the inertial \mathbf{k}_i axis, as shown in Figure 2.4. The resulting intermediary frame is called the body-1 frame in ¹⁰. The yaw angle is designated as ψ .

The pitch angle of a multirotor is defined as a right-handed rotation about the \mathbf{j}_1 in the body-1 frame, as shown in Figure 2.5, and the resulting intermediary frame is called the body-2 frame in ¹¹. The pitch angle is designated as θ .

The roll angle of a multirotor is defined as a right-handed rotation about the body-2 \mathbf{i}_2 axis, as shown in Figure 2.6. The roll angle is designated as ϕ .

Throughout the book, we will designate the vector of Euler angles by

$$\Theta = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix}.$$

Euler angle are often used because they are intuitive to understand and easy to visualize. When attitude needs to be plotted in a graph, the Euler angles are typically used for convenience. However, Euler angles represent a local representation of attitude and are rarely suitable for simulation, estimation, or control. First note that Euler angles do not commute, i.e., a yaw followed by a pitch followed by a roll, is

¹⁰ Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012

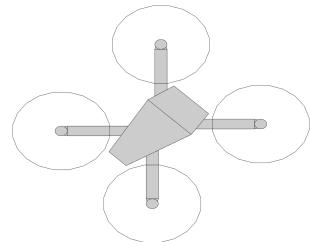


Figure 2.4: The yaw angle is defined as a right-handed rotation about the inertial \mathbf{k}_i axis.

¹¹ Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012

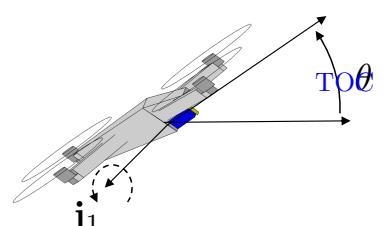


Figure 2.5: The pitch angle is

different than a roll followed by a pitch followed by a yaw. This non-commutativity has many negative implications. To understand that they are a local representation, consider the attitude that would result by first pitching the multirotor by 90 degrees, and then rolling by 90 degrees. This is the same orientation that would result from first yawing by 90 degrees, and then pitching by 90 degrees. Therefore Euler angle representations do not uniquely represent the attitude.

The pointing angle, or attitude of the camera relative to the multirotor can be described by the azimuth and elevation angles. The azimuth angle of the camera is defined as a right-handed rotation about the multirotor body \mathbf{k}_b axis, as shown in Figure 2.7. The resulting intermediary frame is called the gimbal-1 frame in ¹². The azimuth angle is designated in this book as α .

The elevation angle of the camera is defined as a right-handed rotation about the gimbal-1 \mathbf{j}_{g1} axis, as shown in Figure 2.8. The resulting frame is called the gimbal frame. The elevation angle is designated in this book as β .

2.4.2 Rotation Matrices

As discussed previously, the attitude of a multirotor can also be represented by a rotation matrix. A matrix $M \in \mathbb{R}^{n \times n}$ is said to be *orthogonal* if $MM^\top = M^\top M = I$, or equivalently if $M^{-1} = M^\top$. The set of n -dimensional orthogonal matrices are designated by

$$O(n) = \{M \in \mathbb{R}^{n \times n} | M^{-1} = M^\top\}.$$

If $M = (\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_n) \in O(n)$, then

$$M^\top M = \begin{pmatrix} \mathbf{m}_1^\top \mathbf{m}_1 & \dots & \mathbf{m}_1^\top \mathbf{m}_n \\ \vdots & & \vdots \\ \mathbf{m}_n^\top \mathbf{m}_1 & \dots & \mathbf{m}_n^\top \mathbf{m}_n \end{pmatrix} = I$$

implies that the columns of M are orthogonal to each other and have unit length. From properties of the determinate $\det(M^\top) = \det(M)$ and $\det(I) = 1$, we have that

$$\begin{aligned} \det(M^\top M) &= \det(I) \\ \iff \det^2(M) &= 1 \\ \iff \det(M) &= \pm 1. \end{aligned}$$

For 3×3 matrices, Equation (2.8) implies that $\det(M) = \mathbf{m}_1^\top (\mathbf{m}_2 \times \mathbf{m}_3)$. Therefore $\det(M) = 1$ if and only if the columns of M form a right-handed coordinate system ($\mathbf{m}_2 \times \mathbf{m}_3 = \mathbf{m}_1$), otherwise the determinant is equal to -1 . Therefore we define the set of special orthogonal matrices

$$SO(n) = \{M \in \mathbb{R}^{n \times n} | M \in O(n) \text{ and } \det(M) = 1\},$$

¹² Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012

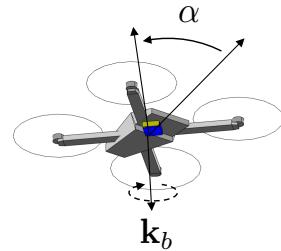


Figure 2.7: The azimuth angle of the camera is defined as a right-handed rotation about the multirotor body \mathbf{k}_b axis.

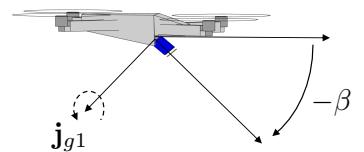


Figure 2.8: The elevation angle of the camera is defined as a right-handed rotation about the gimbal-1 \mathbf{j}_{g1} axis.

and note that $R \in SO(3)$ implies that the columns of R form a right-handed coordinate system. Therefore, there is a one-to-one relationship between the orientation of right-handed coordinate systems and elements of $SO(3)$. We say that R is a rotation matrix if $R \in SO(3)$. Note that the set $SO(3)$ does not form a vector space since $aR_1 + bR_2 \notin SO(3)$ when $R_1, R_2 \in SO(3)$. We will see later that elements of $SO(3)$ form a *group*, and which is called the *special orthogonal group of order 3*.

Therefore, the orientation of a multirotor can be represented by an element of $SO(3)$. If $\mathcal{F}_b = \{\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b\}$ is a right-handed coordinate frame fixed in the multirotor body, then the attitude of \mathcal{F}_b relative to another frame $\mathcal{F}_a = \{\mathbf{i}_a, \mathbf{j}_a, \mathbf{k}_a\}$ is given by

$$R_b^a = \begin{pmatrix} \mathbf{i}_b^a & \mathbf{j}_b^a & \mathbf{k}_b^a \end{pmatrix}$$

where the columns of R_b^a are the unit vectors of frame \mathcal{F}_b expressed relative to frame \mathcal{F}_a . We will be particularly interested in R_b^i , the attitude of the multirotor body relative to the inertial frame, where \mathbf{i}_b points out the nose of the multirotor, \mathbf{j}_b points to the right, and \mathbf{k}_b points down.

To represent the attitude of the gimbal relative to the body, we need to assign a right-handed coordinate system to the gimbal. Define $\mathcal{F}_g = \{\mathbf{i}_g, \mathbf{j}_g, \mathbf{k}_g\}$ such that \mathbf{i}_g points along the optical axis of the camera, \mathbf{k}_g points down when the gimbal is level, and $\mathbf{j}_g = \mathbf{k}_g \times \mathbf{i}_g$ points to the right. The attitude of the gimbal relative to the body is given by

$$R_g^b = \begin{pmatrix} \mathbf{i}_g^b & \mathbf{j}_g^b & \mathbf{k}_g^b \end{pmatrix}.$$

To gain further insights into rotation matrices, note that any vector \mathbf{p} can be represented in \mathcal{F}_r and \mathcal{F}_s as

$$\mathbf{p} = \alpha_r \mathbf{i}_r + \beta_r \mathbf{j}_r + \gamma_r \mathbf{k}_r = \alpha_s \mathbf{i}_s + \beta_s \mathbf{j}_s + \gamma_s \mathbf{k}_s.$$

Taking the inner product of both sides of this equation with the vectors \mathbf{i}_r , \mathbf{j}_r , and \mathbf{k}_r gives

$$\begin{aligned} \alpha_r &= \alpha_s \mathbf{i}_r^\top \mathbf{i}_s + \beta_s \mathbf{i}_r^\top \mathbf{j}_s + \gamma_s \mathbf{i}_r^\top \mathbf{k}_s \\ \beta_r &= \alpha_s \mathbf{j}_r^\top \mathbf{i}_s + \beta_s \mathbf{j}_r^\top \mathbf{j}_s + \gamma_s \mathbf{j}_r^\top \mathbf{k}_s \\ \gamma_r &= \alpha_s \mathbf{k}_r^\top \mathbf{i}_s + \beta_s \mathbf{k}_r^\top \mathbf{j}_s + \gamma_s \mathbf{k}_r^\top \mathbf{k}_s. \end{aligned}$$

Since $\mathbf{p}^r = (\alpha_r, \beta_r, \gamma_r)^\top$ and $\mathbf{p}^s = (\alpha_s, \beta_s, \gamma_s)^\top$ we have

$$\begin{pmatrix} \alpha_r \\ \beta_r \\ \gamma_r \end{pmatrix} = \begin{pmatrix} \mathbf{i}_r^\top \mathbf{i}_s & \mathbf{i}_r^\top \mathbf{j}_s & \mathbf{i}_r^\top \mathbf{k}_s \\ \mathbf{j}_r^\top \mathbf{i}_s & \mathbf{j}_r^\top \mathbf{j}_s & \mathbf{j}_r^\top \mathbf{k}_s \\ \mathbf{k}_r^\top \mathbf{i}_s & \mathbf{k}_r^\top \mathbf{j}_s & \mathbf{k}_r^\top \mathbf{k}_s \end{pmatrix} \begin{pmatrix} \alpha_s \\ \beta_s \\ \gamma_s \end{pmatrix},$$

or in other words $\mathbf{p}^r = R_s^r \mathbf{p}^s$, where we see that since $\mathbf{n}^\top \mathbf{m} = \cos \theta$

The Cauchy-Schwartz equality says that for any two vectors \mathbf{a} and \mathbf{b} , $\mathbf{a}^\top \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$ where θ is the angle between \mathbf{a} and \mathbf{b} . TOC

where θ is the angle between \mathbf{n} and \mathbf{m} , that R_s^r is a matrix of cosines of angles between the unit vectors of frame r and frame s . Therefore, rotation matrices are often called *direction cosine matrices*. Therefore the vector

$$\mathbf{i}_s^r = \begin{pmatrix} \text{angle between } \mathbf{i}_s \text{ and } \mathbf{i}_r \\ \text{angle between } \mathbf{i}_s \text{ and } \mathbf{j}_r \\ \text{angle between } \mathbf{i}_s \text{ and } \mathbf{k}_r \end{pmatrix}.$$

To emphasize some of the concept above, let $\mathcal{R}_r = \{\mathbf{i}_r, \mathbf{j}_r, \mathbf{k}_r\}$ and let $\mathcal{R}_s = \{\mathbf{i}_s, \mathbf{j}_s, \mathbf{k}_s\}$, then note that

$$\mathbf{i}_s^s = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{j}_s^s = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{k}_s^s = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Given the discussion above we have that

$$R_s^r = (\mathbf{i}_s^r \quad \mathbf{j}_s^r \quad \mathbf{k}_s^r),$$

or

$$\mathbf{i}_s^r = R_s^r \mathbf{i}_s^s = (\mathbf{i}_s^r \quad \mathbf{j}_s^r \quad \mathbf{k}_s^r) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Therefore the columns of R_s^r are the unit axes of frame \mathcal{F}_s expressed with respect to frame \mathcal{F}_r .

Let \mathbf{p}_q , \mathbf{p}_r and \mathbf{p}_s be the coordinates of \mathbf{p} when expressed with respect to frames \mathcal{F}_q , \mathcal{F}_r , and \mathcal{F}_s , respectively. Then we can write

$$\begin{aligned} \mathbf{p}^r &= R_s^r \mathbf{p}^s \\ \mathbf{p}^q &= R_r^q \mathbf{p}^r. \end{aligned}$$

Therefore we have that $\mathbf{p}^q = R_r^q R_s^r \mathbf{p}^s$. Since we also know that $\mathbf{p}_q = R_s^q \mathbf{p}_s$, we must have that

$$R_s^q = R_r^q R_s^r.$$

We have shown that rotation matrices satisfy the following properties

$$(R_r^s)^{-1} = (R_r^s)^T = R_s^r \quad (2.29)$$

$$R_r^q R_s^r = R_s^q \quad (2.30)$$

$$\det R_s^r = 1. \quad (2.31)$$

2.4.3 Passive Versus Active Rotations

We are using a convention here, where we define R as a *passive* rotation. Consider Equation ???. The vector quantity \mathbf{r} did not change during this rotation, it is simply being expressed from another point of

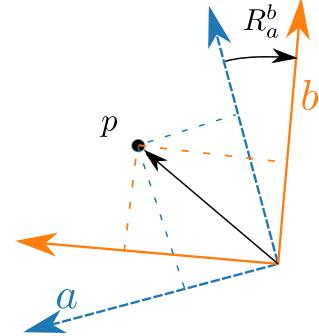


Figure 2.9: Illustration of a passive rotation.

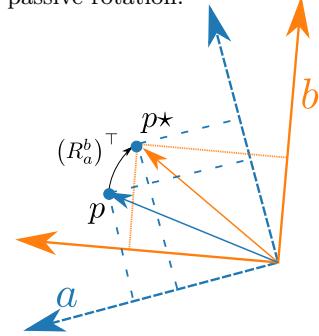


Figure 2.10: Illustration of an active rotation.

view. Some literature defines rotations as *active*. Figures 2.9 and 2.10 illustrates the difference between active and passive rotations. In the passive case, the point p does not change when rotating from frame a to b . It remains fixed while the perspective changes. In contrast, the active case, the point p is moved to become a new object, point \mathbf{p}^* . This interpretation is much more common in the computer graphics literature, where a common operation is to generate some asset at an arbitrary origin, and then move that object to be rendered at some other location. Engineering and robotics literature typically deal with passive rotations, where the goal is often to reason about physical entities that are not always directly controlled.

It can be shown that active and passive rotations are the transpose of one another, as

$$R_{\text{active}} = R_{\text{passive}}^\top.$$

Throughout the book, we will always use passive rotations unless otherwise stated.

While rotation matrices are probably the most straight-forward method to represent rotations, they have some shortcomings. The first is that they require nine parameters to describe three degrees of freedom. This causes larger memory requirements than strictly necessary, and requires more operations than necessary when concatenating matrices. With modern computing resources, this is less of a problem, however the biggest disadvantage is that numerical errors can accumulate in rotation matrices and cause unwanted scaling and shearing as the matrix loses orthonormality. This error can be corrected by Gram-Schmidt orthogonalization, but doing so can be computationally expensive. As a result, rotation matrices are still much more efficient than an Euler angle representation, but they are not as efficient as unit quaternions.

2.4.4 Quaternions

The material in this section loosely follows the notation in ¹³.

One of the disadvantages of rotation matrices is that they require nine parameters to represent a rotation, which is fundamentally a three dimensional entity. Unit quaternions can be used to represent rotations using four parameters. Quaternions are hyper-complex numbers of the form

$$\mathbf{q} = q_0 + iq_x + jq_y + kq_z,$$

where i , j , and k are generalized complex numbers that satisfy

$$i^2 = j^2 = k^2 = ijk = -1. \quad (2.32)$$

¹³ Nikolas Trawny and Stergios I. Roumeliotis. Indirect kalman filter for 3d attitude estimation a tutorial for quaternion algebra. Technical Report 2005-002, Rev. 57, University of Minnesota, March 2005

Because of the reverse order of concatenation, there is actually another convention for quaternions, first used by NASA JPL, where the quaternion imaginary numbers follow the *left-hand* rule instead of the right hand rule. This change makes unit quaternions concatenate right-to-left (to match rotation matrices). The left-handed definition is known as JPL notation, whereas the right-handed definition presented here is known as Hamilton notation. While this change may seem innocent on the surface, it has huge implications when we start talking about Lie Groups. Modern scientific literature in robotics and computer vision that use quaternions seem to almost always use Hamilton notation, although JPL is used in some early quaternion papers relating to spacecraft attitude observers.

Equation (2.32) implies that $ijk^2 = -k$ which implies that $ij = k$, which implies that $ij^2 = kj$, which implies that $kj = -i$. By similar arguments we can show that $ij = -ji = k$, $jk = -kj = i$ and $ki = -ik = j$. Letting $\mathbf{p} = p_0 + ip_x + jp_y + kp_z$, we can multiply \mathbf{p} and \mathbf{q} , and use the above relationships for i, j, k to get

$$\begin{aligned}\mathbf{p}\mathbf{q} &= (p_0 + ip_x + jp_y + kp_z)(q_0 + iq_x + jq_y + kq_z) \\ &= (p_0q_0 - p_xq_x - p_yq_y - p_zq_z) + i(p_0q_x + p_xq_0 + p_yq_z - p_zq_y)\end{aligned}\tag{2.33}$$

$$+ j(p_0q_y - p_xq_z + p_yq_0 + p_zq_x) + k(p_0q_z + p_xq_y - p_yq_x + p_zq_0)\tag{2.34}$$

Throughout this book we will denote quaternions as a four vector

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \bar{q} \end{pmatrix},$$

where q_0 is called the scalar part of the quaternion, and $\bar{q} = (q_x, q_y, q_z)^\top$, representing the complex elements, is called the vector part of the quaternion. In vector notation, the quaternion product in Equation (2.34) becomes

$$\mathbf{p} \otimes \mathbf{q} = \begin{pmatrix} p_0 & -\bar{p}^\top \\ \bar{p} & p_0I + [\bar{p}]_\times \end{pmatrix} \begin{pmatrix} q_0 \\ \bar{q} \end{pmatrix}\tag{2.35}$$

$$= \begin{pmatrix} q_0 & -\bar{q}^\top \\ \bar{q} & q_0I - [\bar{q}]_\times \end{pmatrix} \begin{pmatrix} p_0 \\ \bar{p} \end{pmatrix}.\tag{2.36}$$

It turns out that attitude can be represented using unit quaternions. Let

$$S^3 = \{\mathbf{q} \in \mathbb{R}^4 \mid \|\mathbf{q}\| = 1\}$$

be the 3-dimensional unit sphere in \mathbb{R}^4 . Unit quaternions are elements of S^3 . We showed in Section ?? that a rotation matrix $R \in SO(3)$ has eigenvalues at 1, $e^{\mp j\theta}$, and that R represents a right-handed rotation of angle θ about the unit length eigenvector \mathbf{n} associated with the eigenvalue at $\lambda = 1$. The unit quaternion associated with $R(\mathbf{n}, \theta)$ is given by

$$\mathbf{q} = \begin{pmatrix} \cos \frac{\theta}{2} \\ \mathbf{n} \sin \frac{\theta}{2} \end{pmatrix}.$$

Conversely, any unit length four vector $\bar{\mathbf{q}} = (q_0, q_x, q_y, q_z)^\top \in S^3$ can be thought of as a unit quaternion representing a rotation of angle $\theta = 2\cos^{-1} q_0$ about the axis given by $\mathbf{n} = (q_x, q_y, q_z)^\top / \sqrt{q_x^2 + q_y^2 + q_z^2}$. Unit quaternions are therefore much easier to visualize than rotation matrices. Unit quaternions however, represent a double cover of $SO(3)$

since a rotation θ about \mathbf{n} is identical to a rotation of $2\pi - \theta$ about $-\mathbf{n}$. Therefore the unit quaternion

$$\mathbf{q} = \begin{pmatrix} \cos \frac{2\pi - \theta}{2} \\ (-\mathbf{n}) \sin \frac{2\pi - \theta}{2} \end{pmatrix} = \begin{pmatrix} -\cos \frac{\theta}{2} \\ -\mathbf{n} \sin \frac{\theta}{2} \end{pmatrix} = - \begin{pmatrix} \cos \frac{\theta}{2} \\ \mathbf{n} \sin \frac{\theta}{2} \end{pmatrix}$$

represents the same rotation as $(\cos \frac{\theta}{2}, \mathbf{n}^\top \sin \frac{\theta}{2})^\top = -\mathbf{q}$. To remove the ambiguity, by convention, we will use always use the quaternion with positive first element to represent the rotation.

The unit quaternion conjugate, or inverse is equivalent to conjugating the complex portion of the quaternion to get

$$\mathbf{q}^{-1} = \begin{pmatrix} q_0 \\ -\bar{q} \end{pmatrix}.$$

Unit quaternions, unlike rotation matrices, multiply from left to right, as

$$\mathbf{q}_s^q = \mathbf{q}_s^r \otimes \mathbf{q}_r^q.$$

To rotate a vector using a unit quaternion, a non-unit quaternion is formed with zero real-part and where the imaginary part is equal to the vector. If \mathbf{q}_s^r represents a rotation from frame s to frame r , then we have

$$\begin{pmatrix} 0 \\ \mathbf{r}^r \end{pmatrix} = (\mathbf{q}_s^r)^{-1} \otimes \begin{pmatrix} 0 \\ \mathbf{r}^s \end{pmatrix} \otimes \mathbf{q}_s^r. \quad (2.37)$$

Alternatively, Equation (2.35) can be used to obtain the following condensed expression

$$\begin{aligned} \mathbf{r}^r &= \text{rot}(\mathbf{q}_s^r, \mathbf{r}^s) \\ &= \mathbf{r}^s + 2q_0 [\mathbf{r}^s]_\times \bar{\mathbf{q}}_s^r + 2 [\mathbf{r}^s]_\times \bar{\mathbf{q}}_s^r \bar{\mathbf{q}}_s^r \\ &= [(q_0^2 - \|\bar{\mathbf{q}}_s^r\|^2)I + 2\bar{\mathbf{q}}_s^r \bar{\mathbf{q}}_s^{r\top} + 2q_0 [\bar{\mathbf{q}}_s^r]_\times] \mathbf{r}^s \end{aligned}$$

If implemented efficiently, this method can be just as efficient as the matrix-vector multiplication used when rotating a vector with a rotation matrix.

2.4.5 Angle-axis Representation

In robotics, and especially in computer vision¹⁴, the so-called angle-axis representation is used extensively.

We have seen in Section 2.3 that any rotation matrix can be expressed as

$$R(\mathbf{n}, \theta) = I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2.$$

The angle-axis representation uses the *rotation vector*

$$\mathbf{r} = \theta \mathbf{n}$$

¹⁴ openCV uses angle-axis representation for rotations in many of its routines.

to represent rotations. From Equation (2.27) we have

$$R(\mathbf{r}) = I + \text{sinc}(\|\mathbf{r}\|) [\mathbf{r}]_{\times} + \frac{1}{2} \text{sinc}^2 \left(\frac{\|\mathbf{r}\|}{2} \right) [\mathbf{r}]_{\times}^2. \quad (2.38)$$

In addition, from Lemma 2.3.4 we have

$$R(\mathbf{r}) = \exp([\mathbf{r}]_{\times}).$$

The rotation vector \mathbf{r} is straight forward to visualize since the direction of \mathbf{r} is the rotation vector, and the length of \mathbf{r} specifies the angle or rotation. The disadvantage of using the rotation vector is that it represents a many-to-one mapping from \mathbb{R}^3 to $SO(3)$ since

$$\mathbf{r} = (\theta + 2\pi m)\mathbf{n}$$

represents the same rotation for any integer m .

2.4.6 Transforms between representations

Lemma 2.4.1 (Euler angles to rotation matrix) *Given the Euler angles ϕ, θ, ψ , the corresponding rotation matrix is*

$$R_b^i = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix},$$

where $c_\phi \triangleq \cos \phi$ and $s_\phi \triangleq \sin \phi$.

Proof: The rotation matrix R_b^i begins with a rotation of ψ about the inertial \mathbf{k} axis, followed by a rotation of θ about the body-1 \mathbf{j} axis, followed by a rotation of ϕ about the body \mathbf{i} axis. Therefore

$$\mathcal{R}_b^i = R(\mathbf{e}_1, \phi)R(\mathbf{e}_2, \theta)R(\mathbf{e}_3, \psi) \quad (2.39)$$

$$\begin{aligned} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}. \end{aligned} \quad (2.40)$$

■

Lemma 2.4.2 (Quaternion to rotation vector) *Given the quaternion $\mathbf{q} = (q_0, \bar{\mathbf{q}}^\top)^\top$, the corresponding rotation vector is*

$$\mathbf{r} = \frac{2 \cos^{-1}(q_0)}{\sqrt{1 - q_0^2}} \bar{\mathbf{q}}.$$

Proof: The quaternion is given by

$$\mathbf{q} = \begin{pmatrix} q_0 \\ \bar{\mathbf{q}} \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{n} \end{pmatrix}.$$

Therefore $\theta = 2 \cos^{-1}(q_0)$ and $\mathbf{n} = \frac{1}{\sin(\theta/2)} \bar{q}$. The result follows from the fact that $\mathbf{r} = \theta \mathbf{n}$ and $\sin(\cos^{-1} q_0) = \sqrt{1 - q_0^2}$. ■

Lemma 2.4.3 (Euler angles to rotation vector) *Given the Euler angles ϕ, θ, ψ , the corresponding rotation vector is*

$$\mathbf{r} = \frac{2 \cos^{-1}(q_0)}{\sqrt{1 - q_0^2}} \bar{q}$$

where

$$q_0 = \cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2}$$

$$\bar{q} = \begin{pmatrix} \cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} \\ \cos \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} \\ \sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \end{pmatrix}.$$

Lemma 2.4.4 (Rotation matrix to Euler angles) *Given the rotation matrix*

$$R_b^i = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

Proof given at <https://www.gregslabaugh.net/publications/euler.pdf>.

the associated Euler angles are

$$\phi = \begin{cases} \text{atan2}\left(\frac{r_{32}}{\sqrt{1-r_{31}^2}}, \frac{r_{33}}{\sqrt{1-r_{31}^2}}\right) & r_{31} \neq \pm 1 \\ \text{atan2}(-r_{12}, -r_{13}) & r_{31} = +1 \\ \text{atan2}(r_{12}, r_{13}) & r_{31} = -1 \end{cases}$$

$$\theta = \begin{cases} -\sin^{-1}(r_{31}) & r_{31} \neq \pm 1 \\ -\frac{\pi}{2} & r_{31} = +1 \\ \frac{\pi}{2} & r_{31} = -1 \end{cases}$$

$$\psi = \begin{cases} \text{atan2}\left(\frac{r_{21}}{\sqrt{1-r_{31}^2}}, \frac{r_{11}}{\sqrt{1-r_{31}^2}}\right) & r_{31} \neq \pm 1 \\ 0 & r_{31} = +1 \\ 0 & r_{31} = -1 \end{cases}$$

Lemma 2.4.5 (Quaternion to rotation matrix) *Given the unit quaternion $\mathbf{q} = (q_0, \bar{q}^\top)^\top$, the associated rotation matrix is*

$$R = I + 2q_0 [\bar{q}]_\times + 2 [\bar{q}]_\times^2.$$

Proof: Writing the quaternion as

$$\begin{pmatrix} q_0 \\ \bar{q} \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{n} \end{pmatrix},$$

we have that

$$\begin{aligned}\sin(\Theta) &= \sin\left(\frac{\theta}{2} + \frac{\theta}{2}\right) \\ &= \sin \frac{\theta}{2} \cos \frac{\theta}{2} + \cos \frac{\theta}{2} \sin \frac{\theta}{2} \\ &= 2 \sin \frac{\theta}{2} \cos \frac{\theta}{2}.\end{aligned}$$

Therefore $\sin \theta [\mathbf{n}]_\times = 2 \cos \frac{\theta}{2} \left[\sin \frac{\theta}{2} \mathbf{n} \right]_\times = 2q_0 \bar{q}$. Similarly it is straightforward to show that $1 - \cos \theta = 2 \sin^2 \frac{\theta}{2}$, and therefore that $(1 - \cos \theta) [\mathbf{n}]_\times^2 = 2 \left[\sin \frac{\theta}{2} \mathbf{n} \right]_\times^2$. The result then follows from the Rodrigues formula. \blacksquare

Lemma 2.4.6 (Quaternion to Euler angles) *Given the unit quaternion $\mathbf{q}_b^i = (q_0, \bar{q}^\top)^\top = (q_0, q_x, q_y, q_z)^\top$, the associated Euler angles are*

$$\begin{aligned}\phi &= \tan^{-1} \left(\frac{2(q_y q_z + q_0 q_x)}{q_0^2 - q_x^2 - q_y^2 + q_z^2} \right) \\ \theta &= \sin^{-1} (2(q_0 q_y - q_x q_z)) \\ \psi &= \tan^{-1} \left(\frac{2(q_x q_y + q_0 q_z)}{q_0^2 + q_x^2 - q_y^2 - q_z^2} \right).\end{aligned}\tag{2.41}$$

Proof: From Lemma 2.4.5 we have

$$\begin{aligned}R(\mathbf{q}) &= (2q_0^2 - 1)I - 2q_0 \bar{q}^\wedge + 2\bar{q} \bar{q}^\top \\ &= \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y + q_0 q_z) & 2(q_x q_z - q_0 q_y) \\ 2(q_x q_y - q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z + q_0 q_x) \\ 2(q_x q_z + q_0 q_y) & 2(q_y q_z - q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix},\end{aligned}\tag{2.42}$$

where $R(\mathbf{q})$ is the rotation matrix associated with the unit quaternion \mathbf{q} . Equating equation $R(\mathbf{q})$ to Equation (2.40) gives

$$\begin{aligned}&\begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y + q_0 q_z) & 2(q_x q_z - q_0 q_y) \\ 2(q_x q_y - q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z + q_0 q_x) \\ 2(q_x q_z + q_0 q_y) & 2(q_y q_z - q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \\ &= \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\theta c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{pmatrix},\end{aligned}\tag{2.43}$$

and solving for Euler angles gives Equation (2.41). \blacksquare

Lemma 2.4.7 (Euler angles to unit quaternion) *Given the Euler angles ϕ, θ, ψ , the corresponding unit quaternion is*

$$\mathbf{q}_b^i = \begin{pmatrix} \cos \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \\ \cos \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} - \sin \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} \\ \cos \frac{\psi}{2} \sin \frac{\theta}{2} \cos \frac{\phi}{2} + \sin \frac{\psi}{2} \cos \frac{\theta}{2} \sin \frac{\phi}{2} \\ \sin \frac{\psi}{2} \cos \frac{\theta}{2} \cos \frac{\phi}{2} - \cos \frac{\psi}{2} \sin \frac{\theta}{2} \sin \frac{\phi}{2} \end{pmatrix}. \quad (2.44)$$

Proof: Solving Equation (2.43) for the quaternion (after significant algebra) gives Equation (2.44). \blacksquare

Lemma 2.4.8 (Rotation matrix to quaternion) *Given the rotation matrix*

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

the associated quaternion is¹⁵

$$\begin{aligned} q_0 &= \begin{cases} \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}, & \text{if } r_{11}+r_{22}+r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}-r_{21})^2+(r_{13}-r_{31})^2+(r_{23}-r_{32})^2}{3-r_{11}-r_{22}-r_{33}}}, & \text{otherwise} \end{cases} \\ q_x &= \text{sign}(r_{32}-r_{23}) \begin{cases} \frac{1}{2}\sqrt{1+r_{11}-r_{22}-r_{33}}, & \text{if } r_{11}-r_{22}-r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}+r_{21})^2+(r_{13}+r_{31})^2+(r_{23}+r_{32})^2}{3-r_{11}+r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \\ q_y &= \text{sign}(r_{13}-r_{31}) \begin{cases} \frac{1}{2}\sqrt{1-r_{11}+r_{22}-r_{33}}, & \text{if } -r_{11}+r_{22}-r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}+r_{21})^2+(r_{13}-r_{31})^2+(r_{23}+r_{32})^2}{3+r_{11}-r_{22}+r_{33}}}, & \text{otherwise} \end{cases} \\ q_z &= \text{sign}(r_{21}-r_{12}) \begin{cases} \frac{1}{2}\sqrt{1-r_{11}-r_{22}+r_{33}}, & \text{if } -r_{11}-r_{22}+r_{33} > 0 \\ \frac{1}{2}\sqrt{\frac{(r_{12}-r_{21})^2+(r_{13}+r_{31})^2+(r_{23}+r_{32})^2}{3+r_{11}+r_{22}-r_{33}}}, & \text{otherwise} \end{cases}. \end{aligned}$$

Proof: The result follows by equation R to Equation (2.42) and solving for the elements of \mathbf{q} . \blacksquare

¹⁵ Soheil Sarabandi and Federico Thomas. Accurate computation of quaternions from rotation matrices. In J. Lenarcic and V. Parenti-Castelli, editors, *Advances in Robot Kinematics*, pages 39–46. Springer, 2019.

Lemma 2.4.9 (Rotation matrix to rotation vector) *Given the rotation matrix R , the associated rotation vector is*

$$\mathbf{r} = \left(\frac{\cos^{-1} \left(\frac{1-tr(R)}{2} \right)}{\sqrt{(3-tr(R))(1+tr(R))}} \right) (R - R^\top)^\vee.$$

Proof: Follows from Lemma 2.3.5. \blacksquare

Lemma 2.4.10 (Rotation vector to quaternion) *Given the rotation vector \mathbf{r} , the associated quaternion is*

$$\mathbf{q} = \begin{pmatrix} \cos \left(\frac{\|\mathbf{r}\|}{2} \right) \\ \sin \left(\frac{\|\mathbf{r}\|}{2} \right) \frac{\mathbf{r}}{\|\mathbf{r}\|} \end{pmatrix}.$$

Proof: The result follows by writing $\mathbf{r} = \|\mathbf{r}\| \frac{\mathbf{r}}{\|\mathbf{r}\|} \doteq \theta \mathbf{n}$, and equating with the quaternion formula

$$\mathbf{q} = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{n} \end{pmatrix}.$$

■

Lemma 2.4.11 (Rotation vector to Euler angles) *Given the rotation vector $\mathbf{r} = (r_x, r_y, r_z)^\top$, the associated Euler angles are*

$$\begin{aligned} \phi &= \tan^{-1} \left(\frac{r_x \|\mathbf{r}\| \sin(\|\mathbf{r}\|) + r_y r_z (1 - \cos(\|\mathbf{r}\|))}{(r_x^2 + r_y^2) \cos(\|\mathbf{r}\|) + r_z^2} \right) \\ \theta &= \sin^{-1} \left(\frac{r_y \|\mathbf{r}\| \sin(\|\mathbf{r}\|) - r_x r_z (1 - \cos(\|\mathbf{r}\|))}{\|\mathbf{r}\|^2} \right) \\ \psi &= \tan^{-1} \left(\frac{r_x r_y (1 - \cos(\|\mathbf{r}\|)) + r_z \|\mathbf{r}\| \sin(\|\mathbf{r}\|)}{r_x^2 + (r_y^2 + r_z^2) \cos(\|\mathbf{r}\|)} \right). \end{aligned}$$

Proof: We will derive the result for ϕ and leave the derivation of the other angles to the reader. From Lemma 2.4.10 we have that

$$\mathbf{q}^\top = (q_0, q_x, q_y, q_z) = \left(\cos(\|\mathbf{r}\|/2), \frac{r_x}{\|\mathbf{r}\|} \sin(\|\mathbf{r}\|/2), \frac{r_y}{\|\mathbf{r}\|} \sin(\|\mathbf{r}\|/2), \frac{r_z}{\|\mathbf{r}\|} \sin(\|\mathbf{r}\|/2) \right)^\top,$$

and from Lemma 2.4.6 we have that

$$\phi = \tan^{-1} \left(\frac{2(q_y q_z + q_0 q_x)}{q_0^2 - q_x^2 - q_y^2 + q_z^2} \right).$$

Therefore

$$\begin{aligned} 2(q_y q_z + q_0 q_x) &= 2 \left(\frac{r_y r_z}{\|\mathbf{r}\|^2} \sin^2(\|\mathbf{r}\|/2) + \frac{r_x \|\mathbf{r}\|}{\|\mathbf{r}\|^2} \sin(\|\mathbf{r}\|/2) \cos(\|\mathbf{r}\|/2) \right) \\ &= \frac{1}{\|\mathbf{r}\|^2} [r_y r_z (1 - \cos(\|\mathbf{r}\|)) + r_x \|\mathbf{r}\| \sin(\|\mathbf{r}\|)], \end{aligned}$$

where we have used the facts that $\sin(A/2) \cos(A/2) = \frac{1}{2} \sin(A)$ and $\sin^2(A/2) = \frac{1}{2}(1 - \cos(A))$. We also have that

$$\begin{aligned} q_0^2 - q_x^2 - q_y^2 + q_z^2 &= \cos^2(\|\mathbf{r}\|/2) - \frac{r_x^2}{\|\mathbf{r}\|^2} \sin^2(\|\mathbf{r}\|/2) - \frac{r_y^2}{\|\mathbf{r}\|^2} \sin^2(\|\mathbf{r}\|/2) + \frac{r_z^2}{\|\mathbf{r}\|^2} \sin^2(\|\mathbf{r}\|/2) \\ &= \frac{1}{\|\mathbf{r}\|^2} \left[\|\mathbf{r}\|^2 (1 - \sin^2(\|\mathbf{r}\|/2)) + \sin^2(\|\mathbf{r}\|/2) (-r_x^2 - r_y^2 + r_z^2) \right] \\ &= \frac{1}{\|\mathbf{r}\|^2} [r_x^2 + r_y^2 + r_z^2 + \sin^2(\|\mathbf{r}\|/2) (-2r_x^2 - 2r_y^2)] \\ &= \frac{1}{\|\mathbf{r}\|^2} [r_x^2 + r_y^2 + r_z^2 + (1 - \cos(\|\mathbf{r}\|)) (-2r_x^2 - 2r_y^2)] \\ &= \frac{1}{\|\mathbf{r}\|^2} [(r_x^2 + r_y^2)(1 - \cos(\|\mathbf{r}\|)) + r_z^2]. \end{aligned}$$

■

2.5 Rotational Kinematics

In this section we discuss rotational kinematics, or how the rotational representation evolves in time, given the angular velocity of a frame. We begin with the Equation of Coriolis.

2.5.1 Equation of Coriolis

In this section we derive the famous equation of Coriolis. We will follow the derivation given in Stevens & Lewis¹⁶.

Suppose that we are given two coordinate frames \mathcal{F}_r and \mathcal{F}_s as shown in Figure 2.11. For example, \mathcal{F}_r might represent the inertial frame and \mathcal{F}_s might represent the body frame of a multicopter. Suppose that the vector \mathbf{p} is moving in \mathcal{F}_s and that \mathcal{F}_s is rotating and translating with respect to \mathcal{F}_r . Our objective is to find the time derivative of \mathbf{p} as seen from frame \mathcal{F}_r .

We will derive the appropriate equation through two steps. Assume first that \mathcal{F}_s is not rotating with respect to \mathcal{F}_r . Denoting the time derivative of \mathbf{p} in frame \mathcal{F}_r as $\frac{d\mathbf{p}}{dt_r}$, and the time derivative of \mathbf{p} in frame \mathcal{F}_s as $\frac{d\mathbf{p}}{dt_s}$ we get

$$\frac{d\mathbf{p}}{dt_r} = \frac{d\mathbf{p}}{dt_s}. \quad (2.45)$$

On the other hand, assume that \mathbf{p} is fixed in \mathcal{F}_s but that \mathcal{F}_s is rotating with respect to \mathcal{F}_r , and let $\boldsymbol{\omega}_{s/r}$ be the instantaneous axis of rotation and $\delta\phi$ the (right-handed) rotation angle. Then the Rodrigues formula (2.25) gives

$$\begin{aligned} \mathbf{p} + \delta\mathbf{p} &= \left[I + \sin(\delta\phi) [\mathbf{n}]_\times + (1 - \cos(\delta\phi)) [\mathbf{n}]_\times^2 \right] \mathbf{p} \\ &\approx \left[I + \delta\phi [\mathbf{n}]_\times + (\delta\phi^2) [\mathbf{n}]_\times^2 \right] \mathbf{p} \\ &\approx \left[I + \delta\phi [\mathbf{n}]_\times \right] \mathbf{p} \\ &= \mathbf{p} + (\delta\phi\mathbf{n}) \times \mathbf{p}, \end{aligned}$$

which implies, after dividing both sides by δt that

$$\frac{\delta\mathbf{p}}{\delta t} \approx \frac{\delta\phi}{\delta t} \mathbf{n} \times \mathbf{p}.$$

Taking the limit as $\delta t \rightarrow 0$ and defining the angular velocity of \mathcal{F}_s with respect to \mathcal{F}_r as $\boldsymbol{\omega}_{s/r} \triangleq \dot{\phi}\mathbf{n}$ we get

$$\frac{d\mathbf{p}}{dt_r} = \boldsymbol{\omega}_{s/r} \times \mathbf{p}. \quad (2.46)$$

Since differentiation is a linear operator we can combine Equations (2.45) and (2.46) to obtain

$$\frac{d\mathbf{p}}{dt_r} = \frac{d\mathbf{p}}{dt_s} + \boldsymbol{\omega}_{s/r} \times \mathbf{p}, \quad (2.47)$$

¹⁶ Brian L Stevens and Frank L Lewis. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2003

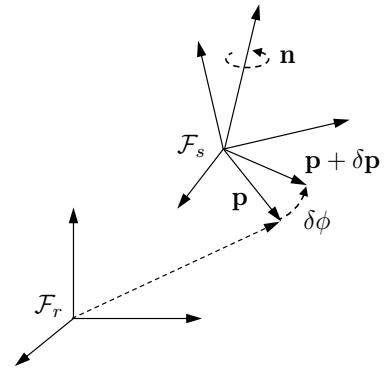


Figure 2.11: Derivation of the equation of Coriolis.

which is the equation of Coriolis.

Note that the vectors in Equation (2.47) have not yet been expressed with respect to a particular coordinate system. In fact, we can express each vector in Equation (2.47) with respect to any given coordinate frame. However, care needs to be taken to express the vectors **after** the differentiation operator. For example, expressing the equations in frame \mathcal{F}_f we have

$$\left(\frac{d\mathbf{p}}{dt_r} \right)^s = \left(\frac{d\mathbf{p}}{dt_s} \right)^s + \boldsymbol{\omega}_{s/r}^s \times \mathbf{p}^s.$$

Note however that $\left(\frac{d\mathbf{p}}{dt_r} \right)^s$ is not necessarily equal to $\frac{d(\mathbf{p}^s)}{dt_r}$. For example, suppose that \mathbf{i}_s is the unit vector along the x -axis of the s -frame that is undergoing a pure rotation denoted by $\boldsymbol{\omega}_{s/r}$. Then it is clear that $\left(\frac{d\mathbf{i}_s}{dt_r} \right)^s = \boldsymbol{\omega}_{s/r}^s \times \mathbf{i}_s^s \neq 0$. However, since $\mathbf{i}_s^s = (1, 0, 0)^\top$ is constant in the s -frame, $\frac{d(\mathbf{i}_s^s)}{dt_r} = 0$. It is true however that $\left(\frac{d\mathbf{p}}{dt_s} \right)^s = \frac{d(\mathbf{p}^s)}{dt_s}$. In general, if the time derivative of the vector is with respect to the same frame in which the vector is being expressed, then the derivative of the vector with respect to frame s expressed in frame s is equal to the derivative of the vector expressed in frame s with respect to frame s .

We also note that the derivative of a scalar is independent of the coordinate frame with which the derivative is being taken. In this book we will use the dot-notation to mean differentiation with respect to time, only when the derivative is applied component wise, and each component is being differentiated as a scalar. In that sense, if $\mathbf{p}^s = (x, y, z)^\top$, then

$$\frac{d\mathbf{p}^s}{dt_s} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}.$$

When vectors are expressed relative to a coordinate frame, then Equation (2.47) can be written as

$$\left(\frac{d\mathbf{p}}{dt_r} \right)^q = \left(\frac{d\mathbf{p}}{dt_s} \right)^q + [\boldsymbol{\omega}_{s/r}^q]_\times \mathbf{p}^q.$$

2.5.2 Rotational Kinematics

The Coriolis equation (2.47) facilitates the derivation of the rotational kinematics of a rigid body. Defining coordinate frame s as $\mathcal{F}_s = \{\mathbf{i}_s, \mathbf{j}_s, \mathbf{k}_s\}$, and suppose that frame s is rotated relative to frame r by angular velocity $\boldsymbol{\omega}_{s/r}$, then the Coriolis formula gives

$$\frac{d\mathbf{i}_s}{dt_r} = \frac{d\mathbf{i}_s}{dt_s} + \boldsymbol{\omega}_{s/r} \times \mathbf{i}_s, \quad \frac{d\mathbf{j}_s}{dt_r} = \frac{d\mathbf{j}_s}{dt_s} + \boldsymbol{\omega}_{s/r} \times \mathbf{j}_s, \quad \frac{d\mathbf{k}_s}{dt_r} = \frac{d\mathbf{k}_s}{dt_s} + \boldsymbol{\omega}_{s/r} \times \mathbf{k}_s.$$

Since the coordinate axes are stationary in their own frame, $\frac{d\mathbf{i}_s}{dt_s} = \frac{d\mathbf{j}_s}{dt_s} = \frac{d\mathbf{k}_s}{dt_s} = 0$, and we have

$$\frac{d\mathbf{i}_s}{dt_r} = \boldsymbol{\omega}_{s/r} \times \mathbf{i}_s, \quad \frac{d\mathbf{j}_s}{dt_r} = \boldsymbol{\omega}_{s/r} \times \mathbf{j}_s, \quad \frac{d\mathbf{k}_s}{dt_r} = \boldsymbol{\omega}_{s/r} \times \mathbf{k}_s.$$

Expressing all vectors in frame r and rewriting in matrix notation gives

$$\begin{aligned} \begin{bmatrix} \frac{d\mathbf{i}_s^r}{dt_r} & \frac{d\mathbf{j}_s^r}{dt_r} & \frac{d\mathbf{k}_s^r}{dt_r} \end{bmatrix} &= \left[[\boldsymbol{\omega}_{s/r}^r]_{\times} \mathbf{i}_s^r, \quad [\boldsymbol{\omega}_{s/r}^r]_{\times} \mathbf{j}_s^r, \quad [\boldsymbol{\omega}_{s/r}^r]_{\times} \mathbf{k}_s^r \right] \\ &= [\boldsymbol{\omega}_{s/r}^r]_{\times} \begin{bmatrix} \mathbf{i}_s^r & \mathbf{j}_s^r & \mathbf{k}_s^r \end{bmatrix} \\ &= [\boldsymbol{\omega}_{s/r}^r]_{\times} R_s^r. \end{aligned}$$

Defining

$$\dot{R}_s^r \triangleq \begin{bmatrix} \frac{d\mathbf{i}_s^r}{dt_r} & \frac{d\mathbf{j}_s^r}{dt_r} & \frac{d\mathbf{k}_s^r}{dt_r} \end{bmatrix}$$

gives the rotational kinematics as

$$\dot{R}_s^r = [\boldsymbol{\omega}_{s/r}^r]_{\times} R_s^r. \quad (2.48)$$

Suppose that we equate the s -frame with the body frame \mathcal{F}_b and the r -frame with the inertial frame \mathcal{F}_i , then Equation (2.48) becomes

$$\dot{R}_b^i = [\boldsymbol{\omega}_{b/i}^i]_{\times} R_b^i,$$

which is given in terms of the angular velocity $\boldsymbol{\omega}_{b/i}$ expressed in the inertial frame. However, the angular velocity can be measured with respect to the body frame using rate gyros. Therefore, it is preferable to write the rotational kinematics in terms of $\boldsymbol{\omega}_{b/i}^b$. Noting that $\boldsymbol{\omega}_{s/r}^r = R_s^r \boldsymbol{\omega}_{s/r}^s$, we get

$$\begin{aligned} \dot{R}_s^r &= [\boldsymbol{\omega}_{s/r}^r]_{\times} R_s^r \\ &= [R_s^r \boldsymbol{\omega}_{s/r}^s]_{\times} R_s^r \\ &= R_s^r [\boldsymbol{\omega}_{s/r}^s]_{\times} (R_s^r)^T R_s^r \\ &= R_s^r [\boldsymbol{\omega}_{s/r}^s]_{\times}. \end{aligned}$$

Note that we have used Equation (2.15) $[(R\mathbf{a})]_{\times} = R [\mathbf{a}]_{\times} R^T$.

We have therefore established the relationship for rotational kinematics as

$$\dot{R}_s^r = R_s^r [\boldsymbol{\omega}_{s/r}^s]_{\times} \quad (2.49)$$

$$= [\boldsymbol{\omega}_{s/r}^r]_{\times} R_s^r \quad (2.50)$$

Applying these formulas to the body and inertial frames we have

$$\dot{R}_b^i = R_b^i [\boldsymbol{\omega}_{b/i}^b]_{\times} = [\boldsymbol{\omega}_{b/i}^i]_{\times} R_b^i \quad (2.51)$$

$$\dot{R}_i^b = -[\boldsymbol{\omega}_{b/i}^b]_{\times} R_i^b = -R_i^b [\boldsymbol{\omega}_{b/i}^i]_{\times}. \quad (2.52)$$

Applying these formulas to the gimbal and body frames we have

$$\begin{aligned}\dot{R}_g^b &= R_g^b \left[\boldsymbol{\omega}_{g/b}^g \right]_\times = \left[\boldsymbol{\omega}_{g/b}^b \right]_\times R_g^b \\ \dot{R}_b^g &= - \left[\boldsymbol{\omega}_{g/b}^g \right]_\times R_b^g = - R_b^g \left[\boldsymbol{\omega}_{g/b}^b \right]_\times.\end{aligned}$$

Using the conclusions of the previous section, we can get the following result.

Lemma 2.5.1 *If \mathbf{p} is a vector and \mathcal{F}^s and \mathcal{F}^r are coordinate frames, with $R_r^s \in SO(3)$ being the rotation between frames, then*

$$\left(\frac{d\mathbf{p}}{dt_r} \right)^s = R_r^s \frac{d(\mathbf{p}^r)}{dt_s}.$$

Proof:

$$\begin{aligned}\left(\frac{d\mathbf{p}}{dt_r} \right)^s &= \left(\frac{d\mathbf{p}}{dt_s} \right)^s + \boldsymbol{\omega}_{s/r}^s \times \mathbf{p}^s \\ &= \frac{d(\mathbf{p}^s)}{dt_s} + \left[\boldsymbol{\omega}_{s/r}^s \right]_\times R_r^s \mathbf{p}^r \\ &= \frac{d(R_r^s \mathbf{p}^r)}{dt_s} + \left[\boldsymbol{\omega}_{s/r}^s \right]_\times R_r^s \mathbf{p}^r \\ &= \frac{dR_r^s}{dt_s} \mathbf{p}^r + R_r^s \frac{d(\mathbf{p}^r)}{dt_s} + \left[\boldsymbol{\omega}_{s/r}^s \right]_\times R_r^s \mathbf{p}^r \\ &= - \left[\boldsymbol{\omega}_{s/r}^s \right]_\times R_r^s \mathbf{p}^r + R_r^s \frac{d(\mathbf{p}^r)}{dt_s} + \left[\boldsymbol{\omega}_{s/r}^s \right]_\times R_r^s \mathbf{p}^r \\ &= R_r^s \frac{d(\mathbf{p}^r)}{dt_s}.\end{aligned}$$

■

2.5.3 Integrating the Rotational Kinematics

In robotics and flying vehicle applications, we often need to integrate the rotational kinematics given samples of the rate gyros. In this section we describe numerical approximations to the equation

$$\dot{R} = R \left[\boldsymbol{\omega} \right]_\times, \quad (2.53)$$

assuming that $\boldsymbol{\omega}$ is either piecewise constant, or piecewise linear between samples. Since R evolves on the set $SO(3)$, the simple Euler integration scheme $R[k] = R[k-1] + T R[k-1] \left[\boldsymbol{\omega}[k] \right]_\times$ is particularly bad since R immediately leaves $SO(3)$. A common strategy is to re-orthonormalize R after every integration step, but this is an ad hoc scheme that may introduce additional drift in the integration process. If $\boldsymbol{\omega}$ is assumed to be piecewise constant or piecewise linear between samples, then the kinematics can be integrated exactly.

We begin by solving the differential equation (2.53).

Lemma 2.5.2 *The solution to Equation (2.53) with initial condition $R(t_0) = R_0$ is given by*

$$R(t) = R(t_0) \exp\left(\left[\int_{t_0}^t \omega(\tau) d\tau\right]_\times\right). \quad (2.54)$$

Proof: Rewrite Equation (2.53) as

$$\dot{R} - R [\omega]_\times = 0,$$

and multiply on the left by the integrating factor $\exp\left(-\left[\int_{t_0}^t \omega(\tau) d\tau\right]_\times\right)$ to get

$$\begin{aligned} (\dot{R} - R [\omega]_\times) \exp\left(-\left[\int_{t_0}^t \omega(\tau) d\tau\right]_\times\right) &= \\ \frac{d}{dt} \left[R \exp\left(-\left[\int_{t_0}^t \omega(\tau) d\tau\right]_\times\right) \right] &= 0. \end{aligned}$$

Integrating both sides and applying the fundamental theorem of Calculus gives

$$R(t) \exp\left(-\left[\int_{t_0}^t \omega(\tau) d\tau\right]_\times\right) - R(t_0) \exp\left(-\left[\int_{t_0}^{t_0} \omega(\tau) d\tau\right]_\times\right) = 0. \quad (2.55)$$

Multiplying on the left by $\exp\left(-\left[\int_{t_0}^t \omega(\tau) d\tau\right]_\times\right)$ gives the result. ■

In the following, we let T_s represent the sample rate, and we will use the notation $R[k] = R(kT_s)$ and $\omega[k] \triangleq \omega(kT_s)$. We say that ω is piecewise constant between samples if

$$\omega(t) = \omega[k]$$

for $(k-1)T_s < t \leq kT_s$, and we say that ω is piecewise linear between samples if When ω is piecewise linear between samples, i.e., when

$$\omega(t) = \left(\frac{kT_s - t}{T_s}\right) \omega[k-1] + \left(\frac{(t - (k-1)T_s)}{T_s}\right) \omega[k]$$

for $(k-1)T_s \leq t < kT_s$.

Lemma 2.5.3 *If the angular velocity is piecewise constant between samples, then from Equation (2.55) we get*

$$R[k] = R[k-1] \exp(T_s [\omega[k]]_\times). \quad (2.56)$$

Similarly, if the angular velocity is piecewise linear between samples, then

$$R[k] = R[k-1] \exp\left(\frac{T_s}{2} [\omega[k] + \omega[k-1]]_\times\right). \quad (2.57)$$

The fundamental theorem of Calculus states that $\int_a^b dF = F(b) - F(a)$.

In deriving this expression, we have used the fact that $\exp(-Mt) = (\exp(Mt))^{-1}$ and $\exp(Mt) \exp(-Mt_0) = \exp(M(t - t_0))$.

Proof: If the sample rate is T_s , then letting $t = kT_s$ and $t_0 = (k-1)T_s$, and using the notation $R[k] = R(kT)$, we get

$$R[k] = R[k-1] \exp \left(\left[\int_{(k-1)T_s}^{kT_s} \boldsymbol{\omega}(\tau) d\tau \right]_\times \right).$$

When $\boldsymbol{\omega}$ is piecewise constant between samples we have

$$\int_{(k-1)T_s}^{kT_s} \boldsymbol{\omega}(\tau) d\tau = \boldsymbol{\omega}[k] T_s,$$

resulting in Equation (2.56). When $\boldsymbol{\omega}$ is piecewise linear between samples we get

$$\begin{aligned} \int_{(k-1)T_s}^{kT_s} \boldsymbol{\omega}(\tau) d\tau &= \int_{(k-1)T_s}^{kT_s} \left[\left(\frac{kT_s - \tau}{T_s} \right) \boldsymbol{\omega}[k-1] + \left(\frac{\tau - (k-1)T_s}{T_s} \right) \boldsymbol{\omega}[k] \right] d\tau \\ &= \frac{T_s}{2} (\boldsymbol{\omega}[k] + \boldsymbol{\omega}[k-1]), \end{aligned}$$

resulting in Equation (2.57). \blacksquare

2.5.4 Numerical Differentiation of the Rotation Matrix

Similarly, if we are given samples of $R(t)$ at sample rate T_s , we would like to approximate the angular velocity $\boldsymbol{\omega}(t)$.

Lemma 2.5.4 *Given the rotational kinematics in Equation (2.53), and measured rotation matrices $R[k]$ and $R[k-1]$, then*

$$\boldsymbol{\omega}[k] \approx \left[\frac{1}{T_s} \log (R^\top[k-1] R[k]) \right]^\vee. \quad (2.58)$$

Proof: From Equation (2.56) we have

$$R[k] = R[k-1] \exp (T_s [\boldsymbol{\omega}[k]]_\times).$$

Solving for $[\boldsymbol{\omega}[k]]_\times$ gives

$$[\boldsymbol{\omega}[k]]_\times = \frac{1}{T_s} \log (R^\top[k-1] R[k])$$

from which we obtain Equation (2.58). \blacksquare

2.5.5 Quaternion Kinematics

RWB: Add this later.

2.5.6 Euler Angle Kinematics

RWB: Add this later.

2.5.7 Rotation Vector Kinematics

RWB: Add this later.

2.6 Homogeneous Transformations

In this section we discuss the use of homogeneous transformations to represent rigid body motion that includes both translation and rotation simultaneously. We will show that 4×4 matrices can be used to represent rigid body motion.

Considering Figure 2.12, suppose that a robot moves from a pose represented by frame \mathcal{F}_a to a pose represented by frame \mathcal{F}_b . If the robot measures the position of point p in frame a and from odometry knows its relative motion represented by the rotation matrix R_a^b and the translation vector $\mathbf{t}_{a/b}^b$, then the position of p relative to frame \mathcal{F}_b is given by

$$\mathbf{r}_{p/b}^b = R_a^b \mathbf{r}_{p/a}^a + \mathbf{t}_{a/b}^b. \quad (2.59)$$

We would like to represent this relative motion using a matrix. To do so, we introduce the use of *homogeneous coordinates* for position. The homogeneous coordinates of the vector \mathbf{p}^c expressed in a general frame \mathcal{F}_c are given by

$$\bar{\mathbf{p}}^c \triangleq \begin{pmatrix} \mathbf{p}^c \\ 1 \end{pmatrix},$$

where 1 has been appended as the last coordinate. Using this notation, Equation (2.59) can be written as

$$\bar{\mathbf{r}}_{p/b}^b = \begin{pmatrix} \mathbf{r}_{p/b}^b \\ 1 \end{pmatrix} = \begin{pmatrix} R_a^b & \mathbf{t}_{a/b}^b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{r}_{p/a}^a \\ 1 \end{pmatrix} = T_a^b \bar{\mathbf{r}}_{p/a}^a,$$

where

$$T_a^b \triangleq \begin{pmatrix} R_a^b & \mathbf{t}_{a/b}^b \\ 0 & 1 \end{pmatrix}$$

is called a homogeneous transformation matrix. It is important to note is that the translation vector is defined in the *destination* frame, rather than the origin frame of the rotation matrix.

2.6.1 Homogeneous Transformation Matrices

Define the set

$$SE(3) = \left\{ T \in \mathbb{R}^{4 \times 4} \mid T = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix}, R \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\},$$

which is called the *Special Euclidean group of dimension 3*. We can derive the following facts about elements of $SE(3)$.

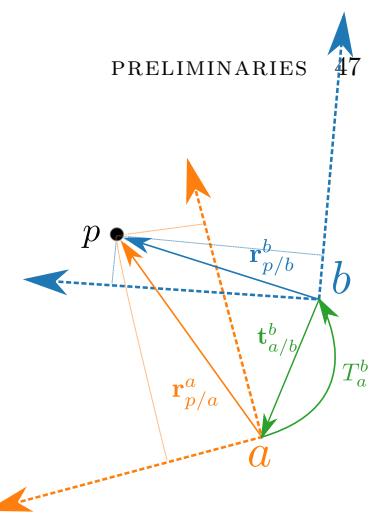


Figure 2.12: Illustration of a passive transformation

Lemma 2.6.1 (Closed under Multiplication) *If $T_1, T_2 \in SE(3)$, then $T_3 = T_2 T_1 \in SE(3)$.*

Proof:

$$\begin{aligned} T_3 &= \begin{pmatrix} R_2 & \mathbf{t}_2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 & \mathbf{t}_1 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_2 R_1 & R_2 \mathbf{t}_1 + \mathbf{t}_2 \\ 0 & 1 \end{pmatrix}, \end{aligned}$$

which is an element of $SE(3)$ since $SO(3)$ is closed under multiplication and $R_2 \mathbf{t}_1 + \mathbf{t}_2 \in \mathbb{R}^3$. \blacksquare

Lemma 2.6.2 (Inverse) *If*

$$T = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in SE(3),$$

then

$$T^{-1} = \begin{pmatrix} R^\top & -R^\top \mathbf{t} \\ 0 & 1 \end{pmatrix} \in SE(3).$$

Proof: The fact that $TT^{-1} = T^{-1}T = I$ can be shown by direct calculation. Since $R^\top \in S(3)$ when $R \in SO(3)$ and $-R^\top \mathbf{t} \in \mathbb{R}^3$ it follows that $T^{-1} \in SE(3)$. \blacksquare

Lemma 2.6.3 (Determinant) *If $T \in SE(3)$, then $\det(T) = 1$.*

Proof: Follows from $\det T = \det(R) \det(1) = 1$. \blacksquare

Lemma 2.6.4 (Concatenation) *If $T_a^b \in SE(3)$ and $T_b^c \in SE(3)$ are homogeneous transformations from frames a to b and b to c , respectively, then the homogeneous transformation from frame a to frame c is given by*

$$\begin{aligned} T_a^c &= T_b^c \cdot T_a^b \\ &= \begin{pmatrix} R_b^c & \mathbf{t}_{c/b}^c \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} R_a^b & \mathbf{t}_{b/a}^b \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} R_b^c R_a^b & R_b^c \mathbf{t}_{b/a}^b + \mathbf{t}_{c/b}^c \\ 0 & 1 \end{pmatrix}. \end{aligned}$$

2.6.2 Passive and Active Transformations

The same notion of passive and active transformations discussed with rotations applies here. To perform a passive transformation with $SE(3)$, we just pad the vector to be rotated with an extra 1 at the bottom, and multiply by the homogeneous transformation matrix.

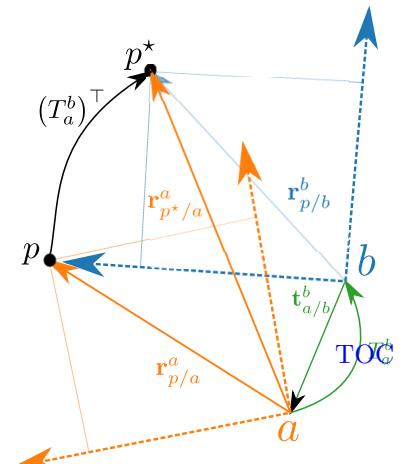


Figure 2.13: Illustration of an active transformation.

An active transformation occurs by multiplying by the inverse of the passive transform as shown in Figure 2.12. Just as with rotations, If you walk through the coordinate frames of an active transformation, you'll see that the coordinate frames don't line up right in our notation. This is because the active transformation does not preserve the location. Instead we have defined a new location, based on the coordinates of p in frame a projected into frame b . This is represented by the p^* notation in Figure 2.13.

2.6.3 Rigid body kinematics

As we saw in Section 2.5, the rotational kinematics are given by

$$\dot{R}_s^r = R_s^r [\omega_{s/r}^s]_\times. \quad (2.60)$$

In this section we will derive a similar expression for homogeneous transformation matrices. Let

$$T_s^r = \begin{pmatrix} R_s^r & \mathbf{t}_{s/r}^r \\ 0 & 1 \end{pmatrix},$$

and differentiate to get

$$\begin{aligned} \dot{T}_s^r &= \begin{pmatrix} \dot{R}_s^r & \dot{\mathbf{t}}_{s/r}^r \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} R_s^r [\omega_{s/r}^s]_\times & R_s^r \dot{\mathbf{t}}_{s/r}^s \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} R_s^r & \mathbf{t}_{s/r}^r \\ 0 & 1 \end{pmatrix} \begin{pmatrix} [\omega_{s/r}^s]_\times & \dot{\mathbf{t}}_{s/r}^s \\ 0 & 0 \end{pmatrix} \end{aligned}$$

Define the relative velocity as

$$\mathbf{v}_{s/r} \triangleq \dot{\mathbf{t}}_{s/r},$$

and the relative twist as

$$\boldsymbol{\nu}_{s/r} = \begin{pmatrix} \mathbf{v}_{s/r} \\ \omega_{s/r} \end{pmatrix} \in \mathbb{R}^6.$$

We also define the “wedge” operator for twist as

$$\begin{pmatrix} \mathbf{v}_{s/r} \\ \omega_{s/r} \end{pmatrix}^\wedge \triangleq \begin{pmatrix} [\omega_{s/r}]_\times & \mathbf{v}_{s/r} \\ 0 & 0 \end{pmatrix},$$

and we define $\mathfrak{se}(3)$ to be the set of all “wedged” twist vectors:

$$\mathfrak{se}(3) = \left\{ \Omega = \begin{pmatrix} [\omega]_\times & \boldsymbol{\nu} \\ 0 & 0 \end{pmatrix} \mid \boldsymbol{\nu}, \omega \in \mathbb{R}^3 \right\}.$$

We have that the homogeneous transformation matrix evolves as

$$\dot{T}_s^r = T_s^r (\boldsymbol{\nu}_{s/r}^s)^\wedge, \quad (2.61)$$

where $T_s^r \in SO(3)$ and $(\boldsymbol{\nu}_{s/r}^s)^\wedge \in \mathfrak{se}(3)$, which has the same form as Equation (2.60). Analogous to Equations (2.51) and (2.52), the kinematics of the multirotor body frame relative to the inertial frame are given by

$$\dot{T}_b^i = T_b^i (\boldsymbol{\nu}_{b/i}^b)^\wedge = (\boldsymbol{\nu}_{b/i}^i)^\wedge T_b^i \quad (2.62)$$

$$\dot{T}_i^b = -(\boldsymbol{\nu}_{b/i}^b)^\wedge T_i^b = -T_i^b (\boldsymbol{\nu}_{b/i}^i)^\wedge. \quad (2.63)$$

The following lemma provide a solution to the differential equation (2.61).

Lemma 2.6.5 *The solution to differential matrix equation*

$$\dot{T} = T \boldsymbol{\nu}^\wedge$$

with initial condition $T(t_0)$ is given by

$$T(t) = T(t_0) \exp \left(\int_{t_0}^t \boldsymbol{\nu}(\tau)^\wedge d\tau \right). \quad (2.64)$$

Proof: Similar to Lemma 2.55. \blacksquare

We now derive an expression for the exponential of a matrix $\Omega \in \mathfrak{se}(3)$.

Lemma 2.6.6 *If $\boldsymbol{\nu}^\wedge = \begin{pmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix} \in \mathfrak{se}(3)$, then*

$$\exp(\boldsymbol{\nu}^\wedge) = \begin{pmatrix} \exp([\boldsymbol{\omega}]_\times) & W([\boldsymbol{\omega}]_\times)\mathbf{v} \\ 0 & 1 \end{pmatrix},$$

where

$$W([\boldsymbol{\omega}]_\times) \triangleq I + \left(\frac{1 - \cos(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^2} \right) [\boldsymbol{\omega}]_\times + \left(\frac{\|\boldsymbol{\omega}\| - \sin(\|\boldsymbol{\omega}\|)}{\|\boldsymbol{\omega}\|^3} \right) ([\boldsymbol{\omega}]_\times)^2.$$

Furthermore $\exp(\boldsymbol{\nu}^\wedge) \in SE(3)$.

Proof:

$$\begin{aligned} \exp \begin{pmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix} &= I + \begin{pmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix} + \frac{1}{2!} \begin{pmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix}^2 + \frac{1}{3!} \begin{pmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix}^3 + \dots \\ &= I + \begin{pmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix} + \frac{1}{2!} \begin{pmatrix} ([\boldsymbol{\omega}]_\times)^2 & [\boldsymbol{\omega}]_\times \mathbf{v} \\ 0 & 0 \end{pmatrix} + \frac{1}{3!} \begin{pmatrix} ([\boldsymbol{\omega}]_\times)^3 & ([\boldsymbol{\omega}]_\times)^2 \mathbf{v} \\ 0 & 0 \end{pmatrix} + \dots \\ &= \begin{pmatrix} \exp([\boldsymbol{\omega}]_\times) & W([\boldsymbol{\omega}]_\times)\mathbf{v} \\ 0 & 0 \end{pmatrix}, \end{aligned}$$

where

$$\begin{aligned} W([\omega]_{\times}) &= I + \frac{1}{2} [\omega]_{\times} + \frac{1}{3!} ([\omega]_{\times})^2 + \dots \\ &= I + \sum_{k=0}^{\infty} \left(\frac{([\omega]_{\times})^{(2k+1)}}{(2k+2)!} + \frac{([\omega]_{\times})^{(2k+2)}}{(2k+3)!} \right). \end{aligned}$$

Recall from the proof of Lemma 2.3.4 that

$$\begin{aligned} [\mathbf{n}]_{\times}^{(2m+1)} &= (-1)^m [\mathbf{n}]_{\times} \\ [\mathbf{n}]_{\times}^{(2m+2)} &= (-1)^m [\mathbf{n}]_{\times}^2, \end{aligned}$$

from which we get

$$\begin{aligned} W([\omega]_{\times}) &= I + \sum_{k=0}^{\infty} \left(\frac{(-1)^k \|\omega\|^{2k}}{(2k+2)!} \right) [\omega]_{\times} + \sum_{k=0}^{\infty} \left(\frac{(-1)^k \|\omega\|^{2k}}{(2k+3)!} \right) ([\omega]_{\times})^2 \\ &= I + \left(\frac{1}{2} - \frac{\|\omega\|^2}{4!} + \frac{\|\omega\|^4}{6!} + \dots \right) [\omega]_{\times} + \left(\frac{1}{3!} - \frac{\|\omega\|^2}{5!} + \frac{\|\omega\|^4}{7!} + \dots \right) ([\omega]_{\times})^2 \\ &= I + \left(\frac{1 - \cos(\|\omega\|)}{\|\omega\|^2} \right) [\omega]_{\times} + \left(\frac{\|\omega\| - \sin(\|\omega\|)}{\|\omega\|^3} \right) ([\omega]_{\times})^2. \end{aligned}$$

The fact that $\exp(\Omega) \in SE(3)$ follows from the definition of $SE(3)$

since $\exp([\omega]_{\times}) \in SO(3)$ and $W([\omega]_{\times})\mathbf{v} \in \mathbb{R}^3$. \blacksquare

Note that

$$\begin{aligned} \lim_{x \rightarrow 0} \frac{1 - \cos(x)}{x^2} &= \lim_{x \rightarrow 0} \frac{1 - (1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots)}{x^2} = \lim_{x \rightarrow 0} \frac{1}{2!} - \frac{x^2}{4!} + \dots = \frac{1}{2} \\ \lim_{x \rightarrow 0} \frac{x - \sin(x)}{x^3} &= \lim_{x \rightarrow 0} \frac{x - (x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots)}{x^3} = \lim_{x \rightarrow 0} \frac{1}{3!} - \frac{x^2}{5!} + \dots = \frac{1}{6}, \end{aligned}$$

and therefore $W([\omega]_{\times})$ is well defined as $\omega \rightarrow 0$.

The logarithm also has a closed form.

Lemma 2.6.7 *If $R \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$, then*

$$\log \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \log(R) & W^{-1}(R)\mathbf{t} \\ 0 & 0 \end{pmatrix},$$

where

$$W^{-1}(R) = I - \frac{1}{2} [\omega]_{\times} + \frac{1}{\|\omega\|^2} \left(1 - \frac{\|\omega\| \sin(\|\omega\|)}{2(1 - \cos(\|\omega\|))} \right) [\omega]_{\times}^2,$$

where $\omega = \log(R)^\vee$.

Proof: From the previous lemma we know that

$$\log \begin{pmatrix} \exp([\omega]_{\times}) & W\mathbf{v} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} [\omega]_{\times} & \mathbf{v} \\ 0 & 0 \end{pmatrix}.$$

Equation $R = \exp([\nu^\wedge]_{\times})$ and $\mathbf{t} = W\mathbf{v}$ shows that $\omega = \log(R)^\vee$ and $\mathbf{v} = W^{-1}\mathbf{t}$. \blacksquare

After multiple uses of L'Hopital's rule, it can be shown that

$$\lim_{\theta \rightarrow 0} \frac{1}{\theta^2} \left(1 - \frac{\theta \sin \theta}{2(1 - \cos \theta)} \right) = \frac{1}{12},$$

which implies that $W^{-1}(R)$ is well defined as $R \rightarrow I$ (i.e., $\omega \rightarrow 0$).

2.7 Lie Groups

Lie Group theory is a very old discipline deeply rooted in mathematics and theoretical physics. It was developed to better understand and model nature at the most fundamental levels. In fact, much of particle physics can be modeled using Lie group theory, including special relativity and quantum dynamics.

Robotics requires us to reason about different coordinate frames. We often need to reason about how they move, and how they relate to one another. We also find ourselves trying to infer things about these coordinate frames given noisy sensor measurements. There are a variety of ways that we could do this, but the real problems show up when we need to do this in real time under realistic computational limitations.

To meet actual computational requirements, we generally want to be able to leverage linear algebra and all of its tools. Unfortunately, however, coordinate frame transformations are not vectors, so it's not completely obvious how we can use these tools in a principled manner. Luckily, we have Lie group theory that can bridge this gap. We will see in this section how we can use Lie Groups to take non-vector group objects, and map them into a vector space where we can perform linear algebra and reason about things in an efficient manner, then map the results back into the group so we can do useful things.

As a further motivating example, consider the case where we might want to represent the uncertainty of the estimate of a rotation matrix. In general, assuming that the uncertainty of some vector quantity \mathbf{x} can be represented with a multivariate Gaussian distribution where the covariance of the distribution is computed as

$$\Sigma_{\mathbf{x}} = E \left[(\mathbf{x} - \hat{\mathbf{x}}) (\mathbf{x} - \hat{\mathbf{x}})^{\top} \right],$$

where $\hat{\mathbf{x}}$ is the mean of the distribution and \mathbf{x} is the true value. However, this equation does not work for rotation matrices in part because $R - \hat{R}$ is not meaningful because subtraction is not a sensible operation for rotation matrices. In addition, R has nine parameters but only three degrees of freedom, and so we expect the uncertainty covariance to be 3×3 . Lie group theory addresses this problem in a very satisfactory way.

Euler angles do not solve the problem, because subtracting two "vectors" of Euler angles is problematic if the angles are larger than 180 degrees.

[TOC](#)

2.7.1 Group Theory

Before we launch into Lie groups, let's consider groups generically. A group is a set of objects \mathcal{G} together with an operation

$$\circ : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{G},$$

with the following properties:

G.1 The group is closed under \circ , meaning that if $G_1, G_2 \in \mathcal{G}$, then $G_1 \circ G_2 \in \mathcal{G}$.

G.2 There exists an identity element $E \in \mathcal{G}$ so that for any other $G \in \mathcal{G}$, $E \circ G = G \circ E = G$.

G.3 For every $G \in \mathcal{G}$, there exists a $G^{-1} \in \mathcal{G}$ such that $G \circ G^{-1} = G^{-1} \circ G = E$.

We use groups all the time, maybe without knowing it. Some commonly used groups in robotics are vectors (closed under vector addition), rotation matrices (closed under matrix multiplication), and quaternions (closed under quaternion multiplication). Some more exotic groups could include binary vectors closed under bit-wise-XOR, positive definite matrices closed under matrix addition and integers closed under addition.

A *Lie group* is a group that is also a differentiable manifold, meaning that every group element G induces a map from one group element B to another $C = G \circ B$ and that this map is differentiable. For example, vectors, rotation matrices, quaternions, and positive definite matrices are Lie Groups, while binary vectors closed under XOR and integers are not, because the mappings are not differentiable.

Each Lie group has an associated Lie algebra, typically indicated by using the fraktur font, (i.e. $\mathfrak{so}(3)$). A Lie algebra is a vector space \mathfrak{g} equipped with a binary operation called the *Lie bracket*, $[,]$, that satisfies the following properties:

- Bilinearity: $[aX + bY, Z] = a[X, Z] + b[Y, Z]$,
- Anticommutativity: $[X, Y] = -[Y, X] \quad \forall X, Y \in \mathfrak{g}$,
- The Jacobi Identity: $[X, [Y, Z]] + [Z, [X, Y]] + [Y, [Z, X]] = 0 \quad \forall X, Y, Z \in \mathfrak{g}$.

For matrix Lie algebras, the Lie bracket is given by

$$[A, B] = AB - BA.$$

Intuitively, the Lie Algebra defines something equivalent to the basis of the Lie Group, except instead of basis vectors, we have *generators*. The generators are the building blocks of the group, and

In case you need another reason to stop using Euler angles, Euler angles aren't even a group, unless, of course, the group operator is defined as "convert to rotation matrix, multiply, and then convert back to Euler angles"

typically encode the degrees of freedom in a orthonormal way. Each member of any Lie Algebra can be expressed as the exponential of a linear combination of the generators.

There is an important theorem that we will use a little later called the Baker-Campbell-Hausdorff Theorem (BCH). This theorem states that the solution Z to

$$e^X e^Y = e^Z,$$

can be expressed as a power series involving commutators X and Y in the Lie bracket. The first couple terms of this series is given as

$$Z = X + Y + \frac{1}{2} [X, Y] + \frac{1}{12} [X, [X, Y]] - \frac{1}{12} [Y, [X, Y]] + \dots$$

This formula is central to many proofs in the Lie group-Lie algebra correspondence.

2.7.2 $SO(2)$: A gentle introduction

To get our feet wet, lets first consider the set of 2×2 rotation matrices. Define

$$SO(2) = \left\{ M \in \mathbb{R}^{2 \times 2} \mid M^\top M = MM^\top = I \text{ and } \det(M) = 1 \right\},$$

where the group operation \circ is defined as matrix multiplication. Note that $M \in SO(2)$ implies that

$$M = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

for some $\theta \in \mathbb{R}$. The set $SO(2)$ is closed under multiplication since

$$\begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{pmatrix} = \begin{pmatrix} \cos(\phi + \psi) & -\sin(\phi + \psi) \\ \sin(\phi + \psi) & \cos(\phi + \psi) \end{pmatrix}.$$

In addition, the identity matrix I forms the identity element of $SO(2)$, and for any $M \in SO(2)$, M^\top is the inverse of M since $MM^\top = M^\top M = I$. Therefore $SO(2)$ is a group under matrix multiplication.

We can visualize $R \in SO(2)$ as an element of the unit circle in \mathbb{R}^2 as shown in Figure 2.14.

The Lie algebra associated with $SO(2)$ is the set

$$\mathfrak{so}(2) = \left\{ M = \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix} \mid \theta \in \mathbb{R} \right\},$$

i.e., the set of skew-symmetric matrices in $\mathbb{R}^{2 \times 2}$. Define the “wedge” and “vee” operators as

$$\theta^\wedge = \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix}^\vee = \theta.$$

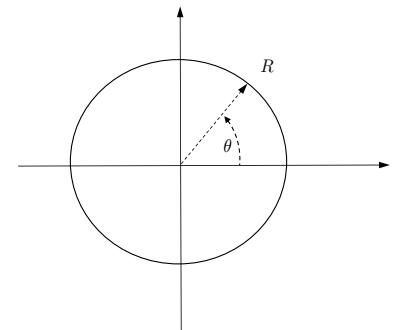


Figure 2.14: Visualization of $SO(2)$ as a point on the unit circle in \mathbb{R}^2 .

By straightforward manipulation, we see that

$$\exp(\theta^\wedge) = I + \theta^\wedge + \frac{1}{2!}(\theta^\wedge)^2 + \frac{1}{3!}(\theta^\wedge)^3 + \dots = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}.$$

Similarly, the log of an element of $SO(2)$ is given by

$$\log(R) = \cos^{-1} \left(\frac{\text{tr}(R)}{2} \right)^\wedge.$$

It is important to note that

$$\begin{aligned} \exp : \mathfrak{so}(2) &\rightarrow SO(2) \\ \log : SO(2) &\rightarrow \mathfrak{so}(2), \end{aligned}$$

therefore it is straightforward to move back and forth between the Lie group $SO(2)$ and its Lie algebra $\mathfrak{so}(2)$. Note that the exponential is a many-to-one map, but that the logarithm maps elements of $SO(2)$ onto only a subset of $\mathfrak{so}(2)$ represented by $[0, \pi]$.

It is interesting to note that $\mathfrak{so}(2)$ is a linear vector space since linear combinations of elements in $\mathfrak{so}(2)$ are also elements of $\mathfrak{so}(2)$, i.e.,

$$a \begin{pmatrix} 0 & -\theta \\ \theta & 0 \end{pmatrix} + b \begin{pmatrix} 0 & -\psi \\ \psi & 0 \end{pmatrix} = \begin{pmatrix} 0 & -(a\theta + b\psi) \\ (a\theta + b\psi) & 0 \end{pmatrix}.$$

Note also that $\mathfrak{so}(2)$ only has dimension of one, and that its elementary basis matrix is

$$J_1 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

In Lie group lingo, the basis vector J_1 is called the generator of $SO(2)$, because any element of $SO(2)$ can be generated by the exponential of a linear combination of the generators. In our case, for any M , there exists a θ such that

$$M = \exp(\theta J_1).$$

Taking the derivative of $M \in SO(2)$ gives

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} &= \begin{pmatrix} -\dot{\theta} \sin \theta & -\dot{\theta} \cos \theta \\ \dot{\theta} \cos \theta & -\dot{\theta} \sin \theta \end{pmatrix} \\ &= \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} 0 & -\dot{\theta} \\ \dot{\theta} & 0 \end{pmatrix}, \end{aligned}$$

or in other words

$$\dot{R} = R \dot{\theta}^\wedge.$$

Defining $\omega = \dot{\theta}$, gives

$$\dot{R} = R \omega^\wedge, \quad (2.65)$$

and the solution (as seen in previous sections) is

$$R(t) = R(t_0) \exp \left(\int_{t_0}^t \omega^\wedge(\tau) d\tau \right).$$

Considering the differential equation $\dot{R} = R\omega^\wedge$, note that the right hand side of the equation represents a vector in the vector space that is tangent to $SO(2)$ at element R . This can be visualized in Figure 2.15. Note that the tangent space is just a rotation by R of the tangent space at the identity element I . The tangent space at I is simply $\mathfrak{so}(2)$, which is the Lie algebra. This is a general property of the Lie algebra of a Lie group, it is the tangent space at the identity element.

Note also that if Equation (2.65) is multiplied on the left by a constant matrix $Q \in SO(2)$, and we let $S = QR$, then

$$\dot{S} = \frac{d}{dt}(QR) = Q\dot{R} = QR\omega^\wedge = S\omega^\wedge.$$

Therefore $S = QR$ satisfies exactly the same differential equation as R . Dynamics that are not affected by a left multiplication of an element of the group are said to be *left-invariant* dynamics. Similarly, if R satisfies $\dot{R} = -\omega^\wedge R$, then multiplying on the right by a constant matrix S does not change the dynamics, which are therefore called *right-invariant* dynamics.

2.7.3 The Lie group $SO(3)$

From the previous section, it should be clear that $SO(3)$ is a Lie group with matrix multiplication as the group operator. The associated Lie algebra is $\mathfrak{so}(3)$, the set of 3×3 skew-symmetric matrices.

The exponential and logarithm functions are defined in Lemmas 2.3.4 and 2.3.5. The “wedge” and “vee” operations are defined as $\omega^\wedge = [\omega]_\times$ and Equation (2.2). Since $\Omega \in \mathfrak{so}(3)$ implies that

$$\Omega = \begin{pmatrix} 0 & -c & b \\ c & 0 & -b \\ -b & a & 0 \end{pmatrix},$$

the basis vectors for $\mathfrak{so}(3)$, or in other words, the generators for $SO(3)$ are

$$J_1 = \mathbf{e}_1^\wedge = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad J_2 = \mathbf{e}_2^\wedge = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad J_3 = \mathbf{e}_3^\wedge = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Therefore $\Omega \in \mathfrak{so}(3)$ implies that

$$\Omega = aJ_1 + bJ_2 + cJ_3,$$

and any element $R \in SO(3)$ can be expressed as

$$R = \exp(aJ_1 + bJ_2 + cJ_3)$$

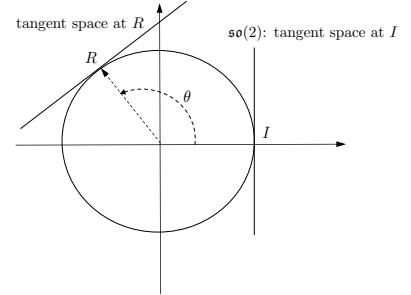


Figure 2.15: Visualization of $\mathfrak{so}(2)$ and the tangent space to $SO(2)$.

Definition of left and right invariant dynamics.

2.7.4 The Lie group $SE(3)$

RWB: rewrite this section

As mentioned before, $SE(3)$ is the set of rigid body transforms. Lie Theory as developed by physicists has very little to say about $SE(3)$, as it is actually a subset of the set of 4×4 matrices, known classically as the Lorentz group. The Lorentz group (or the complexified version, octonians) are used to encode not only rigid body transforms, but also the dilation of space and time due to relativistic effects. In robotics, we can usually restrict ourselves to time and space-preserving transformations, so we can use a simplified version of the Lorentz space generators.¹⁷

The generators of $SE(3)$ are just the basis of infinitesimal transformations along each degree of freedom.

$$J_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad J_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad J_3 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$J_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad J_5 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad J_6 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The use of these generators defines the $(\cdot)^\wedge$ operator for $SE(3)$ as

$$\begin{pmatrix} \omega \\ \mathbf{v} \end{pmatrix}^\wedge = \begin{pmatrix} [\omega]_\times & \mathbf{v} \\ 0 & 0 \end{pmatrix},$$

and $(\cdot)^\vee$ as the inverse operation.

2.7.5 The Lie group S^3

Let's start with the definition of the quaternion exponential

$$\exp(\omega^\wedge) = \sum_{k=0}^{\infty} \frac{(\omega^\wedge)^k}{k!}, \quad \omega^\wedge = \begin{pmatrix} 0 \\ \omega \end{pmatrix} \quad (2.66)$$

where the term $(\mathbf{q})^k$ refers to multiplying \mathbf{q} with itself $(\mathbf{q} \cdot \mathbf{q} \cdots)$ k times. We also note that

$$\begin{aligned} (\omega^\wedge)^2 &= (\omega_x i + \omega_y j + \omega_z k) \cdot (\omega_x i + \omega_y j + \omega_z k) \\ &= -\omega_x^2 - \omega_y^2 - \omega_z^2 \\ &= -\|\omega\|^2 \end{aligned}$$

¹⁷ The bottom row of the 4×4 matrix is used in the Lorentz group, whereas it is not used in $SE(3)$. The Lorentz group generators continue the skew-symmetric pattern we are used to in this bottom row and right column. For more information about using Lie groups to model spacetime, is a phenomenal introductory text to Lie groups in the context of theoretical physics. It includes a very good explanation of the Lorentz group and its double-cover.

Jakob Schwichtenberg. *Physics from symmetry*. Springer International Publishing, 2015

Therefore, if $\theta = \|\boldsymbol{\omega}\|$, then

$$(\boldsymbol{\omega}^\wedge)^2 = -\theta^2, \quad (\boldsymbol{\omega}^\wedge)^3 = -\theta^2 \boldsymbol{\omega}^\wedge \quad (\boldsymbol{\omega}^\wedge)^4 = \theta^4 \dots .$$

Now, we can rewrite the series in Eq. 2.66 as

$$\begin{aligned} \exp(\boldsymbol{\omega}^\wedge) &= \sum_{k=0}^{\infty} \frac{(\boldsymbol{\omega}^\wedge)^k}{k!} \\ &= 1 + \boldsymbol{\omega}^\wedge - \frac{\theta^2}{2!} - \frac{\theta^2}{3!} \boldsymbol{\omega}^\wedge + \frac{\theta^4}{4!} + \frac{\theta^4}{5!} \boldsymbol{\omega}^\wedge - \dots \\ &= 1 + \frac{\theta}{\theta} \boldsymbol{\omega}^\wedge - \frac{\theta^2}{2!} - \frac{\theta^3}{3! \theta} \boldsymbol{\omega}^\wedge + \frac{\theta^4}{4!} + \frac{\theta^5}{5! \theta} \boldsymbol{\omega}^\wedge - \dots \\ &= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} \dots\right) + \frac{1}{\theta} \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} \dots\right) \boldsymbol{\omega}^\wedge \\ &= \cos(\theta) + \frac{\sin(\theta)}{\theta} \boldsymbol{\omega}^\wedge. \end{aligned} \quad (2.67)$$

As with the rotation matrix exponential, we will want to use the Taylor series approximation if $\theta \approx 0$ to avoid numerical errors. (See Table ??).

Computing the closed-form logarithm is done by inverting Eq. 2.67 and is given by

$$\log(\mathbf{q}) = \text{atan2}(\|\vec{\mathbf{q}}\|, q_0) \frac{\mathbf{q}}{\|\vec{\mathbf{q}}\|}.$$

Because of the fact that quaternion dynamics have a $\frac{1}{2}$ in front of them (See Eq. ??), it is common practice in both physics and robotics applications to embed the $\frac{1}{2}$ into the exp function itself such that¹⁸

$$\boxed{\exp(\boldsymbol{\omega}) = \cos\left(\frac{\|\boldsymbol{\omega}\|}{2}\right) + \sin\left(\frac{\|\boldsymbol{\omega}\|}{2}\right) \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}}$$

$$\boxed{\log(\mathbf{q}) = 2 \tan^{-1}\left(\frac{\|\vec{\mathbf{q}}\|}{q_0}\right) \cdot \frac{\vec{\mathbf{q}}}{\|\vec{\mathbf{q}}\|}.}$$

¹⁸ You might recognize this as the axis-angle conversion to unit quaternions (Eq. ??)

This is actually the same as if we redefined the generators of the unit quaternion to all be $\frac{1}{2}J_i$. This is fine, and totally proper, however we just need to be explicit about this choice.

2.8 Adjoint and Jacobians

In this section we will discuss differentiating functions with respect to elements of a Lie group. As a first example, we note that $SO(2)$ is a one-dimensional group because it can be described by a single parameter θ . We typically think of elements of $SO(2)$ as rotation

matrices that transform coordinates or rotate vectors. Therefore, associated with the vector $\mathbf{w} \in \mathbb{R}^2$, we can define the *group action*

$$f_{\mathbf{w}}(R) = R\mathbf{w},$$

where the output of $f_{\mathbf{w}}(R)$ is a vector that represents a transformation of \mathbf{w} . In this section, we think of f as a function of the group element R , with a fixed \mathbf{w} , instead of as a function of \mathbf{w} with a fixed R . It is natural to ask whether we can take the Jacobian of the group action $f_{\mathbf{w}}$ with respect to the group element R .

Toward that end, recall that the Jacobian of a function $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is given by

$$\frac{\partial g}{\partial \mathbf{x}} = \lim_{\varepsilon \rightarrow 0} \begin{pmatrix} \frac{g(\mathbf{x} + \varepsilon \mathbf{e}_1) - g(\mathbf{x})}{\varepsilon} & \dots & \frac{g(\mathbf{x} + \varepsilon \mathbf{e}_n) - g(\mathbf{x})}{\varepsilon} \end{pmatrix},$$

where the i^{th} row describes how g changes with incremental changes in \mathbf{x} along the basis vector \mathbf{e}_i . We can define the $\frac{\partial f}{\partial R}$ similarly, where the i^{th} row of the Jacobian would describe how the group action $f_{\mathbf{w}}(R)$ changes incrementally as R changes incrementally along the basis vector, or generator J_i . Since we cannot add elements in $SO(2)$ we need to perturb R using Lie Group operations. Accordingly, we can perturb R on the right using $R \exp(\varepsilon J_1)$ or we could perturb R on the left using $\exp(\varepsilon J_1)R$. The Jacobian that we obtain will be different in both cases. If we perturb on the right, we will obtain the *right Jacobian*. Alternatively, if we perturb on the left we get the *left Jacobian*.

In our case, the right Jacobian is computed as

$$\begin{aligned} J_r^{f_{\mathbf{w}}}(R) &= \frac{\partial f}{\partial R} = \frac{\partial(R\mathbf{w})}{\partial R} = \lim_{\varepsilon \rightarrow 0} \frac{R \exp(\varepsilon J_1)\mathbf{w} - R\mathbf{w}}{\varepsilon} \\ &= \lim_{\varepsilon \rightarrow 0} \frac{R \left(I + \varepsilon J_1 + \frac{\varepsilon^2}{2!} J_1^2 + \dots \right) \mathbf{w} - R\mathbf{w}}{\varepsilon} \\ &= \lim_{\varepsilon \rightarrow 0} \frac{\varepsilon R J_1 \mathbf{w}}{\varepsilon} = R J_1 \mathbf{w}. \end{aligned}$$

Alternatively, the left Jacobian is computed as

$$\begin{aligned} J_\ell^{f_{\mathbf{w}}}(R) &= \frac{\partial f}{\partial R} = \frac{\partial(R\mathbf{w})}{\partial R} = \lim_{\varepsilon \rightarrow 0} \frac{\exp(\varepsilon J_1)R\mathbf{w} - R\mathbf{w}}{\varepsilon} \\ &= \lim_{\varepsilon \rightarrow 0} \frac{\left(I + \varepsilon J_1 + \frac{\varepsilon^2}{2!} J_1^2 + \dots \right) R\mathbf{w} - R\mathbf{w}}{\varepsilon} \\ &= J_1 R\mathbf{w}. \end{aligned}$$

Note that $\frac{\partial f}{\partial R}$ is a 2×1 vector that because $f_{\mathbf{w}}(R)$ maps to \mathbb{R}^2 (number of rows), and there is only one generator function (number of columns). Therefore, knowing the structure (dimension and generators) of the Lie algebra is critical to the process of taking Jacobians of group actions.

Recall that the derivative of a function is defined as

$$\frac{\partial f}{\partial z} = \lim_{\varepsilon \rightarrow 0} \frac{f(z + \varepsilon) - f(z)}{\varepsilon}.$$

In conversational English, this means, “How does the output of the function f change if the input x is changed by a small amount?”

For vector functions $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, the i^{th} column of the Jacobian matrix indicates how g changes with an incremental change in the i^{th} element of \mathbf{x} .

Similarly, for $SO(3)$, if the group action is defined as $f_{\mathbf{w}}(R) = R\mathbf{w}$, then the right Jacobian of f with respect to R can be computed as

$$\begin{aligned} J_r^{f_{\mathbf{w}}}(R) &= \frac{\partial f_{\mathbf{w}}}{\partial R} = \lim_{\varepsilon \rightarrow 0} \left(\frac{R \exp(\varepsilon J_1)\mathbf{w} - R\mathbf{w}}{\varepsilon} \quad \frac{R \exp(\varepsilon J_2)\mathbf{w} - R\mathbf{w}}{\varepsilon} \quad \frac{R \exp(\varepsilon J_3)\mathbf{w} - R\mathbf{w}}{\varepsilon} \right) \\ &= \lim_{\varepsilon \rightarrow 0} \left(\frac{\varepsilon R J_1 \mathbf{w}}{\varepsilon} \quad \frac{\varepsilon R J_2 \mathbf{w}}{\varepsilon} \quad \frac{\varepsilon R J_3 \mathbf{w}}{\varepsilon} \right) = R \begin{pmatrix} \mathbf{e}_1^\wedge \mathbf{w} & \mathbf{e}_2^\wedge \mathbf{w} & \mathbf{e}_3^\wedge \mathbf{w} \end{pmatrix} \\ &= -R \begin{pmatrix} \mathbf{w}^\wedge \mathbf{e}_1 & \mathbf{w}^\wedge \mathbf{e}_2 & \mathbf{w}^\wedge \mathbf{e}_3 \end{pmatrix} = -R\mathbf{w}^\wedge \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{pmatrix} \\ &= -R\mathbf{w}^\wedge, \end{aligned}$$

and the left Jacobian is computed as

$$\begin{aligned} J_\ell^{f_{\mathbf{w}}}(R) &= \frac{\partial f_{\mathbf{w}}}{\partial R} = \lim_{\varepsilon \rightarrow 0} \left(\frac{\exp(\varepsilon J_1)R\mathbf{w} - R\mathbf{w}}{\varepsilon} \quad \frac{\exp(\varepsilon J_2)R\mathbf{w} - R\mathbf{w}}{\varepsilon} \quad \frac{\exp(\varepsilon J_3)R\mathbf{w} - R\mathbf{w}}{\varepsilon} \right) \\ &= \lim_{\varepsilon \rightarrow 0} \left(\frac{\varepsilon J_1 R\mathbf{w}}{\varepsilon} \quad \frac{\varepsilon J_2 R\mathbf{w}}{\varepsilon} \quad \frac{\varepsilon J_3 R\mathbf{w}}{\varepsilon} \right) = \begin{pmatrix} \mathbf{e}_1^\wedge R\mathbf{w} & \mathbf{e}_2^\wedge R\mathbf{w} & \mathbf{e}_3^\wedge R\mathbf{w} \end{pmatrix} \\ &= - \begin{pmatrix} (R\mathbf{w})^\wedge \mathbf{e}_1 & (R\mathbf{w})^\wedge \mathbf{e}_2 & (R\mathbf{w})^\wedge \mathbf{e}_3 \end{pmatrix} \\ &= -(R\mathbf{w})^\wedge. \end{aligned}$$

2.8.1 The Adjoint Operator and Adjoint Matrix

For a Lie group G , we can think of $g \in G$ as mapping the identity element e to the element g . It is natural to wonder about how small increments in the Lie algebra, i.e., small increments along the generators of the tangent space at e map to small increments along the generators of the tangent space at g . Let TG_g be the tangent space of manifold G at element g , and let $\tau_e^\wedge \in TG_e$ be a small increment in the lie algebra, and $\tau_g^\wedge \in TG_g$ be a small increment in the tangent space at g , then we have that

$$g \exp(\tau_e^\wedge) = \exp(\tau_g^\wedge)g.$$

RWB: Add picture here Therefore, we have that

$$\exp(\tau_g^\wedge) = g \exp(\tau_e^\wedge)g^{-1}$$

For matrix Lie groups we have that

$$\begin{aligned} \exp(g\tau^\wedge g^{-1}) &= I + g\tau^\wedge g^{-1} + \frac{1}{2!}(g\tau^\wedge g^{-1})^2 + \dots \\ &= g(I + \tau^\wedge + \frac{1}{2!}(\tau^\wedge)^2 + \dots)g^{-1} \\ &= g \exp(\tau^\wedge)g^{-1}, \end{aligned}$$

which implies that

$$\begin{aligned} \exp(\tau_g^\wedge) &= \exp(g\tau_e^\wedge g^{-1}) \\ \implies \tau_g^\wedge &= g\tau_e^\wedge g^{-1}. \end{aligned}$$

We denote the *adjoint operator* $\text{Ad}_g : \mathfrak{g} \rightarrow \mathfrak{g}$ as

$$\text{Ad}_g(\boldsymbol{\tau}^\wedge) \stackrel{\triangle}{=} g\boldsymbol{\tau}^\wedge g^{-1},$$

and note that the physical meaning of $\text{Ad}_g(\cdot)$ is that it transforms elements in the tangent space of the input to g to an element in the tangent space at the output of g . For example, if $T_b^i \in SE(3)$ represents the pose in the inertial frame relative to the body frame, then $\text{Ad}_{T_b^i}(\cdot)$ transforms small deviations of the twist in the body frame to small deviations of the twist in the inertial frame, and therefore represents the Jacobian of T_b^i .

Notice that the adjoint is a linear operator since

$$\begin{aligned} \text{Ad}_g(\alpha\boldsymbol{\tau}_1^\wedge + \beta\boldsymbol{\tau}_2^\wedge) &= g(\alpha\boldsymbol{\tau}_1^\wedge + \beta\boldsymbol{\tau}_2^\wedge)g^{-1} \\ &= \alpha g\boldsymbol{\tau}_1^\wedge g^{-1} + \beta g\boldsymbol{\tau}_2^\wedge g^{-1} \\ &= \alpha \text{Ad}_g(\boldsymbol{\tau}_1^\wedge) + \beta \text{Ad}_g(\boldsymbol{\tau}_2^\wedge) \end{aligned}$$

and therefore, there exists a matrix, which in a severe abuse of notation is also denoted as $\text{Ad}_g : \mathbb{R}^{\dim(\mathfrak{g})} \rightarrow \mathbb{R}^{\dim(\mathfrak{g})}$, where

$$\boldsymbol{\tau}_g = \text{Ad}_g \boldsymbol{\tau}_e.$$

As a first example, consider $R \in SO(2)$. The adjoint operator is defined as

$$\begin{aligned} \text{Ad}_R(\omega^\wedge) &= R\omega^\wedge R^\top \\ &= \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix} \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \\ &= \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix}. \end{aligned}$$

The adjoint matrix is computed by

$$(\text{Ad}_R)\omega = \begin{pmatrix} 0 & -\omega \\ \omega & 0 \end{pmatrix}^\vee = \omega = (1) \cdot \omega,$$

implying that $\text{Ad}_R = 1$ for $R \in SO(2)$.

As another example, consider $R_a^b \in SO(3)$. The adjoint operator is defined as

$$\boldsymbol{\omega}^{b\wedge} = \text{Ad}_{R_a^b}(\boldsymbol{\omega}^{a\wedge}) = R_a^b \boldsymbol{\omega}^{a\wedge} R_a^b{}^\top.$$

However, noting that $R_a^b \boldsymbol{\omega}^{a\wedge} R_a^b{}^\top = (R_a^b \boldsymbol{\omega}^a)^\wedge$ we have that $\boldsymbol{\omega}^b = R_a^b \boldsymbol{\omega}^a$. Therefore, for $SO(3)$, the adjoint matrix is

$$\text{Ad}_{R_a^b} = R_a^b.$$

Since the adjoint of R is simply R , the Lie Group $SO(3)$ is self adjoint. Unfortunately, not all Lie Groups are self-adjoint.

Consider now

$$T_a^b = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in SO(2), \quad \boldsymbol{\nu}^{\wedge} = \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix}^{\wedge} = \begin{pmatrix} \omega^{\wedge} & \mathbf{v} \\ 0 & 1 \end{pmatrix}$$

where $R \in SO(2)$, $\mathbf{t}, \mathbf{v} \in \mathbb{R}^2$, and $\omega \in \mathbb{R}$. The adjoint operator is defined as The adjoint operator is defined as

$$\boldsymbol{\nu}^{b\wedge} = Ad_{T_a^b}(\boldsymbol{\nu}^{a\wedge}) = T_a^b \boldsymbol{\nu}^{a\wedge} T_a^{b\top} = \begin{pmatrix} \omega^{\wedge} & -\omega^{\wedge}\mathbf{t} + R\mathbf{v} \\ 0 & 0 \end{pmatrix}.$$

Noting that

$$\begin{pmatrix} \omega^{\wedge} & -\omega^{\wedge}\mathbf{t} + R\mathbf{v} \\ 0 & 0 \end{pmatrix}^{\vee} = \begin{pmatrix} -\omega^{\wedge}\mathbf{t} + R\mathbf{v} \\ \omega \end{pmatrix} = \begin{pmatrix} R & -1^{\wedge}\mathbf{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix},$$

where $1^{\wedge} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, we conclude that for $SE(2)$

$$Ad_T = \begin{pmatrix} R & -1^{\wedge}\mathbf{t} \\ 0 & 1 \end{pmatrix}.$$

For our final example, let

$$T_a^b = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in SE(3), \quad \boldsymbol{\nu}^{\wedge} = \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix}^{\wedge} = \begin{pmatrix} [\omega]_{\times} & \mathbf{v} \\ 0 & 0 \end{pmatrix}.$$

The adjoint operator is defined as

$$\boldsymbol{\nu}^{b\wedge} = Ad_{T_a^b}(\boldsymbol{\nu}^{a\wedge}) = T_a^b \boldsymbol{\nu}^{a\wedge} T_a^{b\top} = \begin{pmatrix} R [\omega]_{\times} R^{\top} & -R [\omega]_{\times} R^{\top} \mathbf{t} + R\mathbf{v} \\ 0 & 0 \end{pmatrix}.$$

Noting that

$$\begin{pmatrix} R [\omega]_{\times} R^{\top} & -R [\omega]_{\times} R^{\top} \mathbf{t} + R\mathbf{v} \\ 0 & 0 \end{pmatrix}^{\vee} = \begin{pmatrix} [\mathbf{t}]_{\times} R\omega + R\mathbf{v} \\ R\omega \end{pmatrix} = \begin{pmatrix} R & [\mathbf{t}]_{\times} R \\ 0 & R \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix},$$

we have that the associated adjoint matrix for $SE(3)$ is

$$Ad_{T_a^b} \triangleq \begin{pmatrix} R & [\mathbf{t}]_{\times} R \\ 0 & R \end{pmatrix}.$$

Again, the physical meaning of the adjoint matrix Ad_g where $g \in G$, is that it maps small deviations in the Euclidean space isomorphic to the tangent space of the manifold G at the input of g , to small deviations in the Euclidean space isomorphic to the tangent space of the manifold G at the output of g . For example, Consider a rod

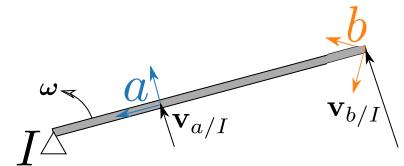


Figure 2.16: Illustration of a rotating rigid body with two co-ordinate frames. The Adjoint of $SE(3)$ will convert ω and \mathbf{v} from frame a to frame b , and account for the change in translation.

rotating about a fixed origin, as shown in Figure 2.16. Suppose we know the linear velocity \mathbf{v}^a and angular velocity $\boldsymbol{\omega}^a$ in coordinate frame a , and want to know these quantities in coordinate frame b . The adjoint matrix in $SE(3)$ provides the transformation. Since $\boldsymbol{\omega}^b$ is only a rotation of $\boldsymbol{\omega}^a$, the bottom row of Ad_T only contains a rotation. However, to compute the linear velocity in frame b , we must account for the “lever arm” and the angular rate, which is what is going on in the two row of Ad_T .

RWB: Add a picture here.

For any Lie group G with associated Lie algebra \mathfrak{g} , the adjoint matrix has the following properties, where $g, h \in G$ and $\tau \in \mathfrak{g}$:

$$\begin{aligned} g \exp(\tau) &= Ad_g(\tau)g, & Ad_g(\cdot) &\text{ as an operator} \\ Ad_{g^{-1}} &= (Ad_g)^{-1}, & Ad_g &\text{ as a matrix} \\ Ad_{gh} &= Ad_g Ad_h, & Ad_* &\text{ as a matrix.} \end{aligned}$$

G	$g \in G$	$\nu^\wedge \in \mathfrak{g}$	$Ad_g : \mathbb{R}^{\dim(\mathfrak{g})} \rightarrow \mathbb{R}^{\dim(\mathfrak{g})}$
$SO(2)$,	$g = R(\theta)$,	$\nu = \omega \in \mathbb{R}$,	$Ad_g = 1$
$SE(2)$,	$g = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix}$,	$\nu = \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} \in \mathbb{R}^3$,	$Ad_g = \begin{pmatrix} R & -\mathbf{1}^\wedge \mathbf{t} \\ 0 & 1 \end{pmatrix}$
$SO(3)$,	$g = R$,	$\nu = \boldsymbol{\omega} \in \mathbb{R}^3$,	$Ad_g = R$
$SE(3)$,	$g = \begin{pmatrix} R & \mathbf{t} \\ 0 & 1 \end{pmatrix}$,	$\nu = \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix} \in \mathbb{R}^6$,	$Ad_g = \begin{pmatrix} R & [\mathbf{t}]_\times R \\ 0 & R \end{pmatrix}$

It turns out that the set of adjoint matrices is itself a matrix Lie group. For example, for $SE(3)$, the set of 6×6 matrices

$$Ad(SE(3)) = \left\{ \begin{pmatrix} R & [\mathbf{t}]_\times R \\ 0 & R \end{pmatrix} \mid R \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\}$$

is a matrix Lie group. Therefore, the matrix Lie group $Ad(SE(3))$ has a Lie algebra, which we will denote as (little ad) $ad(\mathfrak{se}(3))$. The little adjoint for common matrix groups is given in Table ??.

The relationship between the big adjoint and the little adjoint is given by

$$Ad_{\text{Exp}(\xi)}(\exp(\boldsymbol{\nu}^\wedge)) = \exp(ad_\xi(\boldsymbol{\nu}^\wedge)).$$

2.9 Right and Left Jacobian of the Exponential

We have seen that the exponential and logarithmic operator plays especially important roles in Lie group theory. In particular, the exponential operator maps elements of the Lie algebra to elements of the

Table 2.2: Adjoint matrix summary

\mathfrak{g}	$\nu^\wedge \in \mathfrak{g}$	$ad_\nu : \mathbb{R}^{\dim(\mathfrak{g})} \rightarrow \mathbb{R}^{\dim(\mathfrak{g})}$
$\mathfrak{so}(2)$,	$\nu = \omega \in \mathbb{R}$,	$ad_\nu = 0$
$\mathfrak{se}(2)$,	$\nu = \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} \in \mathbb{R}^3$,	$ad_\nu = \begin{pmatrix} \omega^\wedge & -\mathbf{v}^\wedge \\ 0 & 0 \end{pmatrix}$
$\mathfrak{so}(3)$,	$\nu = \boldsymbol{\omega} \in \mathbb{R}^3$,	$ad_\nu = [\boldsymbol{\omega}]_\times$
$\mathfrak{se}(3)$,	$\nu = \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix} \in \mathbb{R}^6$,	$ad_\nu = \begin{pmatrix} [\boldsymbol{\omega}]_\times & [\mathbf{v}]_\times \\ 0 & [\boldsymbol{\omega}]_\times \end{pmatrix}$

Lie group, and the logarithm performs the inverse operation. Therefore, it should come as no surprise that the Jacobian of the exponential and logarithm operators also plays an important role in working with Lie groups.

For a matrix Lie group G with Lie algebra \mathfrak{g} , it is shown in ¹⁹ that the right and left Jacobians of the exponential map $g = \exp(\nu)$, where $\nu \in \mathfrak{g}$, are given by

$$J_r(\nu) = \sum_{n=0}^{\infty} \frac{(-\text{ad}_\nu)^n}{(n+1)!}, \quad (2.68)$$

$$J_\ell(\nu) = \sum_{n=0}^{\infty} \frac{(\text{ad}_\nu)^n}{(n+1)!}. \quad (2.69)$$

The right and left Jacobians are related to each other as

$$\begin{aligned} J_\ell(\nu) &= \text{Exp}(\nu) J_r(\nu) \\ J_\ell(\nu) &= J_r(-\nu). \end{aligned}$$

Right and left Jacobians are extremely useful in robotic applications because of the following property, where $\nu, \delta\nu \in \mathfrak{g}$ and $\|\delta\nu\|$ is small:

$$\text{Exp}(\nu + \delta\nu) \approx \text{Exp}(\nu) \text{Exp}(J_r(\nu)\delta\nu) \quad (2.70a)$$

$$\text{Exp}(\nu + \delta\nu) \approx \text{Exp}(J_\ell(\nu)\delta\nu) \text{Exp}(\nu) \quad (2.70b)$$

$$\text{Exp}(\nu) \text{Exp}(\delta\nu) \approx \text{Exp}(\nu + J_r^{-1}(\nu)\delta\nu) \quad (2.70c)$$

$$\text{Exp}(\delta\nu) \text{Exp}(\nu) \approx \text{Exp}(\nu + J_\ell^{-1}(\nu)\delta\nu). \quad (2.70d)$$

RWB: Add closed form expressions of the exponential, logarithm and Jacobian maps see ²⁰.

2.9.1

he matrix adjoint of $g \in G$ is a representation of g that acts on \mathbf{R}_G and is generically defined as

$$\text{Ad}_g^G : \mathbf{R}_G \rightarrow \mathbf{R}_G; \quad (v) \mapsto \text{Ad}_g^G(v),$$

and represents, for example, a change of coordinates from one location on the manifold to another. A useful property of the adjoint is

$$g \text{Exp}_I^G(v) = \text{Exp}_I^G(\text{Ad}_g^G v) g. \quad (2.71)$$

Table 2.3: Little adjoint matrix summary

¹⁹ Timothy Barfoot. *State Estimation For Robotics*. Cambridge University Press, 2019

²⁰ Joan Solà, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics, 2018. URL <http://arxiv.org/abs/1812.01537>

For $SE(2)$, the matrix adjoint of $g \in SE(2)$ is

$$\text{Ad}_g^{SE(2)} = \begin{bmatrix} R & -[1]_{\times} p \\ 0_{1 \times 2} & 1 \end{bmatrix}. \quad (2.72)$$

The matrix adjoint of $v_1 \in \mathbf{R}_G$ is a representation of \mathbf{R}_G that acts on $v_2 \in \mathbf{R}_G$ generically defined as

$$\text{ad}_{v_1}^G : \mathbf{R}_G \rightarrow \mathbf{R}_G; \quad (v_2) \mapsto \text{ad}_{v_1}^G v_2.$$

For $SE(2)$, the matrix representation of the adjoint is

$$\text{ad}_v^{SE(2)} = \begin{bmatrix} [w]_{\times} & -[1]_{\times} \rho \\ 0_{1 \times 2} & 0 \end{bmatrix}.$$

2.9.2 Jacobian of the Matrix Exponential

When working with Lie groups, we need the differential of the exponential and logarithm functions. These differentials are commonly called the right and left Jacobians. The right and left Jacobians and their inverses are defined to map elements of \mathbf{R}_G to the general linear group (set of invertible matrices) that acts on \mathbf{R}_G . For matrix Lie groups, they are defined as

$$J_r^G(v) = \sum_{n=0}^{\infty} \frac{(-\text{ad}^G(v))^n}{(n+1)!}, \quad J_\ell^G(v) = \sum_{n=0}^{\infty} \frac{(\text{ad}^G(v))^n}{(n+1)!},$$

$$J_r^{G^{-1}}(v) = \sum_{n=0}^{\infty} \frac{B_n (-\text{ad}^G(v))^n}{n!}, \quad J_\ell^{G^{-1}}(v) = \sum_{n=0}^{\infty} \frac{B_n (\text{ad}^G(v))^n}{n!},$$

where B_n are the Bernoulli numbers, the subscripts r/l indicate the right and left Jacobian, and the superscript indicates the corresponding Lie group. The derivation of the left and right Jacobians stems from the Baker-Campbell-Hausdorff formula ²¹. **RWB: Add BCH theorem or more explanation.** The right Jacobian has the properties that for any $v \in \mathbf{R}_G$ and any small $\tilde{v} \in \mathbf{R}_G$,

$$\text{Exp}_I^G(v + \tilde{v}) \approx \text{Exp}_I^G(v) \text{Exp}_I^G(J_r^G(v)\tilde{v}) \quad (2.73a)$$

$$\text{Exp}_I^G(v) \text{Exp}_I^G(\tilde{v}) \approx \text{Exp}_I^G(v + J_r^{G^{-1}}(v)\tilde{v}). \quad (2.73b)$$

Similarly for the left Jacobian,

$$\begin{aligned} \text{Exp}_I^G(v + \tilde{v}) &\approx \text{Exp}_I^G(J_\ell^G(v)\tilde{v}) \text{Exp}_I^G(v) \\ \text{Exp}_I^G(\tilde{v}) \text{Exp}_I^G(v) &\approx \text{Exp}_I^G(v + J_\ell^{G^{-1}}(v)\tilde{v}). \end{aligned}$$

²¹ Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer-Verlag New York, Inc, 2003. ISBN 0-387-40122-9; and Timothy Barfoot. *State Estimation For Robotics*. Cambridge University Press, 2019

For $SE(2)$, the Jacobians have closed form solutions, and the right Jacobian for $SE(2)$ is

$$J_r^{SE(2)}(v) = \begin{bmatrix} W_r(\omega) & D_r(\omega) \rho \\ 0_{1 \times 2} & 1 \end{bmatrix} \quad (2.74)$$

where

$$\begin{aligned} W_r(\omega) &= \frac{\cos \omega - 1}{\omega} [1]_\times + \frac{\sin \omega}{\omega} I \\ D_r(\omega) &= \frac{1 - \cos \omega}{\omega^2} [1]_\times + \frac{\omega - \sin \omega}{\omega^2} I, \end{aligned}$$

and where we again note that $W_r(0)$ and $D_r(0)$ are well defined and that for small ω , $W_r(\omega) \approx I - \frac{1}{2}[\omega]_\times$ and $D_r(\omega) \approx \frac{1}{2}[1]_\times$.

2.10 Useful Tables

Expression	Taylor series
$\sin(\theta)$	$\theta - \frac{1}{3!}\theta^3 + \frac{1}{5!}\theta^5 \dots$
$\cos(\theta)$	$1 - \frac{1}{2}\theta^2 + \frac{1}{4!}\theta^4 \dots$
$\cos\left(\frac{\theta}{2}\right)$	$1 - \frac{1}{8}\theta^2 + \frac{1}{46080}\theta^4 \dots$
$\frac{\sin(\theta)}{\theta}$	$1 - \frac{1}{3!}\theta^2 + \frac{1}{5!}\theta^4 \dots$
$\frac{\sin\left(\frac{\theta}{2}\right)}{\theta}$	$\frac{1}{2} - \frac{1}{48}\theta^2 + \frac{1}{3840}\theta^4 + \dots$
$\frac{\theta}{\sin(\theta)}$	$1 + \frac{1}{3!}\theta^2 + \frac{7}{360}\theta^4 \dots$
$\frac{1-\cos(\theta)}{\theta^2}$	$\frac{1}{2} - \frac{1}{24}\theta^2 + \frac{1}{720}\theta^4 \dots$
$\frac{\cos\theta - \sin\theta}{\theta^2}$	$-\frac{1}{3} + \frac{1}{30}\theta^2 - \frac{1}{840}\theta^4 \dots$
$\frac{\frac{1}{2}\cos\frac{\theta}{2} - \frac{\sin\frac{\theta}{2}}{\theta}}{\theta^2}$	$-\frac{1}{24} + \frac{1}{960}\theta^2 - \frac{1}{107520}\theta^4 \dots$
$\frac{\tan^{-1}\left(\frac{\theta}{y}\right)}{\theta}$	$\frac{1}{y} - \frac{1}{3y^3}\theta^2 + \frac{1}{5y^5}\theta^4 \dots$

Table 2.4: Taylor series expansions

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} \mathbf{y} \cdot \mathbf{x}$	$\text{Ad}(\mathbf{y})^{-1}$	I
$\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^{-1} \cdot \mathbf{y}$	$-\text{Ad}(\mathbf{x})^{-1}$	$-\text{Ad}(\mathbf{x}^{-1} \cdot \mathbf{y})^{-1}$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{y} \cdot \mathbf{x}^{-1}$	$-\text{Ad}(\mathbf{y} \cdot \mathbf{x}^{-1})$	$-\text{Ad}(\mathbf{x})$
$\frac{\partial}{\partial \mathbf{x}} \mathbf{x} \cdot \mathbf{y}$	I	$\text{Ad}(\mathbf{y})^{-1}$

Table 2.5: Group-Group Jacobians.
 $\mathbf{x}, \mathbf{y} \in G$

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} R \cdot \mathbf{v}$	$-[(R \cdot \mathbf{v})]_{\times}$	$-R \cdot [\mathbf{v}]_{\times}$
$\frac{\partial}{\partial \mathbf{x}} R^T \cdot \mathbf{v}$	$R^T [\mathbf{v}]_{\times}$	$[(R^T \cdot \mathbf{v})]_{\times}$
$\frac{\partial}{\partial \boldsymbol{\omega}} \exp([\boldsymbol{\omega}]_{\times})$	$aI + b[\boldsymbol{\omega}]_{\times} + c\boldsymbol{\omega}\boldsymbol{\omega}^T$	$aI - b[\boldsymbol{\omega}]_{\times} + c\boldsymbol{\omega}\boldsymbol{\omega}^T$
$\frac{\partial}{\partial R} \log(R)$	$I - \frac{1}{2}[\boldsymbol{\delta}]_{\times} + e[\boldsymbol{\delta}]_{\times}^2$	$I + \frac{1}{2}[\boldsymbol{\delta}]_{\times} + e[\boldsymbol{\delta}]_{\times}^2$

Table 2.6: Useful Jacobians for $SO(3)$.

$$\begin{aligned} R &\in SO(3), \mathbf{v}, \boldsymbol{\omega} \in \mathbb{R}^3, \\ \boldsymbol{\delta} &= \log(R), \theta = \|\boldsymbol{\delta}\|, \\ a &= \frac{\sin \theta}{\theta}, b = \frac{1-\cos(\theta)}{\theta^2}, c = \frac{1-a}{\theta^2}, \\ e &= \frac{b-2c}{2a} \end{aligned}$$

Expression	Left Jacobian	Right Jacobian
$\frac{\partial}{\partial \mathbf{x}} T \cdot \mathbf{v}$	$\begin{pmatrix} -[(R \cdot \mathbf{v} + \mathbf{t})]_{\times} & I \end{pmatrix}$	$\begin{pmatrix} -R \cdot [\mathbf{v}]_{\times} & R \end{pmatrix}$
$\frac{\partial}{\partial \mathbf{x}} T^{-1} \cdot \mathbf{v}$	$\begin{pmatrix} R^T \cdot [\mathbf{v}]_{\times} & -R^T \end{pmatrix}$	$\begin{pmatrix} [(R^T \cdot (\mathbf{v} - \mathbf{t}))]_{\times} & -I \end{pmatrix}$
$\frac{\partial}{\partial \boldsymbol{\xi}} \exp(\boldsymbol{\xi}^{\wedge})$	$\begin{pmatrix} A & 0 \\ D & A \end{pmatrix}$	$\begin{pmatrix} A^T & 0 \\ D^T & A^T \end{pmatrix}$
$\frac{\partial}{\partial T} \log(T)$	$\begin{pmatrix} E & 0 \\ -E \cdot D \cdot E & E \end{pmatrix}$	$\begin{pmatrix} E^T & 0 \\ -(E \cdot D \cdot E)^T & E^T \end{pmatrix}$

Table 2.7: Useful Jacobians Jacobians for $SE(3)$.

$$\begin{aligned} T &\in SE(3), \mathbf{v} \in \mathbb{R}^3, \\ \boldsymbol{\xi} &= (\boldsymbol{\omega}^T, \mathbf{v}^T)^T \in \mathfrak{se}(3) \\ a &= \frac{\sin \theta}{\theta}, b = \frac{1-\cos \theta}{\theta^2}, \\ c &= \frac{1-a}{\theta^2}, d = \boldsymbol{\omega}^T \mathbf{v} \\ A &= \frac{\partial}{\partial \boldsymbol{\omega}} \exp([\boldsymbol{\omega}]_{\times}) \\ B &= \boldsymbol{\omega} \mathbf{v}^T + \mathbf{v} \boldsymbol{\omega}^T \\ C &= (c-b)I + \left(\frac{a-2b}{\theta^2}\right)[\boldsymbol{\omega}]_{\times} + \left(\frac{b-3c}{\theta^2}\right)\boldsymbol{\omega}\boldsymbol{\omega}^T \\ D &= b[\mathbf{v}]_{\times} + cB + dC \\ E &= \frac{\partial}{\partial R} \log(R) \end{aligned}$$

2.11 Micro Lie Theory

A micro Lie theory for state estimation in robotics

Joan Solà, Jeremie Deray, Dinesh Atchuthan

Abstract—A Lie group is an old mathematical abstract object dating back to the XIX century, when mathematician Sophus Lie laid the foundations of the theory of continuous transformation groups. As it often happens, its usage has spread over diverse areas of science and technology many years later. In robotics, we are recently experiencing an important trend in its usage, at least in the fields of estimation, and particularly in motion estimation for navigation. Yet for a vast majority of roboticians, Lie groups are highly abstract constructions and therefore difficult to understand and to use. This may be due to the fact that most of the literature on Lie theory is written by and for mathematicians and physicists, who might be more used than us to the deep abstractions this theory deals with.

In estimation for robotics it is often not necessary to exploit the full capacity of the theory, and therefore an effort of selection of materials is required. In this paper, we will walk through the most basic principles of the Lie theory, with the aim of conveying clear and useful ideas, and leave a significant corpus of the Lie theory behind. Even with this mutilation, the material included here has proven to be extremely useful in modern estimation algorithms for robotics, especially in the fields of SLAM, visual odometry, and the like.

Alongside this micro Lie theory, we provide a vast reference of formulas for the major Lie groups used in robotics, including most jacobian matrices and the way to easily manipulate them. We also present a new C++ template-only library implementing all the functionality described here.

I. INTRODUCTION

There has been a remarkable effort in the last years in the robotics community to formulate estimation problems properly. This is motivated by an increasing demand for precision, consistency and stability of the solutions. Indeed, a proper modeling of the states and measurements, the functions relating them, and their uncertainties, is crucial to achieve these goals. This has led to designs involving what has been known as ‘manifolds’, which in this context are no less than the smooth topologic surfaces of the Lie groups where the state representations evolve. Relying on the Lie theory (LT) we are able to construct a rigorous calculus corpus to handle uncertainties, derivatives and integrals with precision and ease. Typically, these works have focused on the well known manifolds of rotation SO(3) and rigid motion SE(3).

When being introduced to Lie groups for the first time, it is important to try to regard them from different points of view. The topological viewpoint, see Fig. 1, involves the shape of the manifold and conveys powerful intuitions of its relation to the tangent space and the exponential map. The algebraic viewpoint involves the group operations and their concrete realization, allowing the exploitation of algebraic properties to develop closed-form formulas or to simplify them. The

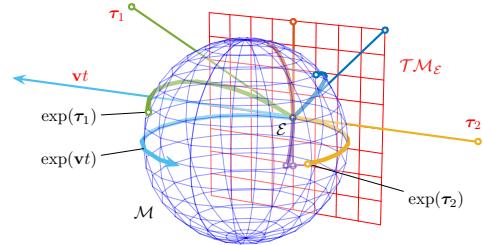


Figure 1. Representation of the relation between the Lie group and the Lie algebra. The Lie algebra (red plane) is the tangent space to the Lie group’s manifold (here represented as a blue sphere) at the identity \mathcal{E} . Through the exponential map, each straight path vt through the origin on the Lie algebra produces a path $\exp(vt)$ over the manifold which runs along the respective geodesic. Conversely, each element of the group has an equivalent in the Lie algebra. This relation is so profound that (nearly) all operations in the group, which is curved and nonlinear, have an exact equivalent in the Lie algebra, which is a linear vector space. Though the sphere in \mathbb{R}^3 is not a Lie group (we just use it as a representation that can be drawn on paper), that in \mathbb{R}^4 is, and describes the group of unit quaternions —see Fig. 4 and Ex. 5.

geometrical viewpoint, particularly useful in robotics, associates group elements to the position, velocity, orientation, and/or other modifications of bodies or reference frames. The origin frame may be identified with the group’s identity, and any other point on the manifold represents a certain ‘local’ frame. By resorting to these analogies, many mathematical abstractions of the LT can be brought closer to intuitive notions in vector spaces, geometry, kinematics and other more classical fields.

Lie theory is by no means simple. To grasp a minimum idea of what LT can be, we may consider the following two references. On one hand, Howe’s “*Very basic Lie theory*” [1] comprises more than 600 pages. On another hand, the more modern Stillwell’s “*Naive Lie theory*” [2] comprises more than 200 pages. With such precedents labeled as ‘very basic’ and ‘naive’, the aim of this paper at merely 16 pages is to simplify Lie theory much more (thus our adjective ‘micro’ in the title). This we do in two forms. First, we select a very restrictive subset of material from the LT. This subset is so small that it merely explores the potential of LT. However, it appears very useful for uncertainty management in the kind of estimation problems we deal with in robotics (*e.g.* inertial pre-integration, odometry and SLAM, visual servoing, and the like), thus enabling elegant and rigorous designs of optimal optimizers. Second, we explain it in a didactical way, with plenty of redundancy so as to reduce the entry gap to LT even more, which we believe is still needed. That is, we insist

on the efforts in this direction of, to name a paradigmatic title, Stillwell's [2], and provide yet a more simplified version. The main text body is generic, though we try to keep the abstraction level to a minimum. Inserted examples serve as grounding base for the general concepts when applied to known groups (rotation and motion matrices, quaternions, etc.). Also, plenty of figures with very verbose captions re-explain the same concepts once again. We put special attention to the computation of Jacobians, which are essential for most optimal optimizers and the source of much trouble when designing new algorithms. And finally, several appendices contain ample reference for the most relevant details of the most commonly used groups in robotics.

Yet the most important simplification is in terms of scope. The following passage from Howe [1] may serve us to illustrate what we leave behind: "*The essential phenomenon of Lie theory is that one may associate in a natural way to a Lie group \mathcal{G} its Lie algebra \mathfrak{g} . The Lie algebra \mathfrak{g} is first of all a vector space and secondly is endowed with a bilinear nonassociative product called the Lie bracket [...]. Amazingly, the group \mathcal{G} is almost completely determined by \mathfrak{g} and its Lie bracket. Thus for many purposes one can replace \mathcal{G} with \mathfrak{g} . [...] This is one source of the power of Lie theory.*" In [2], Stillwell even speaks of "the miracle of Lie theory". In this work we will effectively relegate the Lie algebra to a second plane in favor of its equivalent vector space \mathbb{R}^n , and will not introduce the Lie bracket at all. Therefore, the connection between the Lie group and its Lie algebra will not be made here as profound as it should. Our position is that, given the target application areas that we foresee, this material is often not necessary. Moreover, if included, then we would fail in the objective of being clear and useful, because the reader would have to go into mathematical concepts that, by their abstraction or subtleness, are unnecessarily complicated.

Our effort is in line with other recent works on the subject [3], [4], [5], which have also identified this need of bringing the LT closer to the robotician. Our approach aims at appearing familiar to the target audience of this paper: an audience that is skilled in state estimation (Kalman filtering, graph-based optimization, and the like), but not yet familiar with the theoretical corpus of the Lie theory. We have for this taken some initiatives with regard to notation, especially in the definition of the derivative, bringing it close to the vectorial counterparts, thus making the chain rule clearly visible. As said, we opted to practically avoid the material proper to the Lie algebra, and prefer instead to work on its isomorphic tangent vector space \mathbb{R}^n , which is where we ultimately represent uncertainty or (small) state increments. All these steps are undertaken with absolutely no loss in precision or exactness, and we believe they make the understanding of the LT and the manipulation of its tools easier.

II. A MICRO LIE THEORY

A. The Lie group

The Lie group encompasses the concepts of *group* and *smooth manifold* in a unique body: a Lie group \mathcal{G} is a smooth manifold whose elements satisfy the group axioms. We briefly present these two concepts before joining them together.

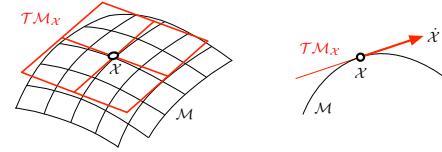


Figure 2. A manifold \mathcal{M} and the vector space $\mathcal{T}\mathcal{M}_x$ (in this case $\simeq \mathbb{R}^2$) tangent at the point x , and a convenient side-cut. The velocity element, $\dot{x} = \partial x / \partial t$, does not belong to the manifold \mathcal{M} but to the tangent space $\mathcal{T}\mathcal{M}_x$.

On one hand, a differentiable or *smooth manifold* is a topological space that locally resembles linear space. The reader should be able to visualize the idea of manifold (Fig. 2): it is like a curved, smooth (hyper)-surface, with no edges or spikes, embedded in a space of higher dimension. In robotics, we say that our state vector evolves on this surface, that is, the manifold describes or is defined by the constraints imposed on the state. For example, vectors with the unit norm constraint define a spherical manifold of radius one. The smoothness of the manifold implies the existence of a unique tangent space at each point. This space is a linear or vector space on which we are allowed to do calculus.

On the other hand, a *group* (\mathcal{G}, \circ) is a set, \mathcal{G} , with a composition operation, \circ , that satisfies the following axioms,

$$\text{Closure under } \circ : \mathcal{X} \circ \mathcal{Y} \in \mathcal{G} \quad (1)$$

$$\text{Identity } \mathcal{E} : \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} \quad (2)$$

$$\text{Inverse } \mathcal{X}^{-1} : \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E} \quad (3)$$

$$\text{Associativity} : (\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z}), \quad (4)$$

for elements $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{G}$.

In a *Lie group*, the manifold looks the same at every point (like *e.g.* in the surface of a sphere, see Exs. 1 and 2), and therefore all tangent spaces at any point are alike. The group structure imposes that the composition of elements of the manifold remains on the manifold, and that each element has an inverse also in the manifold. A special one of these elements is the identity, and thus a special one of the tangent spaces is the tangent at the identity, which we call the Lie algebra of the Lie group. Lie groups join the local properties of smooth manifolds, allowing us to do calculus, with the global properties of groups, enabling nonlinear composition of distant objects.

In this work, for simplicity and as it has been common in robotics works, we will oftentimes refer to Lie groups as just 'manifolds'.

B. The group actions

Importantly, Lie groups come with the power to transform elements of other sets, producing *e.g.* rotations, translations, scalings, and combinations of them. These are extensively used in robotics, both in 2D and 3D.

Given a Lie group \mathcal{M} and a set \mathcal{V} , we note $\mathcal{X} \cdot v$ the *action* of $\mathcal{X} \in \mathcal{M}$ on $v \in \mathcal{V}$,

$$\cdot : \mathcal{M} \times \mathcal{V} \rightarrow \mathcal{V} ; (\mathcal{X}, v) \mapsto \mathcal{X} \cdot v. \quad (5)$$

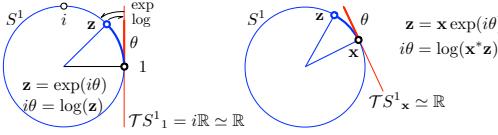


Figure 3. The S^1 manifold is a unit circle (blue) in the plane \mathbb{C} , where the unit complex numbers $z^*z = 1$ live. The Lie algebra $\mathfrak{s}^1 = T\mathcal{S}^1|_{\mathcal{E}}$ is the line of imaginary numbers $i\mathbb{R}$ (red), and any tangent space $T\mathcal{S}^1$ is isomorphic to the line \mathbb{R} (red). Tangent vectors (red segment) wrap the manifold creating the arc of circle (blue arc). Mappings \exp and \log (arrows) map (wrap and unwrap) elements of \mathbb{R} to/from elements of S^1 (blue arc). Increments between unit complex numbers are expressed in the tangent space via composition and the exponential map (and we will define special operators for this). See the text for explanations.

Example 1: The unit complex numbers group S^1

Our first example of Lie group, which is the easiest to visualize, is the group of unit complex numbers under complex multiplication (Fig. 3). Unit complex numbers take the form $z = \cos \theta + i \sin \theta$.

– *Action:* Vectors $x = x + iy$ rotate in the plane by an angle θ , through complex multiplication, $x' = zx$.

– *Group facts:* The product of unit complex numbers is a unit complex number, the identity is 1, and the inverse is the conjugate.

– *Manifold facts:* The unit norm constraint defines the unit circle in the complex plane (which can be viewed as the 1-sphere, and hence the name S^1). This is a 1-DoF curve in 2-dimensional space. Unit complex numbers evolve with time on this circle. The group (the circle) resembles the linear space (the tangent line) locally, but not globally.

For \cdot to be a group action, it must satisfy the axioms,

$$\text{Identity : } \mathcal{E} \cdot v = v \quad (6)$$

$$\text{Compatibility : } (\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v) . \quad (7)$$

Common examples are the groups of rotation matrices $SO(n)$, the group of unit quaternions, and the groups of rigid motion $SE(n)$. Their respective actions on vectors satisfy

$SO(n)$: rotation matrix	$\mathbf{R} \cdot \mathbf{x} \triangleq \mathbf{Rx}$
$SE(n)$: Euclidean matrix	$\mathbf{H} \cdot \mathbf{x} \triangleq \mathbf{Rx} + \mathbf{t}$
S^1 : unit complex	$\mathbf{z} \cdot \mathbf{x} \triangleq \mathbf{zx}$
S^3 : unit quaternion	$\mathbf{q} \cdot \mathbf{x} \triangleq \mathbf{qxq}^*$

See Table I for a more detailed exposition, and the appendices.

The group composition (1) may be viewed as an action of the group on itself, $\circ : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$. Another interesting action is the *adjoint action*, which we will see in Section II-F.

C. The tangent spaces and the Lie algebra

Given $\mathcal{X}(t)$ a point moving on a Lie group's manifold \mathcal{M} , its velocity $\dot{\mathcal{X}} = \partial \mathcal{X}/\partial t$ belongs to the space tangent to \mathcal{M} at \mathcal{X} (Fig. 2), which we note $T\mathcal{M}_{\mathcal{X}}$. The smoothness of the manifold, *i.e.*, the absence of edges or spikes, implies the

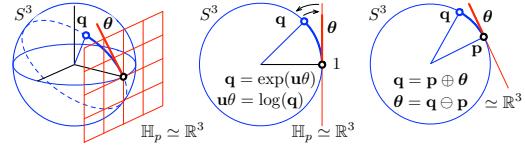


Figure 4. The S^3 manifold is a unit 3-sphere (blue) in the 4-space of quaternions \mathbb{H} , where the unit quaternions $q^*q = 1$ live. The Lie algebra is the space of pure imaginary quaternions $ix + jy + kz \in \mathbb{H}_p$, isomorphic to the hyperplane \mathbb{R}^3 (red grid), and any other tangent space $T\mathcal{S}^3$ is also isomorphic to \mathbb{R}^3 . Tangent vectors (red segment) wrap the manifold over the great arc or *geodesic* (dashed). The centre and right figures show a side-cut through this geodesic (notice how it ressembles S^1 in Fig. 3). Mappings \exp and \log (arrows) map (wrap and unwrap) elements of \mathbb{H}_p to/from elements of S^3 (blue arc). Increments between quaternions are expressed in the tangent space via the operators \oplus, \ominus (see text).

Example 2: The unit quaternions group S^3

A second example of Lie group, which is also relatively easy to visualize, is the group of unit quaternions under quaternion multiplication (Fig. 4). Unit quaternions take the form $\mathbf{q} = \cos(\theta/2) + \mathbf{u} \sin(\theta/2)$, with $\mathbf{u} = iu_x + ju_y + ku_z$ a unitary axis and θ a rotation angle.

– *Action:* Vectors $\mathbf{x} = ix + jy + kz$ rotate in 3D space by an angle θ around the unit axis \mathbf{u} through the double quaternion product $\mathbf{x}' = \mathbf{q} \mathbf{x} \mathbf{q}^*$.

– *Group facts:* The product of unit quaternions is a unit quaternion, the identity is 1, and the inverse is the conjugate.

– *Manifold facts:* The unit norm constraint defines the 3-sphere S^3 , a spherical 3-dimensional surface or *manifold* in 4-dimensional space. Unit quaternions evolve with time on this surface. The group (the sphere) resembles the linear space (the tangent hyperplane $\mathbb{R}^3 \subset \mathbb{R}^4$) locally, but not globally.

existence of a unique tangent space at each point. The structure of such tangent spaces is the same everywhere.

1) *The Lie algebra m:* The tangent space at the identity, $T\mathcal{M}_{\mathcal{E}}$, is called the *Lie algebra* of \mathcal{M} , and noted \mathfrak{m} ,

$$\text{Lie algebra : } \mathfrak{m} \triangleq T\mathcal{M}_{\mathcal{E}} . \quad (8)$$

Every Lie group has an associated Lie algebra. We relate the Lie group with its Lie algebra through the following facts [5] (see Figs. 1 and 6):

- The Lie algebra \mathfrak{m} is a vector space.¹ As such, its elements can be *identified* with vectors in \mathbb{R}^m , whose dimension m is the number of degrees of freedom of \mathcal{M} .
- The *exponential map* $\exp : \mathfrak{m} \rightarrow \mathcal{M}$ exactly converts elements of the Lie algebra into elements of the group. The *log map* is the inverse operation.
- Vectors of the tangent space at \mathcal{X} can be transformed to the tangent space at the identity \mathcal{E} through a linear transform. This transform is called the *adjoint*.

¹In a Lie algebra, the vector space is endowed with a non-associative product called the Lie bracket. In this work, we will not make use of it.

Table I
TYPICAL LIE GROUPS USED IN 2D AND 3D MOTION, INCLUDING THE TRIVIAL \mathbb{R}^n . SEE THE APPENDICES FOR FULL REFERENCE

Lie group \mathcal{M}, \circ	size	dim	$\mathcal{X} \in \mathcal{M}$	Constraint	$\tau^\wedge \in \mathfrak{m}$	$\tau \in \mathbb{R}^m$	Exp(τ)	Comp.	Action
n -D vector	$\mathbb{R}^n, +$	n	n	$\mathbf{v} \in \mathbb{R}^n$	$\mathbf{v} - \mathbf{v} = \mathbf{0}$	$\mathbf{v} \in \mathbb{R}^n$	$\mathbf{v} = \exp(\mathbf{v})$	$\mathbf{v}_1 + \mathbf{v}_2$	$\mathbf{v} + \mathbf{x}$
circle Rotation	S^1, \cdot $SO(2), \cdot$	2 4	1 1	$\mathbf{z} \in \mathbb{C}$ \mathbf{R}	$\mathbf{z}^* \mathbf{z} = 1$ $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$	$i\theta \in i\mathbb{R}$ $[\theta]_\times \in \mathfrak{so}(2)$	$\theta \in \mathbb{R}$ $\theta \in \mathbb{R}$	$\mathbf{z} = \exp(i\theta)$ $\mathbf{R} = \exp([\theta]_\times)$	$\mathbf{z}_1 \mathbf{z}_2$ $\mathbf{R}_1 \mathbf{R}_2$
Rigid motion	$SE(2), \cdot$	9	3	$\mathbf{M} = [\mathbf{R} \ \mathbf{t}]$	$\mathbf{M}^{-1} \mathbf{M} = \mathbf{I}$	$[\theta]_\times \rho \in \mathfrak{se}(2)$	$[\rho] \in \mathbb{R}^3$	$\exp\left(\begin{bmatrix} [\theta]_\times & \rho \\ 0 & 1 \end{bmatrix}\right)$	$M_1 M_2$ $\mathbf{R} \mathbf{x} + \mathbf{t}$
3-sphere Rotation	S^3, \cdot $SO(3), \cdot$	4 9	3 3	$\mathbf{q} \in \mathbb{H}$ \mathbf{R}	$\mathbf{q}^* \mathbf{q} = 1$ $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$	$u\theta/2 \in \mathbb{H}_p$ $[\theta]_\times \in \mathfrak{so}(3)$	$\theta \in \mathbb{R}^3$ $\theta \in \mathbb{R}^3$	$\mathbf{q} = \exp(u\theta/2)$ $\mathbf{R} = \exp([\theta]_\times)$	$\mathbf{q}_1 \mathbf{q}_2$ $\mathbf{R}_1 \mathbf{R}_2$
Rigid motion	$SE(3), \cdot$	16	6	$\mathbf{M} = [\mathbf{R} \ \mathbf{t}]$	$\mathbf{M}^{-1} \mathbf{M} = \mathbf{I}$	$[\theta]_\times \rho \in \mathfrak{se}(3)$	$[\rho] \in \mathbb{R}^6$	$\exp\left(\begin{bmatrix} [\theta]_\times & \rho \\ 0 & 0 \end{bmatrix}\right)$	$M_1 M_2$ $\mathbf{R} \mathbf{x} + \mathbf{t}$

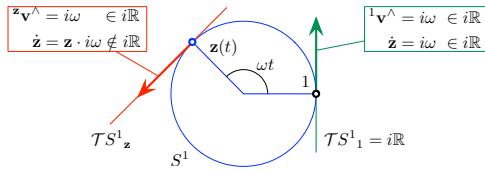


Figure 5. Let a point $\mathbf{z} \in S^1$ move at constant rotation rate ω , $\mathbf{z}(t) = \cos \omega t + i \sin \omega t$. Its velocities when passing through 1 and \mathbf{z} are in the respective tangent spaces, $\mathcal{T}S^1_1$ and $\mathcal{T}S^1_{\mathbf{z}}$. In the case of $\mathcal{T}S^1_{\mathbf{z}}$, the velocity is $\dot{\mathbf{z}} = \mathbf{z} \cdot i\omega = -\omega \sin \omega t + i\omega \cos \omega t$ when expressed in the global coordinates, and ${}^z\mathbf{v}^\wedge = i\omega$ when expressed locally. Their relation is given by ${}^z\mathbf{v}^\wedge = \mathbf{z}^{-1} \dot{\mathbf{z}} = \mathbf{z}^* \dot{\mathbf{z}}$. In the case of $\mathcal{T}S^1_1$, this relation is the identity ${}^1\mathbf{v}^\wedge = \dot{\mathbf{z}} = i\omega$. Clearly, the structure of all tangent spaces is $i\mathbb{R}$, which is the Lie algebra. This is also the structure of $\dot{\mathbf{z}}$ at the identity, and this is why the Lie algebra is defined as the tangent space at the identity.

Lie algebras can be defined locally to a tangent point \mathcal{X} , establishing local coordinates for $\mathcal{T}\mathcal{M}_{\mathcal{X}}$ (Fig. 5). We shall denote elements of the Lie algebras with a ‘hat’ decorator, such as \mathbf{v}^\wedge for velocities or $\tau^\wedge = (\mathbf{v}\mathbf{t})^\wedge = \mathbf{v}^\wedge \mathbf{t}$ for general elements. A left superscript may also be added to specify the precise tangent space, e.g., ${}^{\mathcal{X}}\mathbf{v}^\wedge \in \mathcal{T}\mathcal{M}_{\mathcal{X}}$ and ${}^{\mathcal{X}}\mathbf{v}^\wedge \in \mathcal{T}\mathcal{M}_{\mathcal{E}}$.

The structure of the Lie algebra can be found (see Examples 3 and 5) by time-differentiating the group constraint (3). For multiplicative groups this yields the new constraint $\mathcal{X}^{-1} \dot{\mathcal{X}} + \dot{\mathcal{X}}^{-1} \mathcal{X} = 0$, which applies to the elements tangent at \mathcal{X} (the term \mathcal{X}^{-1} is the derivative of the inverse). The elements of the Lie algebra are therefore of the form,

$$\mathbf{v}^\wedge = \mathcal{X}^{-1} \dot{\mathcal{X}} = -\dot{\mathcal{X}}^{-1} \mathcal{X}. \quad (9)$$

2) The Cartesian vector space \mathbb{R}^m : The elements τ^\wedge of the Lie algebra have non-trivial structures (skew-symmetric matrices, imaginary numbers, pure quaternions, see Table I) but the key aspect for us is that they can be expressed as linear combinations of some base elements E_i , where E_i are called the *generators* of \mathfrak{m} (they are the derivatives of \mathcal{X} around the origin in the i -th direction). It is then handy to manipulate just the coordinates as vectors in \mathbb{R}^m , which we shall note simply τ . We may pass from \mathfrak{m} to \mathbb{R}^m and vice versa through two mutually inverse linear maps or isomorphisms, commonly

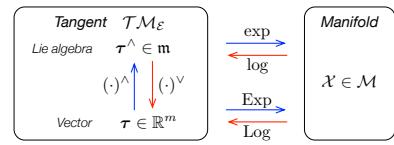


Figure 6. Mappings between the manifold \mathcal{M} and the representations of its tangent space at the origin $\mathcal{T}\mathcal{M}_{\mathcal{E}}$ (Lie algebra \mathfrak{m} and Cartesian \mathbb{R}^m). Maps $\hat{(\cdot)}$ and $\vee(\cdot)$ are the linear invertible maps or *isomorphisms* (10–11), $\exp(\cdot)$ and $\log(\cdot)$ map the Lie algebra to/from the manifold, and $\text{Exp}(\cdot)$ and $\text{Log}(\cdot)$ are shortcuts to map directly the vector space \mathbb{R}^m to/from \mathcal{M} .

called *hat* and *vee* (see Fig. 6),

$$\text{Hat}: \mathbb{R}^m \rightarrow \mathfrak{m}; \quad \tau \mapsto \tau^\wedge = \sum_{i=1}^m \tau_i E_i \quad (10)$$

$$\text{Vee}: \mathfrak{m} \rightarrow \mathbb{R}^m; \quad \tau^\wedge \mapsto (\tau^\wedge)^\vee = \tau = \sum_{i=1}^m \tau_i \mathbf{e}_i, \quad (11)$$

with \mathbf{e}_i the vectors of the base of \mathbb{R}^m (we have $\mathbf{e}_i^\wedge = E_i$). This means that \mathfrak{m} is isomorphic to the vector space \mathbb{R}^m — one writes $\mathfrak{m} \simeq \mathbb{R}^m$, or $\tau^\wedge \simeq \tau$. Vectors $\tau \in \mathbb{R}^m$ are handier for our purposes than their isomorphs $\tau^\wedge \in \mathfrak{m}$, since they can be stacked in larger state vectors, and more importantly, manipulated with linear algebra using matrix operators. In this work, we enforce this preference of \mathbb{R}^m over \mathfrak{m} , to the point that most of the operators and objects that we define (specifically: the adjoint, the Jacobians, the perturbations and their covariances matrices, as we will see soon) are on \mathbb{R}^m .

D. The exponential map

The exponential map $\exp()$ allows us to exactly transfer elements of the Lie algebra to the group (Fig. 1). Intuitively, $\exp()$ wraps the tangent element over the manifold following the great arc or *geodesic* (as when wrapping a string around a ball, Figs. 1, 3 and 4). The inverse map is the $\log()$, i.e., the unwrapping operation. The $\exp()$ map arises naturally by considering the time-derivatives of $\mathcal{X} \in \mathcal{M}$ over the manifold, as follows. From (9) we have,

$$\dot{\mathcal{X}} = \mathcal{X} \mathbf{v}^\wedge. \quad (12)$$

Example 3: The rotation group $SO(3)$, its Lie algebra $\mathfrak{so}(3)$, and the vector space \mathbb{R}^3

In the rotation group $SO(3)$, of 3×3 rotation matrices \mathbf{R} , we have the orthogonality condition $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$. The tangent space may be found by taking the time derivative of this constraint, that is $\mathbf{R}^\top \dot{\mathbf{R}} + \dot{\mathbf{R}}^\top \mathbf{R} = 0$, which we rearrange as

$$\mathbf{R}^\top \dot{\mathbf{R}} = -(\mathbf{R}^\top \dot{\mathbf{R}})^\top.$$

This expression reveals that $\mathbf{R}^\top \dot{\mathbf{R}}$ is a skew-symmetric matrix (the negative of its transpose). Skew-symmetric matrices are often noted $[\omega]_\times$ and have the form

$$[\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$

This gives $\mathbf{R}^\top \dot{\mathbf{R}} = [\omega]_\times$. When $\mathbf{R} = \mathbf{I}$ we have

$$\dot{\mathbf{R}} = [\omega]_\times,$$

that is, $[\omega]_\times$ is in the Lie algebra of $SO(3)$, which we name $\mathfrak{so}(3)$. Since $[\omega]_\times \in \mathfrak{so}(3)$ has 3 DoF, the dimension of $SO(3)$ is $m = 3$. The Lie algebra is a vector space whose elements can be decomposed into

$$[\omega]_\times = \omega_x \mathbf{E}_x + \omega_y \mathbf{E}_y + \omega_z \mathbf{E}_z$$

with $\mathbf{E}_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$, $\mathbf{E}_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$, $\mathbf{E}_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ the generators of $\mathfrak{so}(3)$, and where $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ is the vector of angular velocities. The linear relation above allows us to identify $\mathfrak{so}(3)$ with \mathbb{R}^3 — we write $\mathfrak{so}(3) \simeq \mathbb{R}^3$. We pass from $\mathfrak{so}(3)$ to \mathbb{R}^3 and viceversa using the linear operators *hat* and *vee*,

$$\begin{aligned} \text{Hat : } \quad \mathbb{R}^3 &\rightarrow \mathfrak{so}(3); & \omega &\mapsto \omega^\wedge = [\omega]_\times \\ \text{Vee : } \quad \mathfrak{so}(3) &\rightarrow \mathbb{R}^3; & [\omega]_\times &\mapsto [\omega]_\times^\vee = \omega. \end{aligned}$$

For v constant, this is an ordinary differential equation (ODE) whose solution is

$$\mathcal{X}(t) = \mathcal{X}(0) \exp(v^\wedge t). \quad (13)$$

Since $\mathcal{X}(t)$ and $\mathcal{X}(0)$ are elements of the group, then $\exp(v^\wedge t) = \mathcal{X}(0)^{-1} \mathcal{X}(t)$ must be in the group too, and so $\exp(v^\wedge t)$ maps elements $v^\wedge t$ of the Lie algebra to the group. This is known as the *exponential map*.

In order to provide a more generic definition of the exponential map, let us define the tangent increment $\tau \triangleq vt \in \mathbb{R}^m$ as velocity per time, so that we have $\tau^\wedge = v^\wedge t \in \mathfrak{m}$ a point in the Lie algebra. The exponential map, and its inverse the logarithmic map, can be now written as,

$$\exp : \quad \mathfrak{m} \rightarrow \mathcal{M} \quad ; \quad \tau^\wedge \mapsto \mathcal{X} = \exp(\tau^\wedge) \quad (14)$$

$$\log : \quad \mathcal{M} \rightarrow \mathfrak{m} \quad ; \quad \mathcal{X} \mapsto \tau^\wedge = \log(\mathcal{X}) . \quad (15)$$

Closed forms of the exponential in multiplicative groups are obtained by writing the absolutely convergent Taylor series,

$$\exp(\tau^\wedge) = \mathcal{E} + \tau^\wedge + \frac{1}{2}\tau^{\wedge 2} + \frac{1}{3!}\tau^{\wedge 3} + \dots , \quad (16)$$

Example 4: The exponential map of $SO(3)$

We have seen in Ex. 3 that $\dot{\mathbf{R}} = \mathbf{R}[\omega]_\times \in \mathfrak{so}(3)$. For ω constant, this is an ordinary differential equation (ODE), whose solution is $\mathbf{R}(t) = \mathbf{R}_0 \exp([\omega]_\times t)$. At the origin $\mathbf{R}_0 = \mathbf{I}$ we have the exponential map,

$$\mathbf{R}(t) = \exp([\omega]_\times t) \quad \in SO(3) .$$

We now define the vector $\theta \triangleq \mathbf{u}\theta \triangleq \omega t \in \mathbb{R}^3$ as the integrated rotation in angle-axis form, with angle θ and unit axis \mathbf{u} . Thus $[\theta]_\times \in \mathfrak{so}(3)$ is the total rotation expressed in the Lie algebra. We substitute it above and write the exponential as a power series,

$$\mathbf{R} = \exp([\theta]_\times) = \sum_k \frac{\theta^k}{k!} ([\mathbf{u}]_\times)^k .$$

In order to find a closed-form expression, we write down a few powers of $[\mathbf{u}]_\times$,

$$\begin{aligned} [\mathbf{u}]_\times^0 &= \mathbf{I}, & [\mathbf{u}]_\times^1 &= [\mathbf{u}]_\times, \\ [\mathbf{u}]_\times^2 &= \mathbf{u}\mathbf{u}^\top - \mathbf{I}, & [\mathbf{u}]_\times^3 &= -[\mathbf{u}]_\times, \\ [\mathbf{u}]_\times^4 &= -[\mathbf{u}]_\times^2, & \dots \end{aligned}$$

and realize that all can be expressed as multiples of \mathbf{I} , $[\mathbf{u}]_\times$ or $[\mathbf{u}]_\times^2$. We thus rewrite the series as,

$$\begin{aligned} \mathbf{R} &= \mathbf{I} + [\mathbf{u}]_\times \left(\theta - \frac{1}{3!}\theta^3 + \frac{1}{5!}\theta^5 - \dots \right) \\ &\quad + [\mathbf{u}]_\times^2 \left(\frac{1}{2}\theta^2 - \frac{1}{4!}\theta^4 + \frac{1}{6!}\theta^6 - \dots \right), \end{aligned}$$

where we identify the series of $\sin \theta$ and $\cos \theta$, yielding the closed form,

$$\mathbf{R} = \exp([\mathbf{u}\theta]_\times) = \mathbf{I} + [\mathbf{u}]_\times \sin \theta + [\mathbf{u}]_\times^2 (1 - \cos \theta) .$$

This expression is the well known Rodrigues rotation formula. It can be used as the capitalized exponential just by doing $\mathbf{R} = \text{Exp}(\mathbf{u}\theta) = \exp([\mathbf{u}\theta]_\times)$.

and taking advantage of the algebraic properties of the powers of τ^\wedge (see Ex. 4 and 5 for developments of the exponential map in $SO(3)$ and S^3). These are then inverted to find the logarithmic map. Key properties of the exponential map are

$$\exp((t+s)\tau^\wedge) = \exp(t\tau^\wedge) \exp(s\tau^\wedge) \quad (17)$$

$$\exp(t\tau^\wedge) = \exp(\tau^\wedge)^t \quad (18)$$

$$\exp(-\tau^\wedge) = \exp(\tau^\wedge)^{-1} \quad (19)$$

$$\exp(\mathcal{X}\tau^\wedge \mathcal{X}^{-1}) = \mathcal{X} \exp(\tau^\wedge) \mathcal{X}^{-1} , \quad (20)$$

where (20) can be checked by expanding the Taylor series.

1) *The capitalized exponential map:* The capitalized Exp and Log maps are convenient shortcuts to map vector elements $\tau \in \mathbb{R}^m$ ($\simeq \mathcal{T}\mathcal{M}_{\mathcal{E}}$) directly with elements $\mathcal{X} \in \mathcal{M}$. We have,

$$\text{Exp} : \quad \mathbb{R}^m \rightarrow \mathcal{M} \quad ; \quad \tau \mapsto \mathcal{X} = \text{Exp}(\tau) \quad (21)$$

$$\text{Log} : \quad \mathcal{M} \rightarrow \mathbb{R}^m \quad ; \quad \mathcal{X} \mapsto \tau = \text{Log}(\mathcal{X}) . \quad (22)$$

Example 5: The unit quaternions group S^3 (cont.)

In the group S^3 (recall Ex. 2 and see e.g. [6]), the time derivative of the unit norm condition $\mathbf{q}^*\mathbf{q} = 1$ yields

$$\dot{\mathbf{q}}^*\dot{\mathbf{q}} = -(\mathbf{q}^*\dot{\mathbf{q}})^*.$$

This reveals that $\mathbf{q}^*\dot{\mathbf{q}}$ is a pure quaternion (its real part is zero). Pure quaternions $\mathbf{u}\mathbf{v} \in \mathbb{H}_p$ have the form

$$\mathbf{u}\mathbf{v} = (iu_x + ju_y + ku_z)v = iv_x + jv_y + kv_z,$$

where $\mathbf{u} \triangleq iu_x + ju_y + ku_z$ is pure and unitary, v is the norm, and i, j, k are the generators of the Lie algebra $\mathfrak{s}^3 = \mathbb{H}_p$. Re-writing the condition above we have,

$$\dot{\mathbf{q}} = \mathbf{q} \mathbf{u} v \quad \in \mathcal{T}S^3_{\mathbf{q}_0},$$

which integrates to $\mathbf{q} = \mathbf{q}_0 \exp(\mathbf{u}vt)$. Letting $\mathbf{q}_0 = 1$ and defining $\phi \triangleq \mathbf{u}\phi \triangleq \mathbf{u}vt$ we get the exponential map,

$$\mathbf{q} = \exp(\mathbf{u}\phi) \triangleq \sum \frac{\phi^k}{k!} \mathbf{u}^k \quad \in S^3.$$

The powers of \mathbf{u} follow the pattern $1, \mathbf{u}, -1, -\mathbf{u}, 1, \dots$. We group the terms in 1 and \mathbf{u} and identify the series of $\cos \phi$ and $\sin \phi$. We get the closed form,

$$\mathbf{q} = \exp(\mathbf{u}\phi) = \cos(\phi) + \mathbf{u} \sin(\phi),$$

which is a beautiful extension of the Euler formula, $\exp(i\phi) = \cos \phi + i \sin \phi$. The elements of the Lie algebra $\mathbf{u}\phi \in \mathfrak{s}^3$ can be identified with the rotation vector $\theta \in \mathbb{R}^3$ through the mappings *hat* and *vee*,

$$\begin{aligned} \text{Hat : } & \mathbb{R}^3 \rightarrow \mathfrak{s}^3; \quad \theta \mapsto \theta^\wedge = 2\phi \\ \text{Vee : } & \mathfrak{s}^3 \rightarrow \mathbb{R}^3; \quad \phi \mapsto \phi^\vee = \theta/2, \end{aligned}$$

where the factor 2 accounts for the double effect of the quaternion in the rotation action, $\mathbf{x}' = \mathbf{q} \mathbf{x} \mathbf{q}^*$. With this choice of Hat and Vee, the quaternion exponential

$$\mathbf{q} = \text{Exp}(\mathbf{u}\theta) = \cos(\theta/2) + \mathbf{u} \sin(\theta/2)$$

is equivalent to the rotation matrix $\mathbf{R} = \text{Exp}(\mathbf{u}\theta)$.

Clearly from Fig. 6,

$$\mathcal{X} = \text{Exp}(\boldsymbol{\tau}) \triangleq \exp(\boldsymbol{\tau}^\wedge) \quad (23)$$

$$\boldsymbol{\tau} = \text{Log}(\mathcal{X}) \triangleq \log(\mathcal{X})^\vee. \quad (24)$$

See the Appendices for details on the implementation of these maps for different manifolds.

E. Plus and minus operators

Plus and minus allow us to introduce increments between elements of a (curved) manifold, and express them in its (flat) tangent vector space. Denoted by \oplus and \ominus , they combine one Exp/Log operation with one composition. Because of the non-commutativity of the composition, they are defined in right-

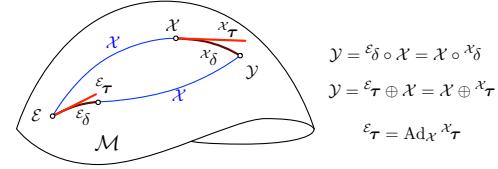


Figure 7. Two paths, $\mathcal{X} \circ \mathcal{X}_\delta$ and $\mathcal{X}_\delta \circ \mathcal{X}$, join the origin \mathcal{E} with the point \mathcal{Y} . They both compose the element \mathcal{X} with increments or ‘deltas’ expressed either in the local frame, \mathcal{X}_δ , or in the origin, \mathcal{E}_δ . Due to non-commutativity, the elements \mathcal{X}_δ and \mathcal{E}_δ are not equal. Their associated tangent vectors $x_\tau = \text{Log}(\mathcal{X}_\delta)$ and $x_\tau = \text{Log}(\mathcal{E}_\delta)$ are therefore unequal too. They are related by a linear transform $\mathcal{E}_\tau = \text{Ad}_{\mathcal{X}} x_\tau$. $\text{Ad}_{\mathcal{X}}$ is the adjoint of \mathcal{M} at \mathcal{X} .

and left- versions depending on the order of the operands. The right operators are (see Fig. 4-right),

$$\text{right-}\oplus : \quad \mathcal{Y} = \mathcal{X} \oplus \mathcal{X}_\tau \triangleq \mathcal{X} \circ \text{Exp}(\mathcal{X}_\tau) \in \mathcal{M} \quad (25)$$

$$\text{right-}\ominus : \quad \mathcal{X}_\tau = \mathcal{Y} \ominus \mathcal{X} \triangleq \text{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in \mathcal{T}\mathcal{M}_\mathcal{X}, \quad (26)$$

and the left operators,

$$\text{left-}\oplus : \quad \mathcal{Y} = \mathcal{E}_\tau \oplus \mathcal{X} \triangleq \text{Exp}(\mathcal{E}_\tau) \circ \mathcal{X} \in \mathcal{M} \quad (27)$$

$$\text{left-}\ominus : \quad \mathcal{E}_\tau = \mathcal{Y} \ominus \mathcal{X} \triangleq \text{Log}(\mathcal{Y} \circ \mathcal{X}^{-1}) \in \mathcal{T}\mathcal{M}_\mathcal{E}. \quad (28)$$

Because in (25) $\text{Exp}(\mathcal{X}_\tau)$ appears at the right hand side of the composition, \mathcal{X}_τ belongs to the tangent space at \mathcal{X} (see (26)): we say by convention² that \mathcal{X}_τ is expressed in the *local* frame at \mathcal{X} — and we note it with a left superscript. Conversely, in (27) $\text{Exp}(\mathcal{E}_\tau)$ is on the left and we have $\mathcal{E}_\tau \in \mathcal{T}\mathcal{M}_\mathcal{E}$: we say that \mathcal{E}_τ is expressed in the *global* frame. Notice that while left- and right- \oplus are distinguished by the operands order, the notation \ominus in (26) and (28) is ambiguous. In this work we express perturbations locally by default and therefore we use the right- forms of \oplus and \ominus by default.

F. The adjoint, and the adjoint matrix

If we identify \mathcal{Y} in (25, 27), we arrive at $\mathcal{E}_\tau \oplus \mathcal{X} = \mathcal{X} \oplus \mathcal{X}_\tau$, which determines a relation between the local and global tangent elements (Fig. 7). This is developed with (20, 25, 27) as

$$\text{Exp}(\mathcal{E}_\tau)\mathcal{X} = \mathcal{X} \text{Exp}(\mathcal{X}_\tau)$$

$$\exp(\mathcal{E}_\tau^\wedge) = \mathcal{X} \exp(\mathcal{X}_\tau^\wedge) \mathcal{X}^{-1} = \exp(\mathcal{X} \mathcal{X}_\tau^\wedge \mathcal{X}^{-1})$$

$$\mathcal{E}_\tau^\wedge = \mathcal{X} \mathcal{X}_\tau^\wedge \mathcal{X}^{-1}$$

We thus define the *adjoint* of \mathcal{M} at \mathcal{X} , noted $\text{Ad}_{\mathcal{X}}$, to be

$$\text{Ad}_{\mathcal{X}} : \mathbf{m} \rightarrow \mathbf{m}; \quad \tau^\wedge \mapsto \text{Ad}_{\mathcal{X}}(\tau^\wedge) \triangleq \mathcal{X} \tau^\wedge \mathcal{X}^{-1}, \quad (29)$$

so that $\mathcal{E}_\tau^\wedge = \text{Ad}_{\mathcal{X}}(\mathcal{X}_\tau^\wedge)$. This defines the *adjoint action* of the group on its own Lie algebra. The adjoint has two interesting (and easy to prove) properties,

$$\begin{aligned} \text{Linear : } \quad \text{Ad}_{\mathcal{X}}(a\tau^\wedge + b\sigma^\wedge) &= a\text{Ad}_{\mathcal{X}}(\tau^\wedge) \\ &\quad + b\text{Ad}_{\mathcal{X}}(\sigma^\wedge) \end{aligned}$$

$$\text{Homomorphism : } \quad \text{Ad}_{\mathcal{X}}(\text{Ad}_{\mathcal{Y}}(\tau^\wedge)) = \text{Ad}_{\mathcal{X}\mathcal{Y}}(\tau^\wedge).$$

²The convention sticks to that of frame transformation, e.g. ${}^G\mathbf{x} = \mathbf{R}^L\mathbf{x}$, where the matrix $\mathbf{R} \in SO(3)$ transforms local vectors into global. Notice that this convention is not shared by all authors, and for example [7] uses the opposite, ${}^L\mathbf{x} = \mathbf{R}^G\mathbf{x}$.

Example 6: The adjoint matrix of SE(3)

The SE(3) group of rigid body motions (see App. D) has group, Lie algebra and vector elements,

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \quad \boldsymbol{\tau}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix}, \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix}.$$

The adjoint matrix is identified by developing (31) as

$$\begin{aligned} \mathbf{Ad}_\mathbf{M} \boldsymbol{\tau} &= (\mathbf{M} \boldsymbol{\tau}^\wedge \mathbf{M}^{-1})^\vee = \dots = \\ &= \left(\begin{bmatrix} \mathbf{R} [\boldsymbol{\theta}]_\times \mathbf{R}^\top & -\mathbf{R} [\boldsymbol{\theta}]_\times \mathbf{R}^\top \mathbf{t} + \mathbf{R} \boldsymbol{\rho} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right)^\vee \\ &= \left(\begin{bmatrix} [\mathbf{R} \boldsymbol{\theta}]_\times & [\mathbf{t}]_\times \mathbf{R} \boldsymbol{\theta} + \mathbf{R} \boldsymbol{\rho} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right)^\vee \\ &= \begin{bmatrix} [\mathbf{t}]_\times \mathbf{R} \boldsymbol{\theta} + \mathbf{R} \boldsymbol{\rho} \\ \mathbf{R} \boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \end{aligned}$$

where we used $[\mathbf{R} \boldsymbol{\theta}]_\times = \mathbf{R} [\boldsymbol{\theta}]_\times \mathbf{R}^\top$ and $[\mathbf{a}]_\times \mathbf{b} = -[\mathbf{b}]_\times \mathbf{a}$. So the adjoint is

$$\mathbf{Ad}_\mathbf{M} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6 \times 6}.$$

Since it is linear, we can find an equivalent matrix operator $\mathbf{Ad}_\mathcal{X}$ that maps the isomorphic tangent vectors,

$$\mathbf{Ad}_\mathcal{X} : \mathbb{R}^m \rightarrow \mathbb{R}^m; \quad \mathcal{X} \boldsymbol{\tau} \mapsto \mathcal{E} \boldsymbol{\tau} = \mathbf{Ad}_\mathcal{X} \mathcal{X} \boldsymbol{\tau}, \quad (30)$$

which we call the *adjoint matrix*. This can be computed by applying $^\vee$ to (29), thus writing

$$\mathbf{Ad}_\mathcal{X} \boldsymbol{\tau} = (\mathcal{X} \boldsymbol{\tau}^\wedge \mathcal{X}^{-1})^\vee, \quad (31)$$

then developing the right hand side to identify the adjoint matrix (see Ex. 6 and the appendices). Additional properties of the adjoint matrix are,

$$\mathcal{X} \oplus \boldsymbol{\tau} = (\mathbf{Ad}_\mathcal{X} \boldsymbol{\tau}) \oplus \mathcal{X} \quad (32)$$

$$\mathbf{Ad}_{\mathcal{X}^{-1}} = \mathbf{Ad}_\mathcal{X}^{-1} \quad (33)$$

$$\mathbf{Ad}_{\mathcal{X} \mathcal{Y}} = \mathbf{Ad}_\mathcal{X} \mathbf{Ad}_\mathcal{Y}. \quad (34)$$

We will use the adjoint matrix often as a way to linearly transform vectors of the tangent space at \mathcal{X} onto vectors of the tangent space at the origin, with $\mathcal{E} \boldsymbol{\tau} = \mathbf{Ad}_\mathcal{X} \mathcal{X} \boldsymbol{\tau}$. In this work, the adjoint matrix will be referred to as simply the adjoint.

G. Derivatives on Lie groups

Among the different ways to define derivatives in the context of Lie groups, we concentrate in those in the form of Jacobian matrices mapping vector tangent spaces. This is sufficient here, since in these spaces uncertainties and increments can be properly and easily defined. Using these Jacobians, the formulas for uncertainty management in Lie groups will largely resemble those in vector spaces.

The Jacobians described hereafter fulfill the chain rule, so that we can easily compute any Jacobians from the partial Jacobians blocks of *inversion*, *composition*, *exponentiation* and *action*. See Section III-A for details and proofs.

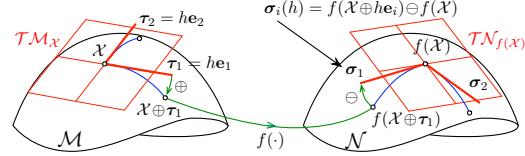


Figure 8. Right Jacobian of a function $f : \mathcal{M} \rightarrow \mathcal{N}$. The perturbation vectors in the canonical directions, $\tau_i = h \mathbf{e}_i \in \mathcal{T}\mathcal{M}_x$, are propagated to perturbation vectors $\sigma_i \in \mathcal{T}\mathcal{N}_{f(\mathcal{X})}$ through the processes of plus, apply $f()$, and minus (arrows), obtaining $\sigma_i(h) = f(\mathcal{X} \oplus h \mathbf{e}_i) \ominus f(\mathcal{X})$. The Jacobian columns are $\mathbf{j}_i = \lim_{h \rightarrow 0} \sigma_i(h)/h$. We write the full Jacobian simply as $\mathbf{J} = \partial f / \partial \mathcal{X} = \lim_{\boldsymbol{\tau} \rightarrow 0} \sigma(\boldsymbol{\tau})/\boldsymbol{\tau}$.

1) Reminder: Jacobians on vector spaces: For a multivariate function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, the Jacobian matrix is defined as the $n \times m$ matrix stacking all partial derivatives,

$$\mathbf{J} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \in \mathbb{R}^{n \times m}. \quad (35)$$

It is handy to define this matrix in the following form. Let us partition $\mathbf{J} = [\mathbf{j}_1 \dots \mathbf{j}_m]$, and let $\mathbf{j}_i = [\frac{\partial f_1}{\partial x_i} \dots \frac{\partial f_n}{\partial x_i}]^\top$ be its i -th column vector. This column vector responds to

$$\mathbf{j}_i = \frac{\partial f(\mathbf{x})}{\partial x_i} \triangleq \lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h \mathbf{e}_i) - f(\mathbf{x})}{h} \in \mathbb{R}^n, \quad (36)$$

where \mathbf{e}_i is the i -th vector of the natural basis of \mathbb{R}^m . Regarding the numerator, notice that the vector

$$\mathbf{v}_i(h) = f(\mathbf{x} + h \mathbf{e}_i) - f(\mathbf{x}) \in \mathbb{R}^n \quad (37)$$

is the variation of $f(\mathbf{x})$ when \mathbf{x} is perturbed in the direction of \mathbf{e}_i , and that the respective Jacobian column is just $\mathbf{j}_i = \partial \mathbf{v}_i(h) / \partial h = \lim_{h \rightarrow 0} \mathbf{v}_i(h)/h$. In this work, for the sake of convenience, we introduce the compact form,

$$\mathbf{J} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \lim_{\mathbf{h} \rightarrow 0} \frac{f(\mathbf{x} + \mathbf{h}) - f(\mathbf{x})}{\mathbf{h}} \in \mathbb{R}^{n \times m}, \quad (38)$$

with $\mathbf{h} \in \mathbb{R}^m$, which aglutinates all columns (36) to form the definition of (35). We remark that (38) is just a notation convenience, since division by the vector \mathbf{h} is undefined and proper computation requires (36). However, this form may be used to identify Jacobians by developing the numerator into a form linear in \mathbf{h} , and identifying the left hand side as the Jacobian, that is,

$$\lim_{\mathbf{h} \rightarrow 0} \frac{f(\mathbf{x} + \mathbf{h}) - f(\mathbf{x})}{\mathbf{h}} = \dots = \lim_{\mathbf{h} \rightarrow 0} \frac{\mathbf{J} \mathbf{h}}{\mathbf{h}} \triangleq \frac{\partial \mathbf{J} \mathbf{h}}{\partial \mathbf{h}} = \mathbf{J}. \quad (39)$$

Notice finally that for small values of \mathbf{h} we have the linear approximation,

$$f(\mathbf{x} + \mathbf{h}) \xrightarrow{\mathbf{h} \rightarrow 0} f(\mathbf{x}) + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \mathbf{h}. \quad (40)$$

2) Right Jacobians on Lie groups: Inspired by the standard derivative definition above, we can now use our right- \oplus and \ominus operators to define Jacobians of functions $f : \mathcal{M} \rightarrow \mathcal{N}$

acting on manifolds (see Fig. 8), obtaining a form akin to the above,

$$\begin{aligned} \frac{\mathcal{X}\partial f(\mathcal{X})}{\partial \mathcal{X}} &\triangleq \lim_{\tau \rightarrow 0} \frac{f(\mathcal{X} \oplus \tau) \ominus f(\mathcal{X})}{\tau} \in \mathbb{R}^{n \times m} \quad (41) \\ &= \lim_{\tau \rightarrow 0} \frac{\text{Log}(f(\mathcal{X})^{-1} \circ f(\mathcal{X} \circ \text{Exp}(\tau)))}{\tau} \\ &= \left. \frac{\partial \text{Log}(f(\mathcal{X})^{-1} \circ f(\mathcal{X} \circ \text{Exp}(\tau)))}{\partial \tau} \right|_{\tau=0}, \end{aligned}$$

where (41) mimics (38), and the other forms are alternative ways to look at it.³ We call this the *right Jacobian* of f . Crucially, and thanks to the way right- \oplus and \ominus operate, variations in \mathcal{X} and $f(\mathcal{X})$ are now expressed as vectors in the local tangent spaces, *i.e.*, tangent respectively at $\mathcal{X} \in \mathcal{M}$ and $f(\mathcal{X}) \in \mathcal{N}$. This derivative is then a proper Jacobian matrix $\mathbb{R}^{n \times m}$ linearly mapping the *local* tangent spaces $\mathcal{T}\mathcal{M}_{\mathcal{X}} \rightarrow \mathcal{T}\mathcal{N}_{f(\mathcal{X})}$ (and we mark the derivative with a local \mathcal{X} superscript). Just as in vector spaces, the columns of this matrix correspond to directional derivatives. That is, the vector (see Fig. 8 again, and compare the following to (37))

$$\sigma_i(h) = f(\mathcal{X} \oplus h\mathbf{e}_i) \ominus f(\mathcal{X}) \in \mathbb{R}^n \quad (42)$$

is the variation of $f(\mathcal{X})$ when \mathcal{X} varies in the direction of \mathbf{e}_i , and the respective Jacobian column is $\mathbf{j}_i = \partial\sigma_i(h)/\partial h$.

As before, we use (41) to effectively find Jacobians by resorting to the same mechanism (39). For example, for a 3D rotation $f : SO(3) \rightarrow \mathbb{R}^3$; $f(\mathbf{R}) = \mathbf{R}\mathbf{p}$, we have $\mathcal{M} = SO(3)$ and $\mathcal{N} = \mathbb{R}^3$ and so,

$$\begin{aligned} \frac{\mathcal{R}\partial\mathbf{R}\mathbf{p}}{\partial\mathbf{R}} &= \lim_{\theta \rightarrow 0} \frac{(\mathbf{R} \oplus \theta)\mathbf{p} \ominus \mathbf{R}\mathbf{p}}{\theta} = \lim_{\theta \rightarrow 0} \frac{\mathbf{R}\text{Exp}(\theta)\mathbf{p} - \mathbf{R}\mathbf{p}}{\theta} \\ &= \lim_{\theta \rightarrow 0} \frac{\mathbf{R}(\mathbf{I} + [\theta]_{\times})\mathbf{p} - \mathbf{R}\mathbf{p}}{\theta} = \lim_{\theta \rightarrow 0} \frac{\mathbf{R}[\theta]_{\times}\mathbf{p}}{\theta} \\ &= \lim_{\theta \rightarrow 0} \frac{-\mathbf{R}[\mathbf{p}]_{\times}\theta}{\theta} = -\mathbf{R}[\mathbf{p}]_{\times} \in \mathbb{R}^{3 \times 3}. \end{aligned}$$

Many examples of this can be observed in Section III and in the appendices. Remark that whenever the function f passes from one manifold to another, the plus and minus operators in (41) must be selected appropriately: \oplus for the domain \mathcal{M} , and \ominus for the codomain or image \mathcal{N} .

For small values of τ , the following approximation holds,

$$f(\mathcal{X} \oplus \mathcal{X}\tau) \xrightarrow{\mathcal{X}\tau \rightarrow 0} f(\mathcal{X}) \oplus \frac{\mathcal{X}\partial f(\mathcal{X})}{\partial \mathcal{X}} \mathcal{X}\tau \in \mathcal{N}. \quad (43)$$

3) *Left Jacobians on Lie groups:* Derivatives can also be defined from the left- plus and minus operators, leading to,

$$\begin{aligned} \frac{\varepsilon\partial f(\mathcal{X})}{\partial \mathcal{X}} &\triangleq \lim_{\tau \rightarrow 0} \frac{f(\tau \oplus \mathcal{X}) \ominus f(\mathcal{X})}{\tau} \in \mathbb{R}^{n \times m} \quad (44) \\ &= \lim_{\tau \rightarrow 0} \frac{\text{Log}(f(\text{Exp}(\tau) \circ \mathcal{X}) \circ f(\mathcal{X})^{-1})}{\tau}, \end{aligned}$$

which we call the *left Jacobian* of f . Notice that now $\tau \in \mathcal{T}\mathcal{M}_{\mathcal{E}}$, and the numerator belongs to $\mathcal{T}\mathcal{N}_{\mathcal{E}}$, thus the left Jacobian is a $n \times m$ matrix mapping the *global* tangent spaces, $\mathcal{T}\mathcal{M}_{\mathcal{E}} \rightarrow \mathcal{T}\mathcal{N}_{\mathcal{E}}$, which are the Lie algebras of \mathcal{M} and \mathcal{N} (and

³The notation $\frac{\partial f}{\partial \mathcal{X}}$ is chosen in front of other alternatives in order to make the chain rule readable, *i.e.*, $\frac{\partial \mathcal{Z}}{\partial \mathcal{X}} = \frac{\partial \mathcal{Z}}{\partial \mathcal{Y}} \frac{\partial \mathcal{Y}}{\partial \mathcal{X}}$.

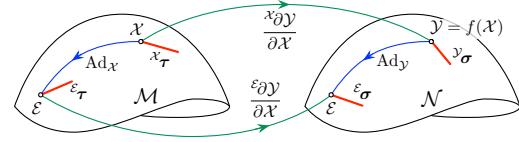


Figure 9. Linear maps between all tangent spaces involved in a function $\mathcal{Y} = f(\mathcal{X})$, from \mathcal{M} to \mathcal{N} . The linear maps $\varepsilon_{\tau} = \text{Ad}_{\mathcal{X}} \mathcal{X}_{\tau}$, $\varepsilon_{\sigma} = \text{Ad}_{\mathcal{Y}} \mathcal{Y}_{\sigma}$, $\varepsilon_{\sigma} = \frac{\varepsilon \partial \mathcal{Y}}{\partial \mathcal{X}} \varepsilon_{\tau}$, and $\mathcal{Y}_{\sigma} = \frac{x \partial \mathcal{Y}}{\partial \mathcal{X}} \mathcal{X}_{\tau}$, form a loop that leads to (46).

we mark the derivative with a global or origin \mathcal{E} superscript). For small values of τ the following holds,

$$f(\varepsilon_{\tau} \oplus \mathcal{X}) \xrightarrow{\varepsilon_{\tau} \rightarrow 0} \frac{\varepsilon \partial f(\mathcal{X})}{\partial \mathcal{X}} \varepsilon_{\tau} \oplus f(\mathcal{X}) \in \mathcal{N}. \quad (45)$$

We can show from (32, 43, 45) (see Fig. 9) that left and right Jacobians are related by the adjoints of \mathcal{M} and \mathcal{N} ,

$$\frac{\varepsilon \partial f(\mathcal{X})}{\partial \mathcal{X}} \text{Ad}_{\mathcal{X}} = \text{Ad}_{f(\mathcal{X})} \frac{\mathcal{X} \partial f(\mathcal{X})}{\partial \mathcal{X}}. \quad (46)$$

4) *Crossed right-left Jacobians:* One can also define Jacobians using right-plus but left-minus, or vice versa. These are useful in some practical cases, since they map local to global tangents or vice versa. To keep it short, we will just relate them to the other Jacobians through the adjoints,

$$\frac{\varepsilon \partial \mathcal{Y}}{\varepsilon \partial \mathcal{X}} = \frac{\varepsilon \partial \mathcal{Y}}{\varepsilon \partial \mathcal{X}} \text{Ad}_{\mathcal{X}} = \text{Ad}_{\mathcal{Y}} \frac{\mathcal{Y} \partial \mathcal{Y}}{\mathcal{Y} \partial \mathcal{X}} \quad (47)$$

$$\frac{\mathcal{Y} \partial \mathcal{Y}}{\mathcal{Y} \partial \mathcal{X}} \text{Ad}_{\mathcal{X}}^{-1} = \text{Ad}_{\mathcal{Y}}^{-1} \frac{\varepsilon \partial \mathcal{Y}}{\varepsilon \partial \mathcal{X}} \quad (48)$$

where $\mathcal{Y} = f(\mathcal{X})$. Now, the sub- and super-scripts indicate the reference frames where the differentials are expressed. Respective small-tau approximations read,

$$f(\mathcal{X} \oplus \mathcal{X}\tau) \xrightarrow{\mathcal{X}\tau \rightarrow 0} \frac{\mathcal{X} \partial f(\mathcal{X})}{\mathcal{X} \partial \mathcal{X}} \mathcal{X}\tau \oplus f(\mathcal{X}) \quad (49)$$

$$f(\varepsilon_{\tau} \oplus \mathcal{X}) \xrightarrow{\varepsilon_{\tau} \rightarrow 0} f(\mathcal{X}) \oplus \frac{f(\mathcal{X}) \partial f(\mathcal{X})}{\varepsilon \partial \mathcal{X}} \varepsilon_{\tau} \quad (50)$$

H. Uncertainty in manifolds, covariance propagation

We define local perturbations τ around a point $\bar{\mathcal{X}} \in \mathcal{M}$ in the tangent vector space $\mathcal{T}\mathcal{M}_{\bar{\mathcal{X}}}$, using right- \oplus and \ominus ,

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \tau, \quad \tau = \mathcal{X} \ominus \bar{\mathcal{X}} \in \mathcal{T}\mathcal{M}_{\bar{\mathcal{X}}}. \quad (51)$$

Covariances matrices can be properly defined on this tangent space at $\bar{\mathcal{X}}$ through the standard expectation operator $\mathbb{E}[\cdot]$,

$$\Sigma_{\mathcal{X}} \triangleq \mathbb{E}[\tau\tau^{\top}] = \mathbb{E}[(\mathcal{X} \ominus \bar{\mathcal{X}})(\mathcal{X} \ominus \bar{\mathcal{X}})^{\top}] \in \mathbb{R}^{n \times n}, \quad (52)$$

allowing us to define Gaussian variables on manifolds, $\mathcal{X} \sim \mathcal{N}(\bar{\mathcal{X}}, \Sigma_{\mathcal{X}})$, see Fig. 10. Notice that although we write $\Sigma_{\mathcal{X}}$, the covariance is rather that of the tangent perturbation τ . Since the dimension m of $\mathcal{T}\mathcal{M}$ matches the degrees of freedom of \mathcal{M} , these covariances are well defined.⁴

⁴A naive definition $\Sigma_{\mathcal{X}} \triangleq \mathbb{E}[(\mathcal{X} - \bar{\mathcal{X}})(\mathcal{X} - \bar{\mathcal{X}})^{\top}]$ is always ill-defined if $\text{size}(\mathcal{X}) > \dim(\mathcal{M})$, which is the case for most non-trivial manifolds.

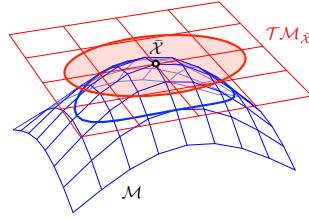


Figure 10. Uncertainty around a point $\bar{\mathcal{X}} \in \mathcal{M}$ is properly expressed as a covariance on the vector space tangent at the point (red). Using \oplus (51), the probability ellipses in the tangent space are wrapped over the manifold (blue), thus illustrating the probability concentration region on the group.

Perturbations can also be expressed in the global reference, that is, in the tangent space at the origin $T\mathcal{M}_{\mathcal{E}}$, using left- \oplus and \ominus ,

$$\mathcal{X} = \tau \oplus \bar{\mathcal{X}}, \quad \tau = \mathcal{X} \ominus \bar{\mathcal{X}} \in T\mathcal{M}_{\mathcal{E}}. \quad (53)$$

This allows global specification of covariance matrices using left-minus in (52). For example, a 3D orientation that is known up to rotations in the horizontal plane can be associated to a covariance ${}^{\mathcal{E}}\Sigma = \text{diag}(\sigma_{\phi}^2, \sigma_{\theta}^2, \infty)$. Since “horizontal” is a global specification, ${}^{\mathcal{E}}\Sigma$ must be specified in the global reference.

Notice finally that there is no consensus on this last point. While [3] and us use local perturbations $\mathcal{X} = \bar{\mathcal{X}} \oplus {}^{\mathcal{X}}\tau$, [5], [8] use perturbations around the origin, $\mathcal{X} = {}^{\mathcal{E}}\tau \oplus \bar{\mathcal{X}}$, yielding global covariance specifications. Since global and local perturbations are related by the adjoint (30), their covariances can be transformed according to

$${}^{\mathcal{E}}\Sigma_{\mathcal{X}} = \mathbf{Ad}_{\mathcal{X}} {}^{\mathcal{X}}\Sigma_{\mathcal{X}} \mathbf{Ad}_{\mathcal{X}}^{\top}. \quad (54)$$

Covariance propagation through a function $f : \mathcal{M} \rightarrow \mathcal{N}; \mathcal{X} \mapsto \mathcal{Y} = f(\mathcal{X})$ just requires the linearization (43) with Jacobian matrices (41) to yield the familiar formula,

$$\Sigma_{\mathcal{Y}} \approx \frac{\partial f}{\partial \mathcal{X}} \Sigma_{\mathcal{X}} \frac{\partial f^{\top}}{\partial \mathcal{X}} \in \mathbb{R}^{m \times m}. \quad (55)$$

I. Discrete integration on manifolds

The exponential map $\mathcal{X}(t) = \mathcal{X}_0 \circ \text{Exp}(\mathbf{v}t)$ performs the continuous-time integral of constant velocities $\mathbf{v} \in T\mathcal{M}_{\mathcal{X}_0}$ onto the manifold. Non-constant velocities $\mathbf{v}(t)$ are typically handled by segmenting them into piecewise constant bits $\mathbf{v}_k \in T\mathcal{M}_{\mathcal{X}_{k-1}}$, of (short) duration δt_k , and writing the discrete integral

$$\mathcal{X}_k = \mathcal{X}_0 \circ \text{Exp}(\mathbf{v}_1 \delta t_1) \circ \text{Exp}(\mathbf{v}_2 \delta t_2) \circ \dots \circ \text{Exp}(\mathbf{v}_k \delta t_k).$$

Equivalently (Fig. 11), we can define $\tau_k = \mathbf{v}_k \delta t_k$ and use \oplus to construct the integral as a “sum” of (small) discrete tangent steps $\tau_k \in T\mathcal{M}_{\mathcal{X}_{k-1}}$, i.e., $\mathcal{X}_k \triangleq \mathcal{X}_0 \oplus \tau_1 \oplus \dots \oplus \tau_k$. We write all these variants in recursive form,

$$\mathcal{X}_k = \mathcal{X}_{k-1} \oplus \tau_k = \mathcal{X}_{k-1} \circ \text{Exp}(\tau_k) = \mathcal{X}_{k-1} \circ \text{Exp}(\mathbf{v}_k \delta t_k). \quad (56)$$

Common examples are the integration of 3D angular rates $\boldsymbol{\omega}$ into the rotation matrix, $\mathbf{R}_k = \mathbf{R}_{k-1} \text{Exp}(\boldsymbol{\omega}_k \delta t)$, or into the quaternion, $\mathbf{q}_k = \mathbf{q}_{k-1} \text{Exp}(\boldsymbol{\omega}_k \delta t)$.

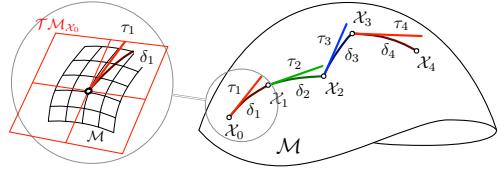


Figure 11. Motion integration on a manifold. Each motion data produces a step $\tau_k \in T\mathcal{M}_{\mathcal{X}_{k-1}}$, which is wrapped to a local motion increment or ‘delta’ $\delta_k = \text{Exp}(\tau_k) \in \mathcal{M}$, and then composed with \mathcal{X}_{k-1} to yield $\mathcal{X}_k = \mathcal{X}_{k-1} \circ \delta_k = \mathcal{X}_{k-1} \circ \text{Exp}(\tau_k) = \mathcal{X}_{k-1} \oplus \tau_k \in \mathcal{M}$.

III. DERIVATION RULES ON MANIFOLDS

For all the typical manifolds \mathcal{M} that we use, we can determine closed forms for the elementary Jacobians of *inversion*, *composition*, *exponentiation* and *action*. Moreover, some of these forms can be related to the adjoint $\mathbf{Ad}_{\mathcal{X}}$, which becomes a central block of the derivation process. Other forms for Log, \oplus and \ominus can be easily derived from them. Once these forms or ‘blocks’ are found, all other Jacobians follow by the chain rule. Except for the so called *left Jacobian*, which we also present below, all acobians developed here are right-Jacobians, i.e., defined by (41). By following the hints here, the interested reader should find no particular difficulties in developing the left-Jacobians. For the reader not willing to do this effort, equation (46) can be used to this end, since

$$\frac{\varepsilon \partial f(\mathcal{X})}{\partial \mathcal{X}} = \mathbf{Ad}_{f(\mathcal{X})} \frac{x \partial f(\mathcal{X})}{\partial \mathcal{X}} \mathbf{Ad}_{\mathcal{X}}^{-1}. \quad (57)$$

We use the notations $\mathbf{J}_{\mathcal{X}}^{f(\mathcal{X})} \triangleq \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}}$ and $\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}} \triangleq \frac{\partial \mathcal{Y}}{\partial \mathcal{X}}$.

A. The chain rule

For $\mathcal{Y} = f(\mathcal{X})$ and $\mathcal{Z} = g(\mathcal{Y})$ we have $\mathcal{Z} = g(f(\mathcal{X}))$. The chain rule simply states,

$$\frac{\partial \mathcal{Z}}{\partial \mathcal{X}} = \frac{\partial \mathcal{Z}}{\partial \mathcal{Y}} \frac{\partial \mathcal{Y}}{\partial \mathcal{X}} \quad \text{or} \quad \mathbf{J}_{\mathcal{X}}^{\mathcal{Z}} = \mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}} \mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}. \quad (58)$$

We prove it here for the right Jacobian using (43) thrice,

$$\begin{aligned} g(f(\mathcal{X})) \oplus \mathbf{J}_{\mathcal{X}}^{\mathcal{Z}} \tau &\leftarrow g(f(\mathcal{X} \oplus \tau)) \rightarrow g(f(\mathcal{X}) \oplus \mathbf{J}_{\mathcal{X}}^{\mathcal{Y}} \tau) \\ &\rightarrow g(f(\mathcal{X})) \oplus \mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}} \mathbf{J}_{\mathcal{X}}^{\mathcal{Y}} \tau \end{aligned}$$

with the arrows indicating limit as $\tau \rightarrow 0$, and so $\mathbf{J}_{\mathcal{X}}^{\mathcal{Z}} = \mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}} \mathbf{J}_{\mathcal{X}}^{\mathcal{Y}}$. The proof for the left and crossed Jacobians is akin, using respectively (45, 49, 50). Notice that when mixing right, left and crossed Jacobians, we need to chain also the reference frames, as in e.g.

$$\frac{z \partial \mathcal{Z}}{\varepsilon \partial \mathcal{X}} = \frac{z \partial \mathcal{Z}}{y \partial \mathcal{Y}} \frac{y \partial \mathcal{Y}}{\varepsilon \partial \mathcal{X}} = \frac{z \partial \mathcal{Z}}{\varepsilon \partial \mathcal{Y}} \frac{\varepsilon \partial \mathcal{Y}}{\varepsilon \partial \mathcal{X}} \quad (59)$$

$$\frac{\varepsilon \partial \mathcal{Z}}{x \partial \mathcal{X}} = \frac{\varepsilon \partial \mathcal{Z}}{y \partial \mathcal{Y}} \frac{y \partial \mathcal{Y}}{x \partial \mathcal{X}} = \frac{\varepsilon \partial \mathcal{Z}}{y \partial \mathcal{Y}} \frac{\varepsilon \partial \mathcal{Y}}{x \partial \mathcal{X}}, \quad (60)$$

Example 7: $SE(n)$ vs. $T(n) \times SO(n)$ vs. $\langle \mathbb{R}^n, SO(n) \rangle$

We consider the space of translations $\mathbf{t} \in \mathbb{R}^n$ and rotations $\mathbf{R} \in SO(n)$. We have for this the well-known $SE(n)$ manifold of rigid motions (see Apps. C and D), which can also be constructed as $T(n) \times SO(n)$ (see Apps. A, B and E). These two are very similar, but have different tangent parametrizations: while $SE(n)$ uses $\boldsymbol{\tau} = (\boldsymbol{\theta}, \boldsymbol{\rho})$ with $\mathcal{X} = \exp(\boldsymbol{\tau}^\wedge)$, $T(n) \times SO(n)$ uses $\boldsymbol{\tau} = (\boldsymbol{\theta}, \mathbf{p})$ with $\mathcal{X} = \exp(\mathbf{p}^\wedge) \exp(\boldsymbol{\theta}^\wedge)$. They share the rotational part $\boldsymbol{\theta}$, but clearly $\boldsymbol{\rho} \neq \mathbf{p}$ (see [9, pag. 35] for further details). In short, $SE(n)$ performs translation and rotation simultaneously as a continuum, while $T(n) \times SO(n)$ performs chained translation+rotation. In radical contrast, in the composite $\langle \mathbb{R}^n, SO(n) \rangle$ rotations and translations do not interact at all. By combining composition with $\text{Exp}()$ we obtain the (right) plus operators,

$$\begin{aligned} SE(n) : \quad \mathcal{X} \oplus \boldsymbol{\tau} &= \begin{bmatrix} \mathbf{R} \text{Exp}(\boldsymbol{\theta}) & \mathbf{t} + \mathbf{R}\mathbf{v}\boldsymbol{\rho} \\ 0 & 1 \end{bmatrix} \\ T(n) \times SO(n) : \quad \mathcal{X} \oplus \boldsymbol{\tau} &= \begin{bmatrix} \mathbf{R} \text{Exp}(\boldsymbol{\theta}) & \mathbf{t} + \mathbf{R}\mathbf{p} \\ 0 & 1 \end{bmatrix} \\ \langle \mathbb{R}^n, SO(n) \rangle : \quad \mathcal{X} \diamond \boldsymbol{\tau} &= \begin{bmatrix} \mathbf{t} + \mathbf{p} \\ \mathbf{R} \text{Exp}(\boldsymbol{\theta}) \end{bmatrix} \end{aligned}$$

where either \oplus may be used for the system dynamics, e.g. motion integration, but usually not \diamond , which might however be used to model perturbations. Their respective minus operators read,

$$\begin{aligned} SE(n) : \quad \mathcal{X}_2 \ominus \mathcal{X}_1 &= \begin{bmatrix} \mathbf{V}_1^{-1} \mathbf{R}_1^\top (\mathbf{p}_2 - \mathbf{p}_1) \\ \text{Log}(\mathbf{R}_1^\top \mathbf{R}_2) \end{bmatrix} \\ T(n) \times SO(n) : \quad \mathcal{X}_2 \ominus \mathcal{X}_1 &= \begin{bmatrix} \mathbf{R}_1^\top (\mathbf{p}_2 - \mathbf{p}_1) \\ \text{Log}(\mathbf{R}_1^\top \mathbf{R}_2) \end{bmatrix} \\ \langle \mathbb{R}^n, SO(n) \rangle : \quad \mathcal{X}_2 \diamond \mathcal{X}_1 &= \begin{bmatrix} \mathbf{p}_2 - \mathbf{p}_1 \\ \text{Log}(\mathbf{R}_1^\top \mathbf{R}_2) \end{bmatrix}, \end{aligned}$$

where now, interestingly, \diamond can be used to evaluate errors and uncertainty. This makes $\oplus, \diamond, \ominus$ valuable operators for computing derivatives and covariances.

thereby fulfilling the group axioms, as well as a non-interacting “exponential map” (notice the angled brackets),

$$\text{Exp}(\boldsymbol{\tau}) \triangleq \begin{bmatrix} \text{Exp}(\boldsymbol{\tau}_1) \\ \vdots \\ \text{Exp}(\boldsymbol{\tau}_M) \end{bmatrix}, \quad \text{Log}(\mathcal{X}) \triangleq \begin{bmatrix} \text{Log}(\mathcal{X}) \\ \vdots \\ \text{Log}(\mathcal{X}_M) \end{bmatrix}, \quad (85)$$

thereby ensuring smoothness. These yield the composite’s right- plus and minus (notice the diamond symbols),

$$\mathcal{X} \diamond \boldsymbol{\tau} \triangleq \mathcal{X} \diamond \text{Exp}(\boldsymbol{\tau}) \quad (86)$$

$$\mathcal{Y} \diamond \mathcal{X} \triangleq \text{Log}(\mathcal{X}^\diamond \diamond \mathcal{Y}). \quad (87)$$

The key consequence of these considerations (see Ex. 7) is that new derivatives can be defined,⁵ using \oplus and \diamond ,

$$\frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \triangleq \lim_{\boldsymbol{\tau} \rightarrow 0} \frac{f(\mathcal{X} \diamond \boldsymbol{\tau}) \diamond f(\mathcal{X})}{\boldsymbol{\tau}}. \quad (88)$$

⁵We assume here right derivatives, but the same applies to left derivatives.

With this derivative, Jacobians of functions $f : \mathcal{M} \rightarrow \mathcal{N}$ acting on composite manifolds can be determined in a per-block basis, which yields simple expressions requiring only knowledge on the manifold blocks of the composite,

$$\frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} = \begin{bmatrix} \frac{\partial f_1}{\partial \mathcal{X}_1} & \cdots & \frac{\partial f_1}{\partial \mathcal{X}_M} \\ \vdots & & \vdots \\ \frac{\partial f_N}{\partial \mathcal{X}_1} & \cdots & \frac{\partial f_N}{\partial \mathcal{X}_M} \end{bmatrix}, \quad (89)$$

where $\partial f_i / \partial \mathcal{X}_j$ are each computed with (41). For small values of $\boldsymbol{\tau}$ the following holds,

$$f(\mathcal{X} \diamond \boldsymbol{\tau}) \xrightarrow{\boldsymbol{\tau} \rightarrow 0} f(\mathcal{X}) \diamond \frac{\partial f(\mathcal{X})}{\partial \mathcal{X}} \boldsymbol{\tau} \in \mathcal{N}. \quad (90)$$

When using these derivatives, covariances and uncertainty propagation must follow the convention. In particular, the covariance matrix (52) becomes

$$\Sigma_{\mathcal{X}} \triangleq \mathbb{E}[(\mathcal{X} \diamond \bar{\mathcal{X}})(\mathcal{X} \diamond \bar{\mathcal{X}})^\top] \in \mathbb{R}^{n \times n}, \quad (91)$$

for which the linearized propagation (55) using (88) applies.

V. LANDMARK-BASED LOCALIZATION AND MAPPING

We provide three applicative examples of the theory for robot localization and mapping. The first one is a Kalman filter for landmark-based localization. The second one is a graph-based smoothing method for simultaneous localization and mapping. The third one adds sensor self-calibration. They are based on a common setup, explained as follows.

We consider a robot in the plane surrounded by a small number of punctual landmarks or *beacons*. The robot receives control actions in the form of axial and angular velocities, and is able to measure the location of the beacons with respect to its own reference frame.

The robot pose is in $SE(2)$ and the beacon positions in \mathbb{R}^2 ,

$$\mathcal{X} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(2), \quad \mathbf{b}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix} \in \mathbb{R}^2.$$

The control signal \mathbf{u} is a twist in $\mathfrak{se}(2)$ comprising longitudinal velocity v and angular velocity ω , with no lateral velocity component, integrated over the sampling time δt . The control is corrupted by additive Gaussian noise $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{W})$. This noise accounts for possible lateral slippages u_s through a value of $\sigma_s \neq 0$,

$$\mathbf{u} = \begin{bmatrix} u_v \\ u_s \\ u_\omega \end{bmatrix} = \begin{bmatrix} v \delta t \\ 0 \\ \omega \delta t \end{bmatrix} + \mathbf{w} \in \mathfrak{se}(2) \quad (92)$$

$$\mathbf{W} = \begin{bmatrix} \sigma_v^2 \delta t & 0 & 0 \\ 0 & \sigma_s^2 \delta t & 0 \\ 0 & 0 & \sigma_\omega^2 \delta t \end{bmatrix} \in \mathbb{R}^{3 \times 3}. \quad (93)$$

At the arrival of a control \mathbf{u}_j at time j , the robot pose is updated with (56),

$$\mathcal{X}_j = \mathcal{X}_i \oplus \mathbf{u}_j \triangleq \mathcal{X}_i \text{Exp}(\mathbf{u}_j). \quad (94)$$

Landmark measurements are of the range and bearing type, though they are put in Cartesian form for simplicity. Their noise $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{N})$ is zero mean Gaussian,

$$\mathbf{y}_k = \mathcal{X}^{-1} \cdot \mathbf{b}_k + \mathbf{n} = \mathbf{R}^\top (\mathbf{b}_k - \mathbf{t}) + \mathbf{n} \in \mathbb{R}^2 \quad (95)$$

$$\mathbf{N} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \in \mathbb{R}^{2 \times 2}, \quad (96)$$

where we notice the rigid motion action $\mathcal{X}^{-1} \cdot \mathbf{b}_k$ (see App. C).

A. Localization with error-state Kalman filter on manifold

We initially consider the beacons \mathbf{b}_k situated at known positions. We define the pose to estimate as $\hat{\mathcal{X}} \in SE(2)$. The estimation error $\delta\mathbf{x}$ and its covariance \mathbf{P} are expressed in the tangent space at $\hat{\mathcal{X}}$ with (51, 52),

$$\begin{aligned} \delta\mathbf{x} &\triangleq \mathcal{X} \ominus \hat{\mathcal{X}} && \in \mathbb{R}^3 \\ \mathbf{P} &\triangleq \mathbb{E}[(\mathcal{X} \ominus \hat{\mathcal{X}})(\mathcal{X} \ominus \hat{\mathcal{X}})^\top] && \in \mathbb{R}^{3 \times 3}. \end{aligned}$$

At each robot motion we apply KF prediction,

$$\hat{\mathcal{X}}_j = \hat{\mathcal{X}}_i \oplus \mathbf{u}_j \quad (97)$$

$$\mathbf{P}_j = \mathbf{J}_{\mathcal{X}_i}^{\mathcal{X}_j} \mathbf{P}_i \mathbf{J}_{\mathcal{X}_i}^{\mathcal{X}_j \top} + \mathbf{J}_{\mathbf{u}_j}^{\mathcal{X}_j} \mathbf{W}_j \mathbf{J}_{\mathbf{u}_j}^{\mathcal{X}_j \top}, \quad (98)$$

with the Jacobians computed from the blocks in App. C,

$$\mathbf{J}_{\mathcal{X}_i}^{\mathcal{X}_j} = \mathbf{J}_{\hat{\mathcal{X}}_i}^{\hat{\mathcal{X}}_j \oplus \mathbf{u}_j} = \text{Ad}_{\text{Exp}(\mathbf{u}_j)}^{-1}$$

$$\mathbf{J}_{\mathbf{u}_j}^{\mathcal{X}_j} = \mathbf{J}_{\hat{\mathcal{X}}_i}^{\hat{\mathcal{X}}_j \oplus \mathbf{u}_j} = \mathbf{J}_r(\mathbf{u}_j).$$

At each beacon measurement \mathbf{y}_k we apply ESKF correction,

$$\text{Innovation : } \mathbf{z} = \mathbf{y}_k - \hat{\mathcal{X}}^{-1} \cdot \mathbf{b}_k$$

$$\text{Innovation cov. : } \mathbf{Z} = \mathbf{H} \mathbf{P} \mathbf{H}^\top + \mathbf{N}$$

$$\text{Kalman gain : } \mathbf{K} = \mathbf{P} \mathbf{H}^\top \mathbf{Z}^{-1}$$

$$\text{Observed error : } \delta\mathbf{x} = \mathbf{K} \mathbf{z}$$

$$\text{State update : } \hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \oplus \delta\mathbf{x} \quad (99)$$

$$\text{Cov. update : } \mathbf{P} \leftarrow \mathbf{P} - \mathbf{K} \mathbf{Z} \mathbf{K}^\top, \quad (100)$$

with the Jacobian computed from the blocks in App. C,

$$\begin{aligned} \mathbf{H} &= \mathbf{J}_{\mathcal{X}}^{\mathcal{X}^{-1} \cdot \mathbf{b}_k} = \mathbf{J}_{\mathcal{X}^{-1}}^{\mathcal{X}^{-1} \cdot \mathbf{b}_k} \mathbf{J}_{\mathcal{X}}^{\mathcal{X}^{-1}} \\ &= [\mathbf{R}^\top \quad \mathbf{R}^\top [1]_\times \mathbf{b}_k] \begin{bmatrix} -\mathbf{R} & [1]_\times \mathbf{t} \\ 0 & -1 \end{bmatrix} \\ &= -[\mathbf{I} \quad \mathbf{R}[1]_\times (\mathbf{b}_k - \mathbf{t})]. \end{aligned}$$

Notice that the only changes with respect to a regular EKF are in (97) and (99), where a regular $+$ is substituted by \oplus . The Jacobians on the contrary are all computed using the Lie theory (see App. C). However, their usage is the same as what we do in standard EKF — see e.g. the equation of the Kalman gain, which is the standard $\mathbf{K} = \mathbf{P} \mathbf{H}^\top (\mathbf{H} \mathbf{P} \mathbf{H}^\top + \mathbf{N})^{-1}$.

B. Smoothing and Mapping with graph-based optimization

We consider now the problem of smoothing and mapping (SAM), where the variables to estimate are the beacons' locations and the robot's trajectory. The solver of choice is a graph-based iterative least-squares optimizer. For simplicity, we assume the trajectory comprised of three robot poses

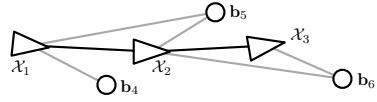


Figure 12. SAM graph problem with 3 poses and 3 beacons. There are 2 motion measurements (black) and 5 beacon measurements (gray).

$\{\mathcal{X}_1 \dots \mathcal{X}_3\}$, and a world with three beacons $\{\mathbf{b}_4 \dots \mathbf{b}_6\}$. The problem state is the composite

$$\mathcal{X} = \langle \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathbf{b}_4, \mathbf{b}_5, \mathbf{b}_6 \rangle, \quad \mathcal{X}_i \in SE(2), \quad \mathbf{b}_k \in \mathbb{R}^2. \quad (101)$$

The resulting graph is shown in Fig. 12. Motion measurements from pose i to j are derived from (94), while measurements of beacon k from pose i respond to (95),

$$\mathbf{u}_{ij} = \mathcal{X}_j \ominus \mathcal{X}_i + \mathbf{w}_{ij} = \text{Log}(\mathcal{X}_i^{-1} \mathcal{X}_j) + \mathbf{w}_{ij} \quad (102)$$

$$\mathbf{y}_{ik} = \mathcal{X}_i^{-1} \cdot \mathbf{b}_k + \mathbf{n}_{ik} \quad (103)$$

Each of these measurements comes with an information matrix, $\Omega_{ij} \triangleq \mathbf{W}_{ij}^{-1}$ and $\Omega_{ik} \triangleq \mathbf{N}_{ik}^{-1}$. For each measurement, we can compute the expectation residual

$$\text{motion residual : } \mathbf{r}_{ij}(\mathcal{X}) = \Omega_{ij}^{\top/2} (\mathbf{u}_{ij} - (\hat{\mathcal{X}}_j \ominus \hat{\mathcal{X}}_i))$$

$$\text{beacon residual : } \mathbf{r}_{ik}(\mathcal{X}) = \Omega_{ik}^{\top/2} (\mathbf{y}_{ik} - \hat{\mathcal{X}}_i^{-1} \cdot \mathbf{b}_k).$$

The optimum update step $\delta\mathbf{x}$ stems from minimizing

$$\delta\mathbf{x}^* = \arg \min_{\delta\mathbf{x}} \sum_{p \in \mathcal{P}} \mathbf{r}_p(\mathcal{X} \oplus \delta\mathbf{x})^\top \mathbf{r}_p(\mathcal{X} \oplus \delta\mathbf{x}) \quad (104)$$

with $\mathcal{P} = \{12, 23, 14, 15, 25, 26, 36\}$ the set of node pairs of each measurement (see Fig. 12). The problem is solved iteratively as follows. Each residual in the sum (104) is linearized to $\mathbf{r}_p(\mathcal{X} \oplus \delta\mathbf{x}) \approx \mathbf{r}_p(\mathcal{X}) + \mathbf{J}_{\mathcal{X}}^{\mathbf{r}_p} \delta\mathbf{x}$ following (90), where $\mathbf{J}_{\mathcal{X}}^{\mathbf{r}_p}$ are sparse Jacobians. The non-zero blocks of these Jacobians, that is $\mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{12}}, \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{23}}, \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{14}}, \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{15}}, \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{23}}, \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{14}}, \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{25}}, \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{14}}, \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{15}}, \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{25}}, \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{26}}, \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{36}}$, can be easily computed following the methods in Section V-A, and noticing that by definition $\mathbf{J}_{\delta\mathbf{x}}^{f(\mathcal{X} \oplus \delta\mathbf{x})}|_{\delta\mathbf{x}=0} = \mathbf{J}_{\mathcal{X}}^{f(\mathcal{X} \oplus \delta\mathbf{x})}|_{\delta\mathbf{x}=0} = \mathbf{J}_{\mathcal{X}}^{f(\mathcal{X})}$. Building the total Jacobian matrix and residual vector,

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{12}} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{12}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{23}} & \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{23}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{14}} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_4}^{\mathbf{r}_{14}} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{15}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_5}^{\mathbf{r}_{15}} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{25}} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_5}^{\mathbf{r}_{25}} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{26}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_6}^{\mathbf{r}_{26}} \\ \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{36}} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_6}^{\mathbf{r}_{36}} \end{bmatrix} \quad \mathbf{r} = \begin{bmatrix} \mathbf{r}_{12} \\ \mathbf{r}_{23} \\ \mathbf{r}_{14} \\ \mathbf{r}_{15} \\ \mathbf{r}_{25} \\ \mathbf{r}_{26} \\ \mathbf{r}_{36} \end{bmatrix} \quad (105)$$

the linearized (104) is now transformed to minimizing

$$\delta\mathbf{x}^* = \arg \min_{\delta\mathbf{x}} \|\mathbf{J} \delta\mathbf{x} + \mathbf{r}\|^2. \quad (106)$$

This is solved via least-squares using the pseudo-inverse of \mathbf{J} (for large problems, QR [10], [11] or Cholesky [12], [13] factorizations are required), yielding the optimal step $\delta\mathbf{x}^*$ used to update the state,

$$\delta\mathbf{x}^* = -(\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top \mathbf{r} \quad (107)$$

$$\mathcal{X} \leftarrow \mathcal{X} \oplus \delta\mathbf{x}^*. \quad (108)$$

The procedure is iterated until convergence.

We highlight here the use of the composite notation in (101), which allows block-wise definitions of the Jacobian (105) and the update (108). We also remark the use of the $SE(2)$ manifold in the motion and measurement models, as we did in the ESKF case in Section V-A.

C. Smoothing and mapping with self-calibration

We consider the same problem as above but with a motion sensor affected by an unknown calibration bias $\mathbf{c} = (c_v, c_\omega)^T$, so that the control is now $\tilde{\mathbf{u}} = (v\delta t + c_v, 0, \omega\delta t + c_\omega)^T + \mathbf{w}$. We define the bias correction function $c()$,

$$\mathbf{u} = c(\tilde{\mathbf{u}}, \mathbf{c}) \triangleq \begin{bmatrix} \tilde{u}_v - c_v \\ \tilde{u}_s \\ \tilde{u}_\omega - c_\omega \end{bmatrix} \in \mathbb{R}^3 \simeq \mathfrak{se}(2) . \quad (109)$$

The state composite is augmented with the unknowns \mathbf{c} ,

$$\begin{aligned} \mathcal{X} &= \langle \mathbf{c}, \mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \mathbf{b}_4, \mathbf{b}_5, \mathbf{b}_6 \rangle , \\ \mathbf{c} &\in \mathbb{R}^2, \quad \mathcal{X}_i \in SE(2), \quad \mathbf{b}_k \in \mathbb{R}^2 , \end{aligned}$$

and the motion residual becomes

$$\mathbf{r}_{ij}(\mathcal{X}) = \Omega_{ij}^{\top/2} (c(\tilde{\mathbf{u}}_{ij}, \mathbf{c}) - (\hat{\mathcal{X}}_j \oplus \hat{\mathcal{X}}_i)) .$$

The procedure is as in Section V-B above, and just the total Jacobian is modified with an extra column on the left,

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{\mathbf{c}}^{\mathbf{r}_{12}} & \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{12}} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{12}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_{\mathbf{c}}^{\mathbf{r}_{23}} & \mathbf{0} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{23}} & \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{23}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{14}} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_4}^{\mathbf{r}_{14}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{\mathcal{X}_1}^{\mathbf{r}_{15}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_5}^{\mathbf{r}_{15}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{25}} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_5}^{\mathbf{r}_{25}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathcal{X}_2}^{\mathbf{r}_{26}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_6}^{\mathbf{r}_{26}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathcal{X}_3}^{\mathbf{r}_{36}} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\mathbf{b}_6}^{\mathbf{r}_{36}} \end{bmatrix} ,$$

where $\mathbf{J}_{\mathbf{c}}^{\mathbf{r}_{ij}} = \Omega_{ij}^{\top/2} \mathbf{J}_{\mathbf{c}}^{c(\mathbf{u}_{ij}, \mathbf{c})}$, with $\mathbf{J}_{\mathbf{c}}^{c(\mathbf{u}_{ij}, \mathbf{c})}$ the 3×2 Jacobian of (109). The optimal solution is obtained with (107, 108). The resulting optimal state \mathcal{X} includes an optimal estimate of \mathbf{c} , that is, the self-calibration of the sensor bias.

APPENDIX A THE 2D ROTATION GROUPS S^1 AND $SO(2)$

The Lie group S^1 is the group of unit complex numbers under complex product. Its topology is the unit circle, or the unit 1-sphere, and therefore the name S^1 . The group, Lie algebra and vector elements have the form,

$$\mathbf{z} = \cos \theta + i \sin \theta, \quad \tau^\wedge = i\theta, \quad \tau = \theta . \quad (110)$$

Inversion and composition are achieved by conjugation $\mathbf{z}^{-1} = \mathbf{z}^*$, and product $\mathbf{z}_a \circ \mathbf{z}_b = \mathbf{z}_a \mathbf{z}_b$.

The group $SO(2)$ is the group of special orthogonal matrices in the plane, or rotation matrices, under matrix multiplication. Group, Lie algebra and vector elements have the form,

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}, \quad \tau^\wedge = [\theta]_\times \triangleq \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}, \quad \tau = \theta . \quad (111)$$

Inversion and composition are achieved by transposition $\mathbf{R}^{-1} = \mathbf{R}^\top$, and product $\mathbf{R}_a \circ \mathbf{R}_b = \mathbf{R}_a \mathbf{R}_b$.

Both groups rotate 2-vectors, and they have isomorphic tangent spaces. We thus study them together.

A. Exp and Log maps

Exp and Log maps may be defined for complex numbers of S^1 and rotation matrices of $SO(2)$. For S^1 we have,

$$\mathbf{z} = \text{Exp}(\theta) = \cos \theta + i \sin \theta \in \mathbb{C} \quad (112)$$

$$\theta = \text{Log}(\mathbf{z}) = \arctan(\text{Im}(\mathbf{z}), \text{Re}(\mathbf{z})) \in \mathbb{R} , \quad (113)$$

where (112) is the Euler formula, whereas for $SO(2)$,

$$\mathbf{R} = \text{Exp}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \in \mathbb{R}^{2 \times 2} \quad (114)$$

$$\theta = \text{Log}(\mathbf{R}) = \arctan(r_{21}, r_{11}) \in \mathbb{R} . \quad (115)$$

B. Inverse, composition

Since our defined derivatives map tangent vector spaces, and these spaces coincide for the planar rotation manifolds of S^1 and $SO(2)$, i.e., $\theta = \text{Log}(\mathbf{z}) = \text{Log}(\mathbf{R})$, it follows that the Jacobians are independent of the representation used (\mathbf{z} or \mathbf{R}). We thus consider generic 2D rotation elements, and note them with the sans-serif font R .

We have

$$R(\theta)^{-1} = R(-\theta) \quad (116)$$

$$Q \circ R = R \circ Q , \quad (117)$$

since planar rotations are commutative. It follows that,

$$\text{Exp}(\theta_1 + \theta_2) = \text{Exp}(\theta_1) \circ \text{Exp}(\theta_2) \quad (118)$$

$$\text{Log}(Q \circ R) = \text{Log}(Q) + \text{Log}(R) \quad (119)$$

$$Q \ominus R = \theta_Q - \theta_R . \quad (120)$$

C. Jacobian blocks

1) Adjoint and other trivial Jacobians: From (41), Section III-B and the properties above, the following scalar derivative blocks become trivial,

$$\mathbf{Ad}_R = 1 \in \mathbb{R} \quad (121)$$

$$\mathbf{J}_R^{R^{-1}} = -1 \in \mathbb{R} \quad (122)$$

$$\mathbf{J}_Q^{Q \circ R} = \mathbf{J}_R^{Q \circ R} = 1 \in \mathbb{R} \quad (123)$$

$$\mathbf{J}_r(\theta) = \mathbf{J}_l(\theta) = 1 \in \mathbb{R} \quad (124)$$

$$\mathbf{J}_R^{R \oplus \theta} = \mathbf{J}_\theta^{R \oplus \theta} = 1 \in \mathbb{R} \quad (125)$$

$$\mathbf{J}_Q^{Q \ominus R} = -\mathbf{J}_R^{Q \ominus R} = 1 \in \mathbb{R} \quad (126)$$

2) Rotation action: For the action $R \cdot v$ we have,

$$\begin{aligned} \mathbf{J}_R^{R \cdot v} &= \lim_{\theta \rightarrow 0} \frac{\mathbf{R} \text{Exp}(\theta)v - \mathbf{R}v}{\theta} \\ &= \lim_{\theta \rightarrow 0} \frac{\mathbf{R}(\mathbf{I} + [\theta]_\times)v - \mathbf{R}v}{\theta} \\ &= \lim_{\theta \rightarrow 0} \frac{\mathbf{R}[\theta]_\times v}{\theta} = \mathbf{R}[1]_\times v \in \mathbb{R}^{2 \times 1} \end{aligned} \quad (127)$$

and

$$\mathbf{J}_v^{R \cdot v} = \frac{\partial \mathbf{R}v}{\partial v} = \mathbf{R} \in \mathbb{R}^{2 \times 2} . \quad (128)$$

APPENDIX B THE 3D ROTATION GROUPS S^3 AND $SO(3)$

The Lie group S^3 is the group of unit quaternions under the quaternion multiplication. Its topology is the unit 3-sphere in \mathbb{R}^4 , and therefore its name S^3 . Quaternions may be represented by either of these equivalent forms,

$$\begin{aligned}\mathbf{q} &= w + ix + jy + kz = w + \mathbf{v} \in \mathbb{H} \\ &= [w \quad x \quad y \quad z]^\top = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} \in \mathbb{H},\end{aligned}\quad (129)$$

where $w, x, y, z \in \mathbb{R}$, and i, j, k are three unit imaginary numbers such that $i^2 = j^2 = k^2 = ijk = -1$. The scalar w is known as the scalar or real part, and $\mathbf{v} \in \mathbb{H}_p$ as the vector or imaginary part. We note \mathbb{H}_p the set of pure quaternions, *i.e.*, of null scalar part, with dimension 3. Inversion and composition are achieved by conjugation $\mathbf{q}^{-1} = \mathbf{q}^*$, where $\mathbf{q}^* \triangleq w - \mathbf{v}$ is the conjugate, and product $\mathbf{q}_a \circ \mathbf{q}_b = \mathbf{q}_a \mathbf{q}_b$.

The group $SO(3)$ is the group of special orthogonal matrices in 3D space, or rotation matrices, under matrix multiplication. Inversion and composition are achieved with transposition and product as in all groups $SO(n)$.

Both groups rotate 3-vectors. They have isomorphic tangent spaces whose elements are identifiable with rotation vectors in \mathbb{R}^3 , so we study them together. It is in this space \mathbb{R}^3 where we define the vectors of rotation rate $\boldsymbol{\omega} \triangleq \mathbf{u}\boldsymbol{\omega}$, angle-axis $\boldsymbol{\theta} \triangleq \mathbf{u}\boldsymbol{\theta}$, and all perturbations and uncertainties.

The quaternion manifold S^3 is a double cover of $SO(3)$, *i.e.*, \mathbf{q} and $-\mathbf{q}$ represent the same rotation \mathbf{R} . The first cover corresponds to quaternions with positive real part $w > 0$. These groups can be considered isomorphic up to the first cover. For an in-depth reference consult [6].

A. Exp and Log maps

The Exp and Log maps may be defined for quaternions of S^3 and rotation matrices of $SO(3)$. For quaternions $\mathbf{q} = (w, \mathbf{v}) \in \mathbb{H}$ we have (see Ex. 5),

$$\mathbf{q} = \text{Exp}(\theta \mathbf{u}) \triangleq \cos(\theta/2) + \mathbf{u} \sin(\theta/2) \in \mathbb{H} \quad (130)$$

$$\theta \mathbf{u} = \text{Log}(\mathbf{q}) \triangleq 2 \frac{\arctan(\|\mathbf{v}\|, w)}{\|\mathbf{v}\|} \in \mathbb{R}^3, \quad (131)$$

We can avoid eventual problems due to the double cover of \mathbf{q} by ensuring that its scalar part w is positive before doing the Log. If it is not, we can substitute \mathbf{q} by $-\mathbf{q}$ before the Log.

For rotation matrices we have (see Ex. 4),

$$\mathbf{R} = \text{Exp}(\theta \mathbf{u}) \triangleq \mathbf{I} + \sin \theta [\mathbf{u}]_\times + (1 - \cos \theta) [\mathbf{u}]_\times^2 \in \mathbb{R}^{3 \times 3} \quad (132)$$

$$\theta \mathbf{u} = \text{Log}(\mathbf{R}) \triangleq \frac{\theta(\mathbf{R} - \mathbf{R}^\top)^\vee}{2 \sin \theta} \in \mathbb{R}^3, \quad (133)$$

with $\theta = \cos^{-1} \left(\frac{\text{trace}(\mathbf{R}) - 1}{2} \right)$.

B. Rotation action

Given the expressions above for the quaternion and the rotation matrix, the rotation action of quaternions on 3-vectors is performed by the double product,

$$\mathbf{x}' = \mathbf{q} \mathbf{x} \mathbf{q}^* \quad (134)$$

while rotation matrices use a single matrix product,

$$\mathbf{x}' = \mathbf{R} \mathbf{x}. \quad (135)$$

Both correspond to a right-hand rotation of θ rad around the axis \mathbf{u} . Identifying in them \mathbf{x} and \mathbf{x}' yields the identity

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & w^2 - x^2 - y^2 + z^2 \end{bmatrix} \quad (136)$$

C. Elementary Jacobian blocks

Since our defined derivatives map tangent vector spaces, and these spaces coincide for the 3D rotation manifolds of S^3 and $SO(3)$, *i.e.*, $\boldsymbol{\theta} = \text{Log}(\mathbf{q}) = \text{Log}(\mathbf{R})$, it follows that the Jacobians are independent of the representation used (\mathbf{q} or \mathbf{R}). We thus consider generic 3D rotation elements, and note them with the sans-serif font \mathbf{R} .

1) *Adjoint:* We have from (31)

$$\mathbf{Ad}_{\mathbf{R}} \boldsymbol{\tau} = (\mathbf{R} [\boldsymbol{\tau}]_\times \mathbf{R}^\top)^\vee = ([(\mathbf{R} \boldsymbol{\tau})]_\times)^\vee = \mathbf{R} \boldsymbol{\tau}$$

therefore

$$\mathbf{Ad}_{\mathbf{R}} = \mathbf{R}, \quad (137)$$

which means, just to clarify it once again, that $\mathbf{Ad}_{\mathbf{q}} = \mathbf{R}(\mathbf{q})$, see (136), and $\mathbf{Ad}_{\mathbf{R}} = \mathbf{R}$.

2) *Inversion, composition:* We have from Section III-B,

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{R}^{-1}} = -\mathbf{Ad}_{\mathbf{R}} = -\mathbf{R} \quad (138)$$

$$\mathbf{J}_{\mathbf{Q}}^{\mathbf{Q}\mathbf{R}} = \mathbf{Ad}_{\mathbf{R}}^{-1} = \mathbf{R}^\top \quad (139)$$

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{Q}\mathbf{R}} = \mathbf{I}. \quad (140)$$

3) *Right and left Jacobians:* They admit the closed forms [9, pag. 40],

$$\mathbf{J}_r(\boldsymbol{\theta}) = \mathbf{I} - \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_\times + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_\times^2 \quad (141)$$

$$\mathbf{J}_r^{-1}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1}{2} [\boldsymbol{\theta}]_\times + \left(\frac{1}{\theta^2} - \frac{1 + \cos \theta}{2\theta \sin \theta} \right) [\boldsymbol{\theta}]_\times^2 \quad (142)$$

$$\mathbf{J}_l(\boldsymbol{\theta}) = \mathbf{I} + \frac{1 - \cos \theta}{\theta^2} [\boldsymbol{\theta}]_\times + \frac{\theta - \sin \theta}{\theta^3} [\boldsymbol{\theta}]_\times^2 \quad (143)$$

$$\mathbf{J}_l^{-1}(\boldsymbol{\theta}) = \mathbf{I} - \frac{1}{2} [\boldsymbol{\theta}]_\times + \left(\frac{1}{\theta^2} - \frac{1 + \cos \theta}{2\theta \sin \theta} \right) [\boldsymbol{\theta}]_\times^2 \quad (144)$$

where we can observe that

$$\mathbf{J}_l = \mathbf{J}_r^\top, \quad \mathbf{J}_l^{-1} = \mathbf{J}_r^{-\top}. \quad (145)$$

4) *Right- plus and minus:* We have for $\boldsymbol{\theta} = \mathbf{Q} \ominus \mathbf{R}$,

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{R} \oplus \boldsymbol{\theta}} = \mathbf{R}(\boldsymbol{\theta})^\top \quad \mathbf{J}_{\boldsymbol{\theta}}^{\mathbf{R} \oplus \boldsymbol{\theta}} = \mathbf{J}_r(\boldsymbol{\theta}) \quad (146)$$

$$\mathbf{J}_{\mathbf{Q}}^{\mathbf{Q} \ominus \mathbf{R}} = \mathbf{J}_r^{-1}(\boldsymbol{\theta}) \quad \mathbf{J}_{\mathbf{R}}^{\mathbf{Q} \ominus \mathbf{R}} = \mathbf{J}_l^{-1}(\boldsymbol{\theta}) \quad (147)$$

5) *Rotation action:* We have

$$\begin{aligned}\mathbf{J}_{\mathbf{R}}^{\mathbf{R} \cdot \mathbf{v}} &\triangleq \lim_{\theta \rightarrow 0} \frac{(\mathbf{R} \oplus \boldsymbol{\theta}) \mathbf{v} - \mathbf{R} \mathbf{v}}{\theta} = \\ \lim_{\theta \rightarrow 0} \frac{\mathbf{R} \text{Exp}(\boldsymbol{\theta}) \mathbf{v} - \mathbf{R} \mathbf{v}}{\theta} &= \lim_{\theta \rightarrow 0} \frac{\mathbf{R}(\mathbf{I} + [\boldsymbol{\theta}]_\times) \mathbf{v} - \mathbf{R} \mathbf{v}}{\theta} \\ &= \lim_{\theta \rightarrow 0} \frac{\mathbf{R} [\boldsymbol{\theta}]_\times \mathbf{v}}{\theta} = \lim_{\theta \rightarrow 0} \frac{-\mathbf{R} [\mathbf{v}]_\times \boldsymbol{\theta}}{\theta} = -\mathbf{R} [\mathbf{v}]_\times\end{aligned} \quad (148)$$

where we used the properties $\text{Exp}(\theta) \approx \mathbf{I} + [\theta]_{\times}$ and $[\mathbf{a}]_{\times} \mathbf{b} = -[\mathbf{b}]_{\times} \mathbf{a}$. The second Jacobian yields,

$$\mathbf{J}_{\mathbf{v}}^{\mathbf{R} \cdot \mathbf{v}} \triangleq \lim_{\partial \mathbf{v} \rightarrow 0} \frac{\mathbf{R}(\mathbf{v} + \partial \mathbf{v}) - \mathbf{R}\mathbf{v}}{\partial \mathbf{v}} = \mathbf{R} . \quad (149)$$

APPENDIX C THE 2D RIGID MOTION GROUP $SE(2)$

We write elements of the rigid motion group $SE(2)$ as

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(2) ,$$

with $\mathbf{R} \in SO(2)$ a rotation and $\mathbf{t} \in \mathbb{R}^2$ a translation. The Lie algebra and vector tangents are formed by elements of the type

$$\boldsymbol{\tau}^{\wedge} = \begin{bmatrix} [\theta]_{\times} & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathfrak{se}(2) , \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix} \in \mathbb{R}^3 .$$

A. Inverse, composition

Inversion and composition are performed respectively with matrix inversion and product,

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad (150)$$

$$\mathbf{H}_a \mathbf{H}_b = \begin{bmatrix} \mathbf{R}_a \mathbf{R}_b & \mathbf{t}_a + \mathbf{R}_a \mathbf{t}_b \\ \mathbf{0} & 1 \end{bmatrix} . \quad (151)$$

B. Exp and Log maps

Exp and Log are implemented via exponential maps directly from the scalar tangent space $\mathbb{R}^3 \simeq \mathfrak{se}(2) = \mathcal{T}SE(2)$ — see [5] for the derivation,

$$\mathbf{H} = \text{Exp}(\boldsymbol{\tau}) \triangleq \begin{bmatrix} \text{Exp}(\theta) & \mathbf{V}(\theta) \boldsymbol{\rho} \\ \mathbf{0} & 1 \end{bmatrix} \quad (152)$$

$$\boldsymbol{\tau} = \text{Log}(\mathbf{H}) \triangleq \begin{bmatrix} \mathbf{V}^{-1}(\theta) \mathbf{p} \\ \text{Log}(\mathbf{R}) \end{bmatrix} . \quad (153)$$

with

$$\mathbf{V}(\theta) = \frac{\sin \theta}{\theta} \mathbf{I} + \frac{1 - \cos \theta}{\theta} [1]_{\times} . \quad (154)$$

C. Jacobian blocks

1) *Adjoint*: The adjoint is easily found from (31) using the fact that planar rotations commute,

$$\mathbf{Ad}_{\mathbf{H}} \boldsymbol{\tau} = (\mathbf{H} \boldsymbol{\tau}^{\wedge} \mathbf{H}^{-1})^{\vee} = \begin{bmatrix} \mathbf{R} \boldsymbol{\rho} - [\theta]_{\times} \mathbf{t} \\ \theta \end{bmatrix} = \mathbf{Ad}_{\mathbf{H}} \begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix} ,$$

leading to

$$\mathbf{Ad}_{\mathbf{H}} = \begin{bmatrix} \mathbf{R} & -[1]_{\times} \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} . \quad (155)$$

All Jacobians follow from (41).

2) *Inversion, composition*: We have from Section III-B,

$$\mathbf{J}_{\mathbf{H}}^{\mathbf{H}^{-1}} = -\mathbf{Ad}_{\mathbf{H}} = \begin{bmatrix} -\mathbf{R} & [1]_{\times} \mathbf{t} \\ \mathbf{0} & -1 \end{bmatrix} \quad (156)$$

$$\mathbf{J}_{\mathbf{H}_a \mathbf{H}_b}^{\mathbf{H}_a \mathbf{H}_b} = \mathbf{Ad}_{\mathbf{H}_b}^{-1} = \begin{bmatrix} \mathbf{R}_b^T & \mathbf{R}_b^T [1]_{\times} \mathbf{t}_b \\ \mathbf{0} & 1 \end{bmatrix} \quad (157)$$

$$\mathbf{J}_{\mathbf{H}_b}^{\mathbf{H}_a \mathbf{H}_b} = \mathbf{I} . \quad (158)$$

3) *Right and left Jacobians*: We have from [9, pag. 36],

$$\mathbf{J}_r = \begin{bmatrix} \sin \theta / \theta & (1 - \cos \theta) / \theta & (\theta v_1 - v_2 + v_2 \cos \theta - v_1 \sin \theta) / \theta^2 \\ (1 - \cos \theta) / \theta & \sin \theta / \theta & (v_1 + \theta v_2 - v_1 \cos \theta - v_2 \sin \theta) / \theta^2 \\ 0 & 0 & 1 \end{bmatrix} \quad (159)$$

$$\mathbf{J}_l = \begin{bmatrix} \sin \theta / \theta & (\cos \theta - 1) / \theta & (\theta v_1 + v_2 - v_2 \cos \theta - v_1 \sin \theta) / \theta^2 \\ (1 - \cos \theta) / \theta & \sin \theta / \theta & (-v_1 + \theta v_2 + v_1 \cos \theta - v_2 \sin \theta) / \theta^2 \\ 0 & 0 & 1 \end{bmatrix} . \quad (160)$$

4) *Rigid motion action*: We have the action on points \mathbf{p} ,

$$\mathbf{H} \cdot \mathbf{p} \triangleq \mathbf{t} + \mathbf{R}\mathbf{p} , \quad (161)$$

therefore from (152),

$$\mathbf{J}_{\mathbf{H}}^{\mathbf{H} \cdot \mathbf{p}} = \lim_{\tau \rightarrow 0} \frac{\mathbf{H} \text{Exp}(\tau) \cdot \mathbf{p} - \mathbf{H} \cdot \mathbf{p}}{\tau} = [\mathbf{R} \quad \mathbf{R} [1]_{\times} \mathbf{p}] \quad (162)$$

$$\mathbf{J}_{\mathbf{p}}^{\mathbf{H} \cdot \mathbf{p}} = \mathbf{R} . \quad (163)$$

APPENDIX D THE 3D RIGID MOTION GROUP $SE(3)$

We write elements of the 3D rigid motion group $SE(3)$ as

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) ,$$

with $\mathbf{R} \in SO(3)$ a rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ a translation vector. The Lie algebra and vector tangents are formed by elements of the type

$$\boldsymbol{\tau}^{\wedge} = \begin{bmatrix} [\theta]_{\times} & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathfrak{se}(3) , \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix} \in \mathbb{R}^6 .$$

A. Inverse, composition

Inversion and composition are performed respectively with matrix inversion and product,

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad (164)$$

$$\mathbf{H}_a \mathbf{H}_b = \begin{bmatrix} \mathbf{R}_a \mathbf{R}_b & \mathbf{t}_a + \mathbf{R}_a \mathbf{t}_b \\ \mathbf{0} & 1 \end{bmatrix} . \quad (165)$$

B. Exp and Log maps

Exp and Log are implemented via exponential maps directly from the vector tangent space $\mathbb{R}^6 \simeq \mathfrak{se}(3) = \mathcal{T}SE(3)$ — see [5] for the derivation,

$$\mathbf{H} = \text{Exp}(\boldsymbol{\tau}) \triangleq \begin{bmatrix} \text{Exp}(\theta) & \mathbf{V}(\theta) \boldsymbol{\rho} \\ \mathbf{0} & 1 \end{bmatrix} \quad (166)$$

$$\boldsymbol{\tau} = \text{Log}(\mathbf{H}) \triangleq \begin{bmatrix} \mathbf{V}^{-1}(\theta) \mathbf{p} \\ \text{Log}(\mathbf{R}) \end{bmatrix} . \quad (167)$$

with

$$\mathbf{V}(\theta) = \mathbf{I} + \frac{1 - \cos \theta}{\theta} [\mathbf{u}]_{\times} + \frac{\theta - \sin \theta}{\theta} [\mathbf{u}]_{\times}^2 . \quad (168)$$

C. Jacobian blocks

1) *Adjoint*: We have,

$$\mathbf{Ad}_{\mathbf{H}} \boldsymbol{\tau} = (\mathbf{H} \boldsymbol{\tau}^{\wedge} \mathbf{H}^{-1})^{\vee} = \begin{bmatrix} \mathbf{R} \boldsymbol{\rho} + [\mathbf{t}]_{\times} \mathbf{R} \theta \\ \mathbf{R} \theta \end{bmatrix} = \mathbf{Ad}_{\mathbf{H}} \begin{bmatrix} \boldsymbol{\rho} \\ \theta \end{bmatrix}$$

therefore,

$$\mathbf{Ad}_{\mathbf{H}} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_{\times} \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6 \times 6} . \quad (169)$$

2) *Inversion, composition:* We have from Section III-B,

$$\mathbf{J}_H^{H^{-1}} = - \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times \mathbf{R} \\ 0 & \mathbf{R} \end{bmatrix} \quad (170)$$

$$\mathbf{J}_{H_a H_b}^{H_a H_b} = \begin{bmatrix} \mathbf{R}_b & [\mathbf{t}_b]_\times \mathbf{R}_b \\ 0 & \mathbf{R}_b \end{bmatrix}^{-1} \quad (171)$$

$$\mathbf{J}_{H_b}^{H_a H_b} = \mathbf{I}_6 . \quad (172)$$

3) *Right and left Jacobians:* A closed form of the left Jacobian is given by Barfoot in [8],

$$\mathbf{J}_l(\rho, \theta) = \begin{bmatrix} \mathbf{J}_l^{-1}(\theta) & -\mathbf{J}_l^{-1}(\theta)\mathbf{Q}(\rho, \theta)\mathbf{J}_l^{-1}(\theta) \\ \mathbf{0} & \mathbf{J}_l^{-1}(\theta) \end{bmatrix} \quad (173)$$

where $\mathbf{J}_l^{-1}(\theta)$ is the inverse left Jacobian of $SO(3)$, see (144), and

$$\begin{aligned} \mathbf{Q}(\rho, \theta) = & \frac{1}{2}\rho_\times + \frac{\theta - \sin\theta}{\theta^3}(\theta_\times\rho_\times + \rho_\times\theta_\times + \theta_\times\rho_\times\theta_\times) \\ & - \frac{1 - \frac{\theta^2}{2} - \cos\theta}{\theta^4}(\theta_\times^2\rho_\times + \rho_\times\theta_\times^2 - 3\theta_\times\rho_\times\theta_\times) \\ & - \frac{1}{2} \left(\frac{1 - \frac{\theta^2}{2} - \cos\theta}{\theta^4} - 3\frac{\theta - \sin\theta - \frac{\theta^3}{6}}{\theta^5} \right) \\ & \times (\theta_\times\rho_\times\theta_\times^2 + \theta_\times^2\rho_\times\theta_\times) . \end{aligned} \quad (174)$$

The right Jacobian is obtained using (76), that is, changing the signs of all ρ , θ and θ above.

4) *Rigid motion action:* We have the action on points \mathbf{p} ,

$$\mathbf{H} \cdot \mathbf{p} \triangleq \mathbf{t} + \mathbf{R}\mathbf{p} , \quad (175)$$

therefore from (166),

$$\mathbf{J}_H^{\mathbf{H} \cdot \mathbf{p}} = \lim_{\tau \rightarrow 0} \frac{\mathbf{H} \text{Exp}(\tau) \cdot \mathbf{p} - \mathbf{H} \cdot \mathbf{p}}{\tau} = [\mathbf{R} \quad -\mathbf{R}[\mathbf{p}]_\times] \quad (176)$$

$$\mathbf{J}_{\mathbf{p}}^{\mathbf{H} \cdot \mathbf{p}} = \mathbf{R} . \quad (177)$$

APPENDIX E

THE TRANSLATION GROUPS $(\mathbb{R}^n, +)$ AND $T(n)$

The group $(\mathbb{R}^n, +)$ is the group of vectors under addition, and can be regarded as a translation group. We deem it *trivial* in the sense that the group elements, the Lie algebra, and the tangent vectors are all the same, so $\mathbf{t} = \mathbf{t}^\wedge = \text{Exp}(\mathbf{t})$. Its equivalent matrix group (under multiplication) is the translation group $T(n)$, whose group, Lie algebra and tangent vector elements are,

$$\mathbf{T} \triangleq \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in T(n), \quad \mathbf{t}^\wedge \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{t} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathfrak{t}(n), \quad \mathbf{t} \in \mathbb{R}^n .$$

Equivalence is easily verified by observing that $\mathbf{T}(\mathbf{0}) = \mathbf{I}$, $\mathbf{T}(-\mathbf{t}) = \mathbf{T}(\mathbf{t})^{-1}$, and that the commutative composition

$$\mathbf{T}_1 \mathbf{T}_2 = \begin{bmatrix} \mathbf{I} & \mathbf{t}_1 + \mathbf{t}_2 \\ \mathbf{0} & 1 \end{bmatrix} ,$$

effectively adds the vectors \mathbf{t}_1 and \mathbf{t}_2 together. Since $T(n)$ is a subgroup of $SE(n)$ with $\mathbf{R} = \mathbf{I}$, we can easily determine its exponential map from generalizations of (152, 166),

$$\text{Exp} : \mathbb{R}^n \rightarrow T(n) \quad \mathbf{T} = \text{Exp}(\mathbf{t}) = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} . \quad (178)$$

This serves as immediate proof for the equivalent exponential of the $(\mathbb{R}^n, +)$ group, which is the identity,

$$\text{Exp} : \mathbb{R}^n \rightarrow \mathbb{R}^n \quad \mathbf{t} = \text{Exp}(\mathbf{t}) . \quad (179)$$

This derives in trivial, commutative plus and minus operators in \mathbb{R}^n ,

$$\mathbf{t}_1 \oplus \mathbf{t}_2 = \mathbf{t}_1 + \mathbf{t}_2 \quad (180)$$

$$\mathbf{t}_2 \ominus \mathbf{t}_1 = \mathbf{t}_2 - \mathbf{t}_1 . \quad (181)$$

A. Jacobian blocks

We express translations as S and T , indistinctly for $T(n)$ and \mathbb{R}^n . The groups are commutative, and so the Jacobians are trivial (compare them with those of $SO(2)$ in Section A-C1).

$$\mathbf{A}\mathbf{d}_T = \mathbf{I} \quad \in \mathbb{R}^{n \times n} \quad (182)$$

$$\mathbf{J}_T^{T^{-1}} = -\mathbf{I} \quad \in \mathbb{R}^{n \times n} \quad (183)$$

$$\mathbf{J}_T^T S = \mathbf{J}_S^T = \mathbf{I} \quad \in \mathbb{R}^{n \times n} \quad (184)$$

$$\mathbf{J}_r = \mathbf{J}_l = \mathbf{I} \quad \in \mathbb{R}^{n \times n} \quad (185)$$

$$\mathbf{J}_T^{T \oplus v} = \mathbf{J}_v^{T \oplus v} = \mathbf{I} \quad \in \mathbb{R}^{n \times n} \quad (186)$$

$$\mathbf{J}_S^{S \ominus T} = -\mathbf{J}_T^{S \ominus T} = \mathbf{I} \quad \in \mathbb{R}^{n \times n} \quad (187)$$

REFERENCES

- [1] R. Howe, "Very basic Lie theory," Yale University, New Haven, USA, Tech. Rep.
- [2] J. Stillwell, *Naive Lie Theory*. Springer-Verlag New York, 2008.
- [3] C. Forster, L. Carbone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.
- [4] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.
- [5] E. Eade, "Lie groups for 2d and 3d transformations," Tech. Rep.
- [6] J. Sola, "Quaternion kinematics for the error-state Kalman filter," *CoRR*, vol. abs/1711.02508, 2017. [Online]. Available: <http://arxiv.org/abs/1711.02508>
- [7] G. Gallego and A. Yezzi, "A compact formula for the derivative of a 3-D rotation in exponential coordinates," Tech. Rep., 2013.
- [8] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, June 2014.
- [9] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups*, ser. Applied and Numerical Harmonic Analysis. Basel: Birkhäuser, 2012, vol. 2: Analytic Methods and Modern Applications.
- [10] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," vol. 25, no. 12, pp. 1181–1203, 2006.
- [11] M. Kaess, H. Johansson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 3281–3288.
- [12] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 3607–3613.
- [13] J. P. V. Ila and J. Andrade-Cetto, "Amortized constant time state estimation in pose slam and hierarchical slam using a mixed kalman-information filter," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 310–318, 2011.

3

Multirotor Equations of Motion

Author: Tim McLain, RWB

3.1 Equations of Motion

Let \mathcal{F}_i denote an earth-fixed inertial reference frame with unit vectors $(\mathbf{i}_i, \mathbf{j}_i, \mathbf{k}_i)$ aligned with the north, east, and down directions. Let \mathcal{F}_b denote a reference frame that is fixed with respect to the multirotor body having unit vectors $(\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b)$ aligned with the forward, right, and down directions in the body frame. The orientation of the multirotor with respect to the inertial reference frame can be expressed by the rotation matrix $R_b^i \in SO(3)$ which can be expressed as¹

$$R_b^i = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix}, \quad (3.1)$$

where we have used the notation $c_x \triangleq \cos x$ and $s_x \triangleq \sin x$. The angles (ϕ, θ, ψ) are the roll, pitch, and yaw Euler angles. The rotation matrix in equation (3.1) corresponds to the yaw-pitch-roll (3-2-1) rotation sequence.

We can represent the rigid-body dynamics of the multirotor with the equations of motion

$$\begin{aligned} \dot{\mathbf{p}}_{b/i}^i &= \mathbf{v}_{b/i}^i \\ \dot{\mathbf{v}}_{b/i}^i &= g\mathbf{k}_i^i + \frac{1}{m} R_b^i \mathbf{F}^b \\ \dot{R}_b^i &= R_b^i \left[\boldsymbol{\omega}_{b/i}^b \right]_\times \\ J \dot{\boldsymbol{\omega}}_{b/i}^b &= -\boldsymbol{\omega}_{b/i}^b \times (\mathbf{J} \boldsymbol{\omega}_{b/i}^b) + \boldsymbol{\tau}^b. \end{aligned}$$

In these equations, m denotes the mass of the multirotor and \mathbf{J} is the inertia matrix of the multirotor with respect to the body-frame axes. The vectors \mathbf{F}^b and $\boldsymbol{\tau}^b$ define the aerodynamic forces and moments experienced by the multirotor and are expressed in the body frame. The

¹ Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012

vectors $v_{b/i}^i$, $\omega_{b/i}^i$, and $p_{b/i}^i$ are the velocity, angular velocity, and position of the multirotor with respect to the inertial reference frame as expressed in the inertial frame, while the variable g denotes the gravitational constant. The notation $[\boldsymbol{\omega}]_\times$ represents the skew-symmetric matrix operator formed from the vector $\boldsymbol{\omega}$ so that $[\boldsymbol{\omega}]_\times \mathbf{v} = \boldsymbol{\omega} \times \mathbf{v}$ for the vector cross product \times and any vector $\mathbf{v} \in \mathbb{R}^3$.

3.1.1 Rotor Forces and Torques

The aerodynamic forces and torques that enable the agile flight of a multirotor come from the dc-motor/propeller modules that we call the rotors. In hover, the rotors of a multirotor each provide a constant and equal thrust to overcome the effects of gravity. The rotors of a multirotor are usually configured in counter-rotating pairs, one rotating clockwise and the other counter-clockwise, so that the motor torques acting on the multirotor cancel out when in a non-rotating hover state. As the rotors rotate, the drag on the propellers produce a torque on the body of the aircraft opposite to the direction of rotation. Figure 3.1 depicts an octocopter with equally spaced counter-rotating rotors as an example of how a multirotor can be configured.

To accelerate the multirotor up or down, the speed of the rotors can be uniformly increased or decreased away from their nominal speed. To pitch up or down about the \mathbf{j}_b axis, the speed differential between the fore and aft rotors can be sped up or slowed down. Similarly, to roll the aircraft right or left about the \mathbf{i}_b axis the speed differential between the left and right rotors can be varied. To yaw the aircraft about the \mathbf{k}_b axis, on the other hand, the speed differential between the clockwise-rotating props and counter-clockwise-rotating props is varied.

In modeling the aerodynamic forces and torques acting on the multirotor, we will consider thrust force F_{thrust} , the drag force F_{drag} , and the torque produced by the rotors. In this section, we will develop the relationship between the thrust force F_{thrust} , the roll, pitch, and yaw torques (τ_x, τ_y, τ_z) and the individual rotor angular speeds. For an N -rotor aircraft, these angular speeds will be given by $(\omega_1, \omega_2, \dots, \omega_N)$ and specified in radians per second. From propeller theory, the thrust and torque produced by a single rotor in hover can be modeled as

$$T_i = C_T \frac{\rho D^4}{4\pi^2} \omega_i^2 \quad (3.2)$$

$$Q_i = C_Q \frac{\rho D^5}{4\pi^2} \omega_i^2, \quad (3.3)$$

where ρ is the density of air, D is the propeller diameter, and C_T and C_Q are non-dimensional aerodynamic coefficients specific to the propeller.

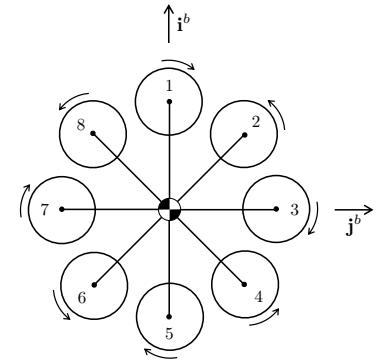


Figure 3.1: Octocopter example of a multirotor configuration.

The total rotor thrust for the multirotor is given by

$$F_{\text{thrust}} = \sum_{i=1}^N T_i, \quad (3.4)$$

where F_i is the thrust force produced by rotor i .

To define the torques produced by the rotors, we need to define the configuration of each rotor relative to the center of mass of the aircraft and its direction of rotation. Figure 3.2 highlights a single rotor i for an arbitrary multirotor configuration. The variable φ_i is the angle between the forward direction of the multirotor, defined by the unit vector \mathbf{i}_b , and the radial direction of the i^{th} rotor. The variable ℓ_i represents the radial distance from the center of mass to the i^{th} rotor. The variable d_i indicates the direction of rotation for each rotor as

$$d_i = \begin{cases} -1 & \text{for CW rotation} \\ +1 & \text{for CCW rotation.} \end{cases} \quad (3.5)$$

From the configuration geometry of Figure 3.2, the total roll torque about the \mathbf{i}_b axis is given by

$$\tau_x = - \sum_{i=1}^N (\ell_i \sin \varphi_i) T_i. \quad (3.6)$$

Similarly, the total pitch torque about the \mathbf{j}_b axis can be calculated as

$$\tau_y = \sum_{i=1}^N (\ell_i \cos \varphi_i) T_i. \quad (3.7)$$

Finally, the total yaw torque about the \mathbf{k}_b axis, which depends on the direction of rotation of each rotor, is given by

$$\tau_z = \sum_{i=1}^N d_i Q_i. \quad (3.8)$$

RWB: Add mixing matrix here. Could fill in the gaps to go to pwm command if ω_i is proportional to pwm.

As with any lifting surface, induced drag forces are generated from the lift forces created by the rotating propellers. The direction of the drag force on a propeller blade at any instant is in the direction opposite of the velocity of the blade tip. Since the blade tip is moving much faster than the airspeed of the multirotor motion. When the blade is advancing into the oncoming apparent wind generated by the combination of multirotor motion and the wind, the induced drag is higher than when the blade is in the retreating portion of its circular motion and moving with the apparent wind. The net effect is that each of the rotors on the aircraft creates an induced drag force that is

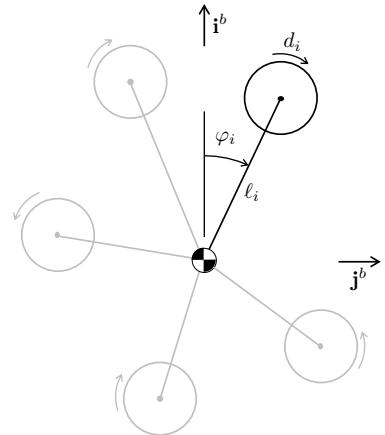
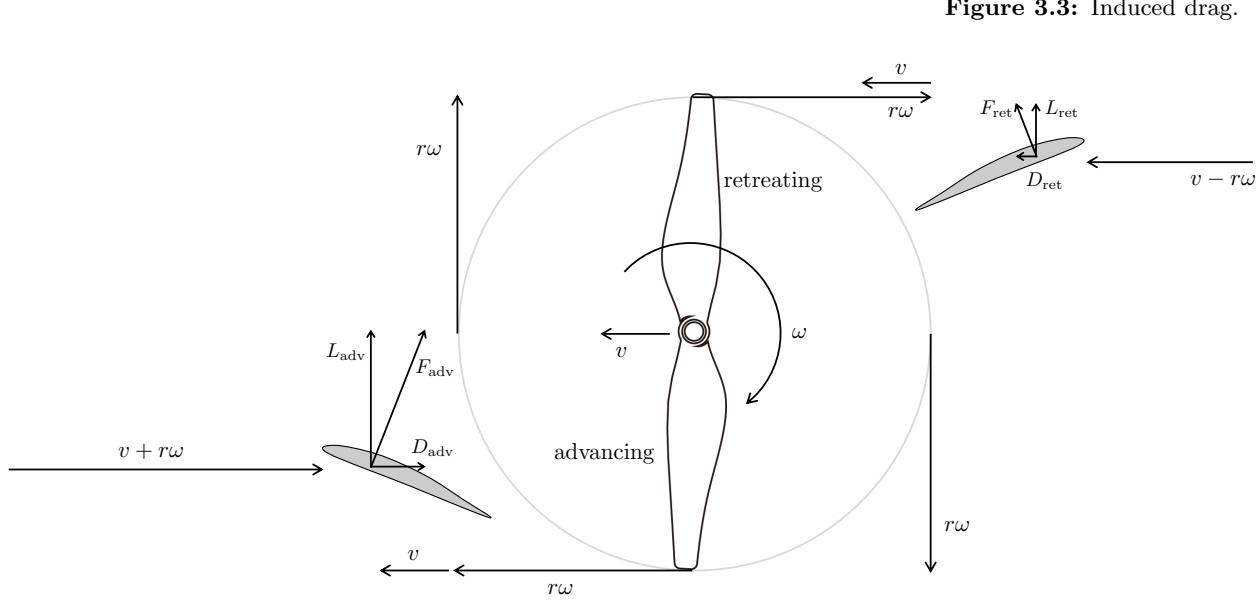


Figure 3.2: Definition of rotor configuration variables.

in the plane of the rotor and proportional to the projection of the lift force onto the plane of the rotor in magnitude and directed opposite to the airspeed vector. The drag force is also proportional to the com-



ponent of the airspeed in the rotor plane.² Accordingly, the drag force can be represented as

$$\begin{aligned}\mathbf{F}_{\text{drag}}^b &\approx -F_{\text{thrust}} C_d \text{diag}(1, 1, 0) \mathbf{v}_{b/i}^b \\ &= \begin{pmatrix} F_{\text{thrust}} C_d & 0 & 0 \\ 0 & F_{\text{thrust}} C_d & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{v}_{b/i}^b.\end{aligned}$$

Since the thrust force is approximately equal to mg , the weight of the aircraft, we can define

$$D = \begin{pmatrix} gC_d & 0 & 0 \\ 0 & gC_d & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and write induced drag in the body frame as

$$\mathbf{F}_{\text{drag}}^b \approx -mDR_b^{i\top} \mathbf{v}_{b/i}^i.$$

In the following we write the thrust force as

$$F_{\text{thrust}} = mT,$$

where T is the throttle setting. Therefore, the total aerodynamic force in the body frame is approximated as

$$\mathbf{F}^b \approx -mTk_b^b - mDR_b^{i\top} \mathbf{v}_{b/i}^i. \quad (3.9)$$

Figure 3.3: Induced drag.

² R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of a quadrotor. *IEEE Robotics & Automation Magazine*, pages 20–32, 2012a

Note that the induced drag force is proportional to velocity v and not to velocity squared v^2 . This is because the induced drag is proportional to $(v + r\omega)^2 - (v - r\omega)^2 = 4(r\omega)v$.

In the inertial frame, the aerodynamic force is

$$\mathbf{F}^i \approx -mTR_b^i \mathbf{e}_3 - mR_b^i D R_b^{i\top} \mathbf{v}_{b/i}^i. \quad (3.10)$$

Recall that $\mathbf{k}_b^b = \mathbf{e}_3 = (0, 0, 1)^\top$

3.2 Multirotor Mass Properties

The mass properties of multicopters of particular interest to us are its mass m , center of mass location, and its inertia matrix \mathbf{J} . The mass of a multicopter is distributed over the aircraft and the inertia matrix models how the mass is distributed and its effect on the rotational dynamics of the aircraft. Because the distribution of the mass of the aircraft is fixed in the body frame, \mathbf{J} is constant with respect to the body axes ($\mathbf{i}_b, \mathbf{j}_b, \mathbf{k}_b$) and can be calculated as

$$\begin{aligned} \mathbf{J} &= \begin{pmatrix} \int(y^2 + z^2) dm & -\int xy dm & -\int xz dm \\ -\int xy dm & \int(x^2 + z^2) dm & -\int yz dm \\ -\int xz dm & -\int yz dm & \int(x^2 + y^2) dm \end{pmatrix} \\ &\triangleq \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix}. \end{aligned}$$

The diagonal terms of \mathbf{J} are called the moments of inertia and the off-diagonal terms are called the products of inertia. Multicopter aircraft tend to be symmetric in their shape and mass distribution and this results in significant simplification in the inertia matrix. In particular, if the aircraft is symmetric with respect to the \mathbf{i}_b - \mathbf{j} , \mathbf{i}_b - \mathbf{k} , and \mathbf{j}_b - \mathbf{k} planes, then the products of inertia, J_{xy} , J_{xz} , and $J_{yz} = 0$, are zero resulting in

$$\mathbf{J} = \begin{pmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{pmatrix}.$$

Importantly, under these symmetry conditions, \mathbf{J} is easily inverted with its inverse expressed as

$$\mathbf{J}^{-1} = \begin{pmatrix} \frac{1}{J_x} & 0 & 0 \\ 0 & \frac{1}{J_y} & 0 \\ 0 & 0 & \frac{1}{J_z} \end{pmatrix}.$$

If a high-quality CAD model is available, the moments and products of inertia and the center of mass location can commonly be calculated numerically using the CAD software. Alternatively, moments of inertia and the center of mass location can be measured experimentally using equipment such as a bifilar pendulum. For simple geometries, the moments of inertia and center of mass can be calculated from moments of inertia of basic shapes and the parallel axis theorem. For example, the moments of inertia for the quadrotor shown

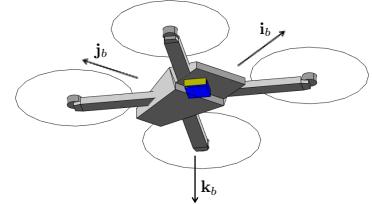


Figure 3.4: Quadrotor with body-frame axes that coincide at the center of mass.

in Figure 3.4 can be calculated by assuming that the quadrotor body is approximated by a cuboid with a specific length, width, mass, and uniform density. The rotor motors, arms, and propellers can be approximated as point masses located at the motor positions.

Summarizing, the equations of motion for the multirotor are given by

$$\dot{\mathbf{p}}_{b/i}^i = \mathbf{v}_{b/i}^i \quad (3.11)$$

$$\dot{\mathbf{v}}_{b/i}^i = g\mathbf{e}_3 - TR_b^i \mathbf{e}_3 - R_b^i D R_b^{i\top} \mathbf{v}_{b/i}^i \quad (3.12)$$

$$\dot{R}_b^i = R_b^i \left[\boldsymbol{\omega}_{b/i}^b \right]_\times \quad (3.13)$$

$$J\dot{\boldsymbol{\omega}}_{b/i}^b = -\boldsymbol{\omega}_{b/i}^b \times (J\boldsymbol{\omega}_{b/i}^b) + \boldsymbol{\tau}^b. \quad (3.14)$$

3.3 Multirotor Aerial Vehicles, Mahony, Kumar, Corke, IEEE RA Mag, Sept, 2020.

The model in this paper is nicely done.



©STOCK PHOTO.COM/© ANDREJS ZAVADSKIS

By Robert Mahony,
Vijay Kumar,
and Peter Corke

T

his article provides a tutorial introduction to modeling, estimation, and control for multirotor aerial vehicles that includes the common four-rotor or quadrotor case.

Aerial robotics is a fast-growing field of robotics and multirotor aircraft, such as the quadrotor (Figure 1), are rapidly growing in popularity. In fact, quadrotor aerial robotic vehicles have become a standard platform for robotics research worldwide. They already have sufficient payload and flight endurance to support a number of indoor and outdoor applications, and the improvements of battery and other technology is rapidly increasing the scope for commercial opportunities. They are highly maneuverable and enable safe and low-cost experimentation in mapping, navigation, and control strategies for robots that move in three-dimensional (3-D) space. This ability to move in 3-D space brings new research challenges compared with the wheeled mobile robots that have driven mobile robotics research over the last decade. Small quadrotors have been demonstrated for exploring and mapping 3-D environments; transporting, manipulating, and assembling objects; and acrobatic tricks such as juggling, balancing, and flips. Additional rotors can be added, leading to generalized N -rotor vehicles, to improve payload and reliability.

Multirotor Aerial Vehicles

*Modeling, Estimation,
and Control of Quadrotor*

Digital Object Identifier 10.1109/MRA.2012.2206474
Date of publication: 27 August 2012

This tutorial describes the fundamentals of the dynamics, estimation, and control for this class of vehicle, with a bias toward electrically powered micro (less than 1 kg)-scale vehicles. The word *helicopter* is derived from the Greek words for spiral (screw) and wing. From a linguistic perspective, since the prefix quad is Latin, the term quadrotor is more correct than quadcopter and more common than tetracopter; hence, we use the term *quadrotor* throughout.

Modeling of Multirotor Vehicles

The most common multirotor aerial platform, the quadrotor vehicle, is a very simple machine. It consists of four individual rotors attached to a rigid cross airframe, as shown in Figure 1. Control of a quadrotor is achieved by differential control of the thrust generated by each rotor. Pitch, roll, and heave (total thrust) control is straightforward to conceptualize. As shown in Figure 2, rotor i rotates anticlockwise (positive about the z axis) if i is even and clockwise if i is odd. Yaw control is obtained by adjusting the average speed of the clockwise and anticlockwise rotating rotors. The system is underactuated, and the remaining degrees of freedom (DoF) corresponding to the translational velocity in the x - y plane must be controlled through the system dynamics.

Rigid-Body Dynamics of the Airframe

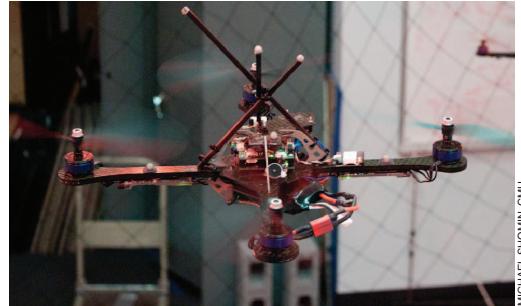
Let $\{\vec{x}, \vec{y}, \vec{z}\}$ be the three coordinate axis unit vectors without a frame of reference. Let $\{A\}$ denote a right-hand inertial frame with unit vectors along the axes denoted by $\{\vec{a}_1, \vec{a}_2, \vec{a}_3\}$ expressed in $\{A\}$. One has algebraically that $\vec{a}_1 = \vec{x}$, $\vec{a}_2 = \vec{y}$, $\vec{a}_3 = \vec{z}$ in $\{A\}$. The vector $r = (x, y, z) \in \{A\}$ denotes the position of the center of mass of the vehicle. Let $\{B\}$ be a (right-hand) body fixed frame for the airframe with unit vectors $\{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$, where these vectors are the axes of frame $\{B\}$ with respect to frame $\{A\}$. The orientation of the rigid body is given by a rotation matrix ${}^A R_B = R = [\vec{b}_1, \vec{b}_2, \vec{b}_3] \in \text{SO}(3)$ in the special orthogonal group. One has $\vec{b}_1 = R\vec{x}$, $\vec{b}_2 = R\vec{y}$, $\vec{b}_3 = R\vec{z}$ by construction.

We will use Z-X-Y Euler angles to model this rotation, as shown in Figure 3. To get from $\{A\}$ to $\{B\}$, we first rotate about \vec{a}_3 by the yaw angle, ψ , and we will call this intermediary frame $\{E\}$ with a basis $\{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$ where \vec{e}_i is expressed with respect to frame $\{A\}$. This is followed by a rotation about the x axis in the rotated frame through the roll angle, ϕ , followed by a third pitch rotation about the new y axis through the pitch angle θ that results in the body-fixed triad $\{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$

$$R = \begin{pmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{pmatrix},$$

where c and s are shorthand forms for cosine and sine, respectively.

Let $v \in \{A\}$ denote the linear velocity of $\{B\}$ with respect to $\{A\}$ expressed in $\{A\}$. Let $\Omega \in \{B\}$ denote the



MICHAEL SHOMIN, CNAU

Figure 1. A quadrotor made by Ascending Technologies with VICON markers for state estimation.

angular velocity of $\{B\}$ with respect to $\{A\}$; this time expressed in $\{B\}$. Let m denote the mass of the rigid object, and $I \in \mathbb{R}^{3 \times 3}$ denote the constant inertia matrix (expressed in the body fixed frame $\{B\}$). The rigid body equations of motion of the airframe are [2] and [3]

$$\dot{\xi} = v, \quad (1a)$$

$$m\dot{v} = mg\vec{a}_3 + RF, \quad (1b)$$

$$\dot{R} = R\Omega_{\times}, \quad (1c)$$

$$I\dot{\Omega} = -\Omega \times I\Omega + \tau. \quad (1d)$$

The notation Ω_{\times} denotes the skew-symmetric matrix, such that $\Omega_{\times}v = \Omega \times v$ for the vector cross product \times and any vector $v \in \mathbb{R}^3$. The vectors $F, \tau \in \{B\}$ combine the principal nonconservative forces and moments applied to the quadrotor airframe by the aerodynamics of the rotors.

Dominant Aerodynamics

The aerodynamics of rotors was extensively studied during the mid 1900s with the development of manned helicopters, and detailed models of rotor aerodynamics are available in the literature [4], [5]. Much of the detail about these aerodynamic models is useful for the design of rotor systems, where the whole range of parameters (rotor

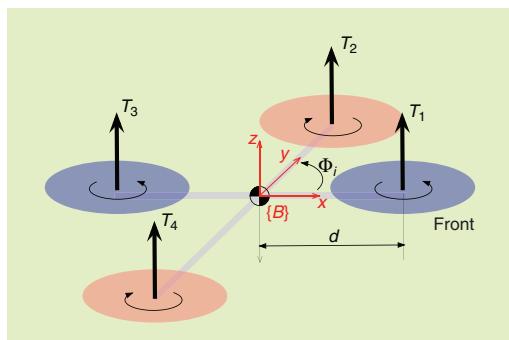


Figure 2. Notation for quadrotor equations of motion. $N = 4$, Φ_i is a multiple of $\pi/4$ (adapted with permission from [1]).

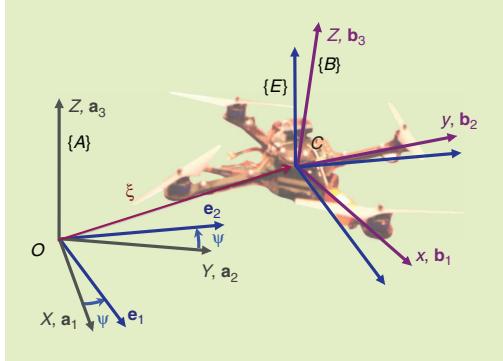


Figure 3. The vehicle model. The position and orientation of the robot in the global frame are denoted by ξ and R , respectively.

geometry, profile, hinge mechanism, and much more) are fundamental to the design problem. For a typical robotic quadrotor vehicle, the rotor design is a question for choosing one among five or six available rotors from the hobby shop, and most of the complexity of aerodynamic modeling is best ignored. Nevertheless, a basic level of aerodynamic modeling is required.

The steady-state thrust generated by a hovering rotor (i.e., a rotor that is not translating horizontally or vertically) in free air may be modeled using momentum theory [5, Sec. 2.26] as

$$T_i := C_T \rho A_{r_i} r_i^2 \varpi_i^2, \quad (2)$$

where, for rotor i , A_{r_i} is the rotor disk area, r_i is the radius, ϖ_i is the angular velocity, C_T is the thrust coefficient that depends on rotor geometry and profile, and ρ is the density of air. In practice, a simple lumped parameter model

$$T_i = c_T \varpi_i^2 \quad (3)$$

is used, where $c_T > 0$ is modeled as a constant that can be easily determined from static thrust tests. Identifying the thrust constant experimentally has the advantage that it will also naturally incorporate the effect of drag on the airframe induced by the rotor flow.

The reaction torque (due to rotor drag) acting on the airframe generated by a hovering rotor in free air may be modeled as [5, Sec. 2.30]

$$Q_i := c_Q \varpi_i^2, \quad (4)$$

where the coefficient c_Q (which also depends on A_{r_i} , r_i , and ρ) can be determined by static thrust tests.

As a first approximation, assume that each rotor thrust is oriented in the z axis of the vehicle, although we note that this assumption does not exactly hold once the rotor begins to rotate and translate through the air, an effect that

is discussed in “Rotor Flapping.” For an N -rotor airframe, we label the rotors $i \in \{1 \dots N\}$ in an anticlockwise direction with rotor 1 lying on the positive x axis of the vehicle (the front), as shown in Figure 2. Each rotor has associated an angle Φ_i between its airframe support arm and the body-fixed frame x axis, and it is the distance d from the central axis of the vehicle. In addition, $\sigma_i \in \{-1, +1\}$ denotes the direction of rotation of the i th rotor: $+1$ corresponding to clockwise and -1 to anticlockwise. The simplest configuration is for N even and the rotors distributed symmetrically around the vehicle axis with adjacent rotors counter rotating.

The total thrust at hover (T_Σ) applied to the airframe is the sum of the thrusts from each individual rotor

$$T_\Sigma = \sum_{i=1}^N |T_i| = c_T \left(\sum_{i=1}^N \varpi_i^2 \right). \quad (5)$$

The hover thrust is the primary component of the exogenous force

$$F = T_\Sigma \vec{z} + \Delta \quad (6)$$

in (1b), where Δ comprises secondary aerodynamic forces that are induced when the assumption that the rotor is in hover is violated. Since F is defined in $\{B\}$, the direction of application is written \vec{z} , although in the frame $\{A\}$ this direction is $\vec{b}_3 = R\vec{z}$.

The net moment arising from the aerodynamics (the combination of the produced rotor forces and air resistances) applied to the N -rotor vehicle use are $\tau = (\tau_1, \tau_2, \tau_3)$.

$$\begin{aligned} \tau_1 &= c_T \sum_{i=1}^N d_i \sin(\Phi_i) \varpi_i^2, \\ \tau_2 &= -c_T \sum_{i=1}^N d_i \cos(\Phi_i) \varpi_i^2, \\ \tau_3 &= c_Q \sum_{i=1}^N \sigma_i \varpi_i^2. \end{aligned} \quad (7)$$

For a quadrotor, we can write this in matrix form

$$\begin{pmatrix} T_\Sigma \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix} = \underbrace{\begin{pmatrix} c_T & c_T & c_T & c_T \\ 0 & dc_T & 0 & -dc_T \\ -dc_T & 0 & dc_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{pmatrix}}_{\Gamma} \begin{pmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \varpi_3^2 \\ \varpi_4^2 \end{pmatrix}, \quad (8)$$

and given the desired thrust and moments, we can solve for the required rotor speeds using the inverse of the constant matrix Γ . In order for the vehicle to hover, one must choose suitable ϖ_i by inverting Γ , such that $\tau = 0$ and $T_\Sigma = mg$.

Blade Flapping and Induced Drag

There are many aerodynamic and gyroscopic effects associated with any rotor craft that modify the simple force model introduced above. Most of these effects cause only minor perturbations and do not warrant consideration for a robotic system, although they are important for the design of a full-sized rotor craft. Blade flapping and induced drag, however, are fundamental effects that are of significant importance in understanding the natural stability of quadrotors and how state observers operate. These effects are particularly relevant since they induce forces in the x - y rotor plane of the quadrotor, the underactuated directions in the dynamics, that cannot be easily dominated by high gain control. In this section, we consider a single rotor and we will drop the subscript i used in the “Dominant Aerodynamics” section to refer to particular rotors.

Quadrotor vehicles are typically equipped with lightweight, fixed-pitch plastic rotors. Such rotors are not rigid, and the aerodynamic and inertial forces applied to a rotor during flight are quite significant and can cause the rotor to flex. In fact, allowing the rotor to bend is an important property of the mechanical design of a quadrotor and fitting rotors that are too rigid can lead to transmission of these aerodynamic forces directly through to the rotor hub and may result in a mechanical failure of the motor mounting or the airframe itself. Having said this, rotors on small vehicles are significantly more rigid relative to the applied aerodynamic forces than rotors on a full-scale rotor craft. Blade-flapping effects are due to the flexing of rotors, while induced drag is associated primarily with the rigidity of the rotor, and a typical quadrotor will experience both. Luckily, their mathematical expression is equivalent and a single term is sufficient to represent both effects in a lumped parameter dynamic model.

When a rotor translates laterally through the air it displays an effect known as rotor flapping (see “Rotor Flapping”). A detailed derivation of rotor flapping involves a mechanical model of the bending of the rotor subject to aerodynamic and centripetal forces as it is swept through a full rotation [5, Sec. 4.5]. The resulting equations of motion are a nonlinear second-order dynamical system with a dominant highly damped oscillatory response at the forced frequency corresponding to the angular velocity of the rotor. For a typical rotor, the flapping dynamics converge to steady state with one cycle of the rotor [5, p. 137], and for the purposes of modeling, only the steady-state response of the flapping dynamics need be considered.

Assuming that the velocity of the vehicle is directly aligned with the X axis in the inertial frame, $v = (v_x, 0, 0)$, a simplified solution is given by

$$\beta := -\frac{\mu A_{lc}}{(1 - \frac{1}{2}\mu^2)}, \quad \beta^\perp := -\frac{\mu A_{ls}}{(1 + \frac{1}{2}\mu^2)} \quad (9)$$

for positive constants A_{lc} and A_{ls} , and where $\mu := |v_x|/\omega r$ is the *advance ratio*, i.e., the ratio of magnitude of

the horizontal velocity of the rotor to the linear velocity of rotor tip. The flapping angle β is the steady-state tilt of the rotor away from the incoming apparent wind and β^\perp is the tilt orthogonal to the incident wind. Here, we use equations (4.46) and (4.47) from [5, p. 138], noting that adding the effects of a virtual rotor hinge model [5, Sec. 4.7] results in additional phase lag between the sine and cosine components of the flapping angles [5, Question 4.7, p. 157] that are absorbed into the constants A_{lc} and A_{ls} in (9).

Rotor flapping is important because the thrust generated by the rotor is perpendicular to the rotor plane and not to the hub of the rotor. Thus, when the rotor disk tilts the rotor thrust is inclined with respect to the airframe and contains a component in the x and y directions of the body-fixed frame.

In practice the rotors are stiff and oppose the aerodynamic force which is lifting the advancing blade so that its increased thrust due to tip velocity is not fully counteracted by a lower angle of attack and lower lift coefficient—the thrust is increased. Conversely for the retreating blade the thrust is reduced. For any airfoil that generates lift (in our case the rotor blade) there is an associated *induced drag*

Rotor Flapping

When a rotor translates horizontally through the air, the advancing blade has a higher absolute tip velocity and will generate more lift than the retreating blade. Thinking of the rotor as a spinning disk, the mismatch in lift generates an overall moment on the rotor disk in the direction of the apparent wind (Figure S1). The high angular momentum of the rotor disk makes it act like a gyroscope, which causes the rotor disk to tilt around the axis given by the cross product of rotor hub axis and the torque axis, i.e., an axis that is offset from the apparent wind by 90° in the horizontal plane of the rotor. Since the motor shaft is vertical, the blade flaps up as it advances into the wind and back down again as it retreats from the apparent wind. Equilibrium is established because the advancing blade rises and decreases its angle of attack, which reduces its lift coefficient, thereby counteracting the additional lift that would have been generated due to its increased tip velocity. Conversely for the retreating blade, the reduced lift due to decreased tip velocity is countered by the increased angle of attack and increased thrust coefficient. In this state, the rotor will have a stable constant tilt away from the apparent wind caused by a translational motion of the rotor. This effect is known as *rotor flapping* and is ubiquitous in rotor vehicles [6].

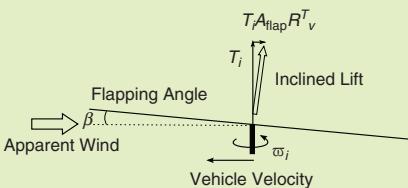


Figure S1

due to the backward inclination of aerodynamic force with respect to the airfoil motion. The induced drag is proportional to the lift generated by the airfoil. In normal hover conditions for a rotor, this force is equally distributed in all directions around the circumference of the rotor and is responsible for the torque Q (4). However, when there is a thrust imbalance, then the sector of the rotor travel with high thrust (for the advancing rotor) will generate more induced drag than the sector where the rotor generates less thrust (for the retreating blade). The net result will be an induced drag that opposes the direction of apparent wind as seen by the rotor, and that is proportional to the velocity of the apparent wind. This effect is often negligible for full scale rotor craft; however, it may be quite significant for small quadrotor vehicles with relatively rigid blades. The consequence of blade flapping and induced drag taken together ensures that there is always a noticeable horizontal drag experienced by a quadrotor even when maneuvering at relatively slow speeds.

We will now use the insight from the discussion above to develop a lumped parameter model for exogenous force generation (6). We assume that all four rotors are identical and rotate at similar speeds so that, at least to a first approximation, the flapping responses of the rotors and the unbalanced aerodynamic forces are the same. It follows that the reactive torques on the airframe transmitted by the rotor masts due to rotor stiffness cancel. For general motion of the vehicle, the apparent wind results in the advance ratio

$$\mu = \sqrt{v'_x^2 + v'_y^2} / \varpi r,$$

where $v' = R^\top v$ is the linear velocity of the vehicle expressed in the body-fixed frame, with v'_x and v'_y being the components in the body-fixed x - y plane. Define

$$A_{\text{flap}} = \frac{1}{\varpi R} \begin{pmatrix} A_{1c} & -A_{1s} & 0 \\ A_{1s} & A_{1c} & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

where ϖ is the set point for the rotor angular velocity. This matrix describes the sensitivities of the flapping angle to the apparent wind in the body-fixed frame, given that μ is small and μ^2 is negligible in the denominators of (9). The first row encodes (9) for the velocity along the body-fixed frame x axis. The second row of A_{flap} is a $\pi/2$ rotation of this response to account for the case where a component of the wind is incoming from the y axis, while the third row projects out velocity in the z axis of the body-fixed frame.

We model the stiffness of the rotor as a simple torsional spring so that the induced drag is directly proportional to this angle and is scaled by the total thrust. The flapping angle is negligible with regard to the orientation of the

induced drag, and in the body-fixed frame the induced drag is

$$D_{\text{ind.}} v' \approx \text{diag}(d_x, d_y, 0) v',$$

where $d_x = d_y$ is the induced drag coefficient.

The exogenous force applied to the rotor can now be modeled by

$$F := T_\Sigma \vec{z} - T_\Sigma D v', \quad (10)$$

where $D = A_{\text{flap}} + \text{diag}(d_x, d_y, 0)$, and T_Σ is the nominal thrust (5).

An important consequence of blade flapping and induced drag is a natural stability of the horizontal dynamics of the quadrotor [7]. Define

$$\mathbb{P}_h := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad (11)$$

to be the projection matrix onto the x - y plane. The horizontal component of a velocity expressed in $\{A\}$ is

$$v_h := \mathbb{P}_h v = (v_x, v_y)^\top \in \mathbb{R}^2. \quad (12)$$

Recalling (1b) and projecting onto the horizontal component of velocity, one has

$$m \dot{v}_h = -T_\Sigma \mathbb{P}_h (\vec{z} + RD v').$$

If the vehicle is flying horizontally, i.e., $v_z = 0$, then $v = \mathbb{P}_h^\top v_h$ and one can write

$$m \dot{v}_h = -T_\Sigma \mathbb{P}_h \vec{z} - \mathbb{P}_h R D R^\top \mathbb{P}_h^\top v_h, \quad (13)$$

where the last term introduces damping since, for a typical system, the matrix D is a positive semidefinite.

A detailed dynamic model of the quadrotor, including flapping and induced drag, is included in the robotics toolbox for MATLAB [8]. This is provided in the form of Simulink library blocks along with a set of inertial and aerodynamic parameters for a particular quadrotor. The graphical output of the animation block is shown in Figure 4. Simulink models, based on these blocks, that illustrate path following and vision-based stabilization are described in detail in [1].

The discussion provided above does not consider several additional aerodynamic effects that are important for high-speed and highly dynamic maneuvers for a quadrotor. In particular, we do not consider translational lift and drag that will effect thrust generation at high speed, axial flow modeling and vortex states that may effect thrust during axial motion, and ground effect that will affect a vehicle flying close to the ground. It should be noted that high gain control can dominate all secondary aerodynamic effects, and high

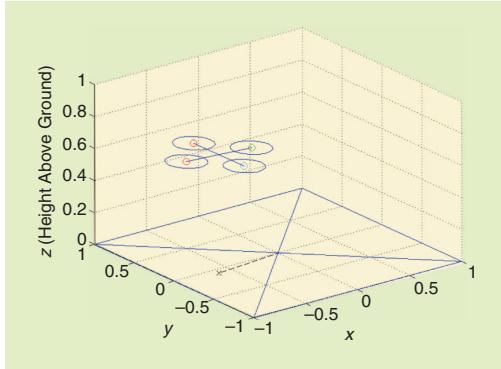


Figure 4. Frame from the Simulink animation of quadrotor dynamics.

performance control of quadrotor vehicles has been demonstrated using the simple static thrust model [23], [24]. The detailed modeling of the blade flapping and induced drag is provided due to its importance in understanding the state estimation algorithms introduced later in the tutorial.

Size, Weight, and Power (SWAP)

Constraints and Scaling Laws

Reducing the scale of the quadrotor has an interesting effect on the inertia, payload, and ultimately the maximum achievable angular and linear acceleration. To gain insight into scaling, it is useful to develop a simple physics model to analyze a quadrotor's ability to produce linear and angular accelerations from a hover state.

If the characteristic length is d , the rotor radius r scales linearly with d . The mass scales as d^3 and the moments of inertia as d^5 . On the other hand, from (3) and (4), it is clear that the lift or thrust, T , and drag, Q , from the rotors scale with the square of the rotor speed, ϖ^2 . In other words, $T \sim \varpi^2 d^4$ and $Q \sim \varpi^2 d^4$, the linear acceleration $a = \ddot{v}$, which depends on the thrust and mass, and the angular acceleration $\alpha = \dot{\Omega}$, which depends on thrust, drag, the moment arm, and the moments of inertia, scale as

$$a \sim \frac{\varpi^2 d^4}{d^3} = \varpi^2 d, \quad \alpha \sim \frac{\varpi^2 d^5}{d^5} = \varpi^2.$$

To explore the scaling of rotor speed with length, it is useful to adopt the two commonly accepted approaches to study scaling in aerial vehicles [9]. Mach scaling is used for compressible flows and essentially assumes that the blade tip speed, v_b , is a constant leading to $\varpi \sim (1/r)$. Froude scaling is used for incompressible flows and assumes that, for similar aircraft configurations, the Froude number, $(v_b^2/dg) = (\varpi^2 r^2/dg)$, is constant. Here, g is the acceleration due to gravity.

Assuming $r \sim d$, we get $\varpi \sim (1/\sqrt{r})$. Thus, Mach scaling predicts

$$a \sim \frac{1}{d}, \quad \alpha \sim \frac{1}{d^2}, \quad (14)$$

while Froude scaling leads to the conclusion

$$a \sim 1, \quad \alpha \sim \frac{1}{d}. \quad (15)$$

Of course, Froude or Mach number similitudes take neither motor characteristics nor battery properties into account. While motor torque increases with length, the operating speed for the rotors is determined by matching the torque-speed characteristics of the motor to the drag versus speed characteristics of the rotors. Further, the motor torque depends on the ability of the battery to source the required current. All these variables are tightly coupled for smaller designs since there are fewer choices available at smaller length scales. Finally, as discussed in the previous subsection, the assumption that rotor blades are rigid may be wrong. Further, the aerodynamics of the blades may be different for blade designs optimized for smaller helicopters and the quadratic scaling of the lift with speed may not be accurate.

In spite of the simplifications in the above similitude analysis, the key insight from both Froude and Mach number similitudes is that smaller quadrotors can produce faster angular accelerations while the linear acceleration is at worst unaffected by scaling. Thus, smaller quadrotors are more agile, a fact that is easily validated from experiments conducted with the Ascending Technologies Pelican quadrotor [10] (approximately 2 kg gross weight when equipped with sensors, 0.75 m diameter, and 5,400 r/min nominal rotor speed at hover), the Ascending Technologies Hummingbird quadrotor [11] (approximately 500 g gross weight, 0.5 m diameter, and 5,000 r/min nominal rotor speed at hover), and laboratory experimental prototypes developed at GRASP laboratory at the University of Pennsylvania (approx. 75 g gross weight, 0.21 m diameter, and approximately 9,000 r/min nominal rotor speed).

Estimating the Vehicle State

The key state estimates required for the control of a quadrotor are its height, attitude, angular velocity, and linear velocity. Of these states, the attitude and angular velocity are the most important as they are the primary variables used in attitude control of the vehicle. The most basic instrumentation carried by any quadrotor is an inertial measurement unit (IMU) often augmented by some form of height measurement, either acoustic, infrared, barometric, or laser based. Many robotics applications require more sophisticated sensor suites such as VICON systems, global positioning system (GPS), camera, Kinect, or scanning laser rangefinder.

Estimating Attitude

A typical IMU includes a three-axis rate gyro, three-axis accelerometer, and three-axis magnetometer. The rate gyro measures the angular velocity of $\{B\}$ relative to $\{A\}$ expressed in the body-fixed frame of reference $\{B\}$

$$\Omega_{\text{IMU}} = \Omega + b_\Omega + \eta \in \{B\},$$

where η denotes the additive measurement noise and b_Ω denotes a constant (or slowly time-varying) gyro bias. Generally, the gyroscopes installed on quadrotor vehicles are lightweight microelectromechanical systems (MEMS) devices that are reasonably robust to noise and quite reliable.

The accelerometers (in a strap down IMU configuration) measure the instantaneous linear acceleration of $\{B\}$ due to exogenous force

$$a_{\text{IMU}} = R^\top(\dot{v} - g\vec{z}) + b_a + \eta_a \in \{B\},$$

where b_a is a bias term, η_a denotes additive measurement noise, and \dot{v} is in the inertial frame. Here, we use the notation $\vec{z} = \vec{a}_3$ since we will need to deal with the algebraic expressions of the coordinate axes throughout this section. Accelerometers are highly susceptible to vibration and, mounted on a quadrotor, they require significant low-pass mechanical and/or electrical filtering to be usable. Most quadrotor avionics will incorporate an analogue anti-aliasing filter on a MEMS accelerometer before the signal is sampled.

A commonly used technique to estimate the bias b_Ω and b_a is to average the output of these sensors for a few seconds while the quadrotor is on the ground and the motors are not yet active. The bias is then assumed constant for the duration of the flight.

The magnetometers provide measurements of the ambient magnetic field

$$m_{\text{IMU}} = R^T A m + B_m + \eta_b \in \{B\},$$

where $A m$ is the Earth's magnetic field vector (expressed in the inertial frame), B_m is a body-fixed frame expression for the local magnetic disturbance, and η_b denotes the measurement noise. The noise η_b is usually low for magnetometer readings; however, the local magnetic disturbance B_m can be significant, especially if the sensor is placed near the power wires to the motors.

The accelerometers and magnetometers can be used to provide absolute attitude information on the vehicle while the rate gyroscope provides complementary angular velocity measurements. The attitude information in the magnetometer signal is straightforward to understand; in the absence of noise and bias, m_{IMU} provides a body-fixed frame measurement of $R^T A m$ and, consequently, constrains two DoF in the rotation R .

The case for using the accelerometer signal for attitude estimation is far more subtle. Using the simplest model (6)

with $\Delta \equiv 0$, $a_{\text{IMU}} = R^\top(\dot{v} - g\vec{z}) = (T_\Sigma/m)\vec{z} \approx g\vec{z}$. This shows that the measured acceleration, for this simple model, would always point in the body-fixed frame direction \vec{z} and provides no attitude information. In practice, it is the blade-flapping component of the thrust that contributes attitude information to the accelerometer signal [7]. Recalling (10) and ignoring bias and noise terms, the model for a_{IMU} can be written as

$$a_{\text{IMU}} = -\frac{T_\Sigma}{m}\vec{z} - \frac{T_\Sigma}{m}DR^\top v. \quad (16)$$

As we show later in the section, only the low-frequency information from the accelerometer signal will be used in the observer construction. Thus, it is only the low-frequency or approximate steady-state response \bar{v} of the velocity v that we need to estimate to build a model for the low-frequency component of a_{IMU} . Setting $\dot{v} = 0$ in (1b), substituting for force (10), and rearranging, we obtain an estimate of the low-frequency component of the velocity signal

$$DR^\top \bar{v} \approx R^\top \vec{z} - \vec{z}.$$

Substituting $DR^\top \bar{v}$ for $DR^\top v$ in (16), we obtain

$$\bar{a}_{\text{IMU}} \approx -\frac{T_\Sigma}{m}R^\top \vec{z}, \quad (17)$$

where \bar{a}_{IMU} denotes the low-frequency component of the accelerometer signal. That is, the low-frequency content of a_{IMU} when the vehicle is near hover is the body-fixed frame expression for the supporting force that is the negative gravitational vector expressed in the body-fixed frame. Most robotics applications involve a quadrotor spending significant periods of time in hover, or slow forward flight, with $\dot{v} \approx 0$, and using the accelerometer reading as an attitude reference during this flight regime has been shown to work well in practice.

The attitude kinematics of the quadrotor are given by (1c). Let \hat{R} denote an estimate for attitude R of the quadrotor vehicle. The following observer [12] fuses accelerometer, magnetometer, and gyroscope data as well as other direct attitude estimates R_E (such as provided by a VICON or other external measurement system) should they be available:

$$\begin{aligned} \dot{\hat{R}} &:= \hat{R} \left(\Omega_{\text{IMU}} - \hat{b} \right)_\times - \alpha, \\ \dot{\hat{b}} &:= k_b \alpha, \\ \alpha &:= \left(\frac{k_a}{g^2} ((\hat{R}^\top \vec{z}) \times \bar{a}_{\text{IMU}}) + \frac{k_m}{|A m|^2} ((\hat{R}^\top A m) \times m_{\text{IMU}}) \right)_\times \\ &\quad + k_E \mathbb{P}_{\text{so}(3)}(\hat{R} R_E^\top), \end{aligned} \quad (18)$$

where k_a, k_m, k_E , and k_b are arbitrary nonnegative observer gains and $\mathbb{P}_{\text{so}(3)}(M) = (M - M^\top)/2$ is the Euclidean

matrix projection onto skew-symmetric matrices. If any one of the measurements in the innovation α are not available or unreliable, then the corresponding gain should be set to zero in the observer. Note that both the attitude \hat{R} and the bias corrected angular velocity $\hat{\Omega} = \Omega_{\text{IMU}} - \hat{b}$ are estimated by this observer. The observer (18) has been extensively studied in the literature [12], [13] and shown to converge exponentially (both theoretically and experimentally) to the desired attitude estimate of attitude with \hat{b} converging to the gyroscope bias b . The filter has a complementary nature, using the high-frequency part of the gyroscope signal and the low-frequency parts of the magnetometer, accelerometer, and external attitude measurements [12]. The roll-off frequencies associated with each of these signals is given by the gains k_a , k_m , and k_E in rad.s $^{-1}$, and good performance of the observer depends on how these gains are tuned. In particular, the accelerometer gains must be tuned to a frequency below the normal bandwidth of the vehicle motion, less than 5 rad.s $^{-1}$ for a typical quadrotor. The magnetometer gain and external gain can be tuned for a higher roll-off frequency depending on the reliability of the signals. The bias gain k_b is typically chosen an order of magnitude slower than the innovation gains $k_b < k_a/10$, leading to a rise time of the bias estimate as slow as 30 s or more. This dynamic response is necessary to track slowly varying bias and decouples the bias estimate from the attitude response; however, it is necessary to initialize the observer with a bias estimate at take off to avoid a long transient in the filter response.

A particular advantage of this observer formulation is that the gains can be adjusted in real time as long as care is taken that the bias gain is small. Adjusting the gains in real time allows one to use the accelerometer during a period when the vehicle is in hover and then set the gain $k_a = 0$ during acrobatic maneuvers when the low-frequency assumptions on \bar{a}_{IMU} no longer hold. The nonlinear robustness, guaranteed asymptotic stability, and flexibility in gain tuning make this observer a preferred candidate for quadrotor attitude estimation compared with classical filters such as the extended Kalman filter (EKF), multiplicative EKF, or more sophisticated stochastic filters.

Estimating Translational Velocity

The blade-flapping response provides a way to build an observer for the horizontal velocity of the vehicle based on the IMU sensors [7], at least while the vehicle is flying in the horizontal plane. Assume that a good estimate of the vehicle attitude \hat{R} is available and that the vehicle is flying at constant height.

Recalling the projector (11), the horizontal component of the inertial acceleration can be measured by

$${}^A a_h := \mathbb{P}_h {}^A a = \mathbb{P}_h R a \approx \mathbb{P}_h \hat{R} a, \quad (19)$$

where the signals a and \hat{R} are available. Since we assume that the vehicle is flying at a constant height, one has

$v_z \approx 0$, and recalling (12), $\mathbb{P}_h^\top v_h \approx v$. Further, the thrust $T_\Sigma \approx mg$ must compensate the weight of the vehicle. Recalling (16) and taking the horizontal component, one has

$${}^A a_h \approx -g \mathbb{P}_h \hat{R} \vec{z} - g \mathbb{P}_h \hat{R} D R^\top \mathbb{P}_h^\top v_h. \quad (20)$$

Assuming that the attitude filter estimate is good, i.e., $\hat{R} = R$, then (19) and (20) can be solved for an estimate of v_h

$$v_h \approx -\frac{1}{g} \left[\mathbb{P}_h \hat{R} D R^\top \mathbb{P}_h^\top \right]^{-1} ({}^A a_h + g \mathbb{P}_h \hat{R} \vec{z}). \quad (21)$$

This estimate of v_h will be well defined as long as the 2×2 matrix $\mathbb{P}_h \hat{R} D R^\top \mathbb{P}_h^\top$ is invertible, a condition that will hold as long as the vehicle pitches or rolls by less than 90° during flight.

Equation (21) provides a measurement of the horizontal velocity; however, since it directly incorporates the unfiltered accelerometer readings, it is generally too noisy to be of much use. Its low-frequency content can, however, be used to drive a velocity complementary observer that uses the attitude estimate and the system model (1b) along with the thrust model (10) for its high-frequency component. Let \hat{v}_h be an estimate of the horizontal component of the inertial velocity of the vehicle. Recalling (1b), we propose the following observer

$$\dot{\hat{v}}_h = -g \mathbb{P}_h^\top (\hat{R} \vec{z} + \hat{R} D \hat{R}^\top \mathbb{P}_h^\top \hat{v}_h) - k_w (\hat{v}_h - v_h), \quad (22)$$

where v_h is given by (21). The gain $k_w > 0$ provides a tuning parameter that adjusts the roll-off frequency for the information from \hat{v}_h that is used in the filter. It also uses an estimated velocity \hat{v}_h to provide an approximation of the more correct $R D R^\top \mathbb{P}_h^\top v_h$ term in the feedforward velocity estimate; however, since the underlying dynamics associated with this term are stable, the observer is stable even with this approximation.

Estimating Position

The final part of state that must be estimated is position, which is typically considered separately as position in the plane and height. Considering the height first, there are in fact two separate heights that are of importance: the first is the absolute height of the vehicle and the second is the relative height over the terrain at a given time. Unfortunately, there is no effective way to use the IMU to estimate absolute height; at best, some low-frequency information from the z axis of the accelerometer provides limited information about vertical motion. Most quadrotors include a barometric sensor that can resolve absolute height to a few centimeters. Absolute height can also be estimated using GPS, VICON, or a full SLAM system. Relative height can be estimated using acoustic, laser-ranging or infrared

sensors. Once a sufficiently accurate height measurement is available, it is better to use this directly in the control than add additional levels of complexity in designing a height observer, especially since, for a typical system, the only feedforward information available is the noisy accelerometer readings.

Position in the plane can also be determined in a relative or absolute way. Absolute position can be obtained from a GPS (few-centimeter accuracy at up to 10 Hz [6]) or an external localization device such as a VICON motion capture system (50 μm accuracy at 375 Hz). However, a GPS does not work indoors and motion-capture systems are expensive, and their sensor array has a limited spatial extent that is impractical to scale up for large indoor environments.

Relative position can be estimated by measuring the distance to objects in the environment from onboard sensors, typically small onboard laser range finders (LRFs) or RGBD camera systems such as the Kinect. Well-known SLAM techniques, borrowing LRF-based techniques similar to those developed for mobile ground robots over the last decade, have been applied to quadrotors [14]. However, LRFs provide only a cross section of the 3-D environment and this scan plane tilts as the vehicle maneuvers, resulting in apparent changes to the distance of walls, and, in extreme cases, the scan plane can intersect the floor or ceiling. LRFs are heavy and power hungry, which prevents their application to the next generation of much smaller quadcopters.

Vision has the advantage that the sensor is small, lightweight, and low power, which will become increasingly important as the size of aerial vehicles decreases. Vision can provide essential navigational competencies such as odometry, attitude estimation, mapping, place and object recognition, and collision detection. There is a long history of applying vision to aerial robotic systems [15]–[19] for indoor and outdoor environments, and the well-known Parrot AR.Drone game device makes strong use of vision for attitude and odometry [20]. Vision can also be used for object recognition based on color, texture, and shape, as well as collision avoidance.

Vision is not without its challenges. First, vision is computationally intense and can result in a low sample rate. Since onboard computational power is limited (by SWAP consumption), most reported systems transmit the images wirelessly to a ground station, which increases system

complexity, control latency, and the susceptibility to interference and dropouts. However, processor speed continues to improve, and we can also utilize the vision and control techniques used by flying insects that perform complex tasks with very limited sensing and neural capability [21]. Second, there is an ambiguity between certain rotational and translational motions, particularly, when a narrow field of view perspective camera is used. Third, the underactuated quadrotor uses the roll and pitch DoF to point the thrust vector in the direction of the desired translational motion. For a camera that is rigidly attached to the quadrotor, this attitude control motion induces a large apparent motion in the image. It is therefore necessary to estimate vehicle attitude at the instant the image was captured by the sensor to eliminate this effect. Biological systems face similar problems, and interestingly, mammals and insects have developed similar solutions: gyroscopic sensors (the vestibular sensors of the inner ear and the halteres, respectively) [22]. Finally, there exists a problem with recovering motion scale when using a single camera. Stereo is possible, but the baseline is constrained, particularly as vehicles get smaller.

Control

The control problem, to track smooth trajectories $(R^*(t), \xi^*(t)) \in \text{SE}(3)$, is challenging for several reasons. First, the system is underactuated: there are four inputs $\mathbf{u} = (T_\Sigma, \tau^\top)^\top$, while $\text{SE}(3)$ is six dimensional. Second, the aerodynamic model described above is only approximate. Finally, the inputs are themselves idealized. In practice, the motor controllers must overcome the drag moments to generate the required speeds and realize the input thrust (T_Σ) and moments (τ). The dynamics of the motors and their interactions with the drag forces on the propellers can be difficult to model, although first-order linear models are a useful approximation.

A hierarchical control approach is common for quadrotors. The lowest level, the highest bandwidth, is in control of the rotor rotational speed. The next level is in control of vehicle attitude, and the top level is in control of position along a trajectory. These levels form nested feedback loops, as shown in Figure 5.

Controlling the Motors

Rotor speed drives the dynamic model of the vehicle according to (8), so high-quality control of the motor speed is fundamentally important for overall control of the vehicle; high bandwidth control of the thrust T_Σ , denoted by u_1 , and the torques (τ_x, τ_y, τ_z) , denoted by \mathbf{u}_2 , lead to high performance attitude and position control. Most quadrotor vehicles are equipped with brushless dc motors that use back electromotive force (EMF) sensing

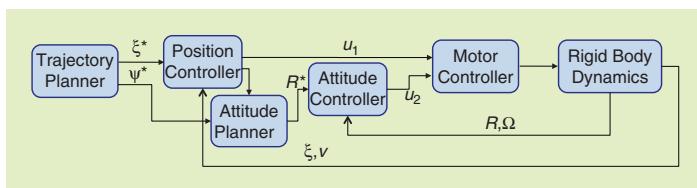


Figure 5. The innermost motor control loop, the intermediate attitude control loop, and the outer position control loop.

for rotor commutation and high-frequency pulselwidth modulation (PWM) to control motor voltage. The simplest systems generally use a direct voltage control of the motors since steady-state motor speed is proportional to voltage; however, the dynamic response is second-order due to the mechanical and electrical dynamics. Improved performance is obtained by incorporating single-input single-output control at the motor/rotor level

$$V_i = k(\varpi_i^* - \varpi_i) + V_{ff}(\varpi_i^*), \quad (23)$$

where V_i is the applied motor voltage, ϖ_i^* is the desired speed, and the actual motor speed ϖ_i can be measured from the electronic commutation in the embedded speed controller. This can help to overcome a common problem where the rotor speed for a given PWM command setting will decrease as the battery voltage reduces during flight. The significant load torque due to aerodynamic drag will lead to a tracking error that can be minimized by high proportional gain (k) and/or a feedforward term. A positive benefit of the drag torque is that the system is heavily damped, which precludes the need for derivative control. The feed-forward term $V_{ff}(\varpi_i^*)$ compensates for the steady-state PWM associated with a given velocity set point by incorporating the best available thrust model determined using static thrust tests and possibly including battery voltage.

The performance of the motor controllers is ultimately limited by the current that can be supplied from the batteries. This may be a significant limiting factor for smaller vehicles. Overly aggressive tuning and extreme maneuvers may cause the voltage bus to drop excessively, reducing the thrust from other rotors and, in extreme cases, causing the onboard electronics to brownout. For this reason, it is common to introduce a saturation, although this destroys the linearity of the motor/rotor response during aggressive maneuvers.

Attitude Control

We first consider the design of an exponentially converging controller in SO(3). Given a desired airframe attitude R^* , we want to first develop a measure of the error in rotations. We choose the measure

$$e_{R_x} = \frac{1}{2}((R^*)^T R - R^T R^*), \quad (24)$$

which yields a skew-symmetric matrix representing the axis of rotation required to go from R to R^* and whose magnitude is equal to the sine of the angle of rotation.

To derive linear controllers, we linearize the dynamics about the nominal hover position at which the roll (ϕ) and pitch (θ) are close to zero and the angular velocities are close to zero. If we write $R = {}^A R_B$ as a product of the yaw

rotation ${}^A R_E(\psi)$ and ${}^E R_B(\phi, \theta)$, which is a composition of the roll and pitch, we can linearize the rotation about $(\psi, \phi, \theta) = (\psi_0, 0, 0)$

$$\begin{aligned} {}^A R_B &= {}^A R_E(\psi_0 + \Delta\psi) {}^E R_B(\Delta\phi, \Delta\theta) \\ &= \begin{pmatrix} \cos\psi & -\sin\psi & \Delta\theta \cos\psi + \Delta\phi \sin\psi \\ \sin\psi & \cos\psi & \Delta\theta \sin\psi - \Delta\phi \cos\psi \\ -\Delta\theta & \Delta\phi & 1 \end{pmatrix}, \end{aligned}$$

where $\psi = \psi_0 + \Delta\psi$. If $R^* = {}^A R_B(\psi_0 + \Delta\psi, \Delta\phi, \Delta\theta)$ and $R = {}^A R_B(\psi_0, 0, 0)$, (24) gives

$$e_{R_x} = \begin{pmatrix} 0 & \Delta\psi & -\Delta\theta \\ -\Delta\psi & 0 & \Delta\phi \\ \Delta\theta & -\Delta\phi & 0 \end{pmatrix}, \quad (25)$$

which, as we expect, corresponds to the error vector

$$e_R = (\Delta\phi, \Delta\theta, \Delta\psi)^\top,$$

with components in the body-fixed frame. If the desired angular velocity vector is zero, we can compute the proportional and derivative error to obtain the PD control law

$$\mathbf{u}_2 = -k_R e_R - k_\Omega e_\Omega, \quad (26)$$

where k_R and k_Ω are positive definite gain matrices. This controller guarantees stability for small deviations from the hover position.

To obtain convergence for larger deviations from the hover position, it is necessary to revert back to (24) without linearization. This allows us to directly compute the error on SO(3). By compensating for the nonlinear inertial terms and by including the correct error term, we obtain

$$\mathbf{u}_2 = J(-k_R e_R - k_\Omega e_\Omega) + \Omega \times J\Omega - J(\Omega \times R^T R^* \Omega^* - R^T R^* \dot{\Omega}^*). \quad (27)$$

This controller is guaranteed to be exponentially stable for almost any rotation [23]. From a practical standpoint, it is possible to neglect the last three terms in the controller and achieve satisfactory performance, but the correct calculation of the error term is important [24].

Trajectory Control

We now turn our attention to the control of the trajectory along a specified trajectory $\xi^*(t)$. As before, we first consider linear controllers by linearizing the dynamics about $\xi = \xi^*(t), \theta = \phi = 0, \psi = \psi^*(t), \dot{\xi} = 0$, and

$\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$, with the nominal input given by $u_1 = mg$, $\mathbf{u}_2 = 0$. Linearizing (1a), we get

$$\begin{aligned}\ddot{\xi}_1 &= g(\Delta\theta \cos \psi^* + \Delta\phi \sin \psi^*), \\ \ddot{\xi}_2 &= g(\Delta\theta \sin \psi^* - \Delta\phi \cos \psi^*), \\ \ddot{\xi}_3 &= \frac{1}{m} u_1 - g.\end{aligned}\quad (28)$$

To exponentially drive all three components of error, we want to command the acceleration vector $\ddot{\xi}^{\text{com}}$ to satisfy

$$(\ddot{\xi}^*(t) - \ddot{\xi}^{\text{com}}) + K_d(\dot{\xi}^*(t) - \dot{\xi}) + K_p(\xi^*(t) - \xi) = 0.$$

From (28), we can immediately write

$$u_1 = m\left(g + \ddot{\xi}_3^* + k_{d,z}(\dot{\xi}_3^* - \dot{\xi}_3) + k_{p,z}(\xi_3^* - \xi_3)\right), \quad (29)$$

to guarantee $(\xi_3(t) - \xi_3^*(t)) \rightarrow 0$. Similarly, for the other two components, we choose to command the appropriate θ^* and ϕ^* to guarantee exponential convergence

$$\phi^* = \frac{1}{g}(\ddot{\xi}_1^* \sin \psi^*(t) - \ddot{\xi}_2^* \cos \psi^*(t)), \quad (30a)$$

$$\theta^* = \frac{1}{g}(\ddot{\xi}_1^* \cos \psi^*(t) + \ddot{\xi}_2^* \sin \psi^*(t)), \quad (30b)$$

where the above equations are obtained by replacing $\Delta\theta$ by θ^* and $\Delta\phi$ by ϕ^* in (28). Finally, $(\psi^*, \phi^*, \theta^*)$ are provided as set points to the attitude controller discussed in the previous section. Thus, as shown in Figure 5, the control problem is addressed by decoupling the position control and attitude control subproblems, and the position control loop provides the attitude set points for the attitude controller.

The position controller can also be obtained without linearization. This is done by projecting the position error (and its derivatives) along \mathbf{b}_3 and applying the input u_1 that cancels the gravitational force and provides the appropriate proportional plus derivative feedback

$$u_1 = m\vec{\mathbf{b}}_3^T \left(\ddot{\xi}^* + K_d(\dot{\xi}^* - \dot{\xi}) + K_p(\xi^* - \xi) + g\vec{\mathbf{a}}_3 \right). \quad (31)$$

Note that the projection operation is a nonlinear function of the roll and pitch angles, and, thus, this is a nonlinear controller. In [23], it is shown that the two nonlinear controllers (27) and (31) result in exponential stability and allow the robot to track trajectories in SE(3).

Trajectory Planning

The quadrotor is underactuated, and this makes it difficult to plan trajectories in 12-dimensional state space (6 DoF position and velocity). However, the problem is considerably simplified if we use the fact that the quadrotor dynamics are differentially flat [25]. To see this, we consider the output position ξ and the yaw angle ψ . We show that we can write all state variables and

inputs as functions of the outputs (ξ, ψ) and their derivatives. Derivatives of ξ yield the velocity v and the acceleration,

$$\dot{v} = \frac{1}{m} u_1 \vec{\mathbf{b}}_3 + g\vec{\mathbf{a}}_3.$$

From Figure 3 we see that

$$\vec{\mathbf{e}}_1 = [\cos \psi, \sin \psi, 0]^T,$$

and the unit vectors for the body-fixed frame can be written in terms of the variables ψ and v as

$$\vec{\mathbf{b}}_3 = \frac{\dot{v} - g\vec{\mathbf{a}}_3}{\|\dot{v} - g\vec{\mathbf{a}}_3\|}, \quad \vec{\mathbf{b}}_2 = \frac{\vec{\mathbf{b}}_3 \times \vec{\mathbf{e}}_1}{\|\vec{\mathbf{b}}_3 \times \vec{\mathbf{e}}_1\|}, \quad \vec{\mathbf{b}}_1 = \vec{\mathbf{b}}_2 \times \vec{\mathbf{b}}_3$$

provided $\vec{\mathbf{b}}_3 \times \vec{\mathbf{e}}_1 \neq 0$. This defines the rotation matrix ${}^A R_B$ as a function of \dot{v} (the second derivative of ξ) and ψ . In this way, we write the angular velocity and the four inputs as functions of position, velocity, acceleration, jerk (γ), and snap, or the derivative of jerk (σ). From these equations, it is possible to verify that there is a diffeomorphism between the 18×1 vector

$$\left(\xi^T, v^T, a^T, \gamma^T, \sigma^T, \psi^T, \dot{\psi}^T, \ddot{\psi}^T \right)^T$$

and

$$R \times \left(\xi^T, \dot{\xi}^T, \ddot{\xi}^T, \Omega^T, u_1, \dot{u}_1, \ddot{u}_1, \mathbf{u}_2^T \right)^T.$$

This property of differential flatness makes it easy to design trajectories that respect the dynamics of the underactuated system. Any four-times-differentiable trajectory in the space of flat outputs, $(\xi^T(t), \psi(t))^T$, corresponds to a feasible trajectory—one that satisfies the equations of motion. All inequality constraints of states and inputs can be expressed as functions of the flat outputs and their derivatives. This mapping to the space of flat outputs can be used to generate trajectories that minimize a cost functional formed by a weighted combination of the different flat outputs and their derivatives:

$$\begin{aligned}\min_{\xi(t), \psi(t)} \int_0^T L(\xi, \dot{\xi}, \ddot{\xi}, \dddot{\xi}, \ddot{\psi}, \dot{\psi}, \ddot{\psi}) dt, \\ g(\xi(t), \psi(t)) \leq 0.\end{aligned}\quad (32)$$

In [24], minimum snap trajectories were generated by minimizing a cost functional derived from the snap and the angular yaw acceleration with

$$L(\xi, \dot{\xi}, \ddot{\xi}, \dddot{\xi}, \ddot{\psi}, \dot{\psi}, \ddot{\psi}) = (1 - \gamma)(\ddot{\xi})^4 + \gamma(\ddot{\psi})^2.$$

By suitable parameterizing trajectories with basis functions in the flat space and by considering linear inequalities in the flat

space to model constraints on states and inputs (e.g., $u_1 \geq 0$), it is possible to turn this optimization into a quadratic program that can be solved in real time for planning.

Finally, as shown in [11], it is possible to combine this controller with attitude-only controllers to fly through vertical windows or land on inclined perches with close to zero normal velocity. A trajectory controller is used by the robot to build up momentum, while the attitude controller enables reorientation while coasting with the generated momentum.

Vision-Based Perception and Control

There are two approaches to the question of controlling an aerial vehicle based on visual information. The first is to use classical robotic SLAM techniques, although with the caveat that the environment and state estimation are inherently 3-D. There are many researchers currently working on this problem, and we will not attempt to discuss this approach further, except to say that should a good-quality environmental estimation and localization algorithm be developed, the control techniques discussed above can be applied. The second approach is direct sensor-based control [26], the most commonly referred to case, being that of image-based visual servo control [27]–[29].

The motion of a point in an image is a function of its coordinate (u, v) and the camera motion

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = J(u, v, Z)v, \quad (33)$$

where Z is the point depth, $v = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)^\top$ is the spatial velocity of the camera (and vehicle), and $J(\cdot)$ is the visual Jacobian or interaction matrix. J can be formulated for a perspective camera [30], where (u, v) are pixel coordinates; or a spherical camera [31] where (u, v) are latitude and longitude angles.

The pitch and roll motion of the vehicle are controlled by the attitude subsystem to maintain a position or to follow a path in space, and this causes image motion. We partition the equations as

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = J_1(u, v)(v_x, v_y, v_z, \omega_z)^\top + J_2(u, v)\begin{pmatrix} \omega_x \\ \omega_y \end{pmatrix}, \quad (34)$$

where the right-most term describes the image motion due to the exogenous roll and pitch motion. Rearranging we can write

$$\begin{pmatrix} \dot{u}' \\ \dot{v}' \end{pmatrix} = \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix}^\top - J_2(u, v)\begin{pmatrix} \omega_x \\ \omega_y \end{pmatrix} \quad (35)$$

$$= J_1(u, v)(v_x, v_y, v_z, \omega_z)^\top, \quad (36)$$

where (u', v') represent image points for which the roll and pitch motion has been removed based on the knowledge of ω_x and ω_y , which can be obtained from gyroscopes.

Now consider a point in the image (u'_i, v'_i) and its desired location in the image (u_i^*, v_i^*) . This desired position might come from a snapshot of the scene taken when the vehicle was at the desired pose that we wish to return to. The desired image motion is therefore $(\dot{u}_i^*, \dot{v}_i^*) = \lambda(u_i^* \ominus u'_i, v_i^* \ominus v'_i)$, where the operator \ominus represents the difference on image plane or sphere. For N points, we can write

$$\begin{aligned} \lambda \begin{pmatrix} \dot{u}_1^* \\ \dot{v}_1^* \\ \vdots \\ \dot{u}_N^* \\ \dot{v}_N^* \end{pmatrix} &- \begin{pmatrix} J_1(u_1, v_1) \\ \vdots \\ J_1(u_N, v_N) \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} J_2(u_1, v_1) \\ \vdots \\ J_2(u_N, v_N) \end{pmatrix}}_B \begin{pmatrix} v_x \\ v_y \\ v_z \\ \omega_z \end{pmatrix}. \end{aligned} \quad (37)$$

If $N > 2$ and the matrix B is nonsingular, we can solve for the required translational and yaw velocity to move the vehicle to a pose where the feature points have the desired image coordinates (u_i^*, v_i^*) . The desired velocity is input to a control system as discussed earlier. This is an example of image-based visual servoing for an underactuated vehicle, and the technique can be applied to a wider variety of problems, such as holding station, path following, obstacle avoidance, and landing.

Conclusions

In this article, we have provided a tutorial introduction to modeling, estimation, and control for multirotor aerial vehicles, with a particular focus on the most common form—the quadrotor. The dynamic model includes the rigid body motion of the vehicle in SE(3), the simple aerodynamics associated with hover, and the extension to the case of forward motion where blade flapping becomes important. State estimation based on accelerometers, gyroscopes, and magnetometers was discussed for attitude and translational velocity, and GPS, motion-capture systems, and cameras for position estimation. A hierarchy of control techniques was discussed, from the individual rotors through attitude control, aggressive trajectory following, and image-based visual control. The future possibilities of highly agile small-scale vehicles were laid with a discussion on dimensional scaling for which vision will be an important sensing modality.

Acknowledgment

This research was partly supported by the Australian Research Council through Future Fellowship FT0991771,

Foundations of Vision Based Control of Robotic Vehicles, the U.S. Army Research Laboratory Grant W911NF-08-2-0004, and the U.S. Office of Naval Research Grants N00014-07-1-0829, N00014-09-1-1051, and N00014-08-1-0696.

References

- [1] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Berlin: Springer-Verlag, 2011.
- [2] T. Hamel, R. Mahony, R. Lozano, and J. Ostrowski, "Dynamic modeling and configuration stabilization for an X4-flyer," in *Proc. Int. Federation of Automatic Control Symp. (IFAC)*, 2002, p. 6.
- [3] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Apr. 26–May 1, 2004, vol. 5, pp. 4393–4398.
- [4] R. W. Prouty, *Helicopter Performance, Stability and Control*. Melbourne, FL: Krieger, 1995 (reprint with additions, original edition 1986).
- [5] J. Leishman. (2000). *Principles of Helicopter Aerodynamics* (Cambridge Aerospace Series). Cambridge, MA: Cambridge Univ. Press. [Online]. Available: <http://books.google.com.au/books?id=nMV-TkaX-9cC>
- [6] H. Huang, G. Hoffmann, S. Waslander, and C. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2009, pp. 3277–3282.
- [7] P. Martin and E. Salaun, "The true role of accelerometer feedback in quadrotor control," in *Proc. IEEE Int. Conf. Robotics and Automation*, Anchorage, AK, May 2010, pp. 1623–1629.
- [8] P. Corke. Robotics toolbox for MATLAB. (2012). [Online]. Available: <http://www.petercorke.com/robot>.
- [9] C. H. Wolowicz, J. S. Bowman, and W. P. Gilbert, "Similitude requirements and scaling relationships as applied to model testing," NASA, Tech. Rep. 1435, Aug. 1979.
- [10] S. Shen, N. Michael, and V. Kumar, "3D estimation and control for autonomous flight with constrained computation," in *Proc. IEEE Int. Conf. Robotics and Automation*, Shanghai, China, May 2011, p. 6.
- [11] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 664–674, Apr. 2012.
- [12] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Non-linear complementary filters on the special orthogonal group," *IEEE Trans. Automat. Contr.*, vol. 53, no. 5, pp. 1203–1218, June 2008.
- [13] S. Bonnabel, P. Martin, and P. Rouchon, "Non-linear symmetry-preserving observers on lie groups," *IEEE Trans. Automat. Contr.*, vol. 54, no. 7, pp. 1709–1713, 2009.
- [14] A. Bachrach, S. Prentice, R. He, and N. Roy, "Range-robust autonomous navigation in GPS-denied environments," *J. Field Robot.*, vol. 28, no. 5, pp. 644–666, 2011.
- [15] O. Amidi, T. Kanade, and R. Miller, "Vision-based autonomous helicopter research at Carnegie Mellon Robotics Institute," in *Proc. Heli Japan*, 1998, vol. 98, p. 6.
- [16] P. Corke, "An inertial and visual sensing system for a small autonomous helicopter," *J. Robot. Syst.*, vol. 21, no. 2, pp. 43–51, Feb. 2004.
- [17] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer," in *Proc. ICRA*, 2011, p. 6.
- [18] C. Kemp, "Visual control of a miniature quad rotor helicopter," Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 2006.
- [19] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *Proc. SPIE Unmanned Systems Technology XI*. Orlando, FL, SPIE, 2009, vol. 7332, p. 10.
- [20] P. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the AR. drone micro UAV," in *Proc. World Congress*, 2011, vol. 18, no. 1, pp. 1477–1484.
- [21] V. Srinivasan and S. Venkatesh, *From Living Eyes to Seeing Machines*. London, U.K.: Oxford Univ. Press, 1997.
- [22] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," *Int. J. Robot. Res.*, vol. 26, no. 6, pp. 519–536, June 2007.
- [23] T. Lee, M. Leok, and N. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proc. IEEE Conf. Decision and Control*, 2010, p. 6.
- [24] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China, May 2011, p. 6.
- [25] M. J. V. Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. J. Robust and Nonlinear Control*, vol. 8, no. 11, pp. 995–1020, 1998.
- [26] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: The Task Function Approach* (The Oxford Engineering Science Series). Oxford, U.K.: Oxford Univ. Press, 1991.
- [27] T. Hamel and R. Mahony, "Visual servoing of an under-actuated dynamic rigid-body system: An image based approach," *IEEE Trans. Robot. Automat.*, vol. 18, no. 2, pp. 187–198, Apr. 2002.
- [28] N. Guenard, T. Hamel, and R. Mahony, "A practical visual servo control for an unmanned aerial vehicle," *IEEE Trans. Robot.*, vol. 24, no. 2, pp. 331–341, Apr. 2008.
- [29] R. Mahony, P. Corke, and T. Hamel, "Dynamic image-based visual servo control using centroid and optic flow features," *J. Dynamic. Syst., Meas. Contr.*, vol. 130, no. 1, p. 12, Jan. 2008.
- [30] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, Oct. 1996.
- [31] P. I. Corke, "Spherical image-based visual servo and structure estimation," in *Proc. IEEE Int. Conf. Robotics and Automation*, Anchorage, AK, May 2010, pp. 5550–5555.

Robert Mahony, Research School of Engineering, Australian National University, Canberra 0200, Australia. E-mail: Robert.Mahony@anu.edu.au

Vijay Kumar, Department of Mechanical Engineering and Applied Mechanics, GRASP Laboratory, University of Pennsylvania, Philadelphia, USA. E-mail: vijay.kumar@grasp.upenn.edu.

Peter Corke, School of Electrical Engineering and Computer Science, Queensland University of Technology, Australia. E-mail: Peter.Corke@qut.edu.au.

4

Trajectory Following

In this chapter we discuss trajectory following algorithms for aircraft.

In particular, given an inertially defined trajectory

$$\mathcal{T}(t) = \{\mathbf{p}_{d/i}^i(t), R_d^i(t)\}$$

where \mathcal{F}_d is the “desired” frame, and where $\mathbf{p}_{d/i}^i(t) \in \mathbb{R}^3$ is the desired trajectory of the center of the frame, and $R_d^i(t) \in SO(3)$ is the desired attitude. The goal is to select the throttle T , and the torque $\boldsymbol{\tau}^b$ so that the multirotor follows the desired trajectory, i.e.,

$$\begin{aligned}\|\mathbf{p}_{b/i}^i(t) - \mathbf{p}_{d/i}^i\| &\rightarrow 0, \\ R_d^{i\top} R_b^i &\rightarrow I.\end{aligned}$$

For completeness we will discuss several options for trajectory stabilization.

4.1 A Primer on Lyapunov Stability Theory

This section provides review of concepts from Lyapunov theory that will be needed to understand the developments in this book, and this chapter in particular. An excellent resource for this material is (Khalil, 1992)¹. Given the controlled nonlinear systems

$$\dot{x} = \bar{f}(x, u),$$

suppose that the control strategy is given by $u = k(x)$, then the system can be written as

$$\dot{x} = \bar{f}(x, k(x)) = f(x).$$

We say that x_e is an equilibrium of the system if $f(x_e) = 0$.

Let \mathcal{X} be the state space, and let $\Omega \subset \mathcal{X}$. If $x_e \in \Omega$, a function $V : \Omega \rightarrow \mathbb{R}$ is said to be *positive definite* on Ω if $V(x) > 0$ for all $x \in \Omega \setminus \{x_e\}$ and $V(x_e) = 0$. The function V is said to be *positive semi-definite* on Ω if $V(x) \geq 0$ for all $x \in \Omega$. Define the level set

$$\Omega_c = \{x \in \mathcal{X} \mid V(x) \leq 0\}.$$

¹ Hassan K Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2002

Theorem 4.1.1 (Lyapunov Theorem) *If $c > 0$ is selected so that Ω_c is closed and bounded, and if $\dot{V}(t)$ is negative definite on Ω_c , then $\|x(t)\| \rightarrow 0$ as $t \rightarrow \infty$.*

As a simple example, let $x \in \mathbb{R}$, and let $\dot{x} = \sin(u)$, where $u = -kx$. Note that $\sin(-k \cdot 0) = 0$ and so $x_e = 0$ is an equilibrium of the system. Define the Lyapunov function candidate $V(x) = \frac{1}{2}x^2$, which results in

$$\dot{V} = x\dot{x} = x \sin(-kx) = -x \sin(kx)$$

which is less than zero if $x \in (-\pi, \pi)$. Therefore, pick any $0 < c < \pi$, and let $\Omega_c = \{x \in \mathbb{R} \mid \frac{1}{2}x^2 \leq c\}$. Clearly Ω_c is closed and bounded, and $\dot{V} < 0$ for all $x \in \Omega_c$. Therefore Theorem 4.1.1 guarantees that $|x(t)| \rightarrow 0$.

However, there are many relatively straightforward controlled systems for which standard Lyapunov theory is deficient. For example, consider the second order double integrator system $\ddot{y} = u$, where the control input is the standard feedforward plus proportional-derivative control

$$u = \ddot{y}_d + k_p(y_d - y) + k_d(\dot{y}_d - \dot{y}).$$

It is well known that this control law ensures that $y(t) \rightarrow y_d(t)$. Suppose that we would like to demonstrate this fact using Theorem 4.1.1. Let $\tilde{y} = y - y_d$, and note that

$$\ddot{\tilde{y}} = \ddot{y} - \ddot{y}_d = u - \ddot{y}_d = -k_p\tilde{y} - k_d\dot{\tilde{y}}. \quad (4.1)$$

Pick the Lyapunov function candidate

$$V(\tilde{y}, \dot{\tilde{y}}) = \frac{k_p}{2}\tilde{y}^2 + \frac{1}{2}\dot{\tilde{y}}^2,$$

which is positive definite if $k_p > 0$. Differentiating we get

$$\begin{aligned} \dot{V} &= k_p\tilde{y}\dot{\tilde{y}} + \dot{\tilde{y}}\dot{\tilde{y}} \\ &= \dot{\tilde{y}}(k_p\tilde{y} - k_p\tilde{y} - k_d\dot{\tilde{y}}) \\ &= -k_d\dot{\tilde{y}}^2. \end{aligned}$$

If $k_d > 0$, then \dot{V} is negative semi-definite since $\dot{\tilde{y}}$ can be nonzero when $\dot{V} = 0$. Letting

$$\Omega_c = \{(\tilde{y}, \dot{\tilde{y}}) \mid V(\tilde{y}, \dot{\tilde{y}}) \leq c\}$$

we note that Ω_c is closed and bounded for any finite c , and that $\dot{V}(x) \leq 0$ on Ω_c . However, Theorem 4.1.1 does not guarantee that $(\tilde{y}(t), \dot{\tilde{y}}(t)) \rightarrow (0, 0)$ because V is not negative definite. Fortunately, the well known LaSalle Invariance Theorem provides an additional tool that helps in this situation. Before stating the theorem, we need two definitions.

Definition 4.1.2 *The set Ω is said to be positively invariant with respect to the dynamics $\dot{x} = f(x)$ if when solutions $x(t)$ enter Ω , they remain in Ω thereafter.*

Definition 4.1.3 *The set Ω is said to be invariant with respect to the dynamics $\dot{x} = f(x)$ the solution begin in Ω at time s , i.e., $x(s) \in \Omega$ implies that $x(t) \in \Omega$ for all time $t \in (-\infty, \infty)$. In other words, the only way for $x(t)$ to be in Ω is if it started there.*

Theorem 4.1.4 (LaSalle Invariance Theorem) . Let $V : \mathcal{X} \mapsto \mathbb{R}$, and select $0 < c < \infty$ so that Ω_c is closed and bounded, and suppose that $\dot{V}(x) \leq 0$ for all $x \in \Omega_c$. Define $E = \Omega_c \cap \{x \in \Omega_c \mid \dot{V}(x) = 0\}$, and define $M \subset E$ to be the set of all invariant points in E . Then all solutions of the differential equation $\dot{x} = f(x)$ approach M asymptotically as $t \rightarrow \infty$.

As an example, consider the previous double integrator problem, where we have shown that $\dot{V} = -k_d \ddot{y}^2 \leq 0$ for all $(\tilde{y}, \dot{\tilde{y}}) \in \Omega_c$, where c is any finite number. Therefore

$$E = \Omega_c \cap \{(\tilde{y}, 0) \mid \tilde{y} \in \mathbb{R}\}.$$

LaSalle's invariance theorem shows that all trajectories $(\tilde{y}(t), \dot{\tilde{y}}(t)) \rightarrow M$, where M is the set of invariant points in E . We will now show that $M = \{(0, 0)\}$. Indeed, if the trajectory $(\tilde{y}(t), \dot{\tilde{y}}(t)) \in M \subset E$, then we know that $\dot{\tilde{y}}(s) = 0$ for all $s \in (-\infty, \infty)$. Therefore $\ddot{\tilde{y}}(s) = 0$ for all $s \in (-\infty, \infty)$. From Equation (4.1) we have that $k_d \ddot{\tilde{y}}(s) = 0$ for all $s \in (-\infty, \infty)$. Since $k_d > 0$ this implies that $\tilde{y}(s) = 0$ for all $s \in (-\infty, \infty)$. Therefore $M = \{(0, 0)\}$ and we have shown that $\|y(t) - y_d(t)\| \rightarrow 0$ as $t \rightarrow \infty$.

4.2 Angular Rate Regulation

We begin by discussing angular rate stabilization. Recall from Equations (3.14) that the angular dynamics of the multirotor are given by

$$J\dot{\omega}_{b/i}^b = -\omega_{b/i}^b \times (J\omega_{b/i}^b) + \tau^b. \quad (4.2)$$

Given a desired angular velocity $\omega_{d/i}^b$ expressed in the body frame, the objective is to select the torque τ^b so that

$$\omega_{b/i}^b \rightarrow \omega_{d/i}^b.$$

Theorem 4.2.1 *If the desired angular velocity is given as $\omega_{d/i}^b(t)$ and is differentiable, and the torque is given by*

$$\tau^b = \omega_{b/i}^b \times (J\omega_{b/i}^b) + J\dot{\omega}_{d/i}^b - JK \left(\omega_{b/i}^b - \omega_{d/i}^b \right), \quad (4.3)$$

where $K = K > 0$, then $\|\omega_{b/i}^b(t) - \omega_{d/i}^b(t)\| \rightarrow 0$.

Proof: Define

$$\tilde{\omega} \triangleq \omega_{b/i}^b - \omega_{d/i}^b,$$

and define the Lyapunov function candidate as

$$V = \frac{1}{2} \tilde{\omega}^\top J \tilde{\omega}.$$

Differentiating we obtain

$$\begin{aligned} \dot{V} &= \tilde{\omega}^\top J \dot{\tilde{\omega}} \\ &= \tilde{\omega}^\top (J \omega_{b/i}^b - J \omega_{d/i}^b) \\ &= \tilde{\omega}^\top (-\omega_{b/i}^b \times (J \omega_{b/i}^b) + \tau^b - J \dot{\omega}_{d/i}^b). \end{aligned}$$

Selecting the torque as in Equation (4.3), gives

$$\dot{V} = -\tilde{\omega}^\top K \tilde{\omega},$$

implying from Theorem 4.1.1, that $\|\tilde{\omega}\| \rightarrow 0$. \blacksquare

Note that the desired angular velocity in Theorem 4.2.1 is given in the body frame. Also note that control strategy (4.3) requires knowledge of the inertia matrix J .

4.3 Attitude Stabilization Using Euler Angles

In near hover conditions, the attitude is adequately represented using the Euler angles, and the dynamics are given by

$$\begin{aligned} \dot{\Theta} &= S(\Theta) \omega_{b/i}^b \\ J \dot{\omega}_{b/i}^b &= -\omega_{b/i}^b \times (J \omega_{b/i}^b) + \tau^b, \end{aligned}$$

where RWB: Need to add the derivation of the top equation to the previous chapter.

$$S(\Theta) = \begin{pmatrix} 1, & \sin \phi \tan \theta, & \cos \phi \tan \theta \\ 0, & \cos \phi, & -\sin \phi \\ 0, & \sin \phi \sec \theta, & \cos \phi \sec \theta \end{pmatrix}$$

is well defined and invertible for small Θ . When the roll angle ϕ and the pitch angle θ are small, we have that $S(\Theta) \approx I$, implying that $\dot{\Theta} \approx \omega_{b/i}^b$. Assuming also that $\omega_{b/i}^b$ is small enough that $-\omega_{b/i}^b \times (J \omega_{b/i}^b)$ can be neglected gives

$$J \ddot{\Theta} = \tau^b.$$

Given the desired attitude as $\dot{\Theta}_d(t)$, the objective is design τ^b so that $\Theta(t) \rightarrow \Theta_d(t)$.

Theorem 4.3.1 Given the desired Euler angles $\Theta^d(t)$, and assuming that $\dot{\Theta}^d$ and $\ddot{\Theta}^d$ are well defined, the control law

$$\boldsymbol{\tau}^b = J\ddot{\Theta}^d - K_p(\Theta - \Theta^d) - K_v(\dot{\Theta} - \dot{\Theta}^d), \quad (4.4)$$

where $K_p = K_p^\top > 0$ and $K_v = K_v^\top > 0$, then $\Theta(t) \rightarrow \Theta^d(t)$ and $\dot{\Theta}(t) \rightarrow \dot{\Theta}^d(t)$.

Proof: Define

$$\tilde{\Theta} \stackrel{\triangle}{=} \Theta - \Theta^d,$$

and define the Lyapunov function

$$V(\tilde{\Theta}, \dot{\tilde{\Theta}}) = \frac{1}{2}\tilde{\Theta}^\top K_p \tilde{\Theta} + \frac{1}{2}\dot{\tilde{\Theta}}^\top J\dot{\tilde{\Theta}}.$$

Differentiating V gives

$$\begin{aligned} \dot{V} &= \dot{\tilde{\Theta}}^\top K_p \tilde{\Theta} + \dot{\tilde{\Theta}}^\top (J\ddot{\Theta} - J\ddot{\Theta}^d) \\ &= \dot{\tilde{\Theta}}^\top (K_p \tilde{\Theta} + \boldsymbol{\tau}^b - J\ddot{\Theta}^d). \end{aligned}$$

Using Equation (4.4) gives

$$\dot{V} = -\dot{\tilde{\Theta}}^\top K_v \dot{\tilde{\Theta}}.$$

Define the set $\Omega_c = \{(\tilde{\Theta}, \dot{\tilde{\Theta}}) \mid V(\tilde{\Theta}, \dot{\tilde{\Theta}}) < c\}$ and note that if the dynamics are satisfied, then all trajectories that start in Ω_c remain in Ω_c . Define

$$E = \Omega_c \cap \{(\tilde{\Theta}, \dot{\tilde{\Theta}}) \mid \dot{V}(\tilde{\Theta}, \dot{\tilde{\Theta}}) = 0\},$$

and let M be the largest invariant set in E . Then for all trajectories in M we must have that $\dot{\tilde{\Theta}} \equiv 0$, which is true if and only if $\ddot{\tilde{\Theta}} \equiv 0$. Therefore $J\ddot{\tilde{\Theta}} = \boldsymbol{\tau}^b - J\ddot{\Theta}^d \equiv 0$. From Equation (4.4) we get that $K_p \tilde{\Theta} \equiv 0$, and therefore that $M = \{(\tilde{\Theta}, \dot{\tilde{\Theta}}) \mid \tilde{\Theta} = 0, \dot{\tilde{\Theta}} = 0\}$, and Theorem 4.1.4 implies that all trajectories that start in Ω_c converge to M . \blacksquare

Note that Equation (4.4) is a standard feedforward plus proportional-derivative feedback control law.

For many multirotor systems, the low-level autopilot is providing rate feedback using a strategy similar to Theorem 4.2.1. In that case, if we assume that the inner rate stabilization loop is much faster than the attitude kinematics, then we can assume that $\omega_{b/i}^b = \omega_c$, where ω_c is the commanded angular velocity, which implies that the equations of motion are simply

$$\dot{\Theta} = \omega_c,$$

leading to the following simplified strategy for attitude control.

Theorem 4.3.2 Given the desired Euler angles $\Theta^d(t)$, and assuming that $\dot{\Theta}^d$ is well defined, the control law

$$\omega_c = \dot{\Theta}^d - K_p(\Theta - \Theta^d), \quad (4.5)$$

where $K_p = K_p^\top > 0$, then $\Theta(t) \rightarrow \Theta^d(t)$.

Proof: Define

$$\tilde{\Theta} \triangleq \Theta - \Theta^d,$$

and define the Lyapunov function

$$V = \frac{1}{2}\tilde{\Theta}^\top\tilde{\Theta}.$$

Differentiating V gives

$$\begin{aligned} \dot{V} &= \tilde{\Theta}^\top (\dot{\Theta} - \dot{\Theta}^d) \\ &= \tilde{\Theta}^\top (\omega_c - \dot{\Theta}^d). \end{aligned}$$

Using Equation (4.5) gives

$$\dot{V} = -\tilde{\Theta}^\top K_p \tilde{\Theta}.$$

From Theorem 4.1.1 we have that $\|\tilde{\Theta}\| \rightarrow 0$. ■

Note that Equation (4.5) is a standard feedforward plus proportional feedback control strategy.

4.4 Attitude Stabilization Using Rotation Matrices

This section derives a control strategy for attitude stabilization using a purely geometric approach. We assume the attitude dynamics given by

$$\dot{R}_b^i = R_b^i \left[\omega_{b/i}^b \right]_\times \quad (4.6)$$

$$J\dot{\omega}_{b/i}^b = -\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b. \quad (4.7)$$

Recall from the Rodrigues formula (2.25) that given a unit vector \mathbf{n} and a scalar θ , the associated rotation matrix is

$$\exp(\theta\mathbf{n}) = R(\theta\mathbf{n}) \doteq I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2.$$

When $\theta = \pi$ we have

Recall that $[\mathbf{n}]_\times^2 = I - \mathbf{n}\mathbf{n}^\top$.

$$\begin{aligned} \exp(\pi\mathbf{n}) &= I + \sin \pi [\mathbf{n}]_\times + (1 - \cos \pi) [\mathbf{n}]_\times^2 \\ x &= I + 2(\mathbf{n}\mathbf{n}^\top - I) \\ &= 2\mathbf{n}\mathbf{n}^\top - I. \end{aligned}$$

Define the set

$$U = \{I\} \cup \left\{ 2\mathbf{n}\mathbf{n}^\top - I \mid \|\mathbf{n}\| = 1 \right\}$$

to be the set consisting of the identity matrix, and the set of all rotations by an angle of π .

Lemma 4.4.1 If $R \in SO(3)$, then $\mathbb{P}_a(R) = 0$ if and only if $R \in U$.²

Proof: Using the Rodrigues formula, the rotation matrix R can be expressed as

$$R = \exp(\theta \mathbf{n}) = I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2, \quad (4.8)$$

for some θ and unit vector \mathbf{n} . Therefore

$$\begin{aligned} \mathbb{P}_a(R) &= \frac{1}{2}(R - R^\top) \\ &= \frac{1}{2}(I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2 \\ &\quad - I - \sin \theta [\mathbf{n}]_\times^\top - (1 - \cos \theta)([\mathbf{n}]_\times^2)^\top) \\ &= 2 \sin \theta [\mathbf{n}]_\times \end{aligned}$$

where we have used the fact that $[\mathbf{n}]_\times^\top = -[\mathbf{n}]_\times$. If $R \in U$, then $\theta = \pi$ and $\mathbb{P}_a(R) = 0$. Alternatively, if $\mathbb{P}_a(R) = 0$ then $\sin \theta = 0$ implying that $\theta = m\pi$, where $m = 0, \pm 1, \pm 2, \dots$. Therefore from Equation (4.8) we have that

$$R = R(m\pi \mathbf{n}) = I + 2(\mathbf{n}\mathbf{n}^\top - I),$$

which establishes the result. \blacksquare

We will also need the following result.

Lemma 4.4.2 If $R \in SO(3)$, then

$$0 \leq \frac{1}{2} \text{tr}(I - R) \leq 2. \quad (4.9)$$

Furthermore $\frac{1}{2} \text{tr}(I - R) = 0$ if and only if $R = I$, and $\frac{1}{2} \text{tr}(I - R) = 2$ if and only if $R \in \{2\mathbf{n}\mathbf{n}^\top - I \mid \|\mathbf{n}\| = 1\}$.

Proof: Using the Rodrigues formula

$$\begin{aligned} \frac{1}{2} \text{tr}(I - R) &= \frac{1}{2} \text{tr} \left(I - (I + \sin \theta [\mathbf{n}]_\times + (1 - \cos \theta) [\mathbf{n}]_\times^2) \right) \\ &= \frac{1}{2} \text{tr} \left(-\sin \theta [\mathbf{n}]_\times - (1 - \cos \theta)(\mathbf{n}\mathbf{n}^\top - I) \right) \\ &= \frac{1}{2} \text{tr} \left((1 - \cos \theta)(I - \mathbf{n}\mathbf{n}^\top) \right) \\ &= \frac{1}{2}(1 - \cos \theta)(\text{tr}(I) - \text{tr}(\mathbf{n}\mathbf{n}^\top)) \\ &= \frac{1}{2}(1 - \cos \theta)(3 - 1) \\ &= 1 - \cos \theta. \end{aligned} \quad (4.10)$$

Clearly Equation (4.9) holds. If $R = I$, then $\theta = 0$ and $\frac{1}{2} \text{tr}(I - R) = 0$. Conversely if $\frac{1}{2} \text{tr}(I - R) = 0$ then $\theta = 2\pi m$, where m is even.

In that case, the Rodrigues formula implies that $R = I$. On the other hand, Equation (4.10) implies that $\frac{1}{2} \text{tr}(I - R) = 2$ if and only if $\theta = \pi$, which from Rodrigues formula is true if and only if $R \in \{2\mathbf{n}\mathbf{n}^\top - I \mid \|\mathbf{n}\| = 1\}$. \blacksquare

Theorem 4.4.3 Suppose that the attitude dynamics are given by

$$\dot{R}_b^i = R_b^i \left[\omega_{b/i}^b \right]_\times \quad (4.11)$$

$$J\dot{\omega}_{b/i}^b = -\omega_{b/i}^\times J\omega_{b/i}^b + \tau^b, \quad (4.12)$$

and that the commanded attitude and angular velocity satisfy

$$\dot{R}_d^i = R_d^i \left[\omega_{d/i}^d \right]_\times$$

where $\dot{\omega}_{d/i}^d$ is continuous. Define

$$\begin{aligned} R_d^b &= R_b^{i\top} R_d^i, \\ \tilde{\omega} &= \omega_{b/i}^b - R_d^b \omega_{d/i}^d \end{aligned}$$

and define

$$\begin{aligned} V(R_d^b, \tilde{\omega}) &\triangleq \frac{k_p}{2} \text{tr}(I - R_d^b) + \frac{1}{2} \tilde{\omega}^\top J \tilde{\omega} \\ \Omega_c &\triangleq \left\{ (R_d^b, \tilde{\omega}) \mid V(R_d^b, \omega) \leq c \right\}, \end{aligned}$$

where $k_p > 0$ and $0 < c < 2k_p$. If the initial conditions satisfy $(R_d^b(0), \omega(0)) \in \Omega_c$ and if the torque is given by

$$\tau^b = \omega_{b/i}^b \times J\omega_{b/i}^b + J \left(-\left[\omega_{b/i}^b \right]_\times R_d^b \omega_{d/i}^d + R_d^b \dot{\omega}_{d/i}^d \right) + k_p \mathbb{P}_a(R_d^b)^\vee - K_d \tilde{\omega} \quad (4.13)$$

where $K_d = K_d^\top > 0$, then $(R_d^b(t), \tilde{\omega}(t)) \rightarrow (I, \mathbf{0})$ asymptotically.

Proof: It is clear that $V(R_d^b, \tilde{\omega})$ is positive definite and radially unbounded in $\tilde{\omega}$. Differentiating V we get

$$\begin{aligned} \dot{V} &= -\frac{k_p}{2} \text{tr}(\dot{R}_b^{i\top} R_d^i + R_b^{i\top} \dot{R}_d^i) + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \\ &= -\frac{k_p}{2} \text{tr} \left(-\left[\omega_{b/i}^b \right]_\times R_b^{i\top} R_d^i + R_b^{i\top} R_d^i \left[\omega_{d/i}^d \right]_\times \right) + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \\ &= -\frac{k_p}{2} \text{tr} \left(-\left[\omega_{b/i}^b \right]_\times R_d^b + R_d^b \left[\omega_{d/i}^d \right]_\times R_d^{b\top} R_d^b \right) + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \\ &= -\frac{k_p}{2} \text{tr} \left(-\left[\omega_{b/i}^b \right]_\times R_d^b + \left[R_d^b \omega_{d/i}^d \right]_\times R_d^b \right) + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \\ &= -\frac{k_p}{2} \text{tr}(-[\tilde{\omega}]_\times R_d^b) + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b), \end{aligned}$$

where we note that

$$\begin{aligned} \omega_{d/i}^b &= \frac{d}{dt} \left(R_b^{i\top} R_d^i \omega_{d/i}^d \right) \\ &= \dot{R}_b^{i\top} R_d^i \omega_{d/i}^d + R_b^{i\top} \dot{R}_d^i \omega_{d/i}^d + R_b^{i\top} R_d^i \dot{\omega}_{d/i}^d \\ &= -\left[\omega_{b/i}^b \right]_\times R_b^{i\top} R_d^i \omega_{d/i}^d + R_b^{i\top} R_d^i \left[\omega_{d/i}^d \right]_\times \omega_{d/i}^d + R_b^{i\top} R_d^i \dot{\omega}_{d/i}^d \\ &= -\left[\omega_{b/i}^b \right]_\times R_b^{i\top} R_d^i \omega_{d/i}^d + R_b^{i\top} R_d^i \dot{\omega}_{d/i}^d. \end{aligned}$$

Using the fact that $R_d^b = \mathbb{P}_s(R_d^b) + \mathbb{P}_a(R_d^b)$ we get

$$\begin{aligned}\dot{V} &= -\frac{k_p}{2} \text{tr} \left(-[\tilde{\omega}]_\times (\mathbb{P}_s(R_d^b) + \mathbb{P}_a(R_d^b)) \right. \\ &\quad \left. + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \right) \\ \dot{V} &= -\frac{k_p}{2} \text{tr} \left(-[\tilde{\omega}]_\times \mathbb{P}_a(R_d^b) \right) + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \\ \dot{V} &= -\tilde{\omega}^\top k_p \mathbb{P}_a(R_d^b)^\vee + \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b) \\ \dot{V} &= \tilde{\omega}^\top (-\omega_{b/i}^b \times J\omega_{b/i}^b + \tau^b - J\dot{\omega}_{d/i}^b - k_p \mathbb{P}_a(R_d^b)^\vee),\end{aligned}$$

where we have used property (2.21) to obtain the second line, and property (2.22) to obtain the third line. Using the torque in Equation (4.13) gives

$$\dot{V} = -\tilde{\omega}^\top K\tilde{\omega},$$

which is negative semi-definite. Define the sets

$$E = \Omega_c \cap \{(R_d^b, \tilde{\omega}) \mid \tilde{\omega}^\top K\tilde{\omega} = 0\}$$

and let M be the largest invariant set in E . Since the system starts in Ω_c , and $\dot{V} \leq 0$ in Ω_c , all system trajectories remain in Ω_c . For trajectories in M , we must have that $\tilde{\omega} \equiv 0$ which implies that $J\dot{\tilde{\omega}} \equiv 0$, which implies that

$$\begin{aligned}J(\omega_{b/i}^b - \dot{\omega}_{d/i}^b) &= -\omega_{b/i}^b \times J\omega_{b/i}^b + \tau - J\dot{\omega}_{d/i}^b \\ &= -k_p \mathbb{P}_a(R_d^b)^\vee - K_d \tilde{\omega} \\ &= -k_p \mathbb{P}_a(R_d^b)^\vee \\ &\equiv 0.\end{aligned}$$

Therefore for all trajectories in M we have that $\mathbb{P}_a(R_d^b) \equiv 0$, which implies that $M = \{I\} \times \{0\}$ from Lemmas 4.4.1 and 4.4.2. The LaSalle's invariance theorem therefore guarantees that trajectories starting in Ω_c approach M asymptotically. ■

For many multirotor systems, the low-level autopilot is providing rate feedback using a strategy similar to Theorem 4.2.1. In that case, if we assume that the inner rate stabilization loop is much faster than the attitude kinematics, then we can assume that $\omega_{b/i}^b = \omega_c^b$, where ω_c^b is the commanded angular velocity expressed in the body frame, which implies that the rotational equations of motion are simply

$$\dot{R}_b^i = R_b^i \left[\omega_c^b \right]_\times \quad (4.14)$$

leading to the following simplified strategy for attitude control.

Recall that:

$$\begin{aligned}[\omega]_\times \omega &= \omega \times \omega = 0 \\ A = A^\top, B = -B^\top &\implies \text{tr}(AB) = 0 \\ \text{tr}([\mathbf{a}]_\times [\mathbf{b}]_\times) &= -2\mathbf{a}^\top \mathbf{b}\end{aligned}$$

Theorem 4.4.4 Suppose that the attitude dynamics are given by Equation (4.14), and that the desired attitude and angular velocity satisfy

$$\dot{R}_d^i = R_d^i \left[\omega_{d/i}^d \right]_\times.$$

Suppose that the initial conditions satisfy

$$\frac{1}{2} \text{tr} \left(I - R_b^{i\top}(0) R_d^i(0) \right) < 2$$

and that the commanded angular velocity satisfies

$$\omega_c^b = R_b^{i\top} R_d^i \omega_{d/i}^d + K_p \mathbb{P}_a(R_b^{i\top} R_d^i)^\vee \quad (4.15)$$

where $K_p = K_p^\top > 0$, then $R_b^i(t) \rightarrow R_d^i(t)$ asymptotically.

Proof: Define the Lyapunov function

$$V = \frac{1}{2} \text{tr} \left(I - R_d^b \right),$$

where $R_d^b = R_b^{i\top} R_d^i$, and take the derivative to get

$$\begin{aligned} \dot{V} &= -\frac{1}{2} \text{tr} \left(\dot{R}_d^b \right) \\ &= -\frac{1}{2} \text{tr} \left(\dot{R}_b^{i\top} R_d^i + R_b^{i\top} \dot{R}_d^i \right) \\ &= -\frac{1}{2} \text{tr} \left(- \left[\omega_c^b \right]_\times R_b^{i\top} R_d^i + R_b^{i\top} R_d^i \left[\omega_{d/i}^d \right]_\times \right) \\ &= -\frac{1}{2} \text{tr} \left(- \left[\omega_c^b \right]_\times R_b^{i\top} R_d^i + R_d^b \left[\omega_{d/i}^d \right]_\times R_d^{b\top} R_d^b \right) \\ &= -\frac{1}{2} \text{tr} \left(- \left[\omega_c^b \right]_\times R_b^{i\top} R_d^i + \left[R_d^b \omega_{d/i}^d \right]_\times R_d^b \right) \\ &= -\frac{1}{2} \text{tr} \left(\left[R_d^b \omega_{d/i}^d - \omega_c^b \right]_\times (\mathbb{P}_s(R_d^b) + \mathbb{P}_a(R_d^b)) \right) \\ &= -\frac{1}{2} \text{tr} \left(\left[R_d^b \omega_{d/i}^d - \omega_c^b \right]_\times \mathbb{P}_a(R_d^b) \right) \\ &= (R_d^b \omega_{d/i}^d - \omega_c^b)^\top \mathbb{P}_a(R_d^b)^\vee. \end{aligned}$$

Using Equation (4.15) gives

$$\dot{V} = -(\mathbb{P}_a(R_d^b)^\vee)^\top K_p \mathbb{P}_a(R_d^b)^\vee,$$

which implies that $\mathbb{P}_a(R_d^b) \rightarrow 0$. The initial conditions and Lemma 4.4.2 ensures that $R_b^{i\top} R_d^i \rightarrow I$. \blacksquare

Note that if $R_d^i(t)$ is given and $\dot{R}_d^i(t)$ is either given or computed numerically, then $\omega_{d/i}^d$ can be computed as

$$\omega_{d/i}^d(t) = \left(R_d^{i\top}(t) \dot{R}_d^i(t) \right)^\vee.$$

4.5 Attitude Stabilization Using Quaternions

4.6 Velocity controller

RWB: Add proof and flesh out section

Theorem 4.6.1 Given the dynamics

$$\dot{\mathbf{v}}_{b/i}^i = g\mathbf{e}_3 - T_c R_d^i \mathbf{e}_3$$

and the desired velocity $\mathbf{v}_{d/i}^i$ and heading direction \mathbf{s}_{ψ_d} , if the throttle and desired rotation matrix are selected as

$$\begin{aligned} T_c &= (g\mathbf{e}_3 - K_d (\mathbf{v}_{d/i}^i - \mathbf{v}_{b/i}^i))^T R_b^i \mathbf{e}_3 \\ \mathbf{r}_{3d} &= \frac{g\mathbf{e}_3 - K_d (\mathbf{v}_{d/i}^i - \mathbf{v}_{b/i}^i)}{\|g\mathbf{e}_3 - K_d (\mathbf{v}_{d/i}^i - \mathbf{v}_{b/i}^i)\|} \\ \mathbf{r}_{2d} &= \frac{\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}}{\|\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}\|} \\ \mathbf{r}_{1d} &= \mathbf{r}_{2d} \times \mathbf{r}_{3d}, \end{aligned}$$

where $K_d = K_d^\top > 0$, and where the desired angular velocity is given by

$$\boldsymbol{\omega}_{d/i}^d = (R_d^{i\top} \dot{R}_d^i)^\vee,$$

then $\|\mathbf{v}_{b/i}^i - \mathbf{v}_{d/i}^i\| \rightarrow 0$ and $\psi \rightarrow \psi_d$.

Proof: RWB: need to add, but similar to stuff we have done before. ■

4.7 Trajectory Tracking Control

Suppose that we are given a desired inertial frame trajectory given by

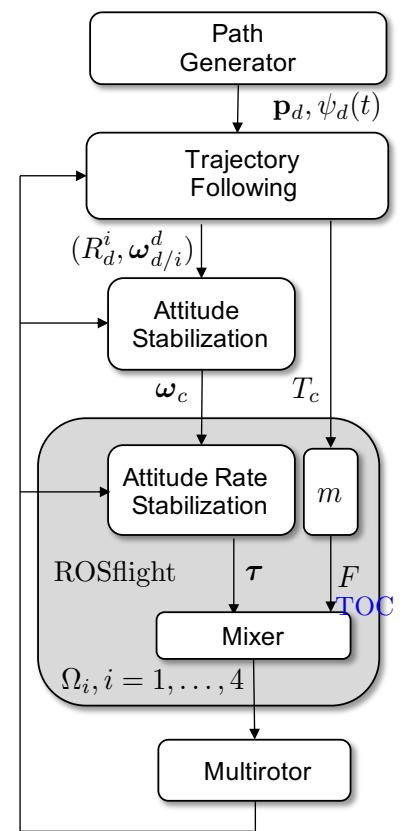
$$\mathcal{T}(t) = \{\mathbf{p}_d(t), \dot{\mathbf{p}}_d(t), \ddot{\mathbf{p}}_d(t), \psi_d(t), \dot{\psi}_d(t)\}$$

the objective is to design a control strategy so that the multicopter follows the trajectory.

4.7.1 Cascaded Control

We will first derive a control law based on the cascaded control architecture shown in Figure 4.1.

We will assume that the low-level autopilot (labeled ROSflight in Figure 4.1) implements attitude rate stabilization and accepts a commanded angular velocity $\boldsymbol{\omega}_c$, which is the commanded angular velocity of the body relative to the inertial frame, expressed in the body frame,



and the commanded throttle T_c , which is the commanded force divided by the mass m . The commanded angular rate is supplied by an attitude stabilization loop similar to that derived in Theorem 4.4.4.

In this section we derive the trajectory following algorithm, where the inputs are the desired inertial position of the multirotor \mathbf{p}_d , and the desired heading angle ψ_d , and their derivatives.

From Equations (3.11)–(3.12), the translational equations of motion are given by

$$\dot{\mathbf{p}}_{b/i}^i = \mathbf{v}_{b/i}^i \quad (4.16)$$

$$\dot{\mathbf{v}}_{b/i}^i = g\mathbf{e}_3 - T_c R_d^i \mathbf{e}_3, \quad (4.17)$$

where we have replaced the throttle T with the commanded throttle T_c , and the rotation matrix R_b^i with the desired attitude R_d^i , and where we have ignored the induced drag term. Define the position error as $\tilde{\mathbf{p}} \doteq \mathbf{p}_{b/i}^i - \mathbf{p}_d$, from which we obtain

$$\ddot{\tilde{\mathbf{p}}} = g\mathbf{e}_3 - T_c R_d^i \mathbf{e}_3 - \dot{\tilde{\mathbf{p}}}. \quad (4.18)$$

Let the desired rotation matrix be given by

$$R_d^i = \begin{pmatrix} \mathbf{r}_{1d} & \mathbf{r}_{2d} & \mathbf{r}_{3d} \end{pmatrix},$$

from which we get

$$\ddot{\tilde{\mathbf{p}}} = g\mathbf{e}_3 - T_c \mathbf{r}_{3d} - \dot{\tilde{\mathbf{p}}}. \quad (4.19)$$

The idea behind the trajectory following scheme is to set $T_c \mathbf{r}_{3d}$ to achieve PD-like following properties, and then to pick \mathbf{r}_{2d} and \mathbf{r}_{1d} to align the front of the multirotor with the direction vector

$$\mathbf{s}_{\psi_d} \doteq \begin{pmatrix} \cos \psi_d \\ \sin \psi_d \\ 0 \end{pmatrix}.$$

Toward that end, we have the following scheme.

Theorem 4.7.1 *Given the dynamics in Equation (4.16) and (4.17), and the desired trajectory given by $\mathbf{p}_d(t)$, $\dot{\mathbf{p}}_d$, $\ddot{\mathbf{p}}_d$, ψ_d , $\dot{\psi}_d$, if the throttle and desired rotation matrix are selected as*

$$T_c = \|\ddot{\tilde{\mathbf{p}}} - g\mathbf{e}_3 - K_p \tilde{\mathbf{p}} - K_d \dot{\tilde{\mathbf{p}}}\| \quad (4.19)$$

$$\mathbf{r}_{3d} = -\frac{\ddot{\tilde{\mathbf{p}}} - g\mathbf{e}_3 - K_p \tilde{\mathbf{p}} - K_d \dot{\tilde{\mathbf{p}}}}{\|\ddot{\tilde{\mathbf{p}}} - g\mathbf{e}_3 - K_p \tilde{\mathbf{p}} - K_d \dot{\tilde{\mathbf{p}}}\|} \quad (4.20)$$

$$\mathbf{r}_{2d} = \frac{\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}}{\|\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}\|} \quad (4.21)$$

$$\mathbf{r}_{1d} = \mathbf{r}_{2d} \times \mathbf{r}_{3d}, \quad (4.22)$$

where $K_p = K_p^\top > 0$ and $K_d = K_d^\top > 0$, and where the desired angular velocity is given by

$$\omega_{d/i}^d = (R_d^{i\top} \dot{R}_d^i)^\vee,$$

then $\|\mathbf{p}_{b/i}^i - \mathbf{p}_d\| \rightarrow 0$ and $\psi \rightarrow \psi_d$.

Note that $R_d^i = (\mathbf{r}_{1d}, \mathbf{r}_{2d}, \mathbf{r}_{3d})$ has been constructed so that the desired body z -axis is aligned with the desired acceleration vector, and the body x -axis has been aligned so that it is in plane defined by the desired heading direction and the desired acceleration vector.

Proof: Substituting Equations (4.19) and (4.20) into Equation (4.23) gives

$$\ddot{\mathbf{p}} = -K_p \tilde{\mathbf{p}} - K_d \dot{\tilde{\mathbf{p}}}. \quad (4.23)$$

Define the Lyapunov function

$$V(\tilde{\mathbf{p}}, \dot{\tilde{\mathbf{p}}}) = \frac{1}{2} \tilde{\mathbf{p}}^\top K_p \tilde{\mathbf{p}} + \frac{1}{2} \dot{\tilde{\mathbf{p}}}^\top \dot{\tilde{\mathbf{p}}},$$

and differentiate to get,

$$\begin{aligned} \dot{V} &= \dot{\tilde{\mathbf{p}}}^\top K_p \tilde{\mathbf{p}} + \dot{\tilde{\mathbf{p}}}^\top \ddot{\mathbf{p}} \\ &= \dot{\tilde{\mathbf{p}}}^\top (\ddot{\mathbf{p}} + K_p \tilde{\mathbf{p}}) \\ &= -\dot{\tilde{\mathbf{p}}}^\top K_d \dot{\tilde{\mathbf{p}}}. \end{aligned}$$

Note that if there are no limits on throttle, then $\Omega_c = \{(\tilde{\mathbf{p}}, \dot{\tilde{\mathbf{p}}}) \mid V \leq c\}$ can be defined by any c . However, if there are throttle limits, then Ω_c is defined by those limits through Equation (4.19). Let

$$E = \Omega_c \cap \{(\tilde{\mathbf{p}}, \dot{\tilde{\mathbf{p}}}) \mid \dot{V} = 0\} = \Omega_c \cap \{(\tilde{\mathbf{p}}, \dot{\tilde{\mathbf{p}}}) \mid \dot{\tilde{\mathbf{p}}} = 0\},$$

and let M be the set of invariant points in E . Then $(\tilde{\mathbf{p}}, \dot{\tilde{\mathbf{p}}}) \in M$ implies that $\dot{\tilde{\mathbf{p}}}(t) \equiv 0$, where the \equiv symbol is used to mean that the equality holds for all time $t \in (-\infty, \infty)$. This implies that $\ddot{\mathbf{p}} \equiv 0$, which implies from (4.23) that $\tilde{\mathbf{p}} \equiv 0$. Therefore from Theorem 4.1.4 $\|\tilde{\mathbf{p}}\| \rightarrow 0$. The attitude stabilization block ensures that $R_b^i(t) \rightarrow R_d^i(t)$, and therefore that $\psi(t) \rightarrow \psi_d(t)$. \blacksquare

Additional stuff:

Let

$$V = \frac{1}{2} \tilde{\mathbf{p}}^\top K_p \tilde{\mathbf{p}} + \frac{1}{2} \dot{\tilde{\mathbf{p}}}^\top \dot{\tilde{\mathbf{p}}}.$$

Differentiate to get

$$\dot{V} = \dot{\tilde{\mathbf{p}}}^\top [K_p \tilde{\mathbf{p}} + g \mathbf{e}_3 - T R_b^i \mathbf{e}_3 - \ddot{\mathbf{p}}_d] \dot{\tilde{\mathbf{p}}} = \dot{\tilde{\mathbf{p}}}^\top [K_p \tilde{\mathbf{p}} + g \mathbf{e}_3 - T \mathbf{r}_3 - \ddot{\mathbf{p}}_d]$$

Add and subtract $T \mathbf{r}_{3d} / (\mathbf{r}_3^\top \mathbf{r}_{3d})$ to get

$$\begin{aligned} \dot{V} &= \dot{\tilde{\mathbf{p}}}^\top \left[K_p \tilde{\mathbf{p}} + g \mathbf{e}_3 - \frac{T \mathbf{r}_{3d}}{(\mathbf{r}_3^\top \mathbf{r}_{3d})} - \ddot{\mathbf{p}}_d \left(\frac{(\frac{T \mathbf{r}_{3d}}{(\mathbf{r}_3^\top \mathbf{r}_{3d})}) T \mathbf{r}_3}{(\mathbf{r}_3^\top \mathbf{r}_{3d})} - \frac{T \mathbf{r}_{3d}}{(\mathbf{r}_3^\top \mathbf{r}_{3d})} \right) \right] \\ &= \dot{\tilde{\mathbf{p}}}^\top \left[K_p \tilde{\mathbf{p}} + g \mathbf{e}_3 - \frac{T \mathbf{r}_{3d}}{(\mathbf{r}_3^\top \mathbf{r}_{3d})} - \ddot{\mathbf{p}}_d \left(\frac{T}{(\mathbf{r}_3^\top \mathbf{r}_{3d})} (I - \mathbf{r}_3 \mathbf{r}_3^\top) \mathbf{r}_{3d} \right) \right]. \end{aligned}$$

Now pick

$$\begin{aligned}\mathbf{a}_d &= -\ddot{\mathbf{p}}_d + g\mathbf{e}_3 + K_p\tilde{\mathbf{p}} + K_d\dot{\tilde{\mathbf{p}}} \\ \mathbf{r}_{3d} &= \frac{\mathbf{a}_d}{\|\mathbf{a}_d\|} \\ T_c &= \mathbf{a}_d^\top \mathbf{r}_3\end{aligned}$$

resulting in

$$\dot{V} = -\dot{\tilde{\mathbf{p}}}^\top K_v \dot{\tilde{\mathbf{p}}} + \dot{\tilde{\mathbf{p}}}^\top (I - \mathbf{r}_3 \mathbf{r}_3^\top) \mathbf{a}_d.$$

RWB: Need to better explain the inner product with $R_b^i \mathbf{e}_3$. Worst case, follow proof of attached paper, but simplify to rotational kinematics.

4.8 Geometric Tracking Control of a Quadrotor UAV on $SE(3)$.

arXiv:1003.2005v1 [math.OC] 10 Mar 2010

Geometric Tracking Control of a Quadrotor UAV on SE(3)

Taeyoung Lee, Melvin Leok*, and N. Harris McClamroch†

Abstract—This paper provides new results for the tracking control of a quadrotor unmanned aerial vehicle (UAV). The UAV has four input degrees of freedom, namely the magnitudes of the four rotor thrusts, that are used to control the six translational and rotational degrees of freedom, and to achieve asymptotic tracking of four outputs, namely, three position variables for the vehicle center of mass and the direction of one vehicle body-fixed axis. A globally defined model of the quadrotor UAV rigid body dynamics is introduced as a basis for the analysis. A nonlinear tracking controller is developed on the special Euclidean group $\text{SE}(3)$ and it is shown to have desirable closed loop properties that are almost global. Several numerical examples, including an example in which the quadrotor recovers from being initially upside down, illustrate the versatility of the controller.

I. INTRODUCTION

A quadrotor unmanned aerial vehicle (UAV) consists of two pairs of counter-rotating rotors and propellers, located at the vertices of a square frame. It is capable of vertical take-off and landing (VTOL), but it does not require complex mechanical linkages, such as swash plates or teeter hinges, that commonly appear in typical helicopters. Due to its simple mechanical structure, it has been envisaged for various applications such as surveillance or mobile sensor networks as well as for educational purposes. There are several university-level projects [1], [2], [3], [4], and commercial products [5], [6], [7] related to the development and application of quadrotor UAVs.

Despite the substantial interest in quadrotor UAVs, little attention has been paid to constructing nonlinear control systems for them, particularly to designing nonlinear tracking controllers. Linear control systems such as proportional-derivative controllers or linear quadratic regulators are widely used to enhance the stability properties of an equilibrium [1], [3], [4], [8], [9]. A nonlinear controller is developed for the linearized dynamics of a quadrotor UAV with saturated positions in [10]. Backstepping and sliding mode techniques are applied in [11]. Since these controllers are based on Euler angles, they exhibit singularities when representing complex rotational maneuvers of a quadrotor UAV, thereby fundamentally restricting their ability to track nontrivial trajectories.

Taeyoung Lee, Mechanical and Aerospace Engineering, Florida Institute of Technology, Melbourne, FL 32901 taeyoung@fit.edu

Melvin Leok, Mathematics, University of California at San Diego, La Jolla, CA 92093 mleok@math.ucsd.edu

N. Harris McClamroch, Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109 nhm@umich.edu

*This research has been supported in part by NSF under grants DMS-0726263, DMS-100152 and DMS-1010687.

†This research has been supported in part by NSF under grant CMS-0555797.

Geometric control is concerned with the development of control systems for dynamic systems evolving on nonlinear manifolds that cannot be globally identified with Euclidean spaces [12], [13], [14]. By characterizing geometric properties of nonlinear manifolds intrinsically, geometric control techniques provide unique insights to control theory that cannot be obtained from dynamic models represented using local coordinates [15]. This approach has been applied to fully actuated rigid body dynamics on Lie groups to achieve almost global asymptotic stability [14], [16], [17], [18].

In this paper, we develop a geometric controller for a quadrotor UAV. The dynamics of a quadrotor UAV is expressed globally on the configuration manifold of the special Euclidean group $\text{SE}(3)$. We construct a tracking controller to follow prescribed trajectories for the center of mass and heading direction. It is shown that this controller exhibits almost global exponential attractiveness to the zero equilibrium of tracking errors. Since this is a coordinate-free control approach, it completely avoids singularities and complexities that arise when using local coordinates.

Compared to other geometric control approaches for rigid body dynamics, this is distinct in the sense that it controls an underactuated quadrotor UAV to stabilize six translational and rotational degrees of freedom using four thrust inputs, while asymptotically tracking four outputs consisting of its position and heading direction. We demonstrate that this controller is particularly useful for complex, acrobatic maneuvers of a quadrotor UAV, such as recovering from being initially upside down.

The paper is organized as follows. We develop a globally defined model for the translational and rotational dynamics of a quadrotor UAV in Section II. Tracking control results on $\text{SE}(3)$ are presented in Section III. Several numerical results are presented in Section IV. Proofs are relegated to the Appendix.

II. QUADROTOR DYNAMICS MODEL

Consider a quadrotor vehicle model illustrated in Fig. 1. This is a system of four identical rotors and propellers located at the vertices of a square, which generate a thrust and torque normal to the plane of this square. We choose an inertial reference frame $\{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$ and a body-fixed frame $\{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$. The origin of the body-fixed frame is located at the center of mass of this vehicle. Define

$$m \in \mathbb{R} \quad \text{the total mass}$$

$$J \in \mathbb{R}^{3 \times 3} \quad \text{the inertia matrix with respect to the body-fixed frame}$$

$$R \in \text{SO}(3) \quad \text{the rotation matrix from the body-fixed frame to the inertial frame}$$

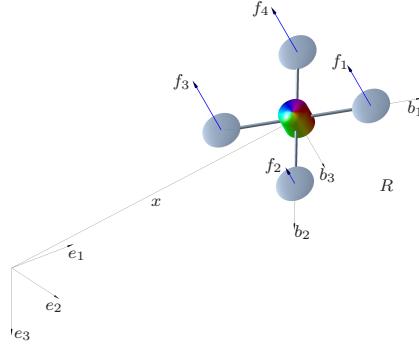


Fig. 1. Quadrotor model

$$\Omega \in \mathbb{R}^3$$

$$x \in \mathbb{R}^3$$

$$v \in \mathbb{R}^3$$

$$d \in \mathbb{R}$$

$$f_i \in \mathbb{R}$$

$$\tau_i \in \mathbb{R}$$

$$f \in \mathbb{R}$$

$$M \in \mathbb{R}^3$$

the angular velocity in the body-fixed frame
the location of the center of mass in the inertial frame

the velocity of the center of mass in the inertial frame

the horizontal distance from the center of mass to the center of a rotor

the thrust generated by the i -th propeller along the $-\vec{b}_3$ axis

the torque generated by the i -th propeller about the $-\vec{b}_3$ axis

the total thrust, i.e., $f = \sum_{i=1}^4 f_i$

the total moment in the body-fixed frame

The configuration of this quadrotor UAV is defined by the location of the center of mass and the attitude with respect to the inertial frame. Therefore, the configuration manifold is the special Euclidean group $\text{SE}(3)$.

We assume that the thrust of each propeller is directly controlled, i.e., we do not consider the dynamics of rotors and propellers. The total thrust is $f = \sum_{i=1}^4 f_i$, which acts along the direction of Re_3 in the inertial frame.

We also assume that the torque generated by each propeller is directly proportional to its thrust. Since it is assumed that the first and the third propellers rotate counterclockwise, and the second and the fourth propellers rotate clockwise, the torque generated by the i -th propeller can be written as $\tau_i = (-1)^i c_{\tau f} f_i$ for a fixed constant $c_{\tau f}$. Then, the total moment in the body-fixed frame is given by

$$M = [d(f_4 - f_2), d(f_1 - f_3), c_{\tau f}(-f_1 + f_2 - f_3 + f_4)].$$

These can be written in matrix form as

$$\begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}. \quad (1)$$

The determinant of the above 4×4 matrix is $8c_{\tau f}d^2$, so it is invertible when $d \neq 0$ and $c_{\tau f} \neq 0$. Therefore, for a given f, M , the thrust of each rotor f_i can be obtained from (1). Using this equation, the total thrust $f \in \mathbb{R}$ and the moment $M \in \mathbb{R}^3$ are considered as control inputs in this paper.

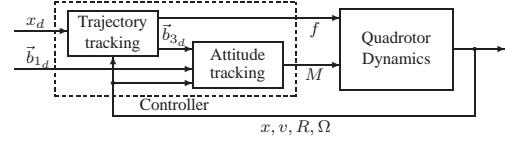


Fig. 2. Controller structure

The equations of motion of this quadrotor UAV can be written as

$$\dot{x} = v, \quad (2)$$

$$m\dot{v} = mge_3 - fRe_3, \quad (3)$$

$$\dot{R} = R\hat{\Omega}, \quad (4)$$

$$J\dot{\Omega} + \Omega \times J\Omega = M, \quad (5)$$

where the *hat map* $\hat{\cdot} : \mathbb{R}^3 \rightarrow \mathfrak{so}(3)$ is defined by the condition that $\hat{xy} = x \times y$ for all $x, y \in \mathbb{R}^3$, and $e_3 = [0; 0; 1] \in \mathbb{R}^3$.

III. GEOMETRIC TRACKING CONTROL ON $\text{SE}(3)$

We develop a controller to follow a prescribed trajectory $x_d(t)$ of the location of the center of mass, and the direction of the body-fixed axis $\vec{b}_{1_d}(t)$, which represents the yawing (or heading) angle of a quadrotor UAV. We develop this controller directly on the nonlinear configuration Lie group and thereby avoid any singularities and complexities that arise in local coordinates. As a result, we are able to achieve almost global exponential attractiveness to the zero equilibrium of tracking errors.

The overall controller structure is illustrated in Fig. 2. The equations of motion (2)-(5) have a cascade structure: the rotational motion of the attitude is decoupled from the translational motion; the translational motion is only dependent on the term fRe_3 in (3). The magnitude of the total thrust f is directly controlled, but the direction of the total thrust Re_3 is determined by the third body-fixed axis \vec{b}_3 . Therefore, in order to change the direction of the total thrust, the attitude should be changed accordingly. Here, we choose the total thrust f and the desired reduced attitude $R_d e_3$ for the third body-fixed axis \vec{b}_3 such that they stabilize the zero equilibrium of the tracking error for the translational dynamics, and the remaining first two columns of R_d representing the direction of the body-fixed axes \vec{b}_1, \vec{b}_2 , are chosen to follow a desired direction \vec{b}_{1_d} . The control moment M is designed to follow the resulting desired attitude R_d obtained by \vec{b}_{1_d} and \vec{b}_{3_d} .

A. Tracking Errors

We define the tracking errors for x, v, R, Ω as follows. The tracking errors for the position and the velocity are given by:

$$e_x = x - x_d, \quad (6)$$

$$e_v = v - v_d. \quad (7)$$

The attitude and angular velocity tracking error should be carefully chosen as they evolve on the tangent bundle of

the nonlinear space $\text{SO}(3)$. The error function on $\text{SO}(3)$ is chosen to be

$$\Psi(R, R_d) = \frac{1}{2} \text{tr}[I - R_d^T R]. \quad (8)$$

This is locally positive-definite about $R_d^T R = I$ within the region where the rotation angle between R and R_d is less than 180° [14]. This set can be represented by the sublevel set of Ψ where $\Psi < 2$, namely $L_2 = \{R_d, R \in \text{SO}(3) \mid \Psi(R, R_d) < 2\}$, which almost covers $\text{SO}(3)$. When the variation of the rotation matrix is expressed as $\delta R = R\hat{\eta}$ for $\eta \in \mathbb{R}^3$, the derivative of the error function is given by

$$\mathbf{D}_R \Psi(R, R_d) \cdot R\hat{\eta} = \frac{1}{2} (R_d^T R - R^T R_d)^\vee \cdot \eta, \quad (9)$$

where the *vee map* $\vee : \mathfrak{so}(3) \rightarrow \mathbb{R}^3$ is the inverse of the hat map. We used the fact that $-\frac{1}{2} \text{tr}[\hat{x}\hat{y}] = x^T y$ for any $x, y \in \mathbb{R}^3$. From this, the attitude tracking error is chosen to be

$$e_R = \frac{1}{2} (R_d^T R - R^T R_d)^\vee. \quad (10)$$

The tangent vectors $\dot{R} \in T_R \text{SO}(3)$ and $\dot{R}_d \in T_{R_d} \text{SO}(3)$ cannot be directly compared since they lie in different tangent spaces. We transform \dot{R}_d into a vector in $T_R \text{SO}(3)$, and we compare it with \dot{R} as follows:

$$\dot{R} - \dot{R}_d (R_d^T R) = R\hat{\Omega} - R_d\hat{\Omega}_d R_d^T R = R(\Omega - R^T R_d \Omega_d)^\wedge.$$

We choose the tracking error for the angular velocity as follows:

$$e_\Omega = \Omega - R^T R_d \Omega_d. \quad (11)$$

We can show that e_Ω is the angular velocity of the rotation matrix $R_d^T R$, represented in the body-fixed frame, since $\frac{d}{dt}(R_d^T R) = (R_d^T R)\hat{e}_\Omega$.

B. Tracking Controller

For given smooth tracking commands $x_d(t), \vec{b}_{1_d}(t)$, and some positive constants k_x, k_v, k_R, k_Ω , the control inputs f, M are chosen as follows:

$$f = -(-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d) \cdot Re_3, \quad (12)$$

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega - J(\hat{\Omega} R^T R_d \Omega_d - R^T R_d \hat{\Omega}_d), \quad (13)$$

where the desired attitude R_d is given by $R_d = [\vec{b}_{1_d}; \vec{b}_{3_d} \times \vec{b}_{1_d}; \vec{b}_{3_d}] \in \text{SO}(3)$, and

$$\vec{b}_{3_d} = R_d e_3 = -\frac{-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d}{\| -k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d \|}. \quad (14)$$

Here, we assume that the denominator of (14) is non-zero,

$$\|-k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d\| \neq 0, \quad (15)$$

and that the desired trajectory satisfies

$$\|-mge_3 + m\ddot{x}_d\| < B \quad (16)$$

for a given positive constant B .

These control inputs f, M are designed as follows. The control moment M given in (13) corresponds to a tracking

controller on $\text{SO}(3)$. For the attitude dynamics of a rigid body described by (4), (5), this controller exponentially stabilizes the zero equilibrium of the attitude tracking errors.

Similarly, the expression in the parentheses in (12) corresponds to a tracking controller for the translational dynamics on \mathbb{R}^3 . The total thrust f and the desired direction \vec{b}_{3_d} of the third body-fixed axis are chosen so that if there is no attitude tracking error, the control input term fRe_3 at the translational dynamics of (3) becomes this tracking controller in \mathbb{R}^3 . Therefore, the trajectory tracking error will converge to zero provided that the attitude tracking error is identically zero.

Certainly, the attitude tracking error may not be zero at any instant. As the attitude tracking error increases, the direction of the control input term fRe_3 of the translational dynamics deviates from the desired direction of $R_d e_3$. This may cause instability on the complete dynamics. In (12), we carefully design the total thrust f so that its magnitude is reduced when there is a larger attitude tracking error. The expression of f includes the dot product of the desired third body-fixed axis $\vec{b}_{3_d} = R_d e_3$ and the current third body-fixed axis $\vec{b}_3 = Re_3$. Therefore, the magnitude of f is reduced when the angle between those two axes becomes larger. These effects are carefully analyzed in the stability proof for the complete dynamics described in the appendix.

In short, this control system is designed so that the position tracking error converges to zero when there is no attitude tracking error, and it is properly adjusted for non-zero attitude tracking errors to achieve asymptotic stability of the complete dynamics.

C. Exponential Asymptotic Stability

We first show exponential stability of the attitude dynamics in the sublevel set $L_2 = \{R_d, R \in \text{SO}(3) \mid \Psi(R, R_d) < 2\}$, and based on this results, we show exponential stability of the complete dynamics in the smaller sublevel set $L_1 = \{R_d, R \in \text{SO}(3) \mid \Psi(R, R_d) < 1\}$

Proposition 1: (Exponential Stability of Attitude Dynamics) Consider the control moment M defined in (13) for any positive constants k_R, k_Ω . Suppose that the initial condition satisfies

$$\Psi(R(0), R_d(0)) < 2, \quad (17)$$

$$\|e_\Omega(0)\|^2 < \frac{2}{\lambda_{\min}(J)} k_R (2 - \Psi(R(0), R_d(0))). \quad (18)$$

Then, the zero equilibrium of the attitude tracking error e_R, e_Ω is exponentially stable. Furthermore, there exist constants $\alpha_2, \beta_2 > 0$ such that

$$\Psi(R(t), R_d(t)) \leq \min \{2, \alpha_2 e^{-\beta_2 t}\}. \quad (19)$$

Proof: The proofs of all the propositions are given in the appendix. ■

In this proposition, (17), (18) represent a region of attraction for the attitude dynamics. This requires that the initial attitude error should be less than 180° . Therefore, the region of attraction for the attitude almost covers $\text{SO}(3)$, and the region of attraction for the angular velocity can be increased by choosing a larger controller gain k_R in (18).

Since the direction of the total thrust is fixed to the third body-fixed axis, the stability of the translational dynamics depends on the attitude tracking error. More precisely, the position tracking performance is affected by the difference between $\vec{b}_3 = R\vec{e}_3$ and $\vec{b}_{3_d} = R_d\vec{e}_3$. In the proceeding stability analysis, it turns out that for the stability of the complete translational and rotational dynamics, the attitude error function Ψ should be less than 1, which states that the initial attitude error should be less than 90° . For the stability of the complete system, we restrict the initial attitude error, and we obtain the following proposition.

Proposition 2: (Exponential Stability of the Complete Dynamics) Consider the control force f and moment M defined at (12), (13). Suppose that the initial condition satisfies

$$\Psi(R(0), R_d(0)) \leq \psi_1 < 1 \quad (20)$$

for a fixed constant ψ_1 . Define $W_1, W_{12}, W_2 \in \mathbb{R}^{2 \times 2}$ to be

$$W_1 = \begin{bmatrix} \frac{c_1 k_x}{m} & -\frac{c_1 k_v}{2m}(1+\alpha) \\ -\frac{c_1 k_v}{2m}(1+\alpha) & k_v(1-\alpha) - c_1 \end{bmatrix}, \quad (21)$$

$$W_{12} = \begin{bmatrix} k_x e_{v_{\max}} + \frac{\alpha}{m} B & 0 \\ B & 0 \end{bmatrix}, \quad (22)$$

$$W_2 = \begin{bmatrix} \frac{c_2 k_R}{\lambda_{\max}(J)} & -\frac{c_2 k_\Omega}{2\lambda_{\min}(J)} \\ -\frac{c_2 k_\Omega}{2\lambda_{\min}(J)} & k_\Omega - c_2 \end{bmatrix}, \quad (23)$$

where $\alpha = \sqrt{\psi_1(2-\psi_1)}$, $e_{v_{\max}} = \max\{\|e_v(0)\|, \frac{B}{k_v(1-\alpha)}\}$, $c_1, c_2 \in \mathbb{R}$. For any positive constants k_x, k_v , we choose positive constants c_1, c_2, k_R, k_Ω such that

$$c_1 < \min \left\{ k_v(1-\alpha), \frac{4mk_x k_v(1-\alpha)}{k_v^2(1+\alpha)^2 + 4mk_x}, \sqrt{k_x m} \right\}, \quad (24)$$

$$c_2 < \min \left\{ k_\Omega, \frac{4k_\Omega k_R \lambda_{\min}(J)^2}{k_\Omega^2 \lambda_{\max}(J) + 4k_R \lambda_{\min}(J)^2}, \sqrt{k_R \lambda_{\min}(J)}, \sqrt{\frac{2}{2-\psi_1} k_R \lambda_{\max}(J)} \right\}, \quad (25)$$

$$\lambda_{\min}(W_2) > \frac{4\|W_{12}\|^2}{\lambda_{\min}(W_1)}. \quad (26)$$

Then, the zero equilibrium of the tracking errors of the complete system is exponentially stable. The region of attraction is characterized by (20) and

$$\|e_\Omega(0)\|^2 < \frac{2}{\lambda_{\min}(J)} k_R (1 - \Psi(R(0), R_d(0))). \quad (27)$$

D. Almost Global Exponential Attractiveness

Proposition 2 requires that the initial attitude error is less than 90° to achieve the exponential stability of the complete dynamics. Suppose that this is not satisfied, i.e. $1 \leq \Psi(R(0), R_d(0)) < 2$. From Proposition 1, we are guaranteed that the attitude error function Ψ exponentially decreases, and therefore, it enters the region of attraction of Proposition 2 in a finite time. Therefore, by combining the results of Proposition 1 and 2, we can show almost global exponential attractiveness.

Definition 1: (Exponential Attractiveness [19]) An equilibrium point $z = 0$ of a dynamic systems is *exponentially attractive* if, for some $\delta > 0$, there exists a constant $\alpha(\delta) > 0$ and $\beta > 0$ such that $\|z(0)\| < \delta \Rightarrow \|z(t)\| \leq \alpha(\delta)e^{-\beta t}$ for any $t > 0$.

This should be distinguished from the stronger notion of exponential stability, in which the constant $\alpha(\delta)$ in the above bound is replaced by $\alpha(\delta) \|z(0)\|$.

Proposition 3: (Almost Global Exponential Attractiveness of the Complete Dynamics) Consider a control system designed according to Proposition 2. Suppose that the initial condition satisfies

$$1 \leq \Psi(R(0), R_d(0)) < 2, \quad (28)$$

$$\|e_\Omega(0)\|^2 < \frac{2}{\lambda_{\min}(J)} k_R (2 - \Psi(R(0), R_d(0))). \quad (29)$$

Then, the zero equilibrium of the tracking errors of the complete dynamics is exponentially attractive.

Since the region of attraction given by (28) for the attitude almost covers $\text{SO}(3)$, this is referred to as almost global exponential attractiveness in this paper. The region of attraction for the angular velocity can be expanded by choosing a larger gain k_R in (29).

E. Properties and Extensions

One of the unique properties of the presented controller is that it is directly developed on $\text{SE}(3)$ using rotation matrices. Therefore, it avoids the complexities and singularities associated with local coordinates of $\text{SO}(3)$, such as Euler angles. It also avoids the ambiguities that arise when using quaternions to represent the attitude dynamics. As the three-sphere S^3 double covers $\text{SO}(3)$, any attitude feedback controller designed in terms of quaternions could yield different control inputs depending on the choice of quaternion vectors. The corresponding stability analysis would need to carefully consider the fact that convergence to a single attitude implies convergence to either of the two antipodal points on S^3 . The use of rotation matrices in the controller design and stability analysis completely eliminates these difficulties.

Another novelty of the presented controller is the choice of the total thrust in (12). This is designed to follow position tracking commands, but it is also carefully designed to guarantee the overall stability of the complete dynamics by feedback control of the direction of the third body-fixed axis. This consideration is natural as each column of a rotation matrix represents the direction of each body-fixed axis. Therefore, another advantage of using rotation matrices is that the controller has a well-defined physical interpretation.

In Propositions 1 and 3, exponential stability and exponential attractiveness are guaranteed for almost all initial attitude errors, respectively. The attitude error function defined in (8) has the following critical points: the identity matrix, and rotation matrices that can be written as $\exp(\pi\hat{v})$ for any $\hat{v} \in S^2$. These non-identity critical points of the attitude error function lie outside of the region of attraction. As it is a two-dimensional subspace of the three-dimensional $\text{SO}(3)$, we claim that the presented controller exhibits *almost global* properties in $\text{SO}(3)$. It is impossible to construct a smooth controller on $\text{SO}(3)$ that has global asymptotic stability. The

two-dimensional family of non-identity critical points can be reduced to four points by modifying the error function to be $\frac{1}{2}\text{tr}[G(I - R_d^T R)]$ for a matrix $G \neq I \in \mathbb{R}^{3 \times 3}$. The presented controller can be modified accordingly.

IV. NUMERICAL EXAMPLE

The parameters of the quadrotor UAV are chosen according to a quadrotor UAV developed in [2].

$$\begin{aligned} J &= [0.0820, 0.0845, 0.1377] \text{ kgm}^2, \quad m = 4.34 \text{ kg} \\ d &= 0.315 \text{ m}, \quad c_{\tau f} = 8.004 \times 10^{-4} \text{ m}. \end{aligned}$$

The controller parameters are chosen as follows:

$$k_x = 16m, \quad k_v = 5.6m, \quad k_R = 8.81, \quad k_\Omega = 2.54.$$

We consider the following two cases.

- (I) This maneuver follows an elliptical helix while rotating the heading direction at a fixed rate. Initial conditions are chosen as

$$\begin{aligned} x(0) &= [0, 0, 0], \quad v(0) = [0, 0, 0], \\ R(0) &= I, \quad \Omega(0) = [0, 0, 0]. \end{aligned}$$

The desired trajectory is as follows.

$$\begin{aligned} x_d(t) &= [0.4t, 0.4 \sin \pi t, 0.6 \cos \pi t], \\ \vec{b}_1(t) &= [\cos \pi t, \sin \pi t, 0]. \end{aligned}$$

- (II) This maneuver recovers from being initially upside down. Initial conditions are chosen as

$$\begin{aligned} x(0) &= [0, 0, 0], \quad v(0) = [0, 0, 0], \\ R(0) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.9995 & -0.0314 \\ 0 & 0.0314 & -0.9995 \end{bmatrix}, \quad \Omega(0) = [0, 0, 0]. \end{aligned}$$

The desired trajectory is as follows.

$$x_d(t) = [0, 0, 0], \quad \vec{b}_1(t) = [1, 0, 0].$$

Simulation results are presented in Figures 3 and 4. For Case (I), the initial value of the attitude error function $\Psi(0)$ is less than 0.15. This satisfies the conditions for Proposition 2, and exponential asymptotic stability is guaranteed. As shown in Figure 3, the tracking errors exponentially converge to zero. This example illustrates that the proposed controlled quadrotor UAV can follow a complex trajectory that involve large angle rotations and nontrivial translations accurately.

In Case (II), the initial attitude error is 178°, which yields the initial attitude error function $\Psi(0) = 1.995 > 1$. This corresponds to Proposition 3, which implies almost global exponential attractiveness. In Figure 4(b), the attitude error function Ψ decreases, and it becomes less than 1 at $t = 0.88$ seconds. After that instant, the position tracking error and the angular velocity error converge to zero as shown in Figures 4(c) and 4(d). The region of attraction of the proposed control system almost covers SO(3), so that the corresponding controlled quadrotor UAV can recover from being initially upside down.

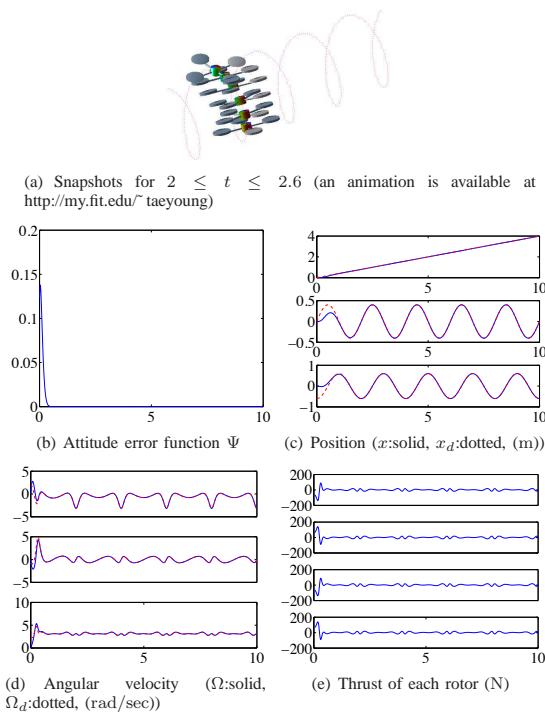


Fig. 3. Case I: following an elliptic helix (horizontal axes represent simulation time in seconds)

V. CONCLUSION

We presented a global dynamic model for a quadrotor UAV, and we developed a geometric tracking controller directly on the special Euclidean group that is intrinsic and coordinate-free, thereby avoiding the singularities of Euler angles and the ambiguities of quaternions in representing attitude. It exhibits exponential stability when the initial attitude error is less than 90°, and it yields almost global exponentially attractiveness when the initial attitude error is less than 180°. These are illustrated by numerical examples.

This controller can be extended as follows. In this paper, four input degrees of freedom are used to track a three-dimensional position, and a one-dimensional heading direction. But, without changing the controller structure, they can be used to follow arbitrary three-dimensional attitude commands. The remaining one input degree of freedom can be used to maintain the altitude as much as possible. By constructing a hybrid controller based on these two tracking modes, we can generate complicated acrobatic maneuvers of a quadrotor UAV.

APPENDIX

Proof of Proposition 1: We first find the error dynamics for e_R, e_Ω , and define a Lyapunov function. Then, we show that under the given conditions, $(R(t), R_d(t))$ always lie in the

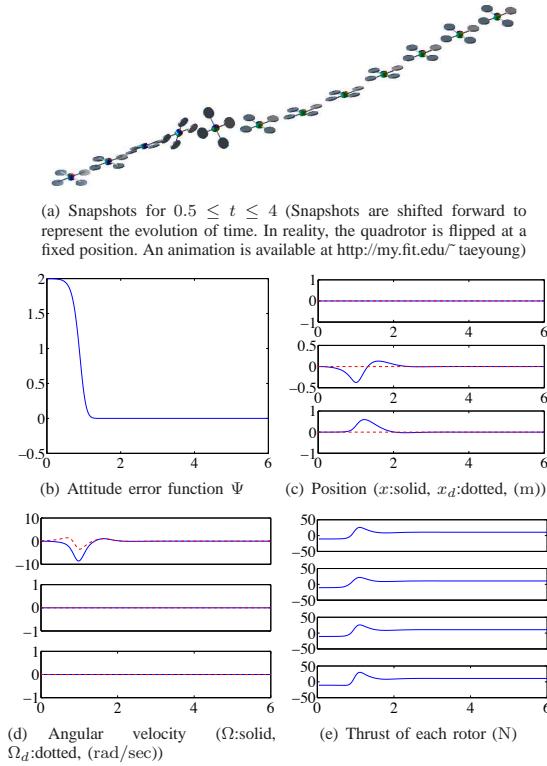


Fig. 4. Case II: recovering from an initially upside down attitude (horizontal axes represent simulation time in seconds)

sublevel set L_2 , which guarantees the positive-definiteness of the attitude error function Ψ . From this, we show the exponential stability of the attitude error dynamics.

a) Error Dynamics: We find the error dynamics for e_R, e_Ω as follows. From the definition of e_Ω in (11), the time derivative of e_R is given by

$$\begin{aligned}\dot{e}_R &= \frac{1}{2}(R_d^T R \hat{e}_\Omega + \hat{e}_\Omega R^T R_d)^V \\ &= \frac{1}{2}(\text{tr}[R^T R_d] I - R^T R_d)e_\Omega \equiv C(R_d^T R)e_\Omega.\end{aligned}\quad (30)$$

We can show that $\|C(R_d^T R)\|_2 \leq 1$ for any $R_d^T R \in \text{SO}(3)$. Thus, $\|\dot{e}_R\| \leq \|e_\Omega\|$. From (11), the time derivative of e_Ω is given by

$$J\dot{e}_\Omega = J\dot{\Omega} + J(\hat{\Omega} R^T R_d \Omega_d - R^T R_d \dot{\Omega}_d).$$

Substituting the equation of motion (5) and the control moment (13), this reduces to

$$J\dot{e}_\Omega = -k_R e_R - k_\Omega e_\Omega. \quad (31)$$

b) Lyapunov Candidate: For a positive constant c_2 , let a Lyapunov candidate \mathcal{V}_2 be

$$\mathcal{V}_2 = \frac{1}{2}e_\Omega \cdot J e_\Omega + k_R \Psi(R, R_d) + c_2 e_R \cdot e_\Omega. \quad (32)$$

From (30), (31), the time derivative of \mathcal{V}_2 is given by

$$\begin{aligned}\dot{\mathcal{V}}_2 &= e_\Omega \cdot J \dot{e}_\Omega - \frac{1}{2}k_R \text{tr}[-\hat{\Omega}_d R_d^T R + R_d^T R \hat{\Omega}] \\ &\quad + c_2 \dot{e}_R \cdot e_\Omega + c_2 e_R \cdot \dot{e}_\Omega \\ &= -k_\Omega \|e_\Omega\|^2 - k_R e_R \cdot e_\Omega - \frac{1}{2}k_R \text{tr}[R_d^T R \hat{e}_\Omega] \\ &\quad + c_2 C(R_d^T R)e_\Omega \cdot e_\Omega + c_2 e_R \cdot J^{-1}(-k_R e_R - k_\Omega e_\Omega).\end{aligned}$$

But, the third term of the above expression can be written as

$$\begin{aligned}\text{tr}[R_d^T R \hat{e}_\Omega] &= \frac{1}{2}\text{tr}[R_d^T R \hat{e}_\Omega - \hat{e}_\Omega R^T R_d] \\ &= \frac{1}{2}\text{tr}[\hat{e}_\Omega (R_d^T R - R^T R_d)] = \text{tr}[\hat{e}_\Omega \hat{e}_R] = -2e_\Omega \cdot e_R.\end{aligned}$$

Therefore, we obtain

$$\dot{\mathcal{V}}_2 = -k_\Omega \|e_\Omega\|^2 - c_2 k_R e_R \cdot J^{-1} e_R + c_2 C(R_d^T R)e_\Omega \cdot e_\Omega - c_2 k_\Omega e_R \cdot J^{-1} e_R. \quad (33)$$

Since $\|C(R_d^T R)\| \leq 1$, this is bounded by

$$\dot{\mathcal{V}}_2 \leq -z_2^T W_2 z_2, \quad (34)$$

where $z_2 = [\|e_R\|, \|e_\Omega\|]^T$, and the matrix $W_2 \in \mathbb{R}^{2 \times 2}$ is given by

$$W_2 = \begin{bmatrix} \frac{c_2 k_R}{\lambda_{\max}(J)} & -\frac{c_2 k_\Omega}{2\lambda_{\min}(J)} \\ -\frac{c_2 k_\Omega}{2\lambda_{\min}(J)} & k_\Omega - c_2 \end{bmatrix}. \quad (35)$$

c) Boundedness of e_R : Suppose that $c_2 = 0$, then from (32), (34), we have

$$\begin{aligned}\mathcal{V}_2|_{c_2=0} &= \frac{1}{2}e_\Omega \cdot J e_\Omega + k_R \Psi(R, R_d), \\ \dot{\mathcal{V}}_2|_{c_2=0} &= -k_\Omega \|e_\Omega\|^2.\end{aligned}$$

This implies that $\mathcal{V}_2|_{c_2=0}$ is non-increasing. Therefore, using (18), the attitude error function is bounded as follows:

$$k_R \Psi(R(t), R_d(t)) \leq \mathcal{V}_2|_{c_2=0}(t) \leq \mathcal{V}_2|_{c_2=0}(0) < 2k_R. \quad (36)$$

This guarantees that there exists a constant ψ_2 such that

$$\Psi(R(t), R_d(t)) \leq \psi_2 < 2, \quad \text{for any } t. \quad (37)$$

Therefore $(R(t), R_d(t))$ always lies in the sublevel set $L_{\psi_2} \subset L_2$.

d) Exponential Stability: Within the sublevel set L_{ψ_2} , the attitude error function is positive-definite, and we obtain

$$\frac{1}{2}\|e_R\|^2 \leq \Psi \leq \frac{1}{2-\psi_2}\|e_R\|^2. \quad (38)$$

Therefore, the Lyapunov function \mathcal{V}_2 is bounded by

$$z_2^T M_{21} z_2 \leq \mathcal{V}_2 \leq z_2^T M_{22} z_2, \quad (39)$$

where

$$M_{21} = \frac{1}{2} \begin{bmatrix} k_R & -c_2 \\ -c_2 & \lambda_{\min}(J) \end{bmatrix}, \quad M_{22} = \frac{1}{2} \begin{bmatrix} \frac{2k_R}{2-\psi_2} & c_2 \\ c_2 & \lambda_{\max}(J) \end{bmatrix}. \quad (40)$$

We choose the positive constant c_2 such that

$$c_2 < \min \left\{ k_\Omega, \frac{4k_\Omega k_R \lambda_{\min}(J)^2}{k_\Omega^2 \lambda_{\max}(J) + 4k_R \lambda_{\min}(J)^2}, \sqrt{k_R \lambda_{\min}(J)}, \sqrt{\frac{2}{2 - \psi_2} k_R \lambda_{\max}(J)} \right\},$$

which makes the matrices W_2, M_{21}, M_{22} positive-definite. Then, the Lyapunov candidate \mathcal{V}_2 and $\dot{\mathcal{V}}_2$ are bounded by

$$\lambda_{\min}(M_{21}) \|z_2\|^2 \leq \mathcal{V}_2 \leq \lambda_{\max}(M_{22}) \|z_2\|^2, \quad (41)$$

$$\dot{\mathcal{V}}_2 \leq -\lambda_{\min}(W_2) \|z_2\|^2. \quad (42)$$

Let $\beta_2 = \frac{\lambda_{\min}(W_2)}{\lambda_{\max}(M_{22})}$. Then, we have

$$\dot{\mathcal{V}}_2 \leq -\beta_2 \mathcal{V}_2. \quad (43)$$

Therefore the zero equilibrium of the tracking error e_R, e_Ω is exponentially stable. Using (38), this implies that

$$\begin{aligned} (2 - \psi_2) \lambda_{\min}(M_{21}) \Psi &\leq \lambda_{\min}(M_{21}) \|e_R\|^2 \\ &\leq \lambda_{\min}(M_{21}) \|z_2\|^2 \leq \mathcal{V}_2(t) \leq \mathcal{V}_2(0) e^{-\beta_2 t}. \end{aligned}$$

So, Ψ exponentially decreases. But, from (37), it is also guaranteed that $\Psi < 2$. This yields (19). ■

Proof of Proposition 2: We first derive the tracking error dynamics. In particular, the velocity error is carefully expressed according to the definition of \vec{b}_{3d} . Using a Lyapunov analysis, we show that the velocity tracking error is uniformly bounded, from which we establish the exponential stability of the complete dynamics.

a) *Attitude Dynamics Error:* The assumptions of Proposition 2, (20), (27) imply (17), (18). The results of Proposition 1 can be directly applied throughout this proof. From (27), equation (36) can be replaced by

$$k_R \Psi(R(t), R_d(t)) \leq \mathcal{V}_2|_{c_2=0}(t) \leq \mathcal{V}_2|_{c_2=0}(0) < k_R. \quad (44)$$

This guarantees that there exist a constant ψ_1 such that

$$\Psi(R(t), R_d(t)) \leq \psi_1 < 1 \quad (45)$$

for all t . This implies that for the given conditions, the attitude error always lies in the sublevel set L_1 , i.e. the attitude error is less than 90° .

b) *Position & Velocity Error Dynamics:* Consider the error dynamics of the translational dynamics. The derivative of e_v is given by

$$\dot{m}\ddot{e}_v = m\ddot{x} - m\ddot{x}_d = mge_3 - fRe_3 - m\ddot{x}_d. \quad (46)$$

Consider a quantity $e_3^T R_d^T Re_3$, which represents the cosine of the angle between \vec{b}_3 and \vec{e}_3 . Since $1 - \Psi$ represents the cosine of the eigen-axis rotation angle between R_d and R , we have $1 > e_3^T R_d^T Re_3 > 1 - \Psi > 0$. Therefore, the quantity $\frac{f}{e_3^T R_d^T Re_3}$ is well-defined. To rewrite this error dynamics of e_v in terms of the attitude error e_R , we add and subtract $\frac{f}{e_3^T R_d^T Re_3} R_d e_3$ to the right hand side of (46) to obtain

$$m\dot{e}_v = mge_3 - m\ddot{x}_d - \frac{f}{e_3^T R_d^T Re_3} R_d e_3 - X, \quad (47)$$

where $X \in \mathbb{R}^3$ is defined by

$$X = \frac{f}{e_3^T R_d^T Re_3} ((e_3^T R_d^T Re_3) R_d e_3 - R_d e_3). \quad (48)$$

Let $A = -k_x e_x - k_v e_v - mge_3 + m\ddot{x}_d$ be the desired control force for the translational dynamics. Then, from (12), (14), we obtain $f = -A \cdot Re_3 = (\|A\| R_d e_3) \cdot Re_3$. Therefore,

$$-\frac{f}{e_3^T R_d^T Re_3} R_d e_3 = -\frac{(\|A\| R_d e_3) \cdot Re_3}{e_3^T R_d^T Re_3} \cdot -\frac{A}{\|A\|} = A.$$

Substituting this into (47), the error dynamics of e_v can be written as

$$m\dot{e}_v = -k_x e_x - k_v e_v - X. \quad (49)$$

c) *Lyapunov Candidate for the Translation Dynamics:* For a positive constant c_1 , let a Lyapunov candidate \mathcal{V}_1 be

$$\mathcal{V}_1 = \frac{1}{2} k_x \|e_x\|^2 + \frac{1}{2} m \|e_v\|^2 + c_1 e_x \cdot e_v. \quad (50)$$

The derivative of \mathcal{V}_1 along the solution of (49) is given by

$$\begin{aligned} \dot{\mathcal{V}}_1 &= -(k_v - c_1) \|e_v\|^2 - \frac{c_1 k_x}{m} \|e_x\|^2 - \frac{c_1 k_v}{m} e_x \cdot e_v \\ &\quad + X \cdot \left\{ \frac{c_1}{m} e_x + e_v \right\}. \end{aligned} \quad (51)$$

We find the bound of X at (48) as follows. Since $f = \|A\| (e_3^T R_d^T Re_3)$, we have

$$\begin{aligned} \|X\| &\leq \|A\| \| (e_3^T R_d^T Re_3) R_d e_3 - R_d e_3 \| \\ &\leq (k_x \|e_x\| + k_v \|e_v\| + B) \| (e_3^T R_d^T Re_3) R_d e_3 - R_d e_3 \|. \end{aligned}$$

Since the last term $\| (e_3^T R_d^T Re_3) R_d e_3 - R_d e_3 \|$ represents the sine of the angle between Re_3 and $R_d e_3$, and $\|e_R\|$ represents the sine of the eigen-axis rotation angle between R_d and R , we have

$$\begin{aligned} \| (e_3^T R_d^T Re_3) R_d e_3 - R_d e_3 \| &\leq \|e_R\| = \sqrt{\Psi(2 - \Psi)} \\ &\leq \sqrt{\psi_1(2 - \psi_1)} < 1. \end{aligned}$$

Therefore, X is bounded by

$$\|X\| \leq (k_x \|e_x\| + k_v \|e_v\| + B) \alpha, \quad (52)$$

where $\alpha = \sqrt{\psi_1(2 - \psi_1)} < 1$. Substituting this into (51),

$$\begin{aligned} \dot{\mathcal{V}}_1 &\leq -(k_v(1 - \alpha) - c_1) \|e_v\|^2 - \frac{c_1 k_x}{m} (1 - \alpha) \|e_x\|^2 \\ &\quad + \frac{c_1 k_v}{m} (1 + \alpha) \|e_x\| \|e_v\| \\ &\quad + \|e_R\| \left\{ k_x \|e_x\| \|e_v\| + \frac{c_1}{m} B \|e_x\| + B \|e_v\| \right\}. \end{aligned} \quad (53)$$

d) *Boundedness of $\|e_v\|$:* In the above expression for $\dot{\mathcal{V}}_1$, there is a third-order error term, namely $k_x \|e_R\| \|e_x\| \|e_v\|$. We find a bound on $\|e_v\|$ to change this term into a second-order error term for the preceding Lyapunov analysis. Suppose $c_1 = 0, k_x = 0$, then from (50), (53), we have

$$\mathcal{V}_1|_{c_1=k_x=0} = \frac{1}{2} m \|e_v\|^2,$$

$$\dot{\mathcal{V}}_1|_{c_1=k_x=0} \leq -k_v(1 - \alpha) \|e_v\|^2 + B \|e_v\|.$$

This implies that when $\|e_v\| > \frac{B}{k_v(1-\alpha)}$, the time derivative of $\|e_v\|$ is negative, and $\|e_v\|$ monotonically decreases. Therefore, $\|e_v\|$ is uniformly bounded by

$$\|e_v(t)\| < \max \left\{ \|e_v(0)\|, \frac{B}{k_v(1-\alpha)} \right\} \equiv e_{v_{\max}}. \quad (54)$$

e) *Lyapunov Candidate for the Complete System::* Let $\mathcal{V} = \mathcal{V}_1 + \mathcal{V}_2$ be the Lyapunov candidate of the complete system.

$$\begin{aligned} \mathcal{V} = & \frac{1}{2} k_x \|e_x\|^2 + \frac{1}{2} m \|e_v\|^2 + c_1 e_x \cdot e_v \\ & + \frac{1}{2} e_\Omega \cdot J e_\Omega + k_R \Psi(R, R_d) + c_2 e_R \cdot e_\Omega. \end{aligned} \quad (55)$$

Using the results of Proposition 1, namely, (33), (39), we can show that the Lyapunov candidate \mathcal{V} is bounded by

$$z_1^T M_{11} z_1 + z_2^T M_{21} z_2 \leq \mathcal{V} \leq z_1^T M_{12} z_1 + z_2^T M'_{22} z_2, \quad (56)$$

where $z_1 = [\|e_x\|, \|e_v\|]^T$, $z_2 = [\|e_R\|, \|e_\Omega\|]^T \in \mathbb{R}^2$, and the matrices $M_{11}, M_{12}, M_{21}, M_{22}$ are given by

$$\begin{aligned} M_{11} &= \frac{1}{2} \begin{bmatrix} k_x & -c_1 \\ -c_1 & m \end{bmatrix}, \quad M_{12} = \frac{1}{2} \begin{bmatrix} k_x & c_1 \\ c_1 & m \end{bmatrix}, \\ M_{21} &= \frac{1}{2} \begin{bmatrix} k_R & -c_2 \\ -c_2 & \lambda_{\min}(J) \end{bmatrix}, \quad M'_{22} = \frac{1}{2} \begin{bmatrix} \frac{2k_R}{2-c_1} & c_2 \\ c_2 & \lambda_{\max}(J) \end{bmatrix}. \end{aligned}$$

Using (34), (53), the time-derivative of \mathcal{V} is given by

$$\dot{\mathcal{V}} \leq -z_1^T W_1 z_1 + z_1^T W_{12} z_2 - z_2^T W_2 z_2, \quad (57)$$

where $W_1, W_{12}, W_2 \in \mathbb{R}^{2 \times 2}$ are defined in (21)-(23).

f) *Exponential Stability:* Under the given conditions (24), (25) of the proposition, all of the matrices $M_{11}, M_{12}, W_1, M_{21}, M_{22}, W_2$, and the Lyapunov candidate \mathcal{V} become positive-definite. The condition given by (26) guarantees that $\dot{\mathcal{V}}$ becomes negative-definite. Therefore, the zero equilibrium of the tracking errors is exponentially stable. ■

Proof of Proposition 3: The given assumptions (28), (29) satisfy the assumption of Proposition 1, from which the tracking error $z_2 = [\|e_R\|, \|e_\Omega\|]$ is guaranteed to exponentially decreases, and to enter the region of attraction of Proposition 2, given by (20), (27), in a finite time t^* .

Therefore, if we show that the tracking error $z_1 = [\|e_x\|, \|e_v\|]$ is bounded in $t \in [0, t^*]$, then the total tracking error $z = [z_1, z_2]$ is uniformly bounded for any $t > 0$, and it exponentially decreases for $t > t^*$. This yields exponential attractiveness.

The boundedness of z_1 is shown as follows. The error dynamics or e_v can be written as

$$m\dot{e}_v = mge_3 - fRe_3 - m\ddot{x}_d.$$

Let \mathcal{V}_3 be a positive-definite function of $\|e_x\|$ and $\|e_v\|$:

$$\mathcal{V}_3 = \frac{1}{2} \|e_x\|^2 + \frac{1}{2} m \|e_v\|^2.$$

Then, we have $\|e_x\| \leq \sqrt{2\mathcal{V}_3}$, $\|e_v\| \leq \sqrt{\frac{2}{m}\mathcal{V}_3}$. The time-derivative of \mathcal{V}_3 is given by

$$\dot{\mathcal{V}}_3 \leq \|e_x\| \|e_v\| + \|e_v\| B + \|e_v\| (k_x \|e_x\| + k_v \|e_v\| + B)$$

$$\begin{aligned} &= k_v \|e_v\|^2 + (2B + (k_x + 1)\|e_x\|) \|e_v\| \\ &\leq d_1 \mathcal{V}_3 + d_2 \sqrt{\mathcal{V}_3}, \end{aligned}$$

where $d_1 = k_v \frac{2}{m} + 2(k_x + 1) \frac{1}{\sqrt{m}}$, $d_2 = 2B \sqrt{\frac{2}{m}}$. Suppose that $\mathcal{V}_3 \geq 1$ for a time interval $[t_a, t_b] \subset [0, t^*]$. In this time interval, we have $\sqrt{\mathcal{V}_3} \leq \mathcal{V}_3$. Therefore,

$$\dot{\mathcal{V}}_3 \leq (d_1 + d_2) \mathcal{V}_3 \Rightarrow \mathcal{V}_3(t) \leq \mathcal{V}_3(t_a) e^{(d_1+d_2)(t-t_a)}.$$

Therefore, for any time interval in which $\mathcal{V}_3 \geq 1$, \mathcal{V}_3 is bounded. This implies that \mathcal{V}_3 is bounded for any $0 \leq t \leq t^*$. ■

REFERENCES

- [1] M. Valenti, B. Bethke, G. Fiore, and J. How, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.
- [2] P. Pounds, R. Mahony, and P. Corke, "Modeling and control of a quadrotor robot," in *Australasian Conference on Robotics and Automation*, 2006.
- [3] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2007, AIAA 2007-6461.
- [4] P. Castillo, R. Lozano, and A. Dzul, "Stabilization of a mini rotorcraft with four rotors," *IEEE Control System Magazine*, pp. 45–55, 2005.
- [5] Mikrokopter. [Online]. Available: <http://www.mikrokopter.de/>
- [6] Microdrone-bulgaria. [Online]. Available: <http://www.microdrones-bulgaria.com/>
- [7] Draganfly innovations. [Online]. Available: <http://www.draganfly.com/>
- [8] S. Bouabdalla, P. Murrieri, and R. Siegward, "Towards autonomous indoor micro VTOL," *Autonomous Robots*, vol. 18, no. 2, pp. 171–183, 2005.
- [9] E. Nice, "Design of a four rotor hovering vehicle," Master's thesis, Cornell University, 2004.
- [10] N. Guenard, T. Hamel, and V. Moreau, "Dynamic modeling and intuitive control strategy for an X4-flyer," in *Proceedings of the IEEE International Conference on Control and Application*, 2005.
- [11] S. Bouabdalla and R. Siegward, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 2259–2264.
- [12] V. Jurdjevic, *Geometric Control Theory*. Cambridge University, 1997.
- [13] A. Bloch, *Nonholonomic Mechanics and Control*, ser. Interdisciplinary Applied Mathematics. Springer-Verlag, 2003, vol. 24.
- [14] F. Bullo and A. Lewis, *Geometric control of mechanical systems*, ser. Texts in Applied Mathematics. New York: Springer-Verlag, 2005, vol. 49, modeling, analysis, and design for simple mechanical control systems.
- [15] S. Bhat and D. Bernstein, "A topological obstruction to global asymptotic stabilization of rotational motion and the unwinding phenomenon," *Proceedings of the American Control Conference*, pp. 2785–2789, 1998.
- [16] D. Maithripala, J. Berg, and W. Dayawansa, "Almost global tracking of simple mechanical systems on a general class of Lie groups," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 216–225, 2006.
- [17] D. Cabecinhas, R. Cunha, and C. Silvestre, "Output-feedback control for almost global stabilization of fully-acuated rigid bodies," in *Proceedings of IEEE Conference on Decision and Control*, 3583–3588, Ed., 2008.
- [18] N. Chaturvedi, N. H. McClamroch, and D. Bernstein, "Asymptotic smooth stabilization of the inverted 3-D pendulum," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1204–1215, 2009.
- [19] Z. Qu, *Robust Control of Nonlinear Uncertain Systems*. New York, NY, USA: John Wiley & Sons, Inc., 1998.

4.9 Trajectory following using LQR

RWB: develop error state trajectory follower using LQR framework

5

Trajectory Generation and Local Planning

The objective of this chapter is to plan trajectories through a map that represents a cluttered world environment. The high-level architecture for this chapter is shown in Figure 5.1.

At the highest level is a voxel representation of the world, local to the multirotor. The voxel grid will evolve dynamically so that the multirotor always remains in voxel at the center of the map. A straight-line path planner is then used to plan paths through the voxel map that minimize the distance to the goal location as well as minimize the associated risk. The output of the straight-line planner is a set of waypoints \mathcal{W} specified as a sequence of desired positions, desired velocities, and desired heading directions. The waypoints are processed by the *minimum-snap trajectory* block to produce a time trajectory of position and heading where the fourth derivative (snap) of the position trajectory is minimized. The trajectory can be fed directly into the trajectory following framework discussed in Chapter 4. However, the framework in Chapter 4 also requires velocity, and angular velocity of the desired multirotor frame. Those elements are naturally produced by the *differential flatness* block shown in Figure 5.1.

5.1 Differential Flatness

Consider the nonlinear system

$$\dot{x} = f(x, u), \quad (5.1)$$

where x is the state and $u \in \mathbb{R}^m$ is the control input, and the output mapping

$$z = h(x), \quad (5.2)$$

where $z \in \mathbb{R}^p$. We say that the system (5.1) is differentially flat with flat output z if the following conditions hold ¹:

1. The components of z are not differentially related over time.

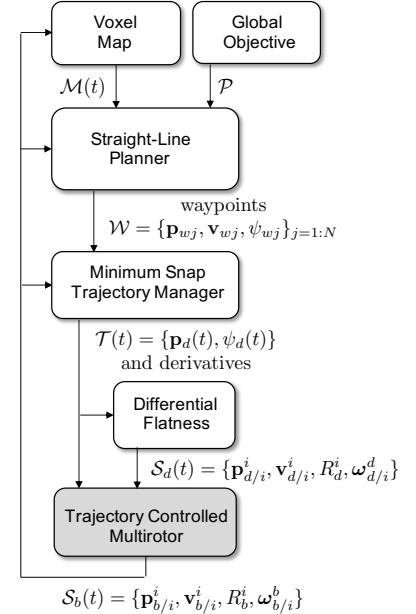


Figure 5.1: The path planning architecture outlined in this chapter.

¹ Ian D Cowling, Oleg A Yakimenko, James F Whidborne, and Alastair K Cooke. A prototype of an autonomous controller for a quadrotor UAV. In *Proceedings of the European Control Conference*, Kos, Greece, jul 2007. [TOC](#)

2. The states x can be written as a function of the flat output and its derivatives, i.e., there exists a function g_1 , and a finite scalar m_1 such that

$$x(t) = g_1 \left(z(t), \frac{d}{dt} z(t), \dots, \frac{d^{m_1}}{dt^{m_1}} z(t) \right). \quad (5.3)$$

3. The control inputs u can be written as a function of the flat output and its derivatives, i.e., there exists a function g_2 and a finite scalar m_2 such that

$$u(t) = g_2 \left(z(t), \frac{d}{dt} z(t), \dots, \frac{d^{m_2}}{dt^{m_2}} z(t) \right). \quad (5.4)$$

We now show that the multi-rotor system is differentially flat.

Theorem 5.1.1 (Multirotor is Differentially Flat) *Suppose that the multirotor system is given by*

$$\dot{\mathbf{p}}_{d/i}^i = \mathbf{v}_{d/i}^i \quad (5.5)$$

$$\dot{\mathbf{v}}_{d/i}^i = g\mathbf{e}_3 + T_d R_d^i \mathbf{e}_3 \quad (5.6)$$

$$\dot{R}_d^i = R_d^i \left[\boldsymbol{\omega}_{d/i}^d \right]_\times, \quad (5.7)$$

and let ψ be the heading angle, i.e., the body x -axis projected onto the inertial $x - y$ plane is $\mathbf{s}_\psi \doteq (\cos \psi, \sin \psi, 0)^\top$.

Then the multirotor system is differentially flat with flat outputs $\{\mathbf{p}_d(t), \psi_d(t)\} \in \mathbb{R}^3 \times \mathbb{R}$.

Proof: To show that the system is differentially flat, the state variables $\mathbf{p}_{d/i}^i$, $\mathbf{v}_{d/i}^i$, R_d^i and the input variables T_d and $\boldsymbol{\omega}_{d/i}^d$ must all be expressed in terms of the flat outputs $\{\mathbf{p}_d(t), \psi_d(t)\}$ and their derivatives.

The position states are directly expressed in the flat outputs as $\mathbf{p}_{d/i}^i = \mathbf{p}_d(t)$. The velocity states are directly expressed in the derivative of the flat outputs as $\dot{\mathbf{v}}_{d/i}^i = \dot{\mathbf{p}}_d(t)$.

Let the rotation matrix be expressed as

$$R_d^i = \begin{pmatrix} \mathbf{r}_{1d} & \mathbf{r}_{2d} & \mathbf{r}_{3d} \end{pmatrix},$$

where r_{jd} is the j^{th} desired body axis of the multirotor expressed in the inertial frame. Therefore Equation (5.6) becomes

$$T_d r_{3d} = \dot{\mathbf{v}}_{d/i}^i - g\mathbf{e}_3,$$

which implies that

$$T_d = \|\dot{\mathbf{v}}_{d/i}^i - g\mathbf{e}_3\|$$

$$\mathbf{r}_{3d} = \frac{\dot{\mathbf{v}}_{d/i}^i - g\mathbf{e}_3}{\|\dot{\mathbf{v}}_{d/i}^i - g\mathbf{e}_3\|}.$$

Therefore the input T_d and the third row of the rotation matrix can be expressed in terms of the flat output as

$$T_d = \|\ddot{\mathbf{p}}_d - g\mathbf{e}_3\|$$

$$\mathbf{r}_{3d} = \frac{\ddot{\mathbf{p}}_d - g\mathbf{e}_3}{\|\ddot{\mathbf{p}}_d - g\mathbf{e}_3\|}.$$

The cross product between the body frame z -axis \mathbf{r}_{3d} and the heading vector \mathbf{s}_{ψ_d} will be in the direction of the body frame y -axis and should always be in the inertial $x - y$ plane. Therefore, \mathbf{r}_{2d} is expressed in terms of the flat outputs as

$$\mathbf{r}_{2d} = \frac{\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}}{\|\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}\|}.$$

Since the body frame is a right-handed coordinate system, the body x -axis is given by

$$\mathbf{r}_{1d} = \mathbf{r}_{2d} \times \mathbf{r}_{3d}.$$

We have shown that R_d^i can be expressed in terms of $\dot{\mathbf{p}}_d$ and ψ_d . By differentiating R_d^i it is clear that \dot{R}_d^i can be expressed in terms of $\dot{\mathbf{p}}_d$, $\ddot{\mathbf{p}}_d$, ψ_d and $\dot{\psi}_d$.

From Equation (5.7), the angular velocity vector is given by

$$\boldsymbol{\omega}_{d/i}^d = \left(R_d^{i\top} \dot{R}_d^i \right)^\vee.$$

Summarizing, the states and outputs are expressed in the flat outputs and their derivatives as

$$\begin{aligned} \mathbf{p}_{d/i}^i(\mathbf{p}_d) &= \mathbf{p}_d(t) \\ \mathbf{v}_{d/i}^i(\dot{\mathbf{p}}_d) &= \dot{\mathbf{p}}_d(t) \\ \mathbf{r}_{3d}(\ddot{\mathbf{p}}_d) &= \frac{\ddot{\mathbf{p}}_d - g\mathbf{e}_3}{\|\ddot{\mathbf{p}}_d - g\mathbf{e}_3\|} \\ \mathbf{r}_{2d}(\ddot{\mathbf{p}}_d, \psi_d) &= \frac{\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}}{\|\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}\|} \\ \mathbf{r}_{1d}(\ddot{\mathbf{p}}_d, \psi_d) &= \mathbf{r}_{2d} \times \mathbf{r}_{3d} \quad (5.8) \\ R_d^i(\ddot{\mathbf{p}}_d, \psi_d) &= \begin{pmatrix} \mathbf{r}_{1d} & \mathbf{r}_{2d} & \mathbf{r}_{3d} \end{pmatrix} \\ T_d(\ddot{\mathbf{p}}_d) &= \|\ddot{\mathbf{p}}_d - g\mathbf{e}_3\| \\ \boldsymbol{\omega}_{d/i}^d(\ddot{\mathbf{p}}_d, \dot{\mathbf{p}}_d, \psi_d, \dot{\psi}_d) &= (R_d^{i\top} \dot{R}_d^i)^\vee. \end{aligned}$$

Since all states and inputs can be expressed in terms of the flat outputs and their derivatives, the system is differentially flat. ■

The inclusion of the induced drag term in the dynamics does not change the differential flatness property, although it does change equations expressions for the states and input variables. The complete derivation is given in ².

²

5.1.1 Analytic calculation of desired angular velocity

Author: Jacob Willis

Rather than numerically differentiating R_d^i as described in Section 2.5.4, this section shows how to find the time derivative of R_d^i analytically. First, recall that

$$R_d^i = \begin{pmatrix} \mathbf{r}_{1d} & \mathbf{r}_{2d} & \mathbf{r}_{3d} \end{pmatrix} \quad (5.9)$$

and therefore that

$$\dot{R}_d^i = \begin{pmatrix} \dot{\mathbf{r}}_{1d} & \dot{\mathbf{r}}_{2d} & \dot{\mathbf{r}}_{3d} \end{pmatrix}. \quad (5.10)$$

To proceed, we need the following two properties.

Property 5.1.2 *The time derivative of the cross product of two time-dependent vectors \mathbf{x} and \mathbf{y} is*

$$\frac{d}{dt}(\mathbf{x} \times \mathbf{y}) = \dot{\mathbf{x}} \times \mathbf{y} + \mathbf{x} \times \dot{\mathbf{y}} \quad (5.11)$$

Proof: This is easiest to see by recalling that $\mathbf{x} \times \mathbf{y} = \mathbf{x}^\wedge \mathbf{y}$ where the result follows from the product rule. ■

Property 5.1.3 *The time derivative of a normalized vector $\frac{\mathbf{x}}{\|\mathbf{x}\|}$ is*

$$\frac{d}{dt} \left(\frac{\mathbf{x}}{\|\mathbf{x}\|} \right) = \Pi_{\frac{\mathbf{x}}{\|\mathbf{x}\|}} \frac{\dot{\mathbf{x}}}{\|\mathbf{x}\|}, \quad (5.12)$$

where $\Pi_{\mathbf{n}} \doteq I - \mathbf{n}\mathbf{n}^\top$ is the projection operator on to the plane orthogonal to the unit vector \mathbf{n} .

Proof: By the quotient rule, we have

$$\frac{d}{dt} \frac{\mathbf{x}}{\|\mathbf{x}\|} = \frac{\dot{\mathbf{x}}\|\mathbf{x}\| - \mathbf{x}\frac{d}{dt}(\|\mathbf{x}\|)}{\|\mathbf{x}\|^2}. \quad (5.13)$$

Now,

$$\frac{d}{dt} \|\mathbf{x}\| = \frac{d}{dt} \sqrt{\mathbf{x}^\top \mathbf{x}} = \frac{1}{2\sqrt{\mathbf{x}^\top \mathbf{x}}} (\dot{\mathbf{x}}^\top \mathbf{x} + \mathbf{x}^\top \dot{\mathbf{x}}) = \frac{\dot{\mathbf{x}}^\top \mathbf{x}}{\|\mathbf{x}\|} \quad (5.14)$$

Plugging in, we get

$$\frac{d}{dt} \frac{\mathbf{x}}{\|\mathbf{x}\|} = \frac{\dot{\mathbf{x}}}{\|\mathbf{x}\|} - \mathbf{x} \frac{\dot{\mathbf{x}}^\top \mathbf{x}}{\|\mathbf{x}\|^3}. \quad (5.15)$$

Rearranging gives

$$\frac{d}{dt} \frac{\mathbf{x}}{\|\mathbf{x}\|} = \left(I - \frac{\mathbf{x}}{\|\mathbf{x}\|} \frac{\mathbf{x}^\top}{\|\mathbf{x}\|} \right) \frac{\dot{\mathbf{x}}}{\|\mathbf{x}\|}.$$

■

The third column of the rotation matrix is given by

$$\mathbf{r}_{3d} = -\frac{\ddot{\mathbf{p}}_d - g\mathbf{e}_3 - K_p\tilde{\mathbf{p}} - K_d\dot{\tilde{\mathbf{p}}}}{\|\ddot{\mathbf{p}}_d - g\mathbf{e}_3 - K_p\tilde{\mathbf{p}} - K_d\dot{\tilde{\mathbf{p}}}\|}. \quad (5.16)$$

Therefore, using the properties above we get that

$$\dot{\mathbf{r}}_{3d} = \Pi_{\mathbf{r}_{3d}} \left(\frac{-\ddot{\mathbf{p}}_d + K_p\dot{\tilde{\mathbf{p}}} + K_d\ddot{\tilde{\mathbf{p}}}}{\|\ddot{\mathbf{p}}_d - g\mathbf{e}_3 - K_p\tilde{\mathbf{p}} - K_d\dot{\tilde{\mathbf{p}}}\|} \right).$$

The second column of the rotation matrix is given by

$$\mathbf{r}_{2d} = \frac{\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}}{\|\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}\|},$$

which implies that

$$\dot{\mathbf{r}}_{2d} = \Pi_{\mathbf{r}_{2d}} \left(\frac{\dot{\mathbf{r}}_{3d} \times \mathbf{s}_{\psi_d} + \dot{\psi}_d \mathbf{r}_{3d} \times (-\sin \psi_d, \cos \psi_d, 0)^\top}{\|\mathbf{r}_{3d} \times \mathbf{s}_{\psi_d}\|} \right).$$

The first column of the rotation matrix is given by

$$\mathbf{r}_{1d} = \mathbf{r}_{2d} \times \mathbf{r}_{3d},$$

which implies that

$$\dot{\mathbf{r}}_{1d} = \dot{\mathbf{r}}_{2d} \times \mathbf{r}_{3d} + \mathbf{r}_{2d} \times \dot{\mathbf{r}}_{3d}.$$

The angular velocity can now be calculated as

$$\boldsymbol{\omega}_{d/i}^d(\mathbf{p}_d, \dot{\mathbf{p}}_d, \ddot{\mathbf{p}}_d, \dot{\psi}_d, \psi_d, \dot{\psi}_d) = \left[\begin{pmatrix} \mathbf{r}_{1d} & \mathbf{r}_{2d} & \mathbf{r}_{3d} \end{pmatrix}^\top \begin{pmatrix} \dot{\mathbf{r}}_{1d} & \dot{\mathbf{r}}_{2d} & \dot{\mathbf{r}}_{3d} \end{pmatrix} \right]^\vee.$$

5.2 An Introduction to b-Spline

The objective of these notes is to explore spline methods for robotic applications. In general, the position of a robot in Euclidian space can be described by a time parametrized trajectory $\mathbf{p}(t) \in \mathbb{R}^3$, $t \in [a, b]$. The time parametrized trajectory can be parameterized using a weighted sum of basis function as

$$\mathbf{p}(t) = \sum_{m=0}^N \mathbf{c}_m \phi_m(t),$$

where $\mathbf{c}_m \in \mathbb{R}^n$, and $\phi_m(t)$ are a set of basis functions. For example, the basis functions could be the set of polynomial power function $\phi_m(t) = t^m/m!$, or the set of sinusoidal function $\phi_m(t) = \sin(\frac{2\pi m}{N}t)$. The disadvantage of both the polynomial power functions and sinusoidal functions is that the basis functions are defined for all $t \in [a, b]$ and so each control points \mathbf{c}_m influences the entire trajectory. Another

disadvantage is that a large number of basis functions may be required to represent complicated trajectories.

In these notes, we will use B-spline basis functions which have a number of very nice properties that we will explore. In particular, a *B-spline* trajectory has the following form

$$\mathbf{p}(t) = \sum_{m=0}^N \mathbf{c}_m b_m^k(t; \mathbf{t}),$$

where $\mathbf{c}_m \in \mathbb{R}^n$ are the control points, $\mathbf{t} = (\tau_0, \tau_1, \tau_2, \dots, \tau_T)$ are called the knot points where $i < j \implies \tau_i \leq \tau_j$, and $b_m^k(t; \mathbf{t})$ are the B-spline basis functions of order k . The spline trajectories will be defined for t in the span of the knot points, i.e., $t \in [\tau_0, \tau_T]$.

Section 5.2.1 defines the B-spline basis function $b_m^k(t; \mathbf{t})$ and describes some of their properties that will be useful for path planning and for estimation. Section ...

5.2.1 B-Spline Basis Functions

The B-spline basis function are defined by the recursive formula:

$$b_m^1(t; \mathbf{t}) = \begin{cases} 1 & \text{if } \tau_m \leq t \leq \tau_{m+1} \\ 0 & \text{otherwise} \end{cases} \quad (5.17)$$

$$b_m^k(t; \mathbf{t}) = w_m^k(t; \mathbf{t}) b_m^{k-1}(t; \mathbf{t}) + [1 - w_m^k(t; \mathbf{t})] b_{m+1}^{k-1}(t; \mathbf{t}), \quad (5.18)$$

$$w_m^k(t; \mathbf{t}) = \begin{cases} \frac{t - \tau_m}{\tau_{m+k} - \tau_m}, & \tau_{m+k} \neq \tau_m \\ 0, & \text{otherwise} \end{cases}. \quad (5.19)$$

First order basis

For example, if the knot vector is given by

$$\mathbf{t} = [\tau_0, \tau_1, \tau_2] \triangleq [0, 1, 2],$$

then there are two basis function of order $k = 1$ given by

$$b_0^1(t; \mathbf{t}) = \begin{cases} 1 & \text{if } \tau_0 \leq t \leq \tau_1 \\ 0 & \text{otherwise} \end{cases}$$

$$b_1^1(t; \mathbf{t}) = \begin{cases} 1 & \text{if } \tau_1 \leq t \leq \tau_2 \\ 0 & \text{otherwise} \end{cases}$$

where $b_0^1(t)$ and $b_1^1(t)$ are shown in Figure 5.2.

Note that $b_1^1(t; \mathbf{t}) = b_0^1(t - 1; \mathbf{t})$, or in other words, all first order basis functions are shifted versions of the central first order basis function $b_0^1(t; \mathbf{t})$. Additional first order basis function can be defined by expanding the knot vector to $\mathbf{t} = [0, \dots, M]$, where $b_m^1(t; \mathbf{t}) = b_0^1(t - m; \mathbf{t})$ for $m \leq M$.

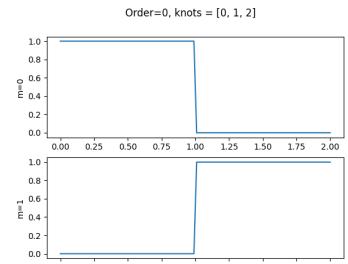


Figure 5.2: First order spline basis

Second order basis

If the knot vector is given by

$$\mathbf{t} = [\tau_0, \tau_1, \tau_2, \tau_3, \tau_4] \stackrel{\Delta}{=} [0, 0, 1, 2, 2],$$

then there are $2k - 1 = 3$ unique basis function of order $k = 2$ given by

$$\begin{aligned} b_0^2(t; \mathbf{t}) &= \frac{t - \tau_0}{\tau_1 - \tau_0} b_0^1(t; \mathbf{t}) + \frac{\tau_2 - t}{\tau_2 - \tau_1} b_1^1(t; \mathbf{t}) = \begin{cases} 0 & \text{if } t_0 = 0 \leq t \leq t_1 = 0 \\ 1 - t & \text{if } t_1 = 0 \leq t \leq t_2 = 1 \end{cases} \\ b_1^2(t; \mathbf{t}) &= \frac{t - \tau_1}{\tau_2 - \tau_1} b_1^1(t; \mathbf{t}) + \frac{\tau_3 - t}{\tau_3 - \tau_2} b_2^1(t; \mathbf{t}) = \begin{cases} t & \text{if } t_1 = 0 \leq t \leq t_2 = 1 \\ 2 - t & t_2 = 1 \leq t \leq t_3 = 2 \\ 0 & \text{otherwise} \end{cases} \\ b_2^2(t; \mathbf{t}) &= \frac{t - \tau_2}{\tau_3 - \tau_2} b_2^1(t; \mathbf{t}) + \frac{\tau_4 - t}{\tau_4 - \tau_3} b_3^1(t; \mathbf{t}) = \begin{cases} t - 1 & \text{if } t_2 = 1 \leq t \leq t_3 = 2 \\ 0 & \text{if } t_3 = 2 \leq t \leq t_4 = 2 \end{cases} \end{aligned}$$

where $b_0^2(t; \mathbf{t})$, $b_1^2(t; \mathbf{t})$, and $b_2^2(t; \mathbf{t})$ are shown on the left in Figure 5.3.

If the knot vector is expanded by one time unit to

$$\mathbf{t}' = [\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5] \stackrel{\Delta}{=} [0, 0, 1, 2, 3, 3],$$

then there are still only three unique basis vectors, but $b_2^2(t; \mathbf{t}')$ is a shifted version $b_1^2(t; \mathbf{t})$ and $b_3^2(t; \mathbf{t}')$ is a shifted version $b_2^2(t; \mathbf{t})$, as shown on the right in Figure 5.3.

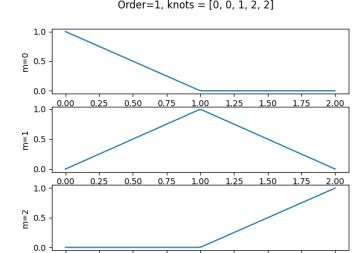


Figure 5.3: Second order spline basis

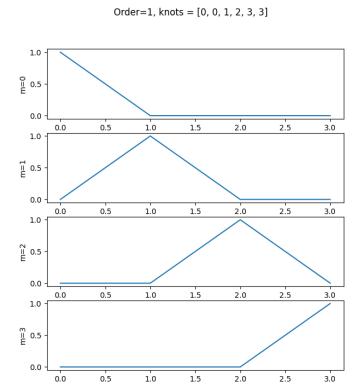


Figure 5.4: Second order spline basis

Third order basis

If the knot vector is given by

$$\mathbf{t} = [\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8] \stackrel{\triangle}{=} [0, 0, 0, 1, 2, 3, 3, 3],$$

then there are $2k - 1 = 5$ unique basis function of order $k = 3$ given by

$$\begin{aligned} b_0^3(t; \mathbf{t}) &= \frac{t - \tau_0}{\tau_2 - \tau_0} b_0^2(t; \mathbf{t}) + \frac{\tau_3 - t}{\tau_3 - \tau_1} b_1^2(t; \mathbf{t}) = \begin{cases} (1-t)^2 & \text{if } 0 \leq t \leq 1 \\ 0 & \text{otherwise} \end{cases} \\ b_1^3(t; \mathbf{t}) &= \frac{t - \tau_1}{\tau_3 - \tau_1} b_1^2(t; \mathbf{t}) + \frac{\tau_4 - t}{\tau_4 - \tau_2} b_2^2(t; \mathbf{t}) = \begin{cases} t(2 - \frac{3}{2}t) & \text{if } 0 \leq t \leq 1 \\ \frac{(2-t)^2}{2} & 1 \leq t \leq 2 \\ 0 & \text{otherwise} \end{cases} \\ b_2^3(t; \mathbf{t}) &= \frac{t - \tau_2}{\tau_4 - \tau_2} b_2^2(t; \mathbf{t}) + \frac{\tau_5 - t}{\tau_5 - \tau_3} b_3^2(t; \mathbf{t}) = \begin{cases} \frac{t^2}{2} & \text{if } 0 \leq t \leq 1 \\ -\frac{3}{2}t^2 + \frac{7}{2}t - \frac{3}{2} & 1 \leq t \leq 2 \\ \frac{(3-t)^2}{2} & 2 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases} \\ b_3^3(t; \mathbf{t}) &= \frac{t - \tau_2}{\tau_4 - \tau_2} b_2^2(t; \mathbf{t}) + \frac{\tau_5 - t}{\tau_5 - \tau_3} b_3^2(t; \mathbf{t}) = \begin{cases} \frac{(t-1)^2}{2} & \text{if } 1 \leq t \leq 2 \\ -\frac{3}{2}t^2 + \frac{15}{2}t - \frac{15}{2} & 2 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases} \\ b_4^3(t; \mathbf{t}) &= \frac{t - \tau_2}{\tau_4 - \tau_2} b_2^2(t; \mathbf{t}) + \frac{\tau_5 - t}{\tau_5 - \tau_3} b_3^2(t; \mathbf{t}) = \begin{cases} (t-2)^2 & \text{if } 2 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases}. \end{aligned}$$

The the unique third order basis function are shown on the left in Figure 5.6.

Expanding the knot vector by one time unit to

$$\mathbf{t}' = [\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5] \stackrel{\triangle}{=} [0, 0, 0, 1, 2, 3, 4, 4, 4],$$

still results in $2k - 1$ unique basis vectors, but $b_3^3(t; \mathbf{t}')$ is a right-shifted version of $b_3^3(t; \mathbf{t})$, and $b_4^3(t; \mathbf{t}')$ and $b_5^3(t; \mathbf{t}')$ are a right-shifted versions of $b_4^3(t; \mathbf{t}')$ and $b_5^3(t; \mathbf{t})$, as shown on the right in Figure 5.6.

Similarly, the unique fourth order and ninth order basis functions are shown in Figures 5.7 and ?? respectively.

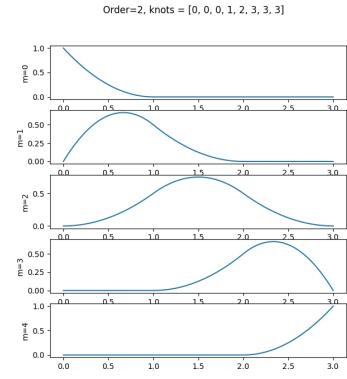


Figure 5.5: Third order spline basis.

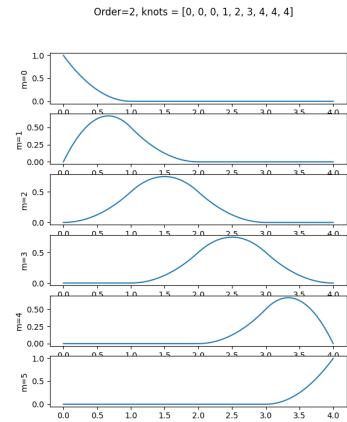


Figure 5.6: Third order spline basis with extra knot

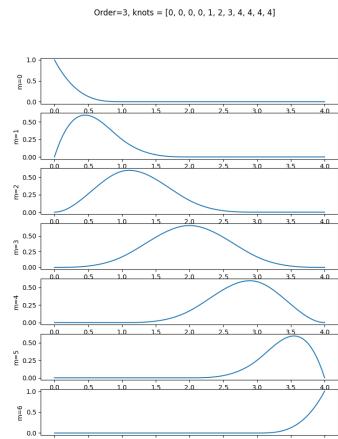
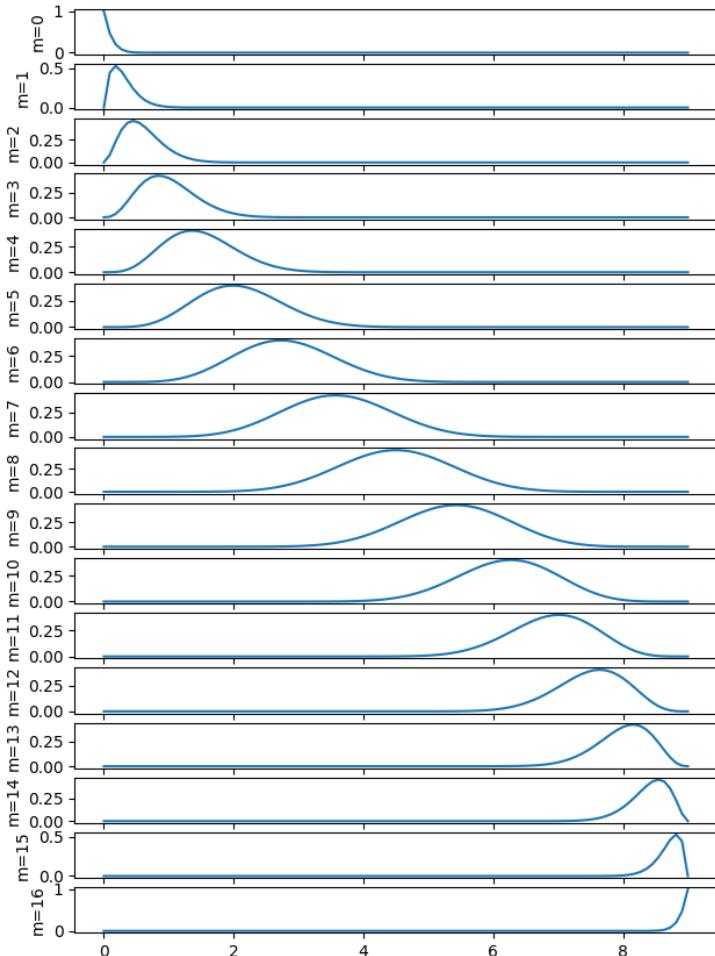


Figure 5.7: Fourth order spline basis

Order=8, knots = [0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9]

Figure 5.8: Ninth order spline basis



Shift and Scale Invariance of the Knot Sequence

Lemma 5.2.1 *Given an arbitrary knot sequence*

$$\mathbf{t} = [\tau_0, \tau_1, \dots, \tau_T],$$

an arbitrary time shift Δ , and the one-sequence $\mathbf{1} \triangleq [1, 1, \dots, 1]$, the B-spline basis functions are shift invariant in the sense that if

$$\mathbf{t} + \Delta \mathbf{1} = [\tau_0 + \Delta, \tau_1 + \Delta, \dots, \tau_T + \Delta]$$

then

$$b_j^k(t; \mathbf{t} + \Delta \mathbf{1}) = b_j^k(t - \Delta; \mathbf{t}),$$

i.e., shifting the knot sequence forward in time is identical to shifting the original B-spline forward in time.

Proof: When the order is $k = 1$, then equation (5.17) gives

$$\begin{aligned} b_j^1(t; \mathbf{t} + \Delta \mathbf{1}) &= \begin{cases} 1 & \text{if } \tau_j + \Delta \leq t \leq \tau_{j+1} + \Delta \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } \tau_j \leq t - \Delta \leq \tau_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ &= b_j^1(t - \Delta; \mathbf{t}). \end{aligned}$$

Assuming that the statement holds when the order is $k - 1 \geq 0$, we get from Equation (5.18) that

$$\begin{aligned} b_j^k(t; \mathbf{t} + \Delta \mathbf{1}) &= \frac{t - \tau_j - \Delta}{\tau_{j+k} + \Delta - \tau_j - \Delta} b_j^{k-1}(t; \mathbf{t} + \Delta \mathbf{1}) + \frac{\tau_{j+k+1} + \Delta - t}{\tau_{j+k+1} + \Delta - \tau_{j+1} - \Delta} b_{j+1}^{k-1}(t; \mathbf{t} + \Delta \mathbf{1}) \\ &= \frac{(t - \Delta) - \tau_j}{\tau_{j+k} - \tau_j} b_j^{k-1}(t - \Delta; \mathbf{t}) + \frac{\tau_{j+k+1} - (t - \Delta)}{\tau_{j+k+1} - \tau_{j+1}} b_{j+1}^{k-1}(t - \Delta; \mathbf{t}) \\ &= b_j^k(t - \Delta; \mathbf{t}). \end{aligned}$$

The proof therefore follows by induction. \blacksquare

Lemma 5.2.2 *Given an arbitrary knot sequence*

$$\mathbf{t} = [\tau_0, \tau_1, \dots, \tau_T]$$

and an arbitrary scaling constant $\alpha \in \mathbb{R}$, the B-spline basis functions are scale invariant in the sense that if

$$\alpha \mathbf{t} = [\alpha \tau_0, \alpha \tau_1, \dots, \alpha \tau_T]$$

then

$$b_j^k(t; \alpha \mathbf{t}) = b_j^k(t/\alpha; \mathbf{t}),$$

i.e., scaling the knot sequence by α is identical to time scaling the original B-spline by $\frac{1}{\alpha}$.

Proof: When the order is $k = 1$, then equation (5.17) gives

$$\begin{aligned} b_j^1(t; \alpha \mathbf{t}) &= \begin{cases} 1 & \text{if } \alpha\tau_j \leq t \leq \alpha\tau_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } \tau_j \leq \frac{t}{\alpha} \leq \tau_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ &= b_j^1(t/\alpha; \mathbf{t}). \end{aligned}$$

Assuming that the statement holds when the order is $k - 1 \geq 0$, we get from Equation (5.18) that

$$\begin{aligned} b_j^k(t; \alpha \mathbf{t}) &= \frac{t - \alpha\tau_j}{\alpha\tau_{j+k} - \alpha\tau_j} b_j^{k-1}(t; \alpha \mathbf{t}) + \frac{\alpha\tau_{j+k+1} - t}{\alpha\tau_{j+k+1} - \alpha\tau_{j+1}} b_{j+1}^{k-1}(t; \alpha \mathbf{t}) \\ &= \frac{(t/\alpha) - \tau_j}{\tau_{j+k} - \tau_j} b_j^{k-1}(t/\alpha; \mathbf{t}) + \frac{\tau_{j+k+1} - (t/\alpha)}{\tau_{j+k+1} - \tau_{j+1}} b_{j+1}^{k-1}(t/\alpha; \mathbf{t}) \\ &= b_j^k(t/\alpha; \mathbf{t}). \end{aligned}$$

The proof therefore follows by induction. \blacksquare

5.2.2 Uniform B-Splines

In the previous section we saw that there is a relationship between the knot point vector \mathbf{t} and the basis functions. We say that the knot vector \mathbf{t} is *uniform* if the knot values are equally spaced, i.e., it has form

$$\mathbf{t} = [\underbrace{\tau_0 - (k-1)\Delta, \dots, \tau_0 - \Delta}_{(k-1)-\text{terms}}, \underbrace{\tau_0, \tau_0 + \Delta, \tau_0 + 2\Delta, \dots, \tau_0 + M\Delta}_{M+1-\text{terms}}, \underbrace{\tau_0 + (M+1)\Delta, \dots, \tau_0 + (M+k-1)\Delta}_{(k-1)-\text{terms}}],$$

where $\Delta > 0$ and $M \geq k$. For example, the following knot vectors are uniform:

$$(t_0 = 0, k = 3, \Delta = 1, M = 3) : \quad \mathbf{t} = [-2, -1, \overline{0}, 1, 2, 3, \overline{4}, 5]$$

$$(t_0 = -3, k = 4, \Delta = 1, M = 5) : \quad \mathbf{t} = [-6, -5, -4, \overline{-3}, -2, -1, 0, 1, 2, \overline{3}, 4, 5]$$

$$(t_0 = 0, k = 4, \Delta = \frac{1}{5}, M = 5) : \quad \mathbf{t} = [-\frac{3}{5}, -\frac{2}{5}, -\frac{1}{5}, \overline{0}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, \overline{\frac{6}{5}, \frac{7}{5}, \frac{8}{5}}].$$

We say that the knot vector $\bar{\mathbf{t}}$ is *uniform and clamped* (the overbar is used to denote clamped) if the knot values are equally spaced and

the first and last $(k - 1)$ -terms are repeated, i.e., it has form

$$\bar{\mathbf{t}} = \left[\underbrace{\tau_0, \tau_0, \dots, \tau_0}_{(k-1)-\text{terms}}, \underbrace{\tau_0, \tau_0 + \Delta, \tau_0 + 2\Delta, \dots, \tau_0 + M\Delta}_{M+1-\text{terms}}, \underbrace{\tau_0 + M\Delta, \dots, \tau_0 + M\Delta}_{(k-1)-\text{terms}} \right],$$

where $\Delta > 0$ and $M \geq k$.

For example, the following knot vectors are clamped and uniform:

$$(t_0 = 0, k = 3, \Delta = 1, M = 3) : \bar{\mathbf{t}} = [0, 0, \vdots 0, 1, 2, 3, \vdots 3, 3]$$

$$(t_0 = -3, k = 4, \Delta = 1, M = 5) : \bar{\mathbf{t}} = [-3, -3, -3, \vdots -3, -2, -1, 0, 1, 2, \vdots 2, 2, 2]$$

$$(t_0 = 0, k = 4, \Delta = \frac{1}{5}, M = 5) : \bar{\mathbf{t}} = [0, 0, 0, \vdots 0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1, \vdots 1, 1, 1].$$

Similarly, the knot vectors shown in Figures 5.2, 5.3, 5.6, 5.7, and ?? are all clamped and uniform.

The length of uniform, and uniform and clamped knot vectors is

$$\text{length}(\mathbf{t}) = M + 1 + 2(k - 1) = M + 2k - 1,$$

and, in both cases, the time interval of interest is $t \in [t_0, t_0 + M\Delta]$, where there are M evenly spaced time intervals.

B-splines associated with uniform knot vectors are called uniform b-splines, and they are only valid on the time interval $t \in [t_0, t_0 + M\Delta]$, similarly B-splines associated with uniform and clamped knot vectors are called uniform and clamped b-splines, and they are also only valid on the time interval $t \in [t_0, t_0 + M\Delta]$.

In the rest of this tutorial, we will focus on the knot vectors

$$\mathbf{t}_M^k \stackrel{\Delta}{=} [-(k - 1), \dots, -1, 0, 1, 2, \dots, M, M + 1, \dots, M + k - 1], \quad (5.20)$$

$$\bar{\mathbf{t}}_M^k \stackrel{\Delta}{=} [\underbrace{0, 0, \dots, 0}_{(k-1)-\text{terms}}, 0, 1, 2, \dots, M, \underbrace{M, \dots, M}_{(k-1)-\text{terms}}], \quad (5.21)$$

and use Lemmas 5.2.1 and 5.2.2 to adapt the results to any desired time interval by shifting and scaling.

There are several important properties of the B-spline basis functions for uniform and clamped knot vectors, which we will lay out in the next several lemmas.

Lemma 5.2.3 *Let \mathbf{t}_M^k be the uniform knot vector defined in Equation (5.20) with $M \geq k$, then the basis function $b_k^{k-1}(t; \mathbf{t}_M^k)$ plays a central role in the sense that*

$$b_{k-1+m}^k(t; \mathbf{t}_M^k) = b_{k-1}^k(t - m; \mathbf{t}_M^k), \quad m = -k, \dots, M - 1.$$

Proof: From Equation (5.17) we give that

$$\begin{aligned} b_m^1(t; \mathbf{t}_M^k) &= \begin{cases} 1 & \text{if } m \leq t \leq m+1 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } 0 \leq t-m \leq 1 \\ 0 & \text{otherwise} \end{cases} \\ &= b_0^1(t-m; \mathbf{t}_M^k). \end{aligned}$$

Suppose that $b_{k-2+m}^{k-1}(t; \mathbf{t}_M^k) = b_{k-2}^{k-1}(t-m; \mathbf{t}_M^k)$, then

$$\begin{aligned} b_{k-1+m}^k(t; \mathbf{t}_M^k) &= \frac{t-(k+m)}{m+2k-(m+k)} b_{k-1+m}^{k-1}(t; \mathbf{t}_M^k) + \frac{m+2k+1-t}{m+2k+1-(m+k+1)} b_{k+m}^{k-1}(t; \mathbf{t}_M^k), \\ &= \frac{(t-m)-k}{k} b_{k-1}^{k-1}(t-m; \mathbf{t}_M^k) + \frac{2k+1-(t-m)}{k} b_{k1}^{k-1}(t-m; \mathbf{t}_M^k), \\ &= b_{k-1}^k(t-m; \mathbf{t}_M^k). \end{aligned}$$

Therefore, the lemma holds by induction. \blacksquare

Lemma 5.2.4 Let \mathbf{t}_M^k be a uniform clamped knot vector defined in Equation (5.20) with $M \geq k$. Then there are exactly $M+k-1$ non-zero basis function of order k , namely $b_m^k(t; \mathbf{t}_M^k)$, $m = 0, \dots, M+k-1$. Furthermore, the basis of support for $b_m^k(t; \mathbf{t}_M^k)$, i.e., the time interval where $b_m^k(t; \mathbf{t}_M^k)$ is non-zero is given by

$$b_m^k(t; \mathbf{t}_M^k) \neq 0 \quad \text{if} \quad \begin{cases} t \in [0, m+1] & 0 \leq m \leq k-1 \\ t \in [m-k+1, m+1] & k \leq m \leq M-1 \\ t \in [m-k+1, M] & M \leq m \leq M+k-2 \end{cases}.$$

Proof: The proof follows from a careful, but straight-forward accounting using Equations (5.17) and (5.18). \blacksquare

A graphical depiction of the region of support for the set of basis functions $\{b_m^k(t; \mathbf{t}_M^k)\}_{m=0}^{M+k-2}$ is shown in Figure 5.9 for $k = 4$ and $M = 8$.

Lemma 5.2.5 Let \mathbf{t}_M^k be a uniform clamped knot vector defined in Equation (5.20) with $M \geq k$. For any instant of time $t \in [0, M]$, the basis vectors at that time sum to unity, i.e., for $t \in [0, M]$,

$$\sum_{m=0}^{M+k-2} b_m^k(t; \mathbf{t}_M^k) = 1.$$

Proof: The proof again follows from a careful, but straight-forward accounting using Equations (5.17) and (5.18). \blacksquare

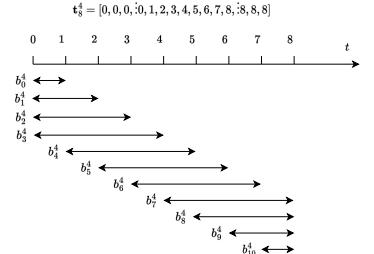


Figure 5.9: Region of support for the B-spline basis functions of order $k = 4$ with knot vector \mathbf{t}_8^4 .

Definition 5.2.6 Let $\{\mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{M+k-2}\}$ be a set of control point vectors in \mathbb{R}^n , and let \mathbf{t}_M^k be a clamped and uniform knot vector, then the function

$$\mathbf{p}(t) = \sum_{m=0}^{M+k-2} \mathbf{c}_m b_m^k(t; \mathbf{t}_M^k), \quad t \in [0, M] \quad (5.22)$$

is said to be a k^{th} order clamped and uniform B-spline.

One of the most important properties of B-splines is the so-called finite-support property, which states that at any instant of time, only a few control points influence the B-spline. This property is formalized in the following lemma.

Lemma 5.2.7 Let \mathbf{t}_M^k be a uniform clamped knot vector defined in Equation (5.20) with $M \geq k$. If $0 \leq j < M$ is an integer, then for any instant of time $t \in [j, j+1] \subset [0, M]$ we have that

$$\mathbf{p}(t) = \sum_{m=0}^{M+k-2} \mathbf{c}_m b_m^k(t; \mathbf{t}_M^k) = \sum_{m=j}^{j+k-1} \mathbf{c}_m b_m^k(t; \mathbf{t}_M^k).$$

In other words, over the interval $t \in [j, j+1]$ there are only k control points that influence $\mathbf{p}(t)$, namely $\{\mathbf{c}_j, \dots, \mathbf{c}_{j+k-1}\}$.

Proof: The proof follows directly from Lemma 5.2.4. \blacksquare

Another important property of B-splines, is that the spline $\mathbf{p}(t)$ is contained in the convex hull of its supporting control points.

Definition 5.2.8 We say that the vector \mathbf{x} is in the convex hull of the vectors $\{\mathbf{q}_1, \dots, \mathbf{q}_n\}$ if

$$\mathbf{x} = \sum_{j=1}^n \alpha_j \mathbf{q}_j, \quad \text{where} \quad \sum_{j=1}^n \alpha_j = 1.$$

Lemma 5.2.9 Let \mathbf{t}_M^k be a uniform or uniform clamped knot vector defined in Equation (5.20) with $M \geq k$. If $0 \leq j < M$ is an integer, then for any instant of time $t \in [j, j+1] \subset [0, M]$ we have that the B-spline

$$\mathbf{p}(t) = \sum_{m=0}^{M+k-2} \mathbf{c}_m b_m^k(t; \mathbf{t}_M^k) = \sum_{m=j}^{j+k-1} \mathbf{c}_m b_m^k(t; \mathbf{t}_M^k)$$

is in the convex hull of the control points $\{\mathbf{c}_j, \dots, \mathbf{c}_{j+k-1}\}$.

Proof: The lemma follows from Lemma 5.2.5 and 5.2.7. \blacksquare

Lemma 5.2.9 is an important result for path planning since we can guarantee collision-free paths by simply checking that the convex hull of control points is collision free.

To simplify notation we will stack the basis function in a vector as

$$\mathbf{b}_M^k(t) \triangleq \begin{pmatrix} b_0^k(t; \mathbf{t}_M^k) \\ b_1^k(t; \mathbf{t}_M^k) \\ \vdots \\ b_{M+k-2}^k(t; \mathbf{t}_M^k) \end{pmatrix}, \quad (5.23)$$

and the control points as a matrix

$$\mathbf{C} = \begin{pmatrix} \mathbf{c}_0 & \mathbf{c}_1 & \dots & \mathbf{c}_{M+k-2} \end{pmatrix}$$

allowing Equation (5.22) to be written as

$$\mathbf{p}(t) = \mathbf{C}\mathbf{b}_M^k(t), \quad t \in [0, M].$$

Given the shift and scale invariance of the B-spline basis functions, $\mathbf{p}(t)$ can be shifted and scaled to represent functions over arbitrary finite time intervals.

SciPy BSpline library The SciPy library has a B-spline library.

The following commands will create a cubic spline.

```
import numpy as np
from scipy.interpolate import BSpline

def uniform_clamped_knots(k, M, t0=np.inf, tf=np.inf):
    # k is the order, M is the number of time intervals
    knots = [0] * k + list(range(0, M + 1)) + [M] * k
    knots = np.asarray(knots) # convert knots to an NP array
    if t0 != np.inf:
        if (tf != np.inf) & (tf > t0):
            knots = (tf-t0)/M * knots
        knots = knots + t0
    return knots

t0 = 1 # initial time
tf = 5 # final time
k = 3 # spline order
M = 3 # number of time intervals
knots = uniform_clamped_knots(k=order, M=M, t0=t0, tf=tf)
# need M+k control points
ctrl_pts = np.array([[0, 0, 0],
                     [0, 1, 0],
                     [0, 0, 1],
```

```

[0, 1, 1],
[1, 1, 0],
[1, 1, 1]])
spl = BSpline(t=knots, c=ctrl_pts, k=order)
plot_spline(spl)

```

Where `plot_spline` is given below.

```

from math import ceil
from scipy.interpolate import BSpline
import matplotlib.pyplot as plt

def plot_spline(spl):
    t0 = spl.t[0] # first knot is t0
    tf = spl.t[-1] # last knot is tf
    # number of points in time vector so spacing is 0.01
    N = ceil((tf - t0)/0.01)
    t = np.linspace(t0, tf, N) # time vector
    position = spl(t)
    # 3D trajectory plot
    fig = plt.figure(1)
    ax = fig.add_subplot(111, projection='3d')
    # plot control points (convert YX(-Z) -> NED)
    ax.plot(spl.c[:, 1], spl.c[:, 0], -spl.c[:, 2],
             '-o', label='control points')
    # plot spline (convert YX(-Z) -> NED)
    ax.plot(position[:, 1], position[:, 0], -position[:, 2],
             'b', label='spline')
    ax.legend()
    ax.set_xlabel('x', fontsize=16, rotation=0)
    ax.set_ylabel('y', fontsize=16, rotation=0)
    ax.set_zlabel('z', fontsize=16, rotation=0)
    #ax.set_xlim3d([-10, 10])
    plt.show()

```

The resulting spline is shown in Figure 5.10.

5.2.3 Derivatives of B-splines

We begin this section by noting that the basis functions and the knot vector do not have to be of the same order. In fact, if the knot vector has higher order than the basis functions, then the first several basis functions will simply be zero. For example, let $\mathbf{t}_3^3 = [0, 0, 0, 1, 2, 3, 3, 3]$, then from Equation (5.17) $b_0^1(t; \mathbf{t}_3^3)$, $b_1^1(t; \mathbf{t}_3^3)$, $b_5^1(t; \mathbf{t}_3^3)$, $b_1^1(t; \mathbf{t}_3^3)$, and $b_5^2(t; \mathbf{t}_3^3)$ are equal to zero since the knot intervals in the denominator for those functions are zero. The remaining basis functions are shown

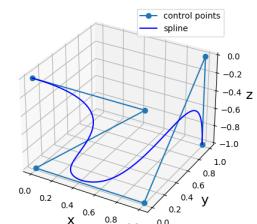


Figure 5.10: Example spline curve

in Figure 5.11.

If on the other hand, the knot vector has one order lower, i.e., $\mathbf{t}_3^2 = [0; 0, 1, 2, 3, 3]$, then the basis vectors are also shown to the right in Figure 5.11. It is clear that reducing the order of the knot vector by one, shifts the index of the basis vectors by one. We can formalize these observations in the following lemma.

Lemma 5.2.10 *For uniform or clamped uniform knot vectors $\mathbf{t}_M^k \in \mathbb{R}^{M+2k-1}$ and $\mathbf{t}_M^{k-1} \in \mathbb{R}^{M+2k-1}$ we have*

$$b_m^j(t; \mathbf{t}_M^k) = b_{m-1}^j(t; \mathbf{t}_M^{k-1}), \quad j \leq k, \quad m = 1, \dots, 2k + M.$$

Using vector notation we have that

$$\underbrace{\mathbf{b}_M^{k-1}(t; \mathbf{t}_M^{k-1})}_{(M+k-1) \times 1} = S_{M+k} \underbrace{\mathbf{b}_M^{k-1}(t; \mathbf{t}_M^k)}_{(M+k) \times 1},$$

where

$$S_N \triangleq \begin{bmatrix} \mathbf{0}_{(N-1) \times 1}, & \mathbf{I}_{(N-1) \times (N-1)} \end{bmatrix}.$$

We have the following lemma which gives the general formula for the derivative of the spline basis.

Lemma 5.2.11 *If \mathbf{t} is a general knot vector, then the ℓ^{th} derivative of the k^{th} order spline function with $0 \leq \ell \leq k$, is given by*

$$\frac{d^\ell}{dt^\ell} b_m^k(t; \mathbf{t}) = (k-1) \left(\frac{\frac{d^{\ell-1}}{dt^{\ell-1}} b_m^{k-1}(t; \mathbf{t})}{\tau_{m+k-1} - \tau_m} - \frac{\frac{d^{\ell-1}}{dt^{\ell-1}} b_{m+1}^{k-1}(t; \mathbf{t})}{\tau_{m+k} - \tau_{m+1}} \right).$$

Using vector notation, we have the following formula for the derivative of a spline function.

Lemma 5.2.12 *Given the k^{th} -order uniform clamped spline function*

$$\mathbf{p}(t) = \mathbf{C} \mathbf{b}_M^k(t), \quad t \in [0, M]$$

we have that

$$\frac{d\mathbf{p}}{dt}(t) = \mathbf{C} D_M^k \mathbf{b}_M^{k-1}(t), \quad t \in [0, M]$$

where D_M^k is the $(M+k-1) \times (M+k-2)$ derivative matrix given by

$$D_M^k = - \begin{bmatrix} \bar{D}_M^k \\ \mathbf{0}_{1 \times (M+k-2)} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{1 \times (M+k-2)} \\ \bar{D}_M^k \end{bmatrix}, \quad (5.24)$$

where

$$\bar{D}_M^k = \text{diag} \left(\frac{k-1}{1}, \frac{k-1}{2}, \dots, \frac{k-2}{k-1}, \underbrace{1, \dots, 1}_{M-k}, \frac{k-1}{k-2}, \dots, \frac{k-1}{2}, \frac{k-1}{1} \right), \quad (5.25)$$

is an $(M+k-2) \times (M+k-2)$ diagonal matrix.

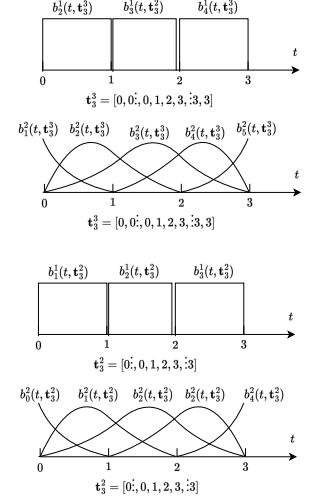


Figure 5.11: Shifting property of uniform clamped B-spline basis functions.

The proof of Lemma 5.2.11 is in .
Les Piegl and Wayne Tiller. *The NURBS Book*. Springer, second edition edition, 1995

Proof: Noting that

$$\mathbf{p}(t) = \mathbf{C}\mathbf{b}_M^k(t) = \sum_{m=0}^{M+k-2} \mathbf{c}_m b_m^k(t; \bar{\mathbf{t}}_M^k),$$

we have that

$$\frac{d\mathbf{p}}{dt} = \sum_{m=0}^{M+k-2} \mathbf{c}_m \frac{db_m^k(t; \bar{\mathbf{t}}_M^k)}{dt}.$$

From Lemma 5.2.11 we get

$$\begin{aligned} \frac{d\mathbf{p}}{dt} &= \sum_{m=0}^{M+k-2} \mathbf{c}_m(k-1) \left(\frac{b_m^{k-1}(t; \bar{\mathbf{t}}_M^k)}{\tau_{m+k-1} - \tau_m} - \frac{b_{m+1}^{k-1}(t; \bar{\mathbf{t}}_M^k)}{\tau_{m+k} - \tau_{m+1}} \right) \\ &= \sum_{m=0}^{M+k-2} \mathbf{c}_m \left[\left(\frac{k-1}{\tau_{m+k-1} - \tau_m} \right) b_m^{k-1}(t; \bar{\mathbf{t}}_M^k) - \left(\frac{k-1}{\tau_{m+k} - \tau_{m+1}} \right) b_{m+1}^{k-1}(t; \bar{\mathbf{t}}_M^k) \right] \\ &= \left(\frac{k-1}{\tau_{k-1} - \tau_0} \right) \mathbf{c}_0 b_0^{k-1}(t; \bar{\mathbf{t}}_M^k) + \sum_{m=1}^{M+k-2} \left(\frac{k-1}{\tau_{m+k-1} - \tau_m} \right) (\mathbf{c}_m - \mathbf{c}_{m-1}) b_m^{k-1}(t; \bar{\mathbf{t}}_M^k) \\ &\quad - \left(\frac{k-1}{\tau_{M+2k-2} - \tau_{M+k-2}} \right) \mathbf{c}_{M+k-2} b_{M+k-1}^{k-1}(t; \bar{\mathbf{t}}_M^k). \end{aligned}$$

Now note that since the first k terms in the knot vector $\bar{\mathbf{t}}_M^k$ are equal to zeros, and the last k terms are equal to M , that $\tau_{k-1} - \tau_0 = \tau_{M+2k-2} - \tau_{M+k-2} = 0$, and therefore the first and last terms are zero. Using the shifting property from Lemma 5.2.10, and changing the index by one, we get

$$\frac{d\mathbf{p}}{dt} = \sum_{m=0}^{M+k-3} \left(\frac{k-1}{\tau_{m+k-2} - \tau_m} \right) (\mathbf{c}_{m+1} - \mathbf{c}_m) b_m^{k-1}(t; \bar{\mathbf{t}}_M^{k-1}), \quad (5.26)$$

where τ_* now corresponds to the knot vector

$$\bar{\mathbf{t}}_M^{k-1} = [\underbrace{0, 0, \dots, 0}_{k-2}, 0, 1, 2, \dots, M, \underbrace{M, \dots, M}_{k-2}].$$

From the definition of $\bar{\mathbf{t}}_M^{k-1}$ we note that

$$\begin{aligned} \frac{k-1}{\tau_{k-1} - \tau_0} &= \frac{k-1}{1}, & \frac{k-1}{\tau_k - \tau_1} &= \frac{k-1}{2}, & \dots & \frac{k-1}{\tau_{2k-3} - \tau_{k-3}} = \frac{k-1}{k-2}, \\ \frac{k-1}{\tau_{2k-2} - \tau_{k-2}} &= \frac{k-1}{k-1}, & \dots & \frac{k-1}{\tau_{M+k-2} - \tau_{M-1}} = \frac{k-1}{k-1}, \\ \frac{k-1}{\tau_{M+k-1} - \tau_M} &= \frac{k-1}{k-2}, & \dots & \frac{k-1}{\tau_{M+2k-3} - \tau_{M+k-3}} = \frac{k-1}{1} \end{aligned}$$

Define the $(M+k-2) \times (M+k-2)$ diagonal matrix \bar{D}_M^k in Equation (5.25), and the $(M+k-1) \times (M+k-2)$ derivative matrix D_M^k in Equation (5.27), then Equation (5.26) can be written as

$$\frac{d\mathbf{p}}{dt} = \mathbf{C}D_M^k \mathbf{b}_M^{k-1}(t).$$

■

Lemma 5.2.13 Given the k^{th} -order uniform spline function

$$\mathbf{p}(t) = \mathbf{C}\mathbf{b}_M^k(t), \quad t \in [0, M]$$

we have that

$$\frac{d\mathbf{p}}{dt}(t) = \mathbf{C}D_M^k \mathbf{b}_M^{k-1}(t), \quad t \in [0, M]$$

where D_M^k is the $(M + k - 1) \times (M + k - 2)$ derivative matrix given by

$$D_M^k = \begin{pmatrix} -1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & & & & \vdots & \\ 0 & 0 & 0 & \ddots & 1 & -1 \\ 0 & 0 & 0 & \ddots & 0 & 1 \end{pmatrix}. \quad (5.27)$$

From Lemma 5.2.12 we can derive a number of useful results, which we summarize in the corollary below.

Corollary 5.2.14 Given the k^{th} -order uniform clamped B-spline function

$$\mathbf{p}(t) = \mathbf{C}\mathbf{b}_M^k(t), \quad t \in [0, M]$$

we can make the following statements:

- (i) The derivative $\frac{d\mathbf{p}}{dt}$ is a $(k - 1)^{\text{th}}$ -order uniform clamped B-spline function.
- (ii) The control points of $\frac{d\mathbf{p}}{dt}$ are given by

$$C' \triangleq CD_M^k = \begin{pmatrix} \frac{k-1}{1}(\mathbf{c}_1 - \mathbf{c}_0)^\top \\ \vdots \\ \frac{k-1}{k-2}(\mathbf{c}_{k-2} - \mathbf{c}_{k-3})^\top \\ (\mathbf{c}_{k-1} - \mathbf{c}_{k-2})^\top \\ \vdots \\ (\mathbf{c}_{M+1} - \mathbf{c}_M)^\top \\ \frac{k-1}{k-2}(\mathbf{c}_{M+2} - \mathbf{c}_{M+1})^\top \\ \vdots \\ \frac{k-1}{1}(\mathbf{c}_{M+k-2} - \mathbf{c}_{M+k-3})^\top \end{pmatrix}^\top \triangleq \begin{pmatrix} \mathbf{c}'_0^\top \\ \vdots \\ \mathbf{c}'_{M+k-3}^\top \end{pmatrix}^\top.$$

- (iii) The ℓ^{th} derivative of $\mathbf{p}(t)$ for $0 \leq \ell \leq k - 2$ is given by

$$\begin{aligned} \frac{d^\ell \mathbf{p}}{dt^\ell} &= CD_M^k D_M^{k-1} \dots D_M^{k-\ell} \mathbf{b}_M^{k-\ell-1}(t), \quad t \in [0, M] \\ &= C^{(\ell)} \mathbf{b}_M^{k-\ell-1}(t), \quad t \in [0, M] \\ &= C\psi^{(\ell)}(t), \quad t \in [0, M] \end{aligned}$$

where the control points of the $(k - \ell - 1)^{th}$ order spline are given by

$$C^{(\ell)} = CD_M^k D_M^{k-1} \dots D_M^{k-\ell}$$

or alternatively, the ℓ^{th} derivative of the basis vector $\mathbf{b}_M^k(t)$ is given by

$$\psi^{(\ell)} \triangleq \frac{d^\ell \mathbf{b}_M^k(t)}{dt^\ell} = D_M^k D_M^{k-1} \dots D_M^{k-\ell} \mathbf{b}_M^{k-\ell-1}(t), \quad t \in [0, M].$$

(iv) The derivative of $\mathbf{p}(t)$ at $t = 0$ is given by the difference of the first two control points as

$$\frac{d\mathbf{p}}{dt}(0) = (k-1)(\mathbf{c}_1 - \mathbf{c}_0).$$

(v) The derivative of $\mathbf{p}(t)$ at $t = M$ is given by the difference of the last two control points as

$$\frac{d\mathbf{p}}{dt}(M) = (k-1)(\mathbf{c}_{M+k-2} - \mathbf{c}_{M+k-3}).$$

(vi) If the desired B-spline trajectory with $k \geq 4$ has the following desired endpoint conditions:

$$\begin{aligned} \text{Initial position: } \mathbf{p}(0) &\stackrel{des}{=} \mathbf{p}_0 \\ \text{Final position: } \mathbf{p}(M) &\stackrel{des}{=} \mathbf{p}_f \\ \text{Initial velocity: } \frac{d\mathbf{p}}{dt}(0) &\stackrel{des}{=} \mathbf{v}_0 \\ \text{Final velocity: } \frac{d\mathbf{p}}{dt}(M) &\stackrel{des}{=} \mathbf{v}_f, \\ \text{Initial acceleration: } \frac{d^2\mathbf{p}}{dt^2}(0) &\stackrel{des}{=} \mathbf{a}_0 \\ \text{Final acceleration: } \frac{d^2\mathbf{p}}{dt^2}(M) &\stackrel{des}{=} \mathbf{a}_f, \end{aligned}$$

then the first and last three control points satisfy

$$\begin{aligned} \mathbf{c}_0 &= \mathbf{p}_0 \\ \mathbf{c}_1 &= \mathbf{p}_0 + \frac{1}{k-1}\mathbf{v}_0 \\ \mathbf{c}_2 &= \mathbf{p}_0 + \frac{3}{k-1}\mathbf{v}_0 + \frac{2}{(k-1)(k-2)}\mathbf{a}_0 \\ \mathbf{c}_{M+k-4} &= \mathbf{p}_f - \frac{3}{k-1}\mathbf{v}_f + \frac{2}{(k-1)(k-2)}\mathbf{a}_f \\ \mathbf{c}_{M+k-3} &= \mathbf{p}_f - \frac{1}{k-1}\mathbf{v}_f \\ \mathbf{c}_{M+k-2} &= \mathbf{p}_f. \end{aligned}$$

5.2.4 B-Spline Planning for Chains of Integrators

5.2.5 B-Splines on Lie Groups

5.2.6 B-Splines Surfaces

Suppose that the objective is to create a one dimensional surface over a two dimensional domain. For example, a terrain map is a one dimensional surface of altitudes over the north-east plane. In this section we will explore the use of B-splines to accomplish this objective. Many of the ideas and notation in this section come from ³.

If \mathbf{t}_M^k and \mathbf{t}_N^k are two different k^{th} order uniform knot vectors of length M and N understood to define knots in the x and y directions, then following Equation (5.22), a clamped uniform B-spline surface is defined as

$$S(x, y) \triangleq \sum_{m=0}^{M+k-2} \sum_{n=0}^{N+k-2} c_{m,n} b_m^k(x; \mathbf{t}_M^k) b_n^k(y; \mathbf{t}_N^k), \quad x \in [0, M], y \in [0, N], \quad (5.28)$$

where $c_{m,n}$, $m = 0, \dots, M + k - 2$, $n = 0, \dots, N + k - 2$ are the control points of the surface. Defining the matrix $\mathbf{C} = \{c_{m,n}\}$, and defining the spatial variable $\mathbf{s} = (s_1, s_2)^\top$, we can write Equation (5.28) as

$$S(\mathbf{s}) = \mathbf{b}_M^k(s_1)^\top \mathbf{C} \mathbf{b}_N^k(s_2), \quad \mathbf{s} \in [0, M] \times [0, N], \quad (5.29)$$

This equation is linear in the spline coefficients \mathbf{C} . In other words, if $S_1(\mathbf{s}) = \mathbf{b}_M^k(s_1)^\top \mathbf{C}_1 \mathbf{b}_N^k(s_2)$, and $S_2(\mathbf{s}) = \mathbf{b}_M^k(s_1)^\top \mathbf{C}_2 \mathbf{b}_N^k(s_2)$ are two different spline surfaces, then

$$S(\mathbf{s}) \triangleq \alpha S_1(\mathbf{s}) + \beta S_2(\mathbf{s}) = \mathbf{b}_M^k(s_1)^\top (\alpha \mathbf{C}_1 + \beta \mathbf{C}_2) \mathbf{b}_N^k(s_2),$$

is also a spline surface.

Suppose that instead of defining the surface over the set $[0, M] \times [0, N]$ we instead would like to define the surface over the set $[\underline{X}, \bar{X}] \times [\underline{Y}, \bar{Y}]$. Given the shift and scaling properties is Lemmas 5.2.1 and 5.2.2 we have

$$S(\mathbf{s}) = \mathbf{b}_M^k \left(\frac{M(s_1 - \underline{X})}{\bar{X} - \underline{X}} \right)^\top \mathbf{C} \mathbf{b}_N^k \left(\frac{N(s_2 - \underline{Y})}{\bar{Y} - \underline{Y}} \right), \quad \mathbf{s} \in [\underline{X}, \bar{X}] \times [\underline{Y}, \bar{Y}]. \quad (5.30)$$

The Python code below plots a randomly defined terrain map over the domain $[-\pi, \pi] \times [-2\pi, 2\pi]$.

```
import numpy as np
import splipy as sp
import matplotlib.pyplot as plt
from bspline_tools import uniform_clamped_knots

def draw_random_surface(order=2, M=10, N=10,
```

³ Romulo T. Rodrigues, Nikolaos Tsiofkas, A. Pedro Aguiar, and Antonio Pascoal. B-spline surfaces for range-based environment mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, pages 10774–10779, Las Vegas, Nevada, October 2020

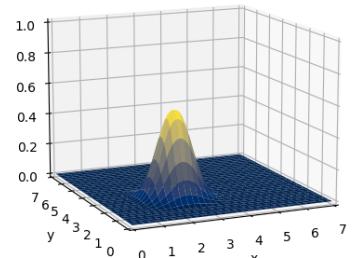


Figure 5.12: A single second order spline bases function $b_3^2(s_1)b_3^2(s_2)$.

```

Xmin=-3.14159, Xmax=3.14159,
Ymin=-2*3.14159, Ymax=2*3.14159):
# define the spline surface
knots_x = uniform_clamped_knots(k=order, M=M)
knots_y = uniform_clamped_knots(k=order, M=N)
basis_x = sp.BSplineBasis(order + 1, knots_x)
basis_y = sp.BSplineBasis(order + 1, knots_y)
C = np.random.rand(M + order, N + order) # random control points
surface = sp.Surface(basis_x, basis_y,
                      np.reshape(C, ((M + order) * (N + order), 1)))
# plot the spline surface
x = np.linspace(Xmin, Xmax, 10 * M)
y = np.linspace(Ymin, Ymax, 10 * N)
S = surface(M*(x-Xmin)/(Xmax-Xmin),
             N*(y-Ymin)/(Ymax-Ymin)) # surface points
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.xlabel('x')
plt.ylabel('y')
x_pts, y_pts = np.meshgrid(x, y) # grid points
ax.plot_surface(x_pts, y_pts, S[:, :, 0])
plt.show()

```

The result of running this code is shown in Figure 5.13

5.3 Minimum Snap Trajectories

Because multirotor dynamics are differentially flat, their motion can be completely parametrized by trajectories in position and heading. It is argued in (Mellinger & Kumar, 2011)⁴ that since acceleration is dependent on third derivative of position, and torque is dependent on the derivative of heading, that smooth trajectories for quadrotors should minimize the fourth derivative (snap) of the position spline, and the second derivative of the yaw spline. In this section, will show to derive quadratic cost functions on the spline coefficients that minimize the appropriate derivative.

Let the position and heading trajectories be defined by k^{th} order splines over the time interval $t \in [0, M]$ as

$$\begin{aligned}\mathbf{p}_d(t) &= \mathbf{C}_p b_M^k(t) \\ \psi_d(t) &= \mathbf{C}_\psi b_M^k(t),\end{aligned}$$

where $\mathbf{C}_p \in \mathbb{R}^{3 \times M+k-1}$, and $\mathbf{C}_\psi \in \mathbb{R}^{1 \times M+k-1}$. As shown in Lemma 5.2.14, the fourth derivative of position and second deriva-

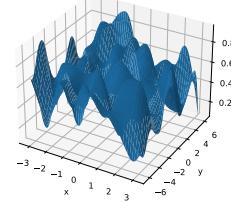


Figure 5.13: Spline surface with randomly generated control points.

⁴ Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525, Shanghai, China, May 2011

tive of heading are given by

$$\begin{aligned}\mathbf{p}_d^{(4)}(t) &= \mathbf{C}_p D_M^k D_M^{k-1} D_M^{k-2} D_M^{k-3} \mathbf{b}_M^{k-4}(t) \\ \psi_d^{(2)}(t) &= \mathbf{C}_\psi D_M^k D_M^{k-1} \mathbf{b}_M^{k-2}(t).\end{aligned}$$

Defining

$$S_M^{k,j} \triangleq D_M^k D_M^{k-1} \cdots D_M^{k-j}$$

gives

$$\begin{aligned}\mathbf{p}_d^{(4)}(t) &= \mathbf{C}_p S_M^{k,3} \mathbf{b}_M^{k-4}(t) \\ \psi_d^{(2)}(t) &= \mathbf{C}_\psi S_M^{k,1} \mathbf{b}_M^{k-2}(t).\end{aligned}$$

The normed integral square of the fourth derivative of $\mathbf{p}(t)$ is given by

$$\begin{aligned}\int_0^M \|\mathbf{p}^{(4)}(t)\|^2 dt &= \int_0^M \mathbf{p}^{(4)\top}(t) \mathbf{p}^{(4)}(t) dt \\ &= \int_0^M \mathbf{b}_M^{k-4}(t)^\top S_M^{k,3\top} \mathbf{C}_p^\top \mathbf{C}_p S_M^{k,3} \mathbf{b}_M^{k-4}(t) dt \\ &= \text{tr} \left(\int_0^M \mathbf{b}_M^{k-4}(t)^\top S_M^{k,3\top} \mathbf{C}_p^\top \mathbf{C}_p S_M^{k,3} \mathbf{b}_M^{k-4}(t) dt \right) \\ &= \text{tr} \left(\int_0^M \mathbf{C}_p S_M^{k,3} \mathbf{b}_M^{k-4}(t) \mathbf{b}_M^{k-4}(t)^\top S_M^{k,3\top} \mathbf{C}_p^\top dt \right) \\ &= \text{tr} \left(\mathbf{C}_p S_M^{k,3} \int_0^M \mathbf{b}_M^{k-4}(t) \mathbf{b}_M^{k-4}(t)^\top dt \ S_M^{k,3\top} \mathbf{C}_p^\top \right).\end{aligned}$$

Define

$$W_M^{k,j} \triangleq S_M^{k,j} \int_0^M \mathbf{b}_M^{k-j}(t) \mathbf{b}_M^{k-j}(t)^\top dt \ S_M^{k,j\top},$$

and note that $W_M^{k,j}$ are constant matrices that can be pre-computed and stored in memory, then

$$\begin{aligned}\int_0^M \|\mathbf{p}^{(4)}(t)\|^2 dt &= \text{tr} \left(\mathbf{C}_p W_M^{k,4} \mathbf{C}_p^\top \right) \\ \int_0^M |\psi^{(2)}(t)|^2 dt &= \text{tr} \left(\mathbf{C}_\psi W_M^{k,1} \mathbf{C}_\psi^\top \right).\end{aligned}$$

The initial and final conditions place constraints on the control points as shown in Corollary 5.2.14. These conditions can be stated as matrix equality constraints on the control points. For example, for fourth order splines, where the initial and final position, velocity, and

acceleration are specified, we have from Corollary 5.2.14 that

$$\begin{aligned}\mathbf{c}_0 &= \mathbf{p}_0 \\ \mathbf{c}_1 &= \mathbf{p}_0 + \frac{1}{k-1} \mathbf{v}_0 \\ \mathbf{c}_2 &= \mathbf{p}_0 + \frac{3}{k-1} \mathbf{v}_0 + \frac{2}{(k-1)(k-2)} \mathbf{a}_0 \\ \mathbf{c}_{M+k-4} &= \mathbf{p}_f - \frac{3}{k-1} \mathbf{v}_f + \frac{2}{(k-1)(k-2)} \mathbf{a}_f \\ \mathbf{c}_{M+k-3} &= \mathbf{p}_f - \frac{1}{k-1} \mathbf{v}_f \\ \mathbf{c}_{M+k-2} &= \mathbf{p}_f.\end{aligned}$$

which can be written in matrix form as

$$\begin{aligned}&\underbrace{\left(\mathbf{c}_0 \quad \dots \mathbf{c}_{M+k-2}\right)}_{\mathbf{C}_p} \underbrace{\left(\mathbf{e}_1 \quad \mathbf{e}_2 \quad \mathbf{e}_3 \quad \mathbf{e}_{M+k-3} \quad \mathbf{e}_{M+k-2} \quad \mathbf{e}_{M+k-1}\right)}_{U_1} \\ &= \underbrace{\left(\mathbf{p}_0 \quad \mathbf{v}_0 \quad \mathbf{a}_0 \quad \mathbf{a}_f \quad \mathbf{v}_f \quad \mathbf{p}_f\right)}_{A_p} \underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & \frac{1}{k-1} & \frac{3}{k-1} & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{(k-1)(k-2)} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{(k-1)(k-2)} & 0 & 0 \\ 0 & 0 & 0 & \frac{-3}{k-1} & \frac{-1}{k-1} & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}}_{B^{k,4}}.\end{aligned}$$

The minimum-snap problem without obstacles, can therefore be written as

$$\begin{aligned}&\min_{\mathbf{C}_p} \text{tr} \left(\mathbf{C}_p W_M^{k,4} \mathbf{C}_p^\top \right) \\ &\text{subject to } \mathbf{C}_p U_1 = A_p B^{k,4}.\end{aligned}\tag{5.31}$$

5.3.1 Minimum snap trajectories without obstacles

Without obstacles, the minimum-snap problem has an analytic solution as derived in the next theorem.

Theorem 5.3.1 *The optimization problem given in Equation (5.31) has an analytic solution given by*

$$\mathbf{C}_p^* = A_p B^{k,4} U_1^\top \left(I - W_M^{k,4} U_2 (U_2^\top W_M^{k,4} U_2)^{-1} U_2^\top \right), \tag{5.32}$$

where

$$U_2 = (\mathbf{e}_4, \dots, \mathbf{e}_{M+k-4}).$$

Proof: We begin by noting that $U_1^\top U_1 = I_{6 \times 6}$ and that $U_2^\top U_1 = 0$. Let $\check{\mathbf{C}}_p = A_p B^{k,4} U_1^\top + Z U_2^\top$ and note that

$$\begin{aligned}\check{\mathbf{C}}_p U_1 &= \left(A_p B^{k,4} U_1^\top + Z U_2^\top \right) U_1 \\ &= A_p B^{k,4} U_1^\top U_1 + Z U_2^\top U_1 \\ &= A_p B^{k,4}.\end{aligned}$$

Therefore $\check{\mathbf{C}}_p$ satisfies the inequality constraints and implies that the optimal solution has the form of

$$\mathbf{C}_p^* = A_p B^{k,4} U_1^\top + Z^* U_2^\top.$$

We have therefore transformed the constrained optimization problem in 5.31 to the unconstrained optimization problem

$$\begin{aligned}&\min_Z \text{tr} \left(\left(A_p B^{k,4} U_1^\top + Z U_2^\top \right) W_M^{k,4} \left(A_p B^{k,4} U_1^\top + Z U_2^\top \right)^\top \right) \\ &= \min_Z \text{tr} \left(A_p B^{k,4} U_1^\top W_M^{k,4} U_1 B^{k,4\top} A_p^\top + Z U_2^\top W_M^{k,4} U_1 B^{k,4\top} A_p^\top + A_p B^{k,4} U_1^\top W_M^{k,4} U_2 Z^\top + Z U_2^\top W_M^{k,4} U_2 Z^\top \right) \\ &= \min_Z \text{tr} \left(A_p B^{k,4} U_1^\top W_M^{k,4} U_1 B^{k,4\top} A_p^\top + 2 Z U_2^\top W_M^{k,4} U_1 B^{k,4\top} A_p^\top + Z U_2^\top W_M^{k,4} U_2 Z^\top \right),\end{aligned}$$

where we have used the fact that $\text{tr}(M^\top) = \text{tr}(M)$. Letting

$$J = \text{tr} \left(A_p B^{k,4} U_1^\top W_M^{k,4} U_1 B^{k,4\top} A_p^\top + 2 Z U_2^\top W_M^{k,4} U_1 B^{k,4\top} A_p^\top + Z U_2^\top W_M^{k,4} U_2 Z^\top \right)$$

and recalling that for matrix equations

$$\begin{aligned}\frac{\partial}{\partial X} \text{tr}(XM) &= M^\top \\ \frac{\partial}{\partial X} \text{tr}(XMX^\top) &= X(M + M^\top),\end{aligned}$$

we get that

$$\frac{\partial J}{\partial Z} = 2 A_p B^{k,4} U_1^\top W_M^{k,4} U_2 + 2 Z U_2^\top W_M^{k,4} U_2,$$

where we have used the fact that $W_M^{k,j}$ is symmetric. Setting $\frac{\partial J}{\partial Z}$ to zero and solving for the optimal Z gives

$$Z^* = -A_p B^{k,4} U_1^\top W_M^{k,4} U_2 (U_2^\top W_M^{k,4} U_2)^{-1}.$$

Therefore

$$\begin{aligned}\mathbf{C}_p^* &= A_p B^{k,4} U_1^\top + Z^* U_2^\top \\ &= A_p B^{k,4} U_1^\top + \left(-A_p B^{k,4} U_1^\top W_M^{k,4} U_2 (U_2^\top W_M^{k,4} U_2)^{-1} \right) U_2^\top \\ &= A_p B^{k,4} U_1^\top \left(I - W_M^{k,4} U_2 (U_2^\top W_M^{k,4} U_2)^{-1} U_2^\top \right).\end{aligned}$$

■

This is a very nice result in the sense that the matrix

$$Q_M^{k,4} = B^{k,4}U_1^\top \left(I - W_M^{k,4}U_2(U_2^\top W_M^{k,4}U_2)^{-1}U_2^\top \right)$$

is a constant, problem independent matrix that can be pre-computed and stored in memory, and the optimal control points can be computed simply from the initial and final conditions as

$$\mathbf{C}_p^* = A_p Q_M^{k,4}.$$

For the heading spline, we assume constraints on the initial and final heading as

$$\begin{aligned}\psi_0 &= \psi_d(0) = \mathbf{C}_\psi b_M^k(0) = c_{\psi,0} \\ \psi_f &= \psi_d(M) = \mathbf{C}_\psi b_M^k(M) = c_{\psi,M+k-2},\end{aligned}$$

which can be written in matrix form as

$$\underbrace{\begin{pmatrix} c_{\psi,0} & \dots & c_{\psi,M+k-2} \end{pmatrix}}_{\mathbf{C}_\psi} \underbrace{\begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_{M+k-1} \end{pmatrix}}_{U_{1,\psi}} = \underbrace{\begin{pmatrix} \psi_0 & \psi_f \end{pmatrix}}_{A_\psi} \underbrace{I_{2 \times 2}}_{B^{k,2}}.$$

Using the same logic as above, we have the following theorem for the heading spline.

Theorem 5.3.2 *The solution to the optimization problem*

$$\begin{aligned}\min_{\mathbf{C}_\psi} & \operatorname{tr} \left(\mathbf{C}_\psi W_M^{k,2} \mathbf{C}_\psi^\top \right) \\ \text{subject to } & \mathbf{C}_\psi U_{1,\psi} = A_\psi\end{aligned}\tag{5.33}$$

is given by

$$\mathbf{C}_\psi^* = A_\psi Q_M^{k,2},\tag{5.34}$$

where

$$\begin{aligned}Q_M^{k,2} &= U_{1,\psi}^\top \left(I - W_M^{k,2}U_{2,\psi}(U_{2,\psi}^\top W_M^{k,2}U_{2,\psi})^{-1}U_{2,\psi}^\top \right) \\ U_{2,\psi} &= (\mathbf{e}_2, \dots, \mathbf{e}_{M+k-2}).\end{aligned}$$

5.3.2 Minimum snap trajectories with obstacles

5.4 Ensuring dynamic feasibility using time scaling

The dynamics of a differentially flat system can be satisfied by planning in the flat output space. However, these dynamic models typically neglect system limitations such as actuator saturation. It is possible to include system limits in the trajectory optimization. The idea is to first plan the trajectory over the time horizon $[0, M]$ and then to check the force and torque constraints and then scale the trajectory so that the constraints are not violated.

Step 1. Plan the position and trajectories

$$\begin{aligned}\mathbf{p}_d(t) &= \mathbf{C}_p b_M^k(t) \\ \psi_d(t) &= \mathbf{C}_\psi b_M^k(t),\end{aligned}$$

to ensure collision avoidance constraints and objective satisfaction.

Step 2. Set the scaling coefficient to $\alpha = 1$, and set $\bar{\alpha} = 1$, $\underline{\alpha} = 0$

Step 3. Using the trajectories

$$\begin{aligned}\mathbf{p}_d(t) &= \mathbf{C}_p b_M^k(t/\alpha) \\ \psi_d(t) &= \mathbf{C}_\psi b_M^k(t/\alpha),\end{aligned}$$

compute the maximum thrust and torque using the differential flatness equations (5.8).

Step 4. If maximum thrust and torque are too large, set

$$\begin{aligned}\underline{\alpha} &\leftarrow \alpha \\ \bar{\alpha} &\leftarrow 2\alpha \\ \textit{alpha} &\leftarrow 2\alpha,\end{aligned}$$

and go to Step 3. If the maximum thrust and torque are too small, go to Step 5.

Step 5. Using the trajectories

$$\begin{aligned}\mathbf{p}_d(t) &= \mathbf{C}_p b_M^k(t/\alpha) \\ \psi_d(t) &= \mathbf{C}_\psi b_M^k(t/\alpha),\end{aligned}$$

compute the maximum thrust and torque using the differential flatness equations (5.8).

Step 6. If maximum thrust and torque are too large, set

$$\begin{aligned}\underline{\alpha} &\leftarrow \alpha \\ \alpha &\leftarrow \frac{\bar{\alpha} + \underline{\alpha}}{2},\end{aligned}$$

and go to Step 5 until convergence.

If the maximum thrust and torque are too small,

$$\begin{aligned}\bar{\alpha} &\leftarrow \alpha \\ \alpha &\leftarrow \frac{\bar{\alpha} + \underline{\alpha}}{2},\end{aligned}$$

and go to Step 5 until convergence.

This is an example of a golden, or bi-section search for the optimal α .

5.4.1 Curvature constraints

If $N > 4$, then Φ^\top has a non-trivial null space that can be exploited to satisfy additional constraints. As an example, we may want to minimize the curvature of the path.

The curvature of $p(\sigma)$ is defined by the formula

$$\kappa(\sigma) = \frac{p'(\sigma) \times p''(\sigma)}{\|p'(\sigma)\|^3}.$$

RWB: Figure out how this constraints the control points.

5.5 Path Planning using a Digital Elevation Map

Add a discussion similar to this paper.

citeManyamCasbeerWeintraub21: This paper uses a delaunay triangulation to parameterize the free space, and then moves control points within the delaunay triangles to satisfy constraints.

⁵: This paper gets a quick and dirty path using the fast marching method, and then creates boxes around the desired trajectory produces a safe path corridor. The b-spline paths are then optimized to stay within the corridor.

⁶: This paper shows how to get quick and dirty paths using the Fast Marching Method (FMM).

⁷: Collision avoidance constraints, as described in ⁸. If the obstacles are ellipsoids: Suppose that the world is modeled by a set of ellipsoidal obstacles

$$\mathcal{O}_i = \{x \in \mathbf{R}^3 : x^\top P_i x \leq 1\}$$

and the total set of obstacles in the world is

$$\mathcal{W} = \bigcup_{i=1}^M \mathcal{O}_i.$$

In this case, we want to ensure that none of the control points is in an obstacle, i.e., that

This paper shows how to formulate the obstacle avoidance problem as a convex problem using dual variables.

⁵ Fei Gao, William Wu, Yi Lin, and Shaojie Shen. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351, Brisbane, Australia, May 2018

⁶ Max Lutz and Thomas Meurer. Efficient formulation of collision avoidance constraints in optimization based trajectory planning and control. In *Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*, pages 228–233, San Diego, CA, August 2021

⁷ Max Lutz and Thomas Meurer. Efficient formulation of collision avoidance constraints in optimization based trajectory planning and control. In *Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*, pages 228–233, San Diego, CA, August 2021

Part II

Vision Based Flight

6

Camera Models and Feature Detection

Figure 6.1 shows a simple representation of a grayscale image. The image is M_x pixels wide, and M_y pixels in height. Image arrays are typically index from the top-left corner. For example in `openCV` the $(0, 0)$ pixel is the top-left pixel of the image, and the (M_x, M_y) pixel is in the bottom right. The center pixel will be denoted $(c_x, c_y) \approx (M_x/2, M_y/2)$.

The intensity of a grayscale image at pixel (m_x, m_y) will be denoted $I(M_x, m_y)$ and is typically represented as an integer value between 0 (black) and 255 (white).

6.1 Pinhole model

The geometry of camera is shown in Figure ???. The camera frame is represented as $\mathcal{F}_c = \{\mathbf{i}_c, \mathbf{j}_c, \mathbf{k}_c\}$, where \mathbf{i}_c is aligned with the m_x -direction in the image plane, i.e., to the right when looking at the image, \mathbf{j}_c is aligned with m_y -direction in the image plane, i.e., down when looking at the image, and \mathbf{k}_c is aligned along the optical axis, or in the pointing direction of the camera, and so that a scaled version of \mathbf{k}_c would intersect the image plane at the center of the image. The image plane is located at a distance Sf from the center of the camera frame, where f is the focal length in units of pixels, and S is a scale factor that converts pixels to meters.

Let $\mathbf{p}_{f/c}^c = (p_x, p_y, p_z)^\top$ be the 3D Euclidian position of a feature point in the environment relative to the camera frame, expressed in the camera frame, as shown in Figure ???. Suppose that the projection of the point $\mathbf{p}_{f/c}^c$ on to the camera frame is at the pixel (m_x, m_y) . The vector that represents the projection of $\mathbf{p}_{f/c}^c$ on the image plane is given by

$$\begin{pmatrix} S(m_x - c_x) \\ S(m_y - c_y) \\ Sf \end{pmatrix}.$$

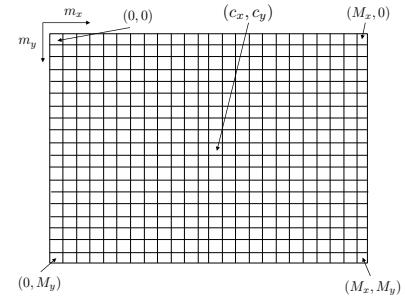


Figure 6.1: Image formation represented as pixels

In this book we will work exclusively with grayscale images. Most cameras return color image, which consists of three $M_x \times M_y$ arrays representing, for example B(blue), G(green), R(red). In `openCV` BGR images can be converted to grayscale using the command `grayimage=cv2.cvtColor(colorimage, cv2.COLOR_BGR2GRAY)`.

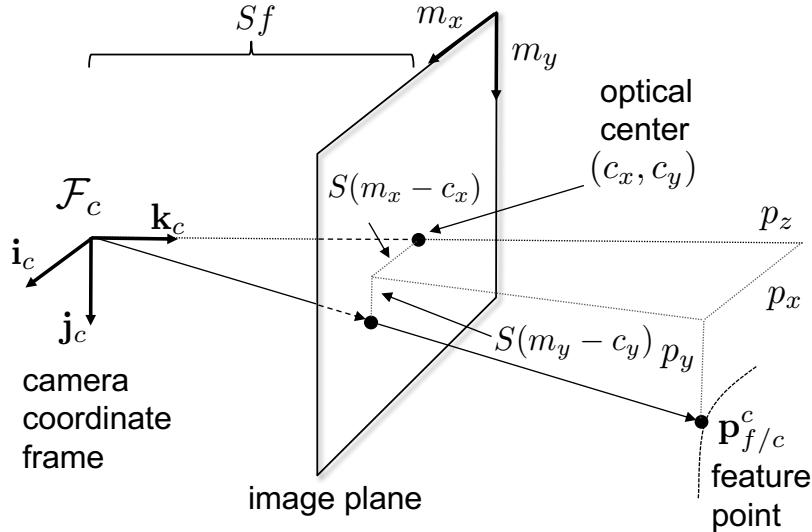


Figure 6.2: Pin hole camera model

From Figure ?? and using similar triangles, we get that

$$\begin{aligned}\frac{S(m_x - c_x)}{Sf} &= \frac{p_x}{p_z} \\ \frac{S(m_y - c_y)}{Sf} &= \frac{p_y}{p_z},\end{aligned}$$

or equivalently

$$\begin{aligned}m_x &= c_x + f \frac{p_x}{p_z} \\ m_y &= c_y + f \frac{p_y}{p_z}.\end{aligned}$$

In order to write the pixel location as vector, we introduce *normalized homogeneous coordinates* to write

$$\begin{aligned}\begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix} &= \begin{pmatrix} c_x(p_z/p_z) + f \frac{p_x}{p_z} \\ c_y(p_z/p_z) + f \frac{p_y}{p_z} \\ (p_z/p_z) \end{pmatrix} \\ &= \frac{1}{p_z} \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}. \quad (6.1)\end{aligned}$$

For real-world cameras, the focal length can be different in the x and y directions. Consider Figure 6.3.

Suppose that the camera field of view in the x direction is φ_x , and

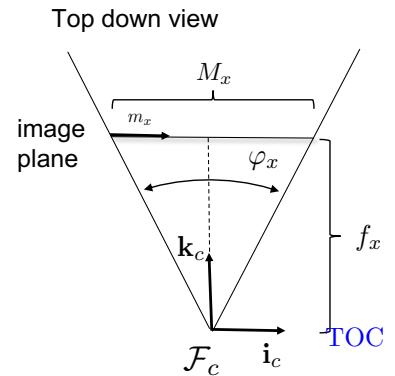


Figure 6.3: Calculation of the focal length in the image x -direction.

that the size of image array in the x -direction is M_x . Then we have

$$\begin{aligned}\tan\left(\frac{\varphi_x}{2}\right) &= \frac{(M_x/2)}{f_x} \\ \implies f_x &= \frac{M_x}{2\tan\left(\frac{\varphi_x}{2}\right)}.\end{aligned}$$

Therefore Equation (6.1) becomes

$$\begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix} = \frac{1}{p_z} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}.$$

It turns out that for real cameras, not only are the pixels not square, but they can also be skewed. The skew factor can be represented by¹

$$\begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix} = \frac{1}{p_z} \begin{pmatrix} f_x & f_\theta & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}.$$

where f_θ is also in units of pixels. Defining

$$K_c \doteq \begin{pmatrix} f_x & f_\theta & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

to be the intrinsic parameters of the camera, we have shown that

$$\lambda_f \begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix} = K_c \mathbf{p}_{f/c}^c, \quad (6.2)$$

where $\lambda_f = p_z$ is the (unknown) distance to the feature along the camera z -axis.

The intrinsic parameters K_c (in addition to radial distortion parameters) can be found through a process known as camera calibration. If the camera has been calibrated, then K_c is known. When K_c is known, then pixel coordinates returned via openCV can be converted to calibrated pixel coordinates as

$$\begin{pmatrix} \epsilon_x \\ \epsilon_y \\ 1 \end{pmatrix} = K_c^{-1} \begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix}, \quad (6.3)$$

where we have the simplified formula²

$$\lambda_f \bar{\epsilon}_{f/c}^c = \mathbf{p}_{f/c}^c.$$

If pixel coordinates are in the form $\mathbf{m} = (m_x, m_y, m_z)^\top$ where m_z is arbitrary, then we say that the pixel is represented in *homogeneous*

¹ Yi Ma, Stefano Soatto, Jan Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003

In ROSflight-Holodeck, the horizontal field-of-view of the camera is 90 degrees from which f_x can be computed. Assume that $f_y = f_x$ and $f_\theta = 0$. The optical center can be computed as $c_* = M_x/2$.

² Note that in this formula ϵ_x and ϵ_y are unitless since K_c , m_x , and m_y have units of pixels.

coordinates. If the last element is equal to one, then we say that the pixel is in *normalized homogeneous coordinates*. If the pixel coordinates have been calibrated as in Equation (6.3), then we say that the pixel is in *calibrated normalized homogeneous coordinates*. Unless otherwise stated, we will assume pixel coordinates are represented using calibrated normalized homogeneous coordinates.

Recall from Section 2.6 that position vectors can be presented in homogeneous coordinates by appending a 1 as the fourth element. In homogeneous coordinates we have³

$$\lambda_f \bar{\epsilon}_{f/c}^c = \Pi_0 \bar{p}_{f/c}^c,$$

where

$$\Pi_0 \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Recall also from Section 2.6 that the position of the feature in the camera frame, can be related to the position of the feature in the inertial frame via the homogeneous transformation

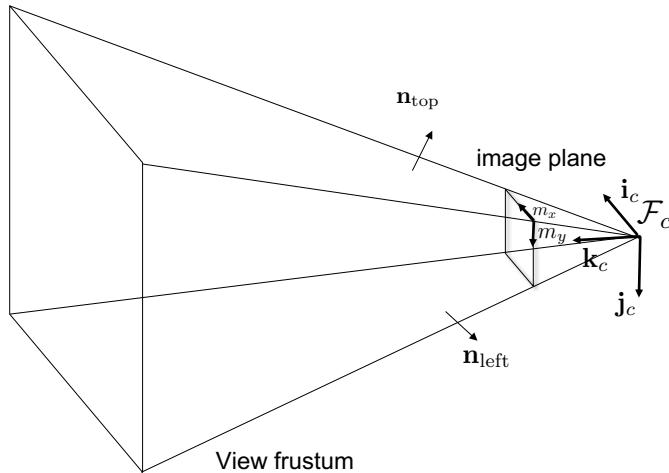
$$\bar{p}_{f/c}^c = T_i^c \bar{p}_{f/i}^i = \begin{pmatrix} R_i^c & \mathbf{t}_{i/c}^c \\ 0 & 1 \end{pmatrix} \bar{p}_{f/i}^i.$$

Therefore, the calibrated camera model becomes

$$\lambda_f \bar{\epsilon}_{f/c}^c = \Pi_0 T_i^c \bar{p}_{f/i}^i. \quad (6.4)$$

6.2 Camera Field of View

Figure ?? shows the field of view of a pin-hole camera. The view



³ Yi Ma, Stefano Soatto, Jan Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003

Figure 6.4: The field of view of a pin-hole camera is a pyramidal frustum.

frustum originates at the origin of the camera coordinate frame \mathcal{F}_c .

The field of view can be represented as the intersection of four half planes in \mathbb{R}^3 . The half planes are described by the equations

$$\begin{aligned}\mathbf{n}_{\text{right}}^{i\top}(\mathbf{r} - \mathbf{p}_{c/i}^i) &\leq 0 \\ \mathbf{n}_{\text{left}}^{i\top}(\mathbf{r} - \mathbf{p}_{c/i}^i) &\leq 0 \\ \mathbf{n}_{\text{top}}^{i\top}(\mathbf{r} - \mathbf{p}_{c/i}^i) &\leq 0 \\ \mathbf{n}_{\text{bottom}}^{i\top}(\mathbf{r} - \mathbf{p}_{c/i}^i) &\leq 0\end{aligned}$$

where $\mathbf{p}_{c/i}^i$ is the position of the origin of the camera frame with respect to the inertial frame, expressed in the inertial frame, and \mathbf{n}_*^i are unit vectors that are perpendicular to the sides of the view frustum, expressed in the inertial frame.

Since the unit vectors \mathbf{n}_* are fixed in the camera frame, we can write

$$\mathbf{n}_*^i = R_c^i \mathbf{n}_*^c.$$

Defining the matrix

$$N = \begin{pmatrix} \mathbf{n}_{\text{right}}^c & \mathbf{n}_{\text{left}}^c & \mathbf{n}_{\text{top}}^c & \mathbf{n}_{\text{bottom}}^c \end{pmatrix},$$

then the view frustum \mathcal{F} is given by

$$\mathcal{F}(\mathbf{p}_{c/i}^i, R_c^i) = \{\mathbf{r} \in \mathbb{R}^3 : N^\top (R_c^i)^\top (\mathbf{r} - \mathbf{p}_{c/i}^i) \leq \mathbf{0}\}.$$

To find the unit vectors \mathbf{n}_* , consider the top down view of the camera field of view shown in Figure ??.

Let M_x be the number of pixels along the camera x -direction, and let f be the focal length given in pixels. Then the angular arc defining the field of view in the x -direction is given by

$$\varphi_x = 2 \tan^{-1} \left(\frac{M_x}{2f} \right).$$

Accordingly, from geometry we have

$$\begin{aligned}\mathbf{n}_{\text{right}} &= \left(\cos \left(\frac{\varphi_x}{2} \right), 0, -\sin \left(\frac{\varphi_x}{2} \right) \right)^\top \\ \mathbf{n}_{\text{left}} &= \left(-\cos \left(\frac{\varphi_x}{2} \right), 0, -\sin \left(\frac{\varphi_x}{2} \right) \right)^\top.\end{aligned}$$

Similar arguments give

$$\begin{aligned}\mathbf{n}_{\text{top}} &= \left(0, -\cos \left(\frac{\varphi_y}{2} \right), -\sin \left(\frac{\varphi_y}{2} \right) \right)^\top \\ \mathbf{n}_{\text{bottom}} &= \left(0, \cos \left(\frac{\varphi_y}{2} \right), -\sin \left(\frac{\varphi_y}{2} \right) \right)^\top,\end{aligned}$$

where

$$\varphi_y = 2 \tan^{-1} \left(\frac{M_y}{2f} \right),$$

and M_y is the number of pixels in the camera y -direction.

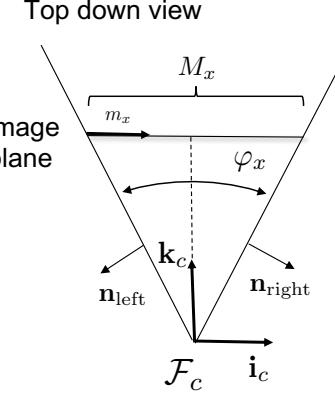


Figure 6.5: A top down view of the field of view.

Lemma 6.2.1 *If the size of the pixel array is given by $M_x \times M_y$, and the camera calibration matrix is given by*

$$K_c = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

then the normalized homogeneous coordinates

$$\boldsymbol{\epsilon} = (\epsilon_x, \epsilon_y, 1)^\top$$

are in the camera field of view if and only if

$$\begin{aligned} -\frac{c_x}{f_x} \leq \epsilon_x \leq \frac{M_x - c_x}{f_x} \\ -\frac{c_y}{f_y} \leq \epsilon_y \leq \frac{M_y - c_y}{f_y}. \end{aligned}$$

Proof: The pixel m_* is in the field of view if and only if

$$0 \leq m_* \leq M_*.$$

The result follows from simple manipulation using the fact that $m_* = f_* \epsilon_* + c_*$. ■

6.3 Some notes on Points and lines in images

See ⁴

A point in an image is actually a line in 3D space. Figure ?? shows a normalized pixel $\boldsymbol{\epsilon}$ in the image plane. Note that in fact the point $\boldsymbol{\epsilon}$ defines a line (or linear space) in \mathbb{R}^3 given by $k\boldsymbol{\epsilon}$ that originates at the origin of \mathcal{F}_c and passes through $\boldsymbol{\epsilon}$. We say that two points given in homogenous coordinates are equivalent if they are equal up to a scale factor, i.e., we write $\mathbf{x} = \mathbf{y}$ if $\mathbf{x} = \lambda\mathbf{y}$ for some λ .

Define the two dimensional projective space \mathbb{P}^2 to be the set of all homogeneous coordinates.⁵ Therefore if $\mathbf{x}, \mathbf{y} \in \mathbb{P}^2$, then $\mathbf{x} = \mathbf{y}$ if $\mathbf{x} = \lambda\mathbf{y}$ for some $\lambda \in \mathbb{R}$. Note that if $\mathbf{x} = \mathbf{y}$, then their vector components will be equal if they are placed in normalized homogeneous coordinates.

The equation for a line in the image plane is given by

$$\epsilon_y = m\epsilon_x + b,$$

where m is the slope and b is the intercept with the y -axis. More generally, a line in the image plane is given by

$$a\epsilon_x + b\epsilon_y + c = 0,$$

⁴ Yi Ma, Stefano Soatto, Jan Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003

⁵ Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003

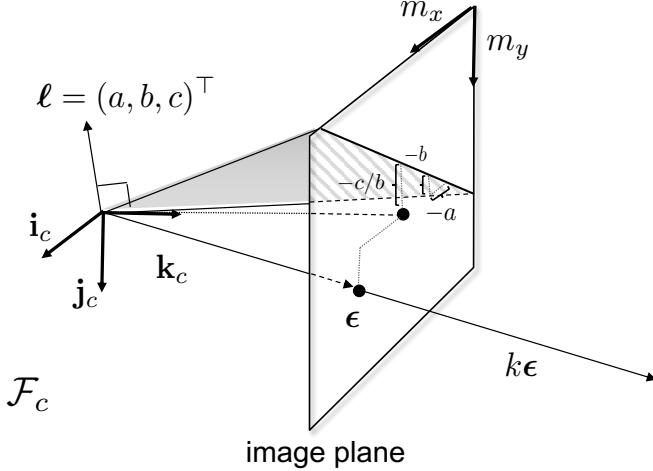


Figure 6.6: A point ϵ in the image is a linear space in \mathbb{R}^3 , and a line ℓ is the image.

thee the slope is $-a/b$ and the y -intercept is $-c/b$. The line is depicted graphically in Figure ???. Let $\ell \stackrel{\triangle}{=} (a, b, c)^\top$ and note that ϵ is on the line ℓ if and only if

$$\ell^\top \epsilon = 0.$$

In 3D space, the line ℓ in the image plane is the projection of a 2D plane in 3D that passes through the origin of \mathcal{F}_c and intersects the image plane at the line. The 3D vector ℓ is normal to this plane. For example, the line along the image x -axis is given by $\epsilon_y = 0$, and the corresponding line is

$$0\epsilon_x + 1\epsilon_y + 0 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}^\top \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ 1 \end{pmatrix} = 0.$$

The line $\ell = (0, 1, 0)^\top$ is perpendicular to the plane that intersects the image plane along the x -axis. Since the scaling of ℓ doesn't change the fact that it is parallel to the corresponding plane, we see that $\ell \in \mathbb{P}^2$ defines a line in the image (i.e., for any scale). Therefore, both lines and points can be represented by homogenous coordinates.

Suppose that there are two lines in the image given by $\ell_1, \ell_2 \in \mathbb{P}^2$, then if the lines are not parallel, the associated planes will intersect at a line given by $\ell_1 \times \ell_2$ which corresponds to the image point $\epsilon = \ell_1 \times \ell_2$, where $\epsilon \in \mathbb{P}^2$ is given in homogeneous coordinates. If the lines are parallel, then their slopes are equal, but the y -intercept may be different. Therefore, parallel lines will have the form

$$\ell_1 = \begin{pmatrix} a \\ b \\ c_1 \end{pmatrix}, \quad \ell_2 = \begin{pmatrix} ka \\ kb \\ kc_2 \end{pmatrix},$$

and the cross product is given by

$$\boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2 = \begin{pmatrix} kb(c_2 - c_1) \\ ka(c_1 - c_2) \\ 0 \end{pmatrix}.$$

Therefore, in homogeneous coordinates, a zero in the third element represents point at infinity.⁶

Given two image points represented in homogeneous coordinates $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2 \in \mathbb{P}^2$, the line in the image that contains both of these points will be defined by a plane that contains both rays. A perpendicular vector to that plane is therefore given by

$$\boldsymbol{\ell} = \boldsymbol{\epsilon}_1 \times \boldsymbol{\epsilon}_2 \in \mathbb{P}^2.$$

We have therefore shown the following theorem.

Theorem 6.3.1 *Let $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2 \in \mathbb{P}^2$ represent two points in the image plane, and let $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \in \mathbb{P}^2$ represent two lines in the image plane. Then*

1. *The point $\boldsymbol{\epsilon}_1$ is on the line $\boldsymbol{\ell}_1$ if and only iff $\boldsymbol{\epsilon}_1^\top \boldsymbol{\ell}_1 = 0$.*
2. *The line that contains both $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ is given in homogeneous coordinates as $\boldsymbol{\ell} = \boldsymbol{\epsilon}_1 \times \boldsymbol{\epsilon}_2$.*
3. *The lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ intersect at the point given in homogeneous coordinates as $\boldsymbol{\epsilon} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$.*

6.4 Feature Detection

RWB: Need to add some images with features to this section.

The image based control and estimation schemes that we will consider in this book will primarily be based on feature detection and feature tracking. In this section we discuss the Harris corner detector and the related Shi-Tomasi corner detector.

The Harris corner detector attempts to find positions in the image that have strong gradients in both the x and y directions. Consider the function

$$E'_{\mathbf{m}}(u, v) = |I(m_x + u, m_y + v) - I(m_x, m_y)|^2$$

which is a function of pixel deviations (u, v) . For example, if $E'_{\mathbf{m}}(1, 0)$ is large, then the image intensity undergoes a large change for one pixel deviation along the x -direction in the image. It is desireable to find features that are well-defined over a window W of pixels. For example, noise may cause strong gradients at one pixel in the image, but the gradient in each direction does not persist for more than a few

⁶ Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003

The `openCV` function `goodfeaturestotrack` uses the Shi-Tomasi algorithm to find good features to track. A nice overview of Features and the Shi-Tomasi feature detector is given at <https://aishack.in/tutorials/features>.

pixels. Define the window $W(\mathbf{m})$ to be an $(N + 1) \times (N + 1)$ box around the pixel \mathbf{m} . Consider the cost function

$$E_{\mathbf{m}}(u, v) = \sum_{(x,y) \in W(\mathbf{m})} |I(x + u, y + v) - I(x, y)|^2,$$

then good features will be those pixels \mathbf{m} where E is large. Using the Taylor series approximation of image intensity $I(x, y)$ we get

$$I(x + u, y + v) = I(x, y) + \frac{\partial I}{\partial x}(x, y)u + \frac{\partial I}{\partial y}(x, y)v + H.O.T. \quad (6.5)$$

Given an image $I(x, y)$, the gradient image along the x -direction can be found by convolving the image with the so-called Sobel edge detector

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Similarly, the gradient along the y -direction can be found by convolving the image with

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Equivalently, the gradient $\frac{\partial I}{\partial \mathbf{m}}$ of an image at pixel $\mathbf{m}^0 = (m_x^0, m_y^0)^\top$ is given by

$$\frac{\partial I}{\partial \mathbf{m}}(\mathbf{m}^0) \triangleq \begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} [I(m_x^0 + 1, m_y^0 - 1) - I(m_x^0 - 1, m_y^0 - 1)] \\ +2[I(m_x^0 + 1, m_y^0) - I(m_x^0 - 1, m_y^0)] \\ +[I(m_x^0 + 1, m_y^0 + 1) - I(m_x^0 - 1, m_y^0 + 1)] \\ [I(m_x^0 - 1, m_y^0 + 1) - I(m_x^0 - 1, m_y^0 - 1)] \\ +2[I(m_x^0, m_y^0 + 1) - I(m_x^0, m_y^0 - 1)] \\ +[I(m_x^0 + 1, m_y^0 + 1) - I(m_x^0 + 1, m_y^0 - 1)] \end{pmatrix}. \quad (6.6)$$

Therefore, dropping all be the constant and linear terms in Equation (6.5) gives

$$\begin{aligned} E_{\mathbf{m}}(u, v) &\approx \sum_{(x,y) \in W(\mathbf{m})} |I_x(x, y)u + I_y(x, y)v|^2, \\ &= \sum_{(x,y) \in W(\mathbf{m})} \begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} (I_x(x, y))^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & (I_y(x, y))^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \\ &= \begin{pmatrix} u & v \end{pmatrix} G(\mathbf{m}) \begin{pmatrix} u \\ v \end{pmatrix}, \end{aligned}$$

where

$$G(\mathbf{m}) = \sum_{(x,y) \in W(\mathbf{m})} \begin{pmatrix} (I_x(x, y))^2 & I_x(x, y)I_y(x, y) \\ I_x(x, y)I_y(x, y) & (I_y(x, y))^2 \end{pmatrix}. \quad (6.7)$$

We are therefore interested in pixels \mathbf{m} where $\mathbf{x}^\top G(\mathbf{m})\mathbf{x}$ is large, where $\mathbf{x} \in \mathbb{R}^2$ is a unit vector. Since $G(\mathbf{m})$ is symmetric, its eigen-decomposition is given by

$$G = U \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} U^\top,$$

where U is orthogonal. Therefore $\mathbf{x}^\top G(\mathbf{m})\mathbf{x}$ is large for all \mathbf{x} when both of the eigenvalues of $G(\mathbf{m})$ are large.

6.4.1 Harris Corner Detector

The so-called Harris corner detector, originally proposed by (Harris & Stephens)⁷ uses the metric

$$R_h = \det G - k \operatorname{tr}(G)$$

to distinguish good features. Since for any matrix

$$\det A = \prod \lambda_i$$

$$\operatorname{tr}(A) = \sum \lambda_i,$$

it follows that

$$R(G) = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2),$$

where k is typically small. Therefore R will be large when both λ_1 and λ_2 are relatively large and close to the same size.

⁷ Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988

6.4.2 Shi-Tomasi Corner Detector

The Shi-Tomasi corner detector, originally proposed in (Shi & Tomasi)⁸ instead uses the metric

$$R_{st}(G) = \min\{\lambda_1, \lambda_2\}$$

⁸ Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, pages 593–600. IEEE, 1994

to distinguish good features. In this case, $R(G)$ will be above a threshold when both eigenvalues are above that threshold. The eigenvalues of G can be found explicitly as

$$\lambda_1 = \frac{1}{2} \left(\sum_W I_x^2 + \sum_W I_y^2 \right) + \frac{1}{2} \sqrt{\left(\sum_W I_x^2 - \sum_W I_y^2 \right)^2 + 4 \left(\sum_W I_x \right)^2 \left(\sum_W I_y \right)^2}$$

$$\lambda_2 = \frac{1}{2} \left(\sum_W I_x^2 + \sum_W I_y^2 \right) - \frac{1}{2} \sqrt{\left(\sum_W I_x^2 - \sum_W I_y^2 \right)^2 + 4 \left(\sum_W I_x \right)^2 \left(\sum_W I_y \right)^2},$$

which implies that

$$R_{st}(G(\mathbf{m})) = \frac{1}{2} \left(\sum_W I_x^2 + \sum_W I_y^2 \right) - \frac{1}{2} \sqrt{\left(\sum_W I_x^2 - \sum_W I_y^2 \right)^2 + 4 \left(\sum_W I_x \right)^2 \left(\sum_W I_y \right)^2}.$$

Python code for finding good features to track is shown here ⁹

⁹ OpenCV function

7

Optical Flow and Feature Tracking

This chapter discusses two related algorithms. The first, discussed in Section 7.2, is the computation of optical flow at a pixel. The optical flow field can then be computed by finding the optical flow at a set of pixel locations. The second algorithm discussed in Section 7.3 is the KLT feature tracking algorithm, that uses an iterated computation of optical flow, to find how specific features have moved from one frame to the next. Section 7.4 describes an extension of the KLT algorithm that uses the IMU to enable tracking across large camera motion, as might occur during aggressive flight. The final section in this chapter, Section 7.5 describes simple guidance algorithms for multirotors that use the optical flow field to maneuvers through canyons and around obstacles.

7.1 Levenberg-Marquardt (LM) Optimization

RWB: Probably put this somewhere else. Maybe in the preliminaries chapter.

The Levenberg-Marquart (LM) algorithm is a standard technique for solving nonlinear least squares problems of the form

$$\beta^* = \arg \min r(z, \beta)^\top r(z, \beta),$$

where $z \in Z$ is a known data vector, $\beta \in B$ is a parameter vector, and $r : Z \times B \rightarrow \mathbb{R}^N$ is a residual vector to be minimized. The optimization problem is solved by iteratively incrementing β by a small δ in a direction that decreases the squared residual

$$S(\beta) = r(x, \beta)^\top r(z, \beta).$$

Using the approximation

$$r(z, \beta + \delta) \approx r(z, \beta) + J(z, \beta)\delta,$$

where

$$J(z, \beta) \triangleq \frac{\partial r}{\partial \beta}(z, \beta)$$

is the Jacobian of r with respect to the parameters β , we get that

$$S(\beta + \delta) \approx [r(z, \beta) - J(z, \beta)\delta]^\top [r(z, \beta) - J(z, \beta)\delta].$$

Taking the partial of $S(\beta + \delta)$ with respect to δ and setting equal to zero gives

$$J^\top r = J^\top J\delta. \quad (7.1)$$

Solving for δ from this equation results in the Gauss-Newton optimization method. In contrast, gradient descent replaces the right hand side of Equation (7.1) by $\lambda\delta$, where λ is a scalar. The LM algorithm is a hybrid between Gauss Newton and gradient descent, requiring δ to satisfy

$$J^\top r = (J^\top J + \lambda I)\delta. \quad (7.2)$$

The general LM optimization algorithm is therefore given by the iteration

$$\beta_{\ell+1} = \beta_\ell + \delta_\ell, \quad (7.3)$$

where

$$\delta_\ell = \left[J(z, \beta_\ell)^\top J(z, \beta_\ell) + \lambda_\ell I \right]^{-1} J(z, \beta_\ell)^\top r(z, \beta_\ell),$$

where λ_ℓ is selected at each iteration to ensure that the residual $r(z, \beta)$ decreases.

7.2 Optical Flow

Suppose that a camera attached to a multirotor collects image I_k at time k and image I_{k+1} at time I_{k+1} , and suppose that $\mathbf{m}_0 = (m_x, m_y)^\top$ is a pixel of interest in image I_k . The objective is to find the corresponding pixel in image I_{k+1} using only image data. We assume that lighting qualities in the scene do not change between images I_k and I_{k+1} .

Let $\boldsymbol{\delta} = (\delta_x, \delta_y)^\top$ be a small change in pixel locations, then the objective is to find $\boldsymbol{\delta}$ to minimize the residual

$$r(I_k, I_{k+1}, \mathbf{m}_0, \boldsymbol{\delta}) = I_{k+1}(\mathbf{m}_0 + \boldsymbol{\delta}) - I_k(\mathbf{m}_0), \quad (7.4)$$

where $\mathbf{m}_0 + \boldsymbol{\delta}$ is the pixel location in the new image of the feature located at \mathbf{m}_0 in the old image. It turns out that finding the optical flow at a single pixel is under-determined and susceptible to noise. Therefore, the residual function is modified to minimize the intensity values over a window surrounding \mathbf{m}_0 as

$$r(I_k, I_{k+1}, \mathbf{m}_0, \boldsymbol{\delta}) = \begin{pmatrix} (I_{k+1}(\mathbf{x}_1 + \boldsymbol{\delta}) - I_k(\mathbf{x}_1)) \\ \vdots \\ (I_{k+1}(\mathbf{x}_n + \boldsymbol{\delta}) - I_k(\mathbf{x}_n)) \end{pmatrix} \quad (7.5)$$

where $\mathbf{x}_1, \dots, \mathbf{x}_n$ are the pixels in the window $W(\mathbf{m}_0)$ around the pixel \mathbf{m}_0 .

Following the previous section and taking the Taylor series expansion of the residual gives

$$\begin{aligned} r(I_k, I_{k+1}, \mathbf{m}_0, \boldsymbol{\delta}) &\approx \begin{pmatrix} I_k(\mathbf{x}_1) + \frac{\partial I_k}{\partial t}(\mathbf{x}_1) + \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x}_1)\boldsymbol{\delta} + H.O.T. - I_k(\mathbf{x}_1) \\ \vdots \\ I_k(\mathbf{x}_n) + \frac{\partial I_k}{\partial t}(\mathbf{x}_n) + \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x}_n)\boldsymbol{\delta} + H.O.T. - I_k(\mathbf{x}_n) \end{pmatrix} \\ &\approx \begin{pmatrix} \frac{\partial I_k}{\partial t}(\mathbf{x}_1) + \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x}_1)\boldsymbol{\delta} \\ \vdots \\ \frac{\partial I_k}{\partial t}(\mathbf{x}_n) + \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x}_n)\boldsymbol{\delta} \end{pmatrix} \end{aligned}$$

where the computation of $\frac{\partial I}{\partial \mathbf{m}}$ is given in Equation (6.6), and where

$$\frac{\partial I_k}{\partial t}(\mathbf{x}_i) \approx I_{k+1}(\mathbf{x}_i) - I_k(\mathbf{x}_i),$$

where the units of $\frac{\partial I_k}{\partial t}$ is intensity/sample, the units of $\frac{\partial I_k}{\partial \mathbf{m}}$ is intensity/pixels, and the units of $\boldsymbol{\delta}$ is pixels/sample.

The optical flow problem can now be stated as the problem of finding the optical flow vector $\boldsymbol{\delta}$ to minimize the squared error of the residual function. Letting $S = r^\top r$ we get

$$\begin{aligned} S(I_k, I_{k+1}, \mathbf{m}_0, \boldsymbol{\delta}) &= \sum_{\mathbf{x} \in M(\mathbf{m}_0)} \left(\frac{\partial I_k}{\partial t}(\mathbf{x}) + \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x})\boldsymbol{\delta} \right)^\top \left(\frac{\partial I_k}{\partial t}(\mathbf{x}) + \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x})\boldsymbol{\delta} \right) \\ &= \boldsymbol{\delta}^\top \left(\sum_{\mathbf{x} \in M(\mathbf{m}_0)} \frac{\partial I_k}{\partial \mathbf{m}}(\mathbf{x}) \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x}) \right) \boldsymbol{\delta} + 2 \left(\sum_{\mathbf{x} \in M(\mathbf{m}_0)} \frac{\partial I_k}{\partial t}(\mathbf{x}) \frac{\partial I_k^\top}{\partial \mathbf{m}}(\mathbf{x}) \right)^\top \boldsymbol{\delta} + \left(\sum_{\mathbf{x} \in M(\mathbf{m}_0)} \frac{\partial I_k^2}{\partial t}(\mathbf{x}) \right) \\ &= \boldsymbol{\delta}^\top G(\mathbf{m}_0)\boldsymbol{\delta} + 2b^\top(\mathbf{m}_0)\boldsymbol{\delta} + c, \end{aligned}$$

where $G(\mathbf{m}_0)$ is given in Equation (6.7), and where

$$\begin{aligned} b(\mathbf{m}_0) &\triangleq \left(\sum_{\mathbf{x} \in W(\mathbf{m}_0)} \frac{\partial I}{\partial \mathbf{m}}(\mathbf{x}) \frac{\partial I}{\partial t} \right), \\ c(\mathbf{m}_0) &\triangleq \sum_{\mathbf{x} \in M(\mathbf{m}_0)} \frac{\partial I_k^2}{\partial t}(\mathbf{x}). \end{aligned}$$

Taking the partial of S with respect to $\boldsymbol{\delta}$ and setting equal to zero gives

$$\boldsymbol{\delta} = G^{-1}(\mathbf{m}_0)b(\mathbf{m}_0). \quad (7.6)$$

Of course this equation requires that $G(\mathbf{m}_0)$ is full rank, i.e., $\text{rank}(G(\mathbf{m}_0)) =$

2. If \mathbf{m}_0 is a `goodfeaturetotrack` then the eigenvalues are both positive and $G(\mathbf{m}_0)$ is full rank. However, for pixel locations that do not

have gradients in both directions, $G(\mathbf{m}_0)$ will lose rank and the optical flow vector cannot be computed at that point.

As described in the previous section, the Gauss-Newton iteration gives

$$\mathbf{m}_{\ell+1} = \mathbf{m}_\ell + G^{-1}(\mathbf{m}_\ell)b(\mathbf{m}_\ell).$$

The method described above assumes that the intensity at pixel \mathbf{m}_0 moves a small distance that is at least contained in the window $W(\mathbf{m}_0)$, and the method can fail for large motions. **RWB: Need to describe image pyramids**. One method for compensating for this problem is to compute a set of image pyramids and to compute the optical flow vector on a lower resolution image in the pyramid, where small optical flow vector now correspond to larger motions in the image. The optical flow algorithm is then computed at higher resolutions using the seed from the lower resolution image.

The `openCV` function `calcOpticalFlowPyrLK()` computes the optical flow at a set of points passed into the function.¹

Our discussion to this point has used uncalibrated pixel coordinates $\mathbf{m} = (m_x, m_y)^\top$. Equation (7.6) gives the optical flow vector and can be written as

$$\dot{\mathbf{m}} = (\dot{m}_x, \dot{m}_y)^\top = \boldsymbol{\delta}(\mathbf{m}).$$

In calibrated normalized pixel coordinates we have

$$\bar{\boldsymbol{\epsilon}} = K_c^{-1}\dot{\mathbf{m}} = K_c^{-1} \begin{pmatrix} m_x \\ m_y \\ 1 \end{pmatrix}.$$

Therefore, the optic flow vector in calibrated normalized pixel coordinates is given by

$$\dot{\boldsymbol{\epsilon}} = \begin{pmatrix} \dot{\epsilon}_x \\ \dot{\epsilon}_y \\ 0 \end{pmatrix} = K_c^{-1} \begin{pmatrix} \dot{m}_x \\ \dot{m}_y \\ 0 \end{pmatrix} = K_c^{-1}\dot{\mathbf{m}},$$

and the units of $\dot{\boldsymbol{\epsilon}}$ are seconds⁻¹.

7.3 KLT Feature Tracking

The KLT feature tracking method, which was first reported in (Lucas & Kanade)², is different than the optical flow method described in Section 7.2 in two ways. First, the optical flow vector is computed at a good-feature-to-track. Therefore, it has already been determined that $G(\mathbf{m})$ is invertible. The second difference is that the optical flow vector is iteratively refined, rather than simply solving Equation (7.6) one time.

¹ `openCV` code example.

² Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the Imaging Understanding Workshop*, pages 121–130, 1981

KLT stands for "Kanade, Lucas, Tomasi" and represents the fact that the method explicitly uses Shi-Tomasi features.

Beginning with the good feature point \mathbf{m} , Equation (7.6) is solved to find the estimated new pixel location

$$\mathbf{m}_1 = \mathbf{m} + \delta(\mathbf{m}) = \mathbf{m} + G^{-1}(\mathbf{m})b(\mathbf{m}).$$

The gradient data G and b are then evaluated at \mathbf{m}_1 and the process is repeated interatively. The KLT feature tracking algorithm can be described using the equation

$$\mathbf{m}_{\ell+1} = \mathbf{m}_\ell + G^{-1}(\mathbf{m}_\ell)b(\mathbf{m}_\ell)$$

The openCV implementation of `calcOpticalFlowPyrLK()` iterative computes the optical flow vector at specified points. When `calcOpticalFlowPyrLK()` is used in conjunction with `goodFeaturesToTrack`, the resulting algorithm is the KLT feature tracker.

7.4 KLT Feature Tracking with IMU

In this section we address the question of whether an IMU can be used to assist optical flow algorithm. The basic idea is the seed the iteration

$$\mathbf{m}_{\ell+1} = \mathbf{m}_\ell + G^{-1}(\mathbf{m}_\ell)b(\mathbf{m}_\ell)$$

with a best guess from the IMU, rather than the feature location \mathbf{m}_0 .

RWB: come back to this. The notation is not quite adequate since I_t has to be computed at both \mathbf{m}_0 and \mathbf{m}_ℓ .

In this section, we address the issue of initializing the optical flow algorithm optimization given in Algorithm ???. We are particularly interested in the robotic situation where an IMU, synchronized to the camera, is available to resolve the scale ambiguity. We will also discuss the case where IMU measurements are not available. The discussion in this section follows in some respects, the development in ³.

For the sake of clarity, in this section we will again explicitly specify the different coordinate frames. Let $R_k^I \in SO(3)$ be the rotation from the camera frame at time k , \mathcal{F}^k to the inertial frame \mathcal{F}^I , and let $\xi_{k/I}^k$ and $v_{k/I}^k$ be the position and velocity of the camera at time k relative to the inertial frame, expressed in the camera frame \mathcal{F}^k , let $a_{k/I}^k$ be the measured specific acceleration of the camera expressed in \mathcal{F}_k , and $\omega_{k/I}^k$ be the measured angular velocity of the camera relative to the inertial frame, as expressed in the camera frame \mathcal{F}^k , where we have assumed that the IMU biases are known and have been removed from the measurements. Then the kinematics for the camera are given by

$$\begin{aligned}\dot{R}_k^I &= R_k^I (\omega_{k/I}^k)_\times \\ \dot{v}_{k/I}^k &= R_k^{I\top} g^I + a_{k/I}^k, \\ \dot{\xi}_{k/I}^k &= v_{k/I}^k\end{aligned}\tag{7.7}$$

³ Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, February 2017

where g^I is the gravity vector in the inertial frame. Let T_s be the sample period of the IMU, and assume that there are m IMU samples between camera frames. We will use the notation $\kappa_0, \kappa_1, \kappa_2, \dots, \kappa_m$ to denote the intermediate sample from $k - 1$ to k , implying that the time instance κ_0 corresponds to the time at image frame $k - 1$, and κ_m corresponds to the time at image frame k . Then, integrating over one sample of the IMU and assuming that the measurements are constant over the sample period, we get

$$\begin{aligned} R_{\kappa_{i+1}}^I &= R_{\kappa_i}^I \exp \left((\omega_{\kappa_i/I}^{\kappa_i}) \times T_s \right) \\ v_{\kappa_{i+1}/I}^{\kappa_{i+1}} &= R_{\kappa_i}^{\kappa_{i+1}} v_{\kappa_i/I}^{\kappa_i} + R_{\kappa_{i+1}}^{I\top} g^I T_s + R_{\kappa_i}^{\kappa_{i+1}} a_{\kappa_i/I}^{\kappa_i} T_s \\ \xi_{\kappa_{i+1}/I}^{\kappa_{i+1}} &= R_{\kappa_i}^{\kappa_{i+1}} \xi_{\kappa_i/I}^{\kappa_i} + v_{\kappa_{i+1}/I}^{\kappa_{i+1}} T_s, \end{aligned}$$

where

$$R_{\kappa_i}^{\kappa_{i+1}} = R_{\kappa_{i+1}}^{I\top} R_{\kappa_i}^I = \exp \left((\omega_{\kappa_i/I}^{\kappa_i}) \times T_s \right)^\top.$$

The predicted pose after m IMU samples is therefore

$$\tilde{R} = R_{\kappa_0}^{\kappa_m} = R_{\kappa_m}^{I\top} R_{\kappa_0}^I \quad (7.8)$$

$$\tilde{q} = \frac{R_{\kappa_0}^{\kappa_m} \xi_{\kappa_0/I}^{\kappa_0} - \xi_{\kappa_m/I}^{\kappa_m}}{\|R_{\kappa_0}^{\kappa_m} \xi_{\kappa_0/I}^{\kappa_0} - \xi_{\kappa_m/I}^{\kappa_m}\|}. \quad (7.9)$$

The predicted relative pose (\tilde{R}, \tilde{q}) is used to initialize Algorithm ??.

When an IMU is not available, Algorithm ?? can be seeded with the identity rotation $\tilde{R} = I$ and a randomly selected translation unit vector \tilde{q} . This method we will call the “random initialization” method. Another alternative is to initialize Algorithm ?? with the best hypothesis from the previous time step. We will denote this method as the “prior” method. A third alternative, that can also be used with or without the IMU, is to initialize Algorithm ?? with the best RANSAC/LMedS hypothesis that has been found so far from previous LM iterations. Since each hypothesis depends on the previous best hypothesis, when the first hypothesis is selected randomly, we will denote this method as the “random recursive” method. When the first hypothesis is the best hypothesis from the IMU, or the prior time step in the case of no IMU, we call it the “prior recursive” method. In Section ?? we will show results using each of these initialization techniques.

M. Hwangbo, J.-S. Kim, and T. Kanade, "Inertial-aided KLT feature tracking for a moving camera," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (St. Louis), pp. 1909–1916, October, 2009.

Inertial-Aided KLT Feature Tracking for a Moving Camera

Myung Hwangbo, Jun-Sik Kim, and Takeo Kanade

Abstract—We propose a novel inertial-aided KLT feature tracking method robust to camera ego-motions. The conventional KLT uses images only and its working condition is inherently limited to small appearance change between images. When big optical flows are induced by a camera-ego motion, an inertial sensor attached to the camera can provide a good prediction to preserve the tracking performance. We use a low-grade MEMS-based gyroscope to refine an initial condition of the nonlinear optimization in the KLT. It increases the possibility for warping parameters to be in the convergence region of the KLT.

For longer tracking with less drift, we use the affine photometric model and it can effectively deal with camera rolling and outdoor illumination change. Extra computational cost caused by this higher-order motion model is alleviated by restraining the Hessian update and GPU acceleration. Experimental results are provided for both indoor and outdoor scenes and GPU implementation issues are discussed.

I. INTRODUCTION

Feature tracking is a front-end to many vision applications from optical flow to object tracking and 3D reconstruction. Higher-level computer vision algorithms require, therefore, robust tracking performance no matter how a camera moves. KLT (Kanade-Lucas-Tomasi) feature tracking which uses template image alignment techniques has been extensively studied from the seminal work by Lucas and Kanade[1], Shi and Tomasi[2], to the unifying work by Baker and Matthews [3].

The fundamental assumption of KLT feature tracking is small spatial and temporal change of appearance across image sequence. Hence it is naturally vulnerable to fast rotational and big shake motions of a camera. Figure 1 shows two common tracking failures of interest: The first is when the pan motion of a camera induces large optical flows, which easily occur in a hand-held device. It fails because the amount of translation is out of the search region even a multi-scale method can afford. The second is when the roll motion of a camera produces large translation as well as rotational deformation of a template. In this case the translation motion model is not enough to explain template deformation but more general warping models are required such as an affine-photometric model [4]. However, it causes extra computational cost in registering and tracking steps due to increased numbers of the parameters.

These two examples address two issues we focus on in this paper: *search region* and *tracking motion model* (image warping function) of the KLT when a camera moves fast. Mathematically speaking the KLT is a solver of nonlinear

M. Hwangbo, J. Kim, and T. Kanade are with Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh 15213, USA
{myung, kimjs, tk}@cs.cmu.edu

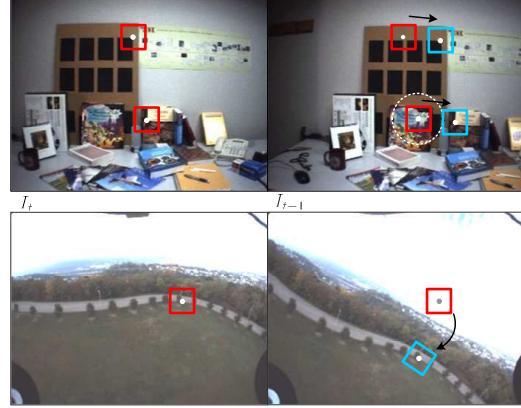


Fig. 1. (Top) An abrupt change of the pan angle in a hand-held camera generates big optical flows that even a multi-scale feature tracker cannot cover. (Bottom) The camera rolling occurred in an aerial vehicle requires a rotational component in addition to translation in a tracking motion model.

optimization problem and has a limited convergence region for a true global minimum. So more assurance for avoiding local minima is offered when a new search region is given from an external inertial sensor instead of simply using the last tracking state. A higher-order tracking model is preferred for our inertial-aided tracking method since image sequences of interest contain more appearance changes than image-only methods expect.

An image sensor is no longer the only sensor found in up-to-date camera systems. For example, image stabilization (IS) driven by integrated inertial sensors becomes *de facto* in small gadgets to remove unwanted jitters. The higher level of integration for visual and inertial sensors has been studied for gaze stabilization [5], object detection [6], and structure from motion [7]. They are based on how humans make the vestibular system collaborate with visual movements [8]. Particularly for camera motion estimation drawn from feature tracking, there have been two main approaches on how to combine them as seen in Figure 2. The first is that camera motions considered independently from both sensors are fused in a certain way (*e.g.*, Kalman filter), and then feed-back to feature tracking. In visual SLAM [9] and Augmented Reality [10][11][12] applications, feature tracking or feature matching uses the knowledge of reconstructed or given position of 3D visual landmarks. The second is that inertial measurements are deeply coupled with feature tracking algorithms in order to improve a parameter search

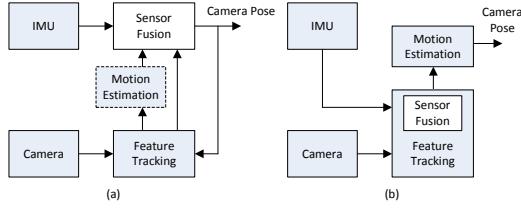


Fig. 2. Two main approaches for feature-based motion estimation that fuses inertial and visual sensors: (a) Camera motion is considered independently from both sensors and then fused in a certain way. (b) Inertial measurements are deeply coupled with a feature tracking algorithm to improve a parameter search problem.

problem [13]. Gray and Veth [14] use an inertial sensor to aid the search for matched SIFT descriptors in a predicted feature space. Makadia and Daniilidis [15] use a sensed gravity direction to reduce the number of model parameters and use the Hough transform for direct estimation of camera motion instead of feature correspondence. Our proposed method also uses instantaneous rotation obtained from inertial sensors to update the parameter of a tracking motion model. Here we only focus on gyroscopic sensors because large optical flows is mainly due to camera rotation in video-rate feature tracking applications.

This paper is organized as follows. Section II describes the derivation of tracker update rules for a high-order tracking model we choose. Section III shows how to embed directly inertial sensor (gyroscope) measurements into the KLT tracking algorithm. Section IV gives brief discussion on GPU implementation for realtime video-rate tracking and Section V demonstrates the improved performance of our method.

II. FEATURE TRACKING PROBLEM

Feature tracking is a sequence of search operation that locates a feature point along incoming images. For small appearance change between temporally adjacent images it can be formulated as a nonlinear optimization problem (1) and a set of parameters $\delta\mathbf{q}$ will be sought that minimizes the cost of intensity difference e between the template T and a new image I_{t+1} .

$$e = \sum_{\mathbf{x} \in \mathcal{A}} s(\mathbf{x}) [T(\mathbf{x}) - I_{t+1}(\mathbf{w}(\mathbf{x}; \mathbf{p}_t, \delta\mathbf{p}))]^2 \quad (1)$$

where $\mathbf{x} = (x, y)^\top$ is a pixel coordinate, $\mathbf{w}(\cdot)$ is a tracking motion model (image warping function), \mathbf{p}_t is a vector of warping parameters, s is a weighting function, and \mathcal{A} is the area of a template window.

There are two main iterative image alignment approaches for template matching: *forward* and *inverse* methods [3]. Both are equivalent up to a first-order approximation but key differences are which image is used as a reference and how to update a warping parameter during iterations. The original Lucas-Kanade algorithm [1] is the forward method and every iteration it recomputes the Hessian from the gradient of

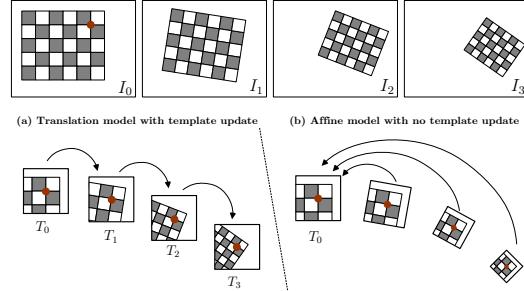


Fig. 3. For long-term feature tracking template update is necessary. A low-order warping function (e.g. translation only) requires frequent template updates to keep up with the appearance change. A high-order warping function (e.g. affine) is more flexible to template deformation but its Hessian computation is expensive.

the warped image of I_{t+1} . On the contrary the inverse method uses a fixed Hessian from the template gradient and consequently it leads to significant computational efficiency over the forward method.

A. Inverse Compositional Image Alignment

This alignment method starts with switching the roles of the image $I_{t+1}(=I)$ and the template T from (1) to (2). The Gauss-Newton method to solve for $\delta\mathbf{p}$ involves two steps: (i) linearize the cost e by the first-order Taylor expansion and (ii) find a local minimum by taking the partial derivative with respect to $\delta\mathbf{p}$. Linearizing at the current warping parameter, i.e., $\delta\mathbf{p} = \mathbf{0}$, gives:

$$e = \sum_{\mathbf{x} \in \mathcal{A}} [I(\mathbf{w}(\mathbf{x}; \mathbf{p}_t)) - T(\mathbf{w}(\mathbf{x}; \delta\mathbf{p}))]^2 \quad (2)$$

$$\approx \sum_{\mathbf{x} \in \mathcal{A}} [I(\mathbf{w}_{\mathbf{x}}(\mathbf{p}_t)) - T(\mathbf{w}_{\mathbf{x}}(\mathbf{0})) - \mathbf{J}(\mathbf{x})\delta\mathbf{p}]^2 \quad (3)$$

where $\mathbf{w}(\mathbf{x}; \cdot) = \mathbf{w}_{\mathbf{x}}(\cdot)$ for brevity and $s(\mathbf{x}) = 1$ here. $\mathbf{J} = \frac{\partial T}{\partial \mathbf{p}}|_{\mathbf{p}=0}$ is called the steepest decent image and it remains unchanged until the template T is updated. The partial derivative of (3) with respect to $\delta\mathbf{p}$ is

$$\frac{\partial e}{\partial \delta\mathbf{p}} = 2 \sum_{\mathbf{x} \in \mathcal{A}} \mathbf{J}^\top [I(\mathbf{w}_{\mathbf{x}}(\mathbf{p}_t)) - T(\mathbf{x}) - \mathbf{J}(\mathbf{x})\delta\mathbf{p}] = 0 \quad (4)$$

Rearranging (4) gives a closed form solution for $\delta\mathbf{p}$ at a local minimum. It iterates until $\|\delta\mathbf{p}\| < \epsilon$ for a small ϵ due to linearization error.

$$\delta\mathbf{p} = \mathbf{H}^{-1} \sum_{\mathbf{x} \in \mathcal{A}} \mathbf{J}^\top [I(\mathbf{w}_{\mathbf{x}}(\mathbf{p}_t)) - T(\mathbf{x})] \quad (5)$$

$$\mathbf{w}_{\mathbf{x}}(\mathbf{p}_{t+1}) \leftarrow \mathbf{w}_{\mathbf{x}}(\mathbf{p}_t) \circ \mathbf{w}_{\mathbf{x}}^{-1}(\delta\mathbf{p}) \quad (6)$$

where $\mathbf{H} = \sum \mathbf{J}^\top \mathbf{J}$ is called the *Hessian* (precisely a first-order approximation to the Hessian).

The inverse method takes computational benefit from fixed \mathbf{J} and \mathbf{H} while in the forward method derived from (1) they changes every iteration. See [3] for more details.

B. Affine Photometric Warping

The choice of a tracking motion model determines the level of allowable image deformation of a template window. We employ the affine-photometric warping proposed by [4] for more robust tracking to camera rotation and outdoor illumination condition. This model has total 8 parameters, $\mathbf{p} = (a_1, \dots, a_6, \alpha, \beta)$ in (7). The affine warp (\mathbf{A}, \mathbf{b}) is for spatial deformation mainly to deal with translation and rolling motion of a camera. For illumination change the scale-and-offset model (α, β) treats contrast variation of a template image. The scale α compensates for the change of an ambient light and the bias β does for the change of a directed light.

$$T(\mathbf{x}; \mathbf{p}) = (\alpha + 1) T(\mathbf{Ax} + \mathbf{b}) + \beta \quad (7)$$

$$\text{where } \mathbf{A} = \begin{bmatrix} 1 + a_1 & a_2 \\ a_3 & 1 + a_4 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} a_5 \\ a_6 \end{bmatrix}$$

By using the chain rule the steepest decent image \mathbf{J} can be computed from the partial derivative at $\mathbf{p} = \mathbf{0}$. $\nabla T = (T_x, T_y)$ is the template gradient image.

$$\mathbf{J} = \frac{\partial T}{\partial \mathbf{p}}|_{\mathbf{p}=\mathbf{0}} = \begin{bmatrix} \frac{\partial T}{\partial \mathbf{a}} & \frac{\partial T}{\partial \alpha} & \frac{\partial T}{\partial \beta} \end{bmatrix} = \begin{bmatrix} \frac{\partial T}{\partial \mathbf{a}} & T & 1 \end{bmatrix} \quad (8)$$

$$\frac{\partial T}{\partial \mathbf{a}}|_{\mathbf{a}=\mathbf{0}} = \frac{\partial T}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \mathbf{a}}|_{\mathbf{a}=\mathbf{0}} = \frac{\partial T}{\partial \mathbf{x}} \frac{\partial \mathbf{w}}{\partial \mathbf{a}} = \nabla T \frac{\partial \mathbf{w}}{\partial \mathbf{a}} \quad (9)$$

$$= \nabla T \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} \quad (10)$$

Then

$$\mathbf{J} = [xT_x \quad yT_x \quad xT_y \quad yT_y \quad T_x \quad T_y \quad T \quad 1] \quad (11)$$

In (5) the update, $\delta\mathbf{p}$, requires the inversion of the symmetric 8×8 Hessian matrix for this model, $\mathbf{H} = \sum_{\mathcal{A}} \mathbf{J}^\top \mathbf{J}$. So much more expensive computation, $\mathcal{O}(n^3)$ with Gaussian elimination method, is involved when compared with a 2×2 Hessian in the translation model.

$$(\mathbf{A}, \mathbf{b})_{t+1} \leftarrow (\mathbf{A}_t \delta \mathbf{A}^{-1}, \mathbf{b}_t - \mathbf{A}_t \delta \mathbf{b}) \quad (12)$$

$$\alpha_{t+1} \leftarrow (\alpha_t + 1) / (\delta\alpha + 1) \quad (13)$$

$$\beta_{t+1} \leftarrow \beta_t - (\alpha_t + 1) \delta\beta \quad (14)$$

Finally from the inverse compositional update rule in (6) the warping parameters \mathbf{p}_{t+1} of the affine-photometric model are propagated as above.

C. Template Update

For long-term feature tracking template update is a mandatory step to keep a plausible track. It usually occurs when the current view point of a camera is quite different from that the template has been taken from and thus it is hard to fit correctly with a chosen tracking motion model. An inherent problem with the template update is accumulation of the localization error of a feature. There exists computational trade-off between frequency of the template update and the order of a motion model. Figure 3 shows two extreme cases

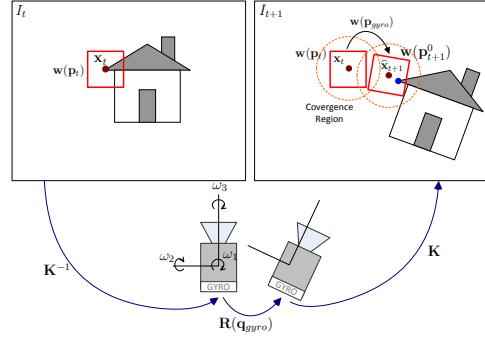


Fig. 4. The inertial-aid KLT for a new image I_{t+1} starts at \mathbf{p}_{t+1}^0 instead of \mathbf{p}_t which is not in the convergence region \mathcal{C} . The initial condition is refined by the 2D homography \mathcal{H} computed from the instantaneous camera rotation $\mathbf{q}_t^{t+1}(gyro)$ from the IMU and a camera calibration matrix \mathbf{K} .

of which difference is whether the template is renewed every certain frames or remains unchanged across images. Less template update requires a tracking motion model to have higher degrees of freedom to adjust bigger image deformation in an incoming image. Nonetheless the Hessian is updated less frequently of which computation is expensive for a higher order model.

We choose carefully the times of the template update by monitoring how good a current tracking is. The measures for tracking quality are squared error, normalized correlation, and shearness. Since tracking a feature location is the main goal of this paper rather than the template itself, the window around the last feature position is captured as a new template if required.

III. INERTIAL FUSION FOR ROBUST TRACKING

One fundamental assumption for all the KLT-based tracking algorithms is a small inter-frame change (e.g. less than three-pixel translation with a 9×9 template window). Otherwise the nonlinear optimization (1) may fail to converge because the convergence region \mathcal{C} for $\delta\mathbf{p}$ is limited to a concave region in which the first-order approximation (4) is close enough for a correct update direction. In the inverse image alignment method, therefore, the success of tracking given T and I_{t+1} heavily depends on whether a starting point \mathbf{p}_{t+1}^0 of the optimization is in \mathcal{C} or not. The image-only tracking method simply uses $\mathbf{p}_{t+1}^0 = \mathbf{p}_t$.

Our goal is to increase a chance of the convergence by relocating \mathbf{p}_{t+1}^0 to \mathcal{C} as close as possible with the aid from the inertial sensor. Here we only focus on gyroscopic sensors because large optical flows is mainly due to camera rotation in video-rate feature tracking applications. The knowledge of the instantaneous camera rotation $\mathbf{q}_t^{t+1}(gyro)$ from a tri-axial gyroscope is directly coupled with the current warping parameter \mathbf{p}_t to predict \mathbf{p}_{t+1}^0 .

A. Camera Ego-Motion Compensation

Figure 4 illustrates the main idea of the paper about how the gyroscope helps feature tracking more robust to big optical flows. If the two images I_t and I_{t+1} are assumed to be taken by a pure camera rotation \mathbf{R}_t^{t+1} the motions of all the feature points can be described by a single 2D homography \mathcal{H} once a camera calibration matrix \mathbf{K} is known [16].

$$\mathcal{H} = \mathbf{K}^{-1} \mathbf{R}_t^{t+1} \mathbf{K} \quad (15)$$

The homography \mathcal{H} is a higher-order deformation than the affine warp (\mathbf{A}, \mathbf{b}) we use for the spatial tracking motion model. Hence the prediction affine warp is appropriately extracted componentwise from the homography. At first the homography needs to be normalized by \mathcal{H}_{33} . The linear transformation part \mathbf{A}_{pred} is the same as an upper 2×2 matrix of \mathcal{H} in order to copy affine components (rotation, scaling, and shear) but projectivity. The translation part \mathbf{b}_{pred} is the transfer amount of position by the 2D homography. $\mathbf{x}_{\mathcal{H}} \equiv \mathcal{H}[\mathbf{x} \ 1]^{\top}$ is a transferred position of \mathbf{x} by \mathcal{H} . The photometric parameters remains unaffected.

$$\mathbf{A}_{pred} = \mathcal{H}_{2 \times 2} \quad (16)$$

$$\mathbf{b}_{pred} = \mathbf{x}_{\mathcal{H}} - \mathbf{x} \quad (17)$$

$$\alpha_{pred} = \beta_{pred} = 0 \quad (18)$$

Given the prediction warp by the inertial sensor the initial parameter \mathbf{p}_{t+1}^0 is obtained by the forward composition of the warping function with \mathbf{p}_t .

$$\begin{aligned} \mathbf{w}(\mathbf{x}; \mathbf{p}_{t+1}^0) &= \mathbf{w}(\mathbf{x}; \mathbf{p}_{pred}) \circ \mathbf{w}(\mathbf{x}; \mathbf{p}_t) \\ &= \mathbf{w}(\mathbf{w}(\mathbf{x}; \mathbf{p}_{pred}), \mathbf{p}_t) \end{aligned} \quad (19)$$

The chance of $\mathbf{p}_{t+1}^0 \in \mathcal{C}$ would be greater than that of $\mathbf{p}_t \in \mathcal{C}$ as the bigger camera rotation is involved.

B. Camera-IMU Synchronization

High-precision synchronization of raw sensors usually requires hardware-level triggering based on a precise clock. For a COTS-based sensor solution with a generic computer, however, exact sampling time of sensor data is obscure. It is hindered by unknown delays in data bridges such as communication and buffering between low-level embedded systems.

If the delays between the sensors and a computer that fuses both data are unknown but fixed, Figure 6 shows an easy and simple way to identify the time lag between two measurement sequences. We first repeat a small sinusoidal motion to the camera in one direction and then compute the average of the optical flow magnitude between images from many feature tracks. Note that the optical flow and gyroscope data have an identical phase since they are induced by the same camera motion. The phase lag ϕ_o between the two signals reveals the time offset t_o of both sensors. The FFT can be used to derive a correct phase lag from repeated camera motions.

Once t_o is obtained, the camera rotation \mathbf{q}_t^{t+1} can be computed properly from the interpolation of asynchronous gyroscope measurements. Let $\boldsymbol{\omega} = [\omega_1 \ \omega_2 \ \omega_3]^{\top}$ be angular

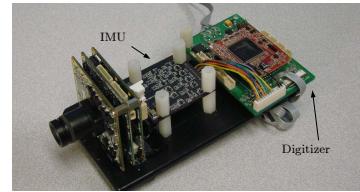


Fig. 5. The camera-IMU system for the inertial-aided feature tracking

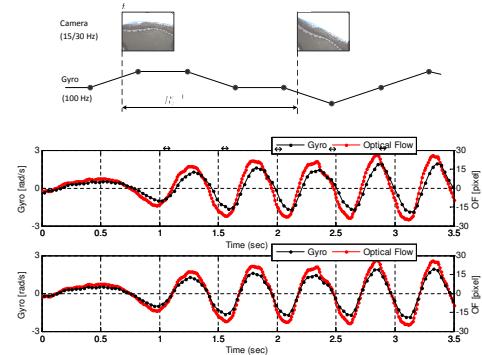


Fig. 6. Synchronization of the camera and the IMU: (Top) Asynchronous gyroscope data with image timestamps are linearly interpolated to get the rotation between I_t and I_{t+1} . (Bottom) By applying a sinusoidal camera motion, the time offset t_o in synchronization can be identified from the phase difference ϕ_o between the IMU and optical flow signals.

rates from gyroscopes. For a strap-down IMU configuration the use of quaternion in (20) is fast and effective for integrating $\boldsymbol{\omega}$ to rotation angle \mathbf{q} .

$$\dot{\mathbf{q}} = \frac{1}{2} \boldsymbol{\Omega}(\boldsymbol{\omega}) \mathbf{q} = \frac{1}{2} \begin{bmatrix} \mathbf{0} & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix} \mathbf{q} \quad (20)$$

From the initial $\mathbf{q}_t = [1 \ 0 \ 0 \ 0]^{\top}$ \mathbf{q}_{t+1} is computed by a numerical integration with proper quaternion normalization.

C. IMU Noise and Camera-IMU Calibration Error

The prediction error in \mathbf{p}_{pred} is caused by two main factors: modeling error and calibration error. Modeling error means that neither camera translation nor projectivity is considered in the prediction warp. We intentionally ignore camera translation since noisy MEMS-based accelerometers is insufficient to provide robust traveling distance and direction of the camera. The other includes misalignment of camera/IMU, camera calibration error, bias and variance of IMU noise. Particularly we focus how much IMU noise is allowable in a given convergence region \mathcal{C} of feature tracking.

For the simplicity of analysis, the error of translation distance $\|\mathbf{b}_{pred}\|$ due to IMU noise is investigated in a pin-hole camera model. For a small rotation, the variances for

Algorithm 1: Inertial-Aid KLT Feature Tracking

```

 $n_{iter}, n_{fmin}, p_{max} \leftarrow$  fixed numbers
Compute the gradient  $\nabla I_{t+1}$ 
if  $n_{feature} < n_{fmin}$  then
| Find new features from cornerness of  $\nabla I_{t+1}$ 
| Compute  $\mathbf{H}$  and  $\mathbf{H}^{-1}$ 
| Fill in lost slots of feature table
Get camera rotation  $\mathbf{R}(\mathbf{q})$  from IMU
for pyramid level =  $p_{max}$  to 0 do
| forall features do
| | Update initial warp  $\mathbf{w}_{t+1}$  from  $\mathbf{w}_{imu}$  using (19)
| | Warp image  $I_{t+1}(\mathbf{w}(\mathbf{x}, \mathbf{p}_{t+1}))$ 
| | Compute error  $e = T - I_{t+1}(\mathbf{w}(\mathbf{x}, \mathbf{p}_{t+1}))$ 
| | Compute update direction  $\delta \mathbf{p}$  using (5)
| | for  $k = 1$  to  $n_{iter}$  do
| | | Line search for best scale  $s^*$ 
| | | Update parameter with  $s^* \delta \mathbf{p}$  using (12)-(14)
| | Remove features that  $e > e_{thresh}$ 

```

camera panning and rolling can be approximated from (15) and (17) respectively for a small rotation.

$$\sigma_{\|\mathbf{b}\|,pred}^2 = (kf)^2 \Delta t_s \sigma_{\omega,x}^2 \quad (21)$$

$$\sigma_{\|\mathbf{b}\|,pred}^2 = r^2 \Delta t_s \sigma_{\omega,z}^2 \quad (22)$$

where k is a pixel scale factor, f is a focal length, r is the distance in pixels from a camera center, and Δt_s is a image sampling time interval. Let the camera be a 1/4" CCD sensor and 640×480 resolution ($k = 126.0$) at 30Hz with a lens of 60° field-of-view ($f = 2\text{mm}$). Given a typical variance of MEMS-based gyroscopes $\sigma_\omega = 0.01 \text{ rad/s}$, the variances are $\sigma_{\|\mathbf{b}\|,pred} = 0.4$ pixel for pan or tilt motion and $\sigma_{\|\mathbf{b}\|,pred} = 0.8$ pixel for roll motion at 120 pixels from the center ($r = 120$). For the 15×15 template size, the errors are negligible when compared with the convergence region shown in Figure 11.

IV. GPU IMPLEMENTATION

In practice the affine-photometric tracking is barely used due to the computational complexity. In template registration time complexity of the Hessian inverse is generally $O(n^3)$ where n is the number of parameters. Compared with the translation model ($n = 2$), therefore, computation load of the affine-photometric model ($n = 8$) increases at least 64 times. In template tracking the complexity is also $O(n^2)$ in updating warping parameters. When several hundreds of features are tracked, the increased burden can be alleviated by parallelized hardware computation. We implemented our high order model on a GPU as Sinha *et al.*[17] has done with the translation warping model which is valid for the ground vehicle application.

However, note that the GPU tend to lose its computational efficiency and is even worse than the CPU when a target algorithm has complicated decision flows depending on the current evaluation of input data. In this case the GPU has

difficulty to use concurrent threads that have the same length of instructions with different inputs. It violates the SIMD principle of parallel computing. Algorithm 1 contains two routines of which internal iterations depend on inputs: sorting cornerness measures and inversion of the Hessian. Gaussian elimination involved with the 8×8 Hessian inverse is a quite complicated thread in the GPU side. Even with extra burden of CPU-GPU data transfer, they run faster on the CPU. Based on NVIDIA CUDA library [18], therfore, we adopt a CPU-GPU hybrid approach in which particularly the cornerness sort and the Hessian inversion are computed in the CPU.

V. EXPERIMENTS

The experiments are performed on two kinds of scenes: the desk scene (640×480 at 30Hz) from a hand-held device and the outdoor aerial scene (320×240 at 15Hz) from a MAV. Both uses the same camera system in Figure 5. We use the multi-resolution pyramid approach to increase the search region as explained in the GPU implementation.

A. Camera-IMU system

The IMU is placed right behind of the camera and the relation between both sensors are calibrated. The camera is the SENTECH USB 2.0 color board camera and the IMU is the O-NAVI GYROSCUBE tri-axis MEMS analog module. Three orthogonal angular rates in the range of $\pm 150^\circ$ are sampled at 100 Hz with a 11-bit resolution digitizer.

B. Results

We have run feature tracking algorithms in two ways for the given scenes. One is purely vision-based tracking that does not use the IMU information but the images only. We call it the *image-only* method here. The other is our inertial-aided method that the camera-ego rotation is compensated. The advantages of the inertial-aided feature tracker over the image-only method are summarized as follows.

- Higher tracking success rate
- Longer tracking length with less drift

Figure 7 compares the inertial-aided and the image-only feature tracking on the desk scene. We rotates the camera so that it undergoes three different rotations - panning, tilting, and rolling sequentially. We apply this set of the camera rotations both slowly and fast in order to see when the IMU fusion makes the difference between two methods. The IMU measurements are plotted together with the number of tracked features for this purpose. In Figure 7(d) at the slow camera motion no big difference in the number of tracked features is noticeable between both methods. When the magnitude of the camera motion increases twice in Figure 7(e), however, the image-only method starts to drop features significantly whenever there is a peak in angular rates and it has good tracks only when the camera nearly stops to switch the direction. On the contrary the inertial-aided method maintains the good performance only losing 10% ~ 20% of features. The robust performance of the inertial-aided method are shown in Figure 7(a)-(c) when the



Fig. 9. The templates are well tracked by the affine motion model when the camera moves forward with a rolling motion in the desk scene. All the degrees of freedom of the affine warping are revealed in template deformations colored in green.

fast camera motions occur. The green dots indicate predicted feature locations by the IMU. See how close the green dots are to the red dots - the tracked points - to verify the effect of the better initialization for feature tracking.

The spatial warping of a square template by the affine transformation is shown in Figure 9. All the degrees of freedom of this motion model can be clearly seen when the camera moves forward with a rolling motion. Here the templates translate and rotate due to the camera rotation. They scale down as the camera approaches to the desk and get sheared as the angle to the desk plane changes. It demonstrates the affine motion model is good enough to track templates deformed by camera rotation. Figure 10 compares the tracking length histograms of both image-only and inertial-aided methods for the desk scene. It evaluates the capability for a long-frame tracking. In this experiment, no new features are registered once 500 features are selected at the first image frame. At the high frame length bins marked by the curly bracket, the inertial-aided tracker has more features. In other words, the image-only tracker tends to lose tracking quickly at the lower frame length.

Figure 8 shows outdoor experiments taken by a micro aerial vehicle. The scene has been taken when the aerial vehicle is banking to the left and back to a level flight. Large optical flows occur when the vehicle changes its rolling angle. The inertial-aided method provides quite better optical flows sufficient to perceive the current vehicle motion relative to the ground plane. This scene has the regions difficult to track. A group of trees has repeated textures and the grass field is very homogeneous. In these regions a tracker is susceptible to get in a wrong local minimum. Since the inertial-aided tracker narrows down a search area, however, stable tracking results are available.

C. GPU Performance

The upper row of Figure 12 shows the comparison when the numbers of features changes with 25×25 template size and all four levels of pyramids. One can see that the tracking time has no increase on most GPUs when the number of features increases. It is because the processing time on a GPU is determined by the worst case. Note that even the

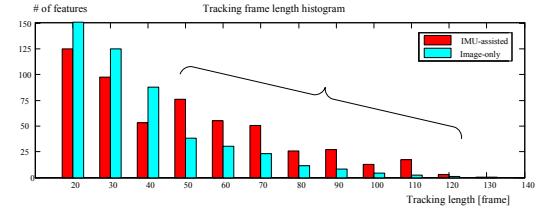


Fig. 10. The tracking length histogram of 500 features in the desk scene with no new registration. The inertial-aided KLT has more features in regions of the longer tracking length as marked by the curly bracket.

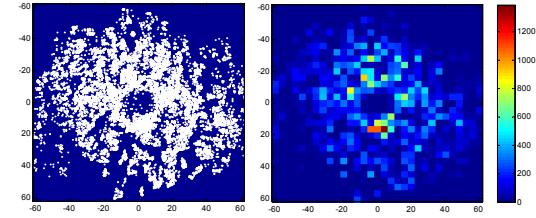


Fig. 11. Feature translations rescued by the inertial-aid tracking in the desk scene: Tested by 15×15 template size. No points appear for small translations because the image-only KLT can also track. The inertial-aid KLT can cover up to 60 pixel translation.

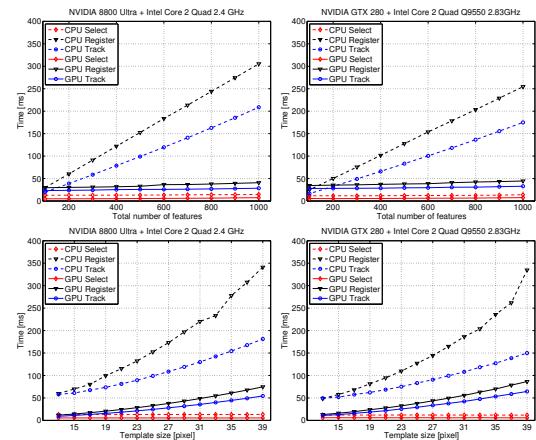


Fig. 12. Performance comparison between the CPU and GPU implementations on various platforms with respect to the number of features using 25×25 templates (upper row), and the template size with 500 features (lower row). Dashed lines are for the CPU and solid lines for the GPU. The GPU has no performance degrade when the number of features increases.

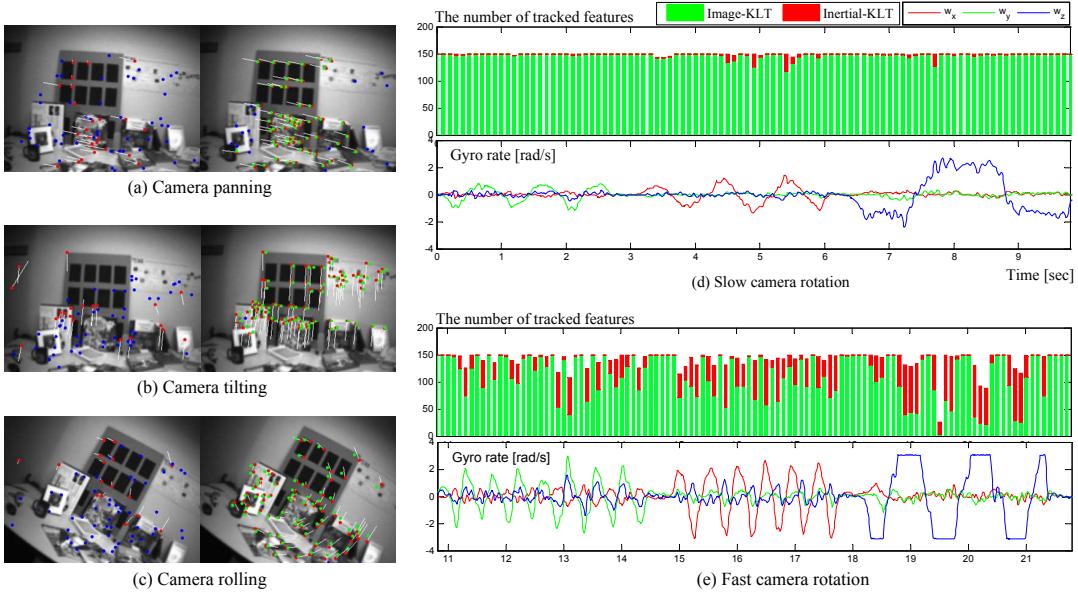


Fig. 7. Inertial-aided vs. Image-only KLT on the desk scene: The camera is rotated in (a) pan, (b) tilt, and (c) roll motions sequentially. Two different motion sets, (d) and (e), are applied in terms of the frequency and the magnitude of the angular rates. When the magnitude of motions increases twice in (e), significant amount of features fails in the image-only KLT but the inertial-aided KLT maintains the good performance in terms of the number of tracked features. (Right graphs) The red area in the bar indicates the number of features rescued by the inertial-aid KLT. (Left images) The green dots are predicted locations by the IMU. See how close the green dots are to the red ones from the purpose of the better initialization for feature tracking.

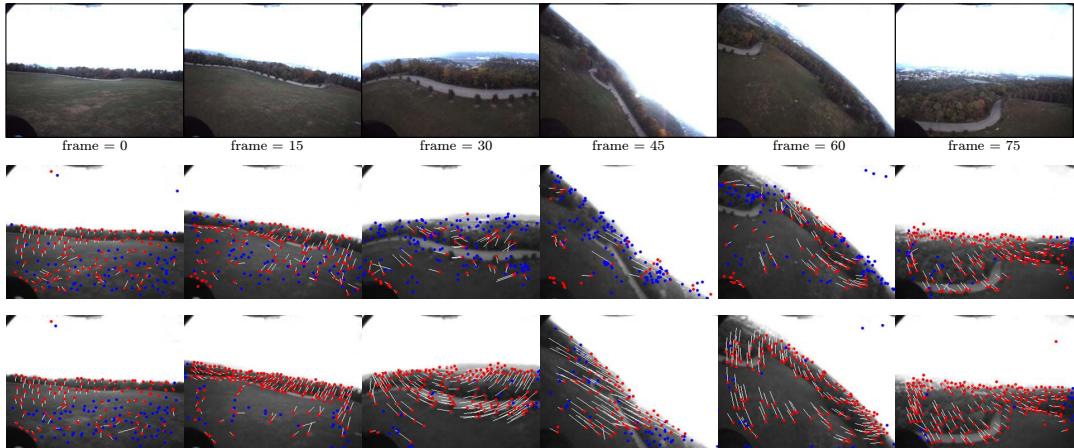


Fig. 8. Inertial-aided vs. Image-only feature tracking on the aerial scene: (Top) Input image sequence (Middle) The image-only feature tracking results. (Bottom) The inertial-aided feature tracking results. The aerial vehicle is banking to the left and back to a level flight. The inertial-aided method generates quite better optical flows sufficient to perceive the current vehicle motion relative to the ground plane. Red dots are for good tracks and blue dots are for lost ones.

worst GPU runs faster than the best CPU. In the lower row of Figure 12, the processing time of tracking 500 features increases quadratically with respect to the template size but its rate is much smaller than that of the CPU.

VI. CONCLUSION

We propose an enhanced KLT feature tracking method assisted by a low-cost IMU for a freely moving camera in 3D. We use the rotational information from the IMU to give better initial estimates of motion parameters for template warping. By compensating a camera-ego motion, search ranges for the motion parameters become much narrower. This simple assistance from the IMU brings significant improvement: longer feature tracking under abrupt camera motions, overall speed-up due to less frequent feature selection. With the affine-photometric motion model our inertial-aided tracker shows very stable and robust results under big shake and fast rolling motions of the camera. We also show that this high-order motion model is implemented for a video-rate feature tracking up to 1000 features by the SIMD property of a GPU.

REFERENCES

- [1] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, (Vancouver, Canada), pp. 674–679, 1981.
- [2] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, (Seattle), June 1994.
- [3] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *Int. J. Comput. Vision*, vol. 56, no. 3, pp. 221–255, 2004.
- [4] H. Jin, P. Favaro, and S. Soatto, "Real-time feature tracking and outlier rejection with changes in illumination," in *Proc. of the Int'l. Conf. on Computer Vision (ICCV)*, July 2001.
- [5] F. Panerai and G. Sandini, "Visual and inertial integration for gaze stabilization," in *Proc. Int'l Symp. Intelligent Robotic Systems (IROS)*, 1997.
- [6] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1597–1608, 2003.
- [7] T. Okatani and K. Deguchi, "Robust estimation of camera translation between two images using a camera with a 3d orientation sensor," in *Proc. IEEE Int'l Conf. on Pattern Recognition (ICPR)*, 2002.
- [8] R. H. Carpenter, *Movements of the Eyes*. London Pion Limited, 2 ed., 1988.
- [9] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [10] J. Chandaria, G. Thomas, B. Bartczak, K. Koeser, R. Koch, M. Becker, G. Bleser, D. Stricker, C. Wohlleber, M. Felsberg, F. Gustafsson, J. Hol, T. B. Schn, J. Skoglund, P. J. Slycke, and S. Smeitz, "Real-time camera tracking in the MATRIS project," in *IBC2006*, (Amsterdam), 2006.
- [11] G. Bleser, C. Wohlleber, M. Becker, and D. Stricker, "Fast and stable tracking for ar fusing video and inertial sensor data," in *Int'l Conf in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, (Plzen, Czech Republic), pp. 109–115, 2006.
- [12] Y. Yokokohji, Y. Sugawara, and T. Yoshikawa, "Accurate image overlay on video see-through HMDs using vision and accelerometers," in *Proc. IEEE Virtual Reality*, 2000.
- [13] S. You, U. Neumann, and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," in *Proc. IEEE Virtual Reality*, 1999.
- [14] J. Gray and M. Veth, "Deeply-integrated feature tracking for embedded navigation," in *ION International Technical Meeting Program*, (Anaheim, USA), 2009.
- [15] A. Makadia and K. Daniilidis, "Correspondenceless ego-motion estimation using an IMU," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 3534–3539, 2005.
- [16] R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge, 2 ed., 2003.
- [17] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Gpu-based video feature tracking and matching," Tech. Rep. Technical Report 06-012, Department of Computer Science, UNC Chapel Hill, May 2006.
- [18] *NVIDIA CUDA Compute Unified Device Architecture - Programming Guide*, 2008.

7.5 Multirotor guidance algorithms using optical flow

The objective of this section is to develop guidance strategies for the multirotor that use optical flow. The interface to the multirotor is using a velocity controller as described in Section 4.6.

7.5.1 Simple relationships between moving camera and optical flow

RWB: I moved the more complicated relationships to the visual servoing chapter. Could move it back here.

We begin by deriving several relationships between a moving camera and the optical flow. Consider the picture shown in Figure 7.1, where the multirotor is moving parallel to a wall that contains features, with a front-looking camera.

Features on the wall will move in the camera x -direction.

Theorem 7.5.1 Suppose that a multirotor is moving at velocity $\mathbf{v}^i = v\mathbf{k}_c$ parallel to a wall that contains a feature at $\mathbf{P}_{f/c}^c$. Then the calibrated optical flow of the feature in the x -direction is given by

$$\dot{\epsilon}_x = \frac{v}{D} \frac{\epsilon_x}{\sqrt{\epsilon_x^2 + 1}}, \quad (7.10)$$

where D is the distance to the feature projected onto the camera $x - z$ plane.

Proof: The distance from the camera frame to the world feature in the horizontal plane is D and the distance from the camera frame to the projection of the feature on the image plane in the horizontal plane is $d = \sqrt{\epsilon_x^2 + 1}$. From Figure 7.1 we see that

$$\dot{\varphi} = \frac{\dot{\epsilon}_x}{d \cos \varphi} = \frac{v \sin \varphi}{D}.$$

Therefore

$$\dot{\epsilon}_x = \frac{dv \cos \varphi \sin \varphi}{D}.$$

Since the distance from the camera center to the image plane is equal to one, we have that $\cos \varphi = 1/d$ and $\sin \varphi = \epsilon_x/d$. Therefore

$$\dot{\epsilon}_x = \frac{v \epsilon_x}{D d} = \frac{v \epsilon_x}{D \sqrt{\epsilon_x^2 + 1}}.$$

Consider the picture shown in Figure 7.2, where the multirotor is moving parallel to the ground that contains features, with a forward-looking camera.

Features on the ground will move in the camera y -direction.

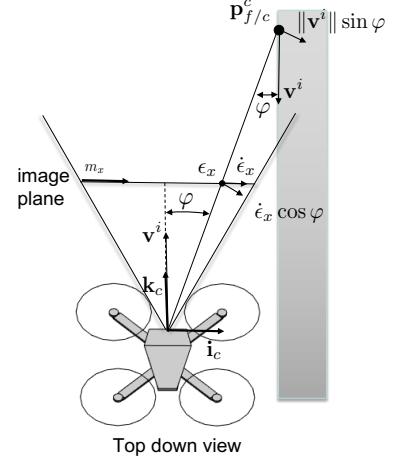


Figure 7.1: Optical flow of a wall for forward looking camera.

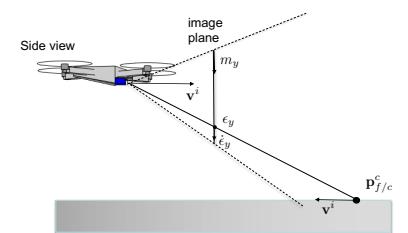


Figure 7.2: Optical flow of a feature on the ground for forward looking camera.

Theorem 7.5.2 Suppose that a multirotor is moving at velocity $\mathbf{v}^i = v\mathbf{k}_c$ parallel to the ground that contains a feature at $\mathbf{p}_{f/c}^c$. Then the calibrated optical flow of the feature in the y -direction is given by

$$\dot{\epsilon}_y = \frac{v}{D} \frac{\epsilon_y}{\sqrt{\epsilon_y^2 + 1}}, \quad (7.11)$$

where D is the distance to the feature projected on to the camera $y-z$ plane.

Proof: Similar to the proof of Theorem 7.5.1. \blacksquare

Consider the picture shown in Figure 7.3, where the multirotor is moving toward a flat wall that contains features, with a forward-looking camera. Define the time-to-collision τ_c to be the time that it will take for the multirotor to intersect the wall.

Theorem 7.5.3 Suppose that a multirotor is moving at velocity $\mathbf{v}^i = v\mathbf{k}_c$ and let \mathbb{P} be the plane in \mathbb{R}^3 with normal vector \mathbf{k}_c that contains a feature $\mathbf{p}_{f/c}^c$, and let τ_c be the time at which the multirotor will intersect \mathbb{P} . Then if $\mathbf{p}_{f/c}^c$ is not on the ray defined by \mathbf{k}_c , then

$$\tau_c = \frac{\epsilon_x}{\dot{\epsilon}_x} = \frac{\epsilon_y}{\dot{\epsilon}_y}. \quad (7.12)$$

Proof: Let $\mathbf{p}_{f/c}^c = (p_x, p_y, p_z)^\top$. Then the calibrated pixel in the x -direction is given by

$$\epsilon_x = \frac{p_x}{p_z}.$$

Differentiating we get

$$\begin{aligned} \dot{\epsilon}_x &= \frac{p_z \dot{p}_x - p_x \dot{p}_z}{p_z^2} \\ &= -\frac{p_x}{p_z} \left(\frac{\dot{p}_z}{p_z} \right) \end{aligned} \quad (7.13)$$

$$= -\epsilon_x \frac{\dot{p}_z}{p_z} \quad (7.14)$$

where the second line follows from the fact that the object does not move perpendicular to the motion, i.e., $\dot{p}_x = 0$. Similarly it can be shown that

$$\dot{\epsilon}_y = -\epsilon_y \frac{\dot{p}_z}{p_z}$$

The time-to-collision will be the distance to the plane divided by the closing velocity, i.e.,

$$\tau_c = -\frac{p_z}{\dot{p}_z},$$

where we note that \dot{p}_z is negative since p_z is decreasing. Equation (7.12) follows from Equation (7.14). \blacksquare

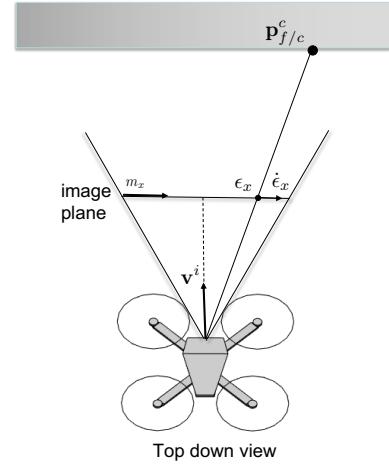


Figure 7.3: Optical flow of a feature on a wall for forward looking camera.

If instead of tracking feature points, the computer vision algorithm tracks the area of an object, then time to collision with the plane containing the object can also be computed as shown in the next theorem.

Theorem 7.5.4 Suppose that a multicopter is moving at velocity $\mathbf{v}^i = v\mathbf{k}_c$ toward a plane defined by the normal vector \mathbf{k}_c , that contains an object of area A . Let ϵ_A be the calibrated pixel area on the image plane, and let $\dot{\epsilon}_A$ be the expansion rate of the pixel area. Then the time-to-collision with the plane containing the object is

$$\tau_c = \frac{\epsilon_A}{\dot{\epsilon}_A}. \quad (7.15)$$

Proof: The pixel area is given by

$$\frac{\epsilon_A}{1} = \frac{A}{p_z}.$$

Differentiating we get

$$\begin{aligned} \dot{\epsilon}_A &= -\frac{A}{p_z} \left(\frac{\dot{p}_z}{p_z} \right) \\ &= -\epsilon_A \frac{\dot{p}_z}{p_z}. \end{aligned}$$

Therefore, the time-to-collision is

$$\tau_c = -\frac{p_z}{\dot{p}_z} = \frac{\epsilon_A}{\dot{\epsilon}_A}.$$

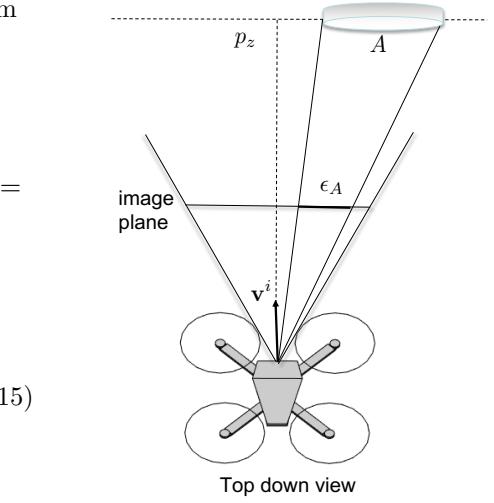


Figure 7.4: Time-to-collision calculation for an object with area A

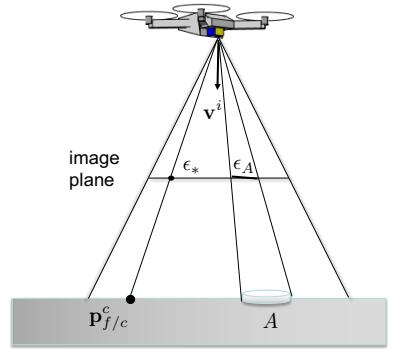


Figure 7.5: Time-to-land calculation.

Corollary 7.5.5 With reference to Figure 7.5 suppose that a multicopter is moving with velocity $\mathbf{v}^i = v\mathbf{k}_c^i = v\mathbf{k}_b^i$ with a downward looking camera, and suppose that it is tracking either a feature at $\mathbf{p}_{f/c}^c$ that is not on the optical axis, or an area A , both on a horizontal ground plane, then the time-to-land is given by

$$\tau_L = \frac{\epsilon_A}{\dot{\epsilon}_A} = \frac{\epsilon_x}{\dot{\epsilon}_x} = \frac{\epsilon_y}{\dot{\epsilon}_y}.$$

Note that if $e_A[k]$ is the area of the object in frame k , then

$$\dot{\epsilon}_A \approx \frac{e_A[k] - e_A[k-1]}{T_s},$$

where T_s is the sample time between frames. Then the time to collision at frame k can be approximated as

$$\tau_c[k] \approx T_s \frac{e_A[k]}{e_A[k] - e_A[k-1]}.$$

7.5.2 Landing strategy

From Corollary 7.5.5 we can derive the following landing strategy.

Assuming a downward facing camera where \mathbf{k}_c is aligned with the body z -axis, let the commanded velocity be given by

$$\mathbf{v}_d^i = (\alpha\tau_L + \beta)\mathbf{k}_c^i,$$

where α has units of m/s^2 and defines the descent rate, and β has units of m/s and defines the landing velocity when the multirotor contacts the ground.

7.5.3 Collision avoidance strategy

Consider the region of interest (ROI) shown in Figure 7.6.

Let $\mathbf{v}_{d,nom}^i$ be a desired nominal velocity.

Step 0. Set $\mathbf{v}_d^i \leftarrow \mathbf{v}_{d,nom}^i$.

Step 1. Find `goodFeaturesToTrack` in ROI and compute optical flow at those features.

Step 2. Compute time-to-collision for all features in the ROI, and find feature ϵ_{min} with smallest time-to-collision, τ_{min} .

Step 3. If $\tau_{min} < \tau_{threshold}$ and $\epsilon_{min} \in ROI_r$, set

$$\mathbf{v}_d^i \leftarrow \mathbf{v}_{d,nom}^i - v_s \mathbf{j}_b^i.$$

If $\tau_{min} < \tau_{threshold}$ and $\epsilon_{min} \in ROI_\ell$, set

$$\mathbf{v}_d^i \leftarrow \mathbf{v}_{d,nom}^i + v_s \mathbf{j}_b^i.$$

where v_s is a side velocity to avoid the obstacle.

7.5.4 Terrain following strategy

Consider the region of interest (ROI) shown in Figure 7.7.

From Theorem 7.5.2 we have seen that

$$\dot{\epsilon}_y = \frac{v}{D} \frac{\epsilon_y}{\sqrt{\epsilon_y^2 + 1}}.$$

where D is the distance to feature in the camera $x - y$ plane. Therefore

$$\frac{D}{v} = \frac{\epsilon_y}{\dot{\epsilon}_y \sqrt{\epsilon_y^2 + 1}}.$$

The idea with terrain following is to regulate this value to some constant $(D/v)_d$. When v increases, multirotor will rise, when v decreases, multirotor will descend. If v is known, then we could regulate D directly.

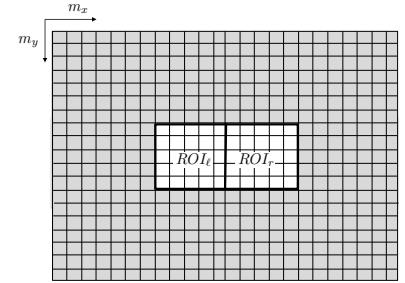


Figure 7.6: Region of interest for collision avoidance.

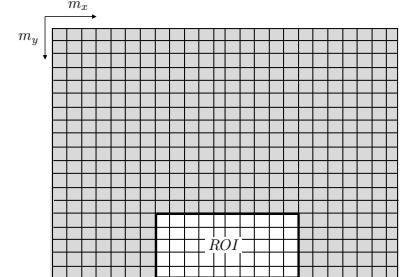


Figure 7.7: Region of interest for terrain following.

Step 1. Find `goodFeaturesToTrack` in ROI and compute optical flow at those features.

Step 1. Compute the average D/v as

$$\left(\frac{D}{v}\right)_{\text{ave}} = \sum_{\epsilon \in ROI} \frac{\epsilon_y}{\dot{\epsilon}_y \sqrt{\epsilon_y^2 + 1}}$$

Step 2. Adjust the desired velocity as

$$\mathbf{v}_d^i \leftarrow \mathbf{v}_{d,nom}^i - k \left[\left(\frac{D}{v}\right)_d - \left(\frac{D}{v}\right)_{\text{ave}} \right] \mathbf{k}_b^i,$$

where $k > 0$ is a tunable control gain.

7.5.5 Wall following strategy

Consider the region of interest (ROI) shown in Figure 7.8.

From Theorem 7.5.1 we have seen that

$$\dot{\epsilon}_x = \frac{v}{D} \frac{\epsilon_x}{\sqrt{\epsilon_x^2 + 1}}.$$

where D is the distance to the wall feature in the camera $x - z$ plane.

Therefore

$$\frac{D}{v} = \frac{\epsilon_x}{\dot{\epsilon}_x \sqrt{\epsilon_x^2 + 1}}.$$

The idea with wall following is to regulate this value to some constant $(D/v)_d$. When v increases, multirotor will move away from the wall, when v decreases, the multirotor will move closer to the wall. If v is known, then we could regulate D directly.

Step 1. Find `goodFeaturesToTrack` in ROI and compute optical flow at those features.

Step 1. Compute the average D/v as

$$\left(\frac{D}{v}\right)_{\text{ave}} = \sum_{\epsilon \in ROI_r} \frac{\epsilon_x}{\dot{\epsilon}_x \sqrt{\epsilon_x^2 + 1}}$$

Step 2. Adjust the desired velocity as

$$\mathbf{v}_d^i \leftarrow \mathbf{v}_{d,nom}^i - k \left[\left(\frac{D}{v}\right)_d - \left(\frac{D}{v}\right)_{\text{ave}} \right] \mathbf{j}_b^i,$$

where $k > 0$ is a tunable control gain.

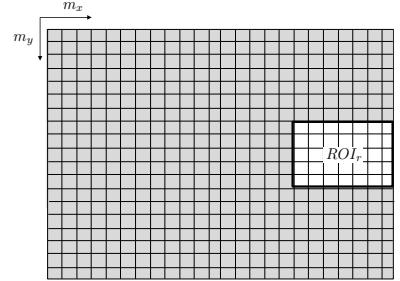


Figure 7.8: Region of interest for wall following.

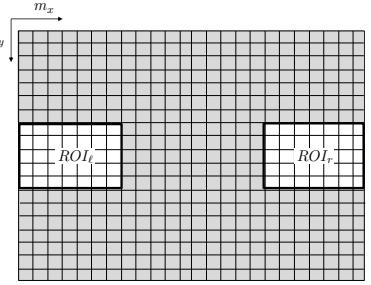


Figure 7.9: Region of interest for canyon following.

Step 1. Find `goodFeaturesToTrack` in ROI_l and ROI_r and compute optical flow at those features.

Step 1. Compute the average optical flow in each region

$$\dot{\epsilon}_{\ell,\text{ave}} = \frac{1}{N} \sum_{\epsilon \in ROI_l} \dot{\epsilon}$$

$$\dot{\epsilon}_{r,\text{ave}} = \frac{1}{N} \sum_{\epsilon \in ROI_r} \dot{\epsilon},$$

where N is the number of pixels in ROI .

Step 2. Adjust the desired velocity as

$$\mathbf{v}_d^i \leftarrow \mathbf{v}_{d,nom}^i + k \left[\frac{\|\dot{\epsilon}_{\ell,\text{ave}}\| - \|\dot{\epsilon}_{r,\text{ave}}\|}{\|\dot{\epsilon}_{\ell,\text{ave}}\| + \|\dot{\epsilon}_{r,\text{ave}}\|} \right] \mathbf{j}_b^i,$$

where k is a control gain.

7.6 Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft, TRO, 2006

Fly-Inspired Visual Steering of an Ultralight Indoor Aircraft

Jean-Christophe Zufferey, *Member, IEEE*, and Dario Floreano, *Member, IEEE*

Abstract—We aim at developing autonomous microflyers capable of navigating within houses or small indoor environments using vision as the principal source of information. Due to severe weight and energy constraints, inspiration is taken from the fly for the selection of sensors, for signal processing, and for the control strategy. The current 30-g prototype is capable of autonomous steering in a 16×16 m textured environment. This paper describes models and algorithms which allow for efficient course stabilization and collision avoidance using optic flow and inertial information.

Index Terms—Collision avoidance, indoor flying robot, optic flow (OF), steering control.

I. INTRODUCTION

THIS paper describes a 30-g airplane capable of autonomous steering and collision avoidance in a 16×16 m experimental room using only visual and inertial information. It represents a major step toward our goal of developing autonomous microflyers capable of navigating within houses or small indoor environments. Flying indoors involves a number of challenges, such as small size and slow speed for maneuverability, light weight to stay airborne, low-power electronics, and smart sensing and control to fly in cluttered environments. One of the main issues when developing such a small flying device is that sensors commonly employed in robotics, in particular, active range finders, are too heavy and energy-consuming to match the limited payload (less than 10 g) of an indoor aircraft. Efficient flight control under drastic weight and energy constraints is an issue that is successfully solved by flying insects like flies. Therefore, we take inspiration from these remarkable and well-studied biological systems for the selection of sensory modalities and design of navigational strategies.

The fly's primary sensor for flight control consists of two compound eyes that span an almost omnidirectional field of view (FOV) with coarse resolution [2]. Their optic lobes contain motion-sensitive neurons which respond to retinal image shifts, the so-called optic flow (OF), induced by the motion of the eyes relative to the surroundings [3]. The fly has several OF-sensitive neurons that have been linked to specific visually guided behaviors (for a review, see [4]). OF is a linear combination of

two components, one resulting from rotation (RotOF) and one from translation (TransOF). TransOF is the only component that depends on the distance from an object [5]. If the RotOF component can be cancelled, OF provides useful information for tasks related to depth perception, such as collision avoidance. This may be the reason why flies navigate using a series of straight segments separated by rapid turns, known as saccades [6], [7]. Straight trajectories allow them to experience pure TransOF, which they use to decide when to initiate a saccade. During saccades, flies seem to ignore the visual information, which is dominated by the RotOF component. Flies also possess sensitive mechanosensory structures, called halteres, that detect rotations of the body, allowing them to maintain equilibrium in flight [8]. The working principle of these biological sensors resembles micro-electro-mechanical systems (MEMS) piezo-electric rate gyros, which sense Coriolis forces that act on oscillating mechanical parts. Halteres have also been shown to play an important role in gaze stabilization [9], which may serve to cancel residual RotOF due to turbulence while flying on a straight trajectory.

The design of our autonomous microflyer takes inspiration from the fly's characteristics at three levels. At the sensory level, the flying robot is equipped only with low-resolution visual sensors and a MEMS rate gyro. At the signal processing level, it estimates OF and fuses it with inertial information in order to provide the control system with the same kind of signals used by flies. At the behavioral level, the control system is designed to produce straight trajectories interspersed with fast turning actions when imminent collision is detected by the visual system.

The rest of this paper is organized as follows. After a short overview in Section II of related projects on miniature aerial robotics and bioinspired vision-based navigation, Section III describes our 30-g flying platform and its electronic equipment. Section IV gives the implementation details of the lightweight OF detectors on this aircraft. Section V outlines the control strategy inspired from biological models, and Section VI presents the results obtained in flight. At the end is a final discussion and a few ideas about future work.

II. RELATED WORK

A. Bioinspired Vision-Based Navigation on Wheels

Several bioinspired strategies for vision-based navigation have been developed on wheeled robots for obstacle avoidance [10]–[13] or corridor following [14]–[17]. However, most of those systems either use wheel encoders, or, more generally, their contact with the ground to reduce interference from RotOF and/or implement active gaze stabilization. With small

Manuscript received November 12, 2004; revised April 11, 2005. This paper was recommended for publication by Associate Editor G. Sukhatme and Editor F. Park upon evaluation of the reviewers' comments. This work was supported by the Swiss National Science Foundation. This paper was presented in part at the IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005.

The authors are with the Ecole Polytechnique Fédérale de Lausanne (EPFL), Laboratory of Intelligent Systems, CH-1015 Lausanne, Switzerland (e-mail: jean-christophe.zufferey@epfl.ch; dario.floreano@epfl.ch).

Digital Object Identifier 10.1109/TRO.2005.858857

flying robots, it is, of course, not possible to rely on wheel encoders, and the tight weight budget excludes the use of mobile cameras. Most of the above-mentioned robots, except the one by Franceschini and colleagues [10], rely on off-board image processing and control. Recently, we proposed a set of vision-based control strategies enabling collision avoidance and altitude control [1], [18]. Those control strategies were tested on a small wheeled robot equipped with electronic components very similar to those used on the 30-g aircraft described here. Image processing and control were fully implemented in the onboard 8-b microcontroller, paving the way toward fully autonomous indoor microflyers.

B. Bioinspired Micromechanical Flying Robots

Other projects have aimed at developing ultralight, unconventional micro flying machines with centimeter-scale rotors [19] or flapping wings [20], [21]. Although those devices represent a remarkable micromechatronic challenge, none of them are yet capable of autonomous navigation; neither are they equipped with exteroceptive sensors.

In an even smaller scale, R. Fearings' team is currently attempting to create a micro flying robot that replicates the wing mechanics and dynamics of flies [22]. The expected weight of the final device is approximately 100 mg for a 25-mm wingspan. So far, a single wing on a test rig, linked to an off-board power supply, has generated about 0.5 mN average lift [23]. The team is currently working on a biomimetic sensor suite for attitude control [24], [25], but in-flight tests have not yet been reported.

In our project, we opted for a more classical flying scheme (an airplane), in order to be able to concentrate on control strategy and navigational autonomy, rather than micromechanical developments and aerodynamic studies. However, in order to reasonably maneuver indoors, a flying robot should be able to fly around 1 m/s and have a minimum turning radius of about 1 m. Therefore, a significant effort has been made to optimize wing profile and minimize the weight of the airframe and the actuators in order to reduce the wing loading as much as possible, thereby minimizing stall speed and improving maneuverability [26].

C. Bioinspired Vision-Based Aerial Navigation

Specific studies on insect-inspired altitude control have been conducted in simulation [27], with tethered helicopters [28], and with outdoor unmanned air vehicles [29]. Full 3-D navigation (attitude control, obstacle avoidance, course stabilization, and altitude control) using OF has been demonstrated in simulation [30]. In that case, the dynamics of the simulated agent was based on a minimalist model of an insect without inertia or detailed aerodynamics. Another work with a simulated helicopter showed efficient navigation and obstacle avoidance in urban canyons using high-resolution, 2-D OF and inertial measurements [31]. Experiments of fly-inspired guidance have also been carried out on a robotic gantry [32], successfully demonstrating an artificial implementation of a model of the fly's visual system for autonomous steering. Although these studies on simulated or tethered robots are important proofs of concept, they cannot totally replace experiments on free-flying devices,

which generally have more uncontrolled parameters and richer dynamics.

Experiments about obstacle avoidance and altitude control have been carried out by Centeye (Washington, DC) with hobbyist model planes equipped with mixed-mode very large scale integration (VLSI) OF detectors [33], [34]. Building upon this preliminary work, Oh *et al.* were the first to demonstrate autonomous landing [35] and collision avoidance [36] with light-weight, fixed-wing aircrafts in indoor environments [37]. For each experiment, the flying robot was equipped with only one 1-D OF detector from Centeye so that it could either act on the pitch angle while looking downward, or steer away from obstacles occurring on one side by looking either right or left. No inertial sensors were used, and the RotOF component was assumed to be small, but it is unclear whether this was actually the case. The flying robot was not equipped with wireless communication capabilities, therefore, no data could be logged during flight tests and no *a posteriori* analysis was possible. Those experiments were carried out in unchanged indoor environments (usually basketball courts), but no continuous autonomous flight has been reported so far. The only data available is video recordings of, for example, one-shot avoidance of a basketball basket in front of an otherwise uniform background.

In our experiment, we focus on control strategy for continuous autonomous steering in an enclosed environment of 16 × 16 m. The goal is not yet to develop adaptive OF detectors capable of providing reliable data in the presence of strongly varying visual characteristics (background light, contrast, spatial frequency). Therefore, we are using very simple, commercially available cameras as the visual sensor front-end. Since those cameras are not very sensitive, the experimental arena is equipped with high contrast textures on the walls. The 30-g airplane has two 1-D OF detectors, such that it can symmetrically avoid frontal and lateral obstacles on both sides while actively deciding the optimum direction of the avoidance maneuver. Inertial measurements are used to enhance OF signals by cancelling RotOF generated by turbulence or corrective rudder actions. Finally, a wireless communication device allows for plotting sensory data for *a posteriori* analysis.

III. FLYING PLATFORM

A. Airframe

The current version of our microflyer (Fig. 1) is made of carbon-fiber rods and balsa wood for the frame, and thin plastic film (2.2 g/m²) for the lifting surfaces. It is propelled by a miniature 6-mm DC motor with a gearbox driving a balsa-wood propeller. Two miniature servos from Didel (Belmont/Lausanne, Switzerland) are placed at the back end of the fuselage to control the rudder and elevator. The plane has a wingspan of 86 cm. Table I gives an overview of the weight distribution for the configuration used in the experiment described in this paper.

The speed range during flight lies between 1.2–2.5 m/s, and its yaw rotation speed is in the range ±100°/s. At 2 m/s flight speed, the curvature radius can be less than 1.3 m. This aircraft has no ailerons on the wings. It uses its vertical rudder to steer. This is sufficient, since yaw motion is highly coupled with roll

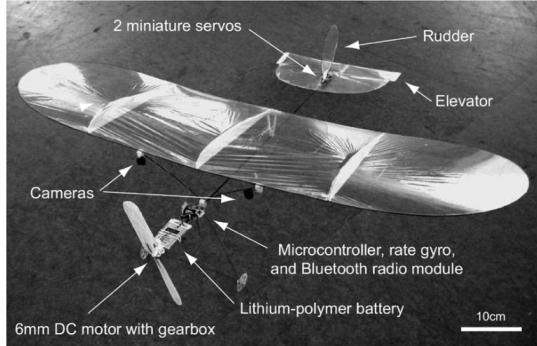


Fig. 1. 30-g aircraft with description of electronic components, sensors, and actuators.

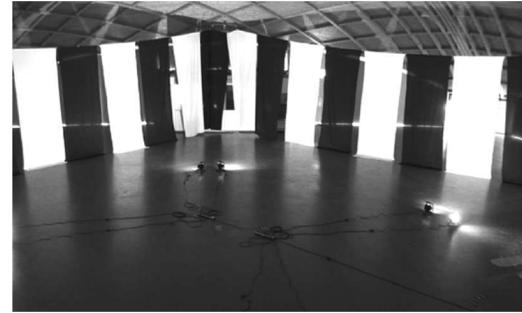


Fig. 2. Experimental room measures 16×16 m and is equipped with contrasting walls made of black and white curtains. Note that the regularity of the pattern is due to the default size of the material from the store.

TABLE I
WEIGHT DISTRIBUTION OF THE 30-G AIRPLANE

Fuselage and tail	: 4.7g
Wing	: 4.5g
Landing gear	: 1.2g
Motor, gearbox, propeller	: 2.7g
Two servos	: 2.7g
Battery	: 6.9g
Electronic board with gyroscope and radio	: 4.0g
Two cameras	: 2.2g
Miscellaneous (cables, glue)	: 1.1g

rotation, so that rudder deflection on the left provokes a leftward roll resulting in a leftward turn. Moreover, the small angle appearing between the left and right wings when the plane is aloft (the wing bends upward due to aerodynamic lift) provides passive roll stability (this is known as the dihedral effect in aerodynamics), so that when the rudder is in its neutral position, the plane tends to cancel any bank and comes back to horizontal automatically after a few seconds.

Due to its very low inertia, this lightweight aircraft is seldom damaged when crashing into obstacles, which is useful during the parameter-tuning phase. In order to further limit the risk of damaging the aircraft during development of the control strategies, the experimental room was covered with black and white curtains that also provided visual texture (Fig. 2).

B. Electronics

The custom-designed embedded electronic board features an 8-b microcontroller running at 20 MHz (Microchip (Chandler, AZ) PIC18F6720, 3840 B of RAM, 64 kwords of program memory) for vision processing and control, a MEMS piezoelectric rate gyro (AnalogDevices (Norwood, MA) ADXRS150, angular velocity range $150^\circ/\text{s}$) for yaw rotation, and a Bluetooth module (Mitsumi (Tokyo, Japan) WML-C10-AHR) for bidirectional communication with a ground station. On-board energy is provided by a 310 mAh lithium-polymer battery. The power consumption of the electronics (including wireless communication) is about 300 mW, whereas overall peak consumption reaches 2 W. The in-flight average consumption allows for an energetic autonomy of about 30 min.

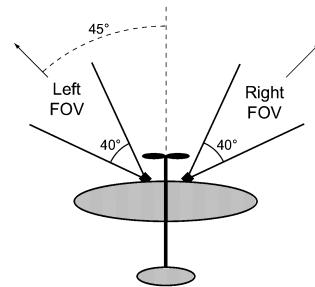


Fig. 3. Top view of the airplane with superposition of orientation and FOV of the 1-D cameras.

C. Vision System

The choice of a suitable vision system for such an aircraft is not trivial. On the one hand, it is well known that motion fields spanning wide FOV are easier to interpret [38], and indeed, nearly omnidirectional vision is a characteristic feature of most flying insects, which possess wide-field, motion-sensitive neurons [39]. On the other hand, artificial vision systems with wide FOV are likely to be too heavy for our application, because they need either a special mirror or fish-eye optics with multiple glass lenses. Such subsystems would also require too much processing power from the onboard microcontroller, which should grab 2-D images and process them in real time. The tradeoff we adopted consists of using multiple lightweight 1-D cameras. The underlying idea is that several of these restricted FOV cameras can be mounted on the aerial robot in order to look in specific directions (see Section V-A) according to the particular task to be solved, while maintaining the total weight and computational requirements within the limits of the aircraft. For the collision-avoidance experiment described here, we use only two horizontal cameras oriented at 45° off the longitudinal axis of the plane (Fig. 3). Those cameras (TAOS (Plano, TX) TSL3301) feature a 1-D array of 102 grey-level pixels, out of which only 28 pixels in the middle are used for the OF detection, and a horizontal FOV of about 40° (see [40] for further details on this camera, possible optics, and comparison with other vision systems).

IV. OPTIC FLOW DETECTION

A. Algorithm

In order to measure OF with the onboard microcontroller, we adapted the image interpolation algorithm (I2A) proposed by Srinivasan [41], which estimates the global motion in a given region of the image by a single-stage, noniterative process. This algorithm computes the displacement s that provides the best fit between a linear combination of two shifted versions of a reference image and a new image acquired after a small delay ΔT . The ratio $s / \Delta T$ (in pixel/s) is proportional to the actual OF (in $^{\circ}/s$), and ΔT can be used as a parameter to scale the OF output (see Section IV-B).

The algorithm, which has been adapted to 1-D images, works as follows. Let $I(n)$ denote the grey level of the n th pixel in the 1-D image array. The algorithm computes the amplitude of the translation s between an image (or a subpart of it) captured at time t , $I_t(n)$, which is called “reference image,” and a later image captured at time $t+1$, $I_{t+1}(n)$. It assumes that, for small displacements of the image, $I_{t+1}(n)$ can be approximated by $\hat{I}_{t+1}(n)$, which is a weighted linear combination of the reference image and of two shifted versions $I_t(n \pm k)$ of that same image

$$\hat{I}_{t+1}(n) = I_t(n) + s \frac{I_t(n - k) - I_t(n + k)}{2k} \quad (1)$$

where k is a small reference shift in pixels. The image displacement s is then computed by minimizing the mean square error E between the estimated image $\hat{I}_{t+1}(n)$ and the new image $I_{t+1}(n)$ with respect to s

$$E = \sum_n [I_{t+1}(n) - \hat{I}_{t+1}(n)]^2 \quad \text{and} \quad \frac{dE}{ds} = 0 \Leftrightarrow \quad (2)$$

$$s = 2k \frac{\sum_n [I_{t+1}(n) - I_t(n)][I_t(n - k) - I_t(n + k)]}{\sum_n [I_t(n - k) - I_t(n + k)]^2}. \quad (3)$$

In our case, the shift amplitude k is set to one pixel, and the delay ΔT between t and $t+1$ is chosen such to ensure that the actual shift does not exceed ± 1 pixel. $I_t(n \pm 1)$ are thus artificially generated by translating the reference image by one pixel to the left and to the right, respectively. More details about this algorithm, as well as a theoretical assessment of its robustness against different perturbations occurring in real-world conditions, can be found in [18, p. 86].

B. Implementation Issues

Equation (3) has been implemented in the embedded microcontroller and applied separately on images coming from the left and right cameras. The microcontroller grabs two successive images corresponding to the reference image $I_t(n)$ and the new one $I_{t+1}(n)$ with a delay ΔT of a few milliseconds. Every pixel intensity is encoded on 8 b, and the variables used in (3) are 32-b integers. The computation is performed in fixed point and lasts only 0.9 ms for 28 pixels, whereas grabbing an image takes less than 0.2 ms.

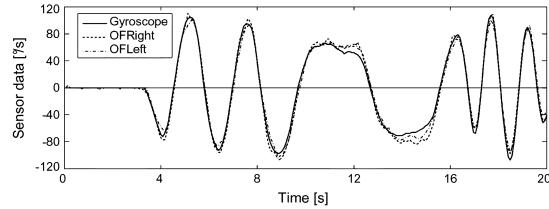


Fig. 4. Match between gyroscopic values and OF estimates of the right (OFRight) and left (OFLef) OF detectors. The data have been recorded every 80 ms while the plane was held by hand in the experimental arena and randomly rotated around its yaw axis.

The delay ΔT is adjusted so that the RotOF values match the rate gyro output when the robot undergoes pure yawing rotation. This empirical calibration provides an optimal ΔT of 6.4 ms. Fig. 4 shows the resulting good match between rotations estimated by the OF detectors and by the rate gyro, while the aircraft is manually rotated about its yaw axis in the experimental arena (Fig. 2).

V. CONTROL STRATEGY

The control strategy for autonomous steering is largely based on the behavior of the fly. For a long time, biologists have been studying the visual cues that affect steering in free-flying flies [6]. A recent paper by Tammero and Dickinson [7] suggests that OFDiv is responsible for triggering saccades, the direction of the saccades (left or right) is opposite to the side experiencing larger OF, and the saccade does not depend on visual feedback. Similarly, our aircraft flies along straight trajectories interspersed with rapid and short turning actions whose duration is fixed. The purpose of this section is to describe the sensory information and control laws that govern this behavior.

The control strategy can be divided into two states: (A) maintain straight trajectory; (B) turn as quickly as possible. The first one is easily implemented by means of a proportional feedback loop connecting the rate gyro to the rudder servomotor, whereas the second one, i.e., the saccade itself, is programmed as a series of rudder commands of a fixed amount of time (1 s). Vision is only required for initiating the saccade, i.e., for the transition from state (A) to state (B). This transition is based on OF estimates acquired during straight flight and, more precisely, on two criteria described in the following paragraph.

A. Triggering a Saccade

In order to find the typical OF signals that the controller can use to trigger a saccade, let us start by studying typical TransOF patterns as they could be experienced by the plane in three different cases of straight flight at constant forward velocity: (a) perpendicularly approaching a flat wall; (b) approaching a flat wall at 30° ; and (c) approaching a corner at 45° . Those situations have been chosen as representative samples of flight in an empty rectangular room. Fig. 5 shows the resulting TransOF fields obtained from the formal description of the

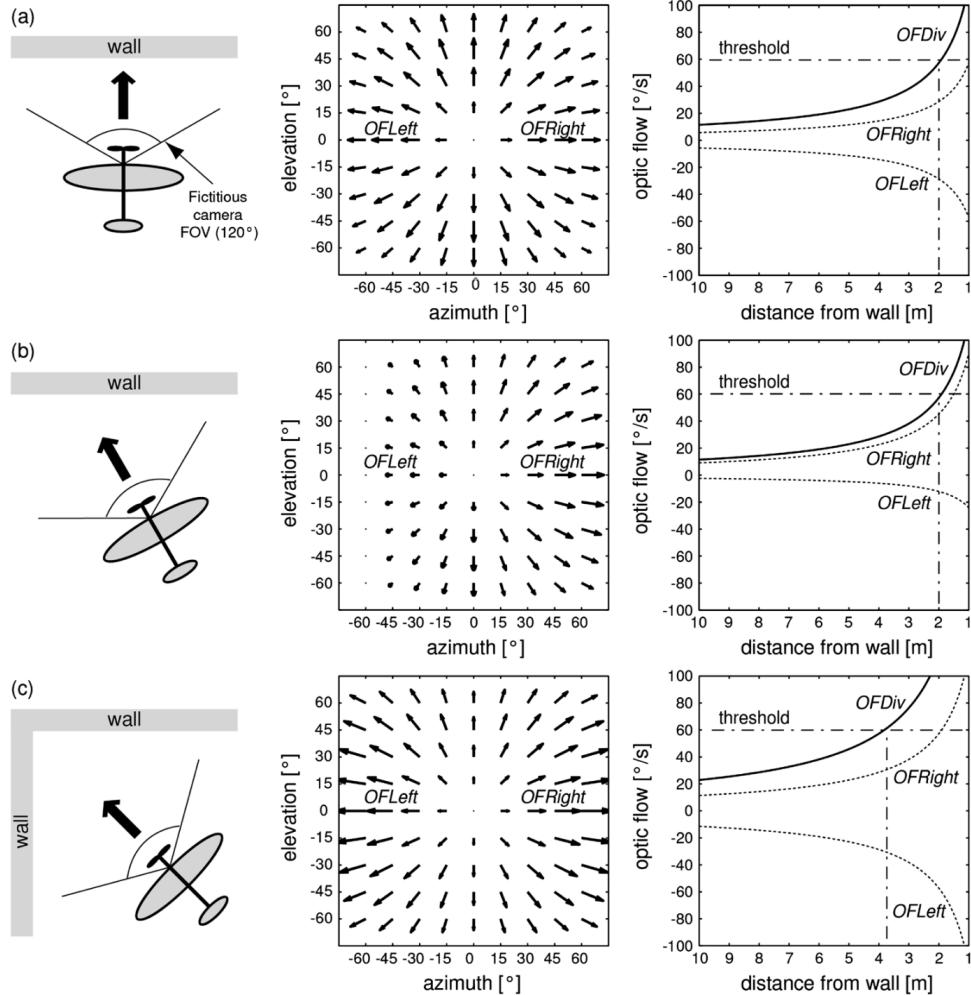


Fig. 5. Ideal motion fields as viewed by a single (fictitious and forward looking) 2-D camera with 120° FOV. (a) Frontal approach toward a wall. (b) Approach at 30°. (c) Approach at 45° to a corner. The first column depicts the fictitious camera position and orientation, as well as the airplane trajectory. In these examples, the aircraft is flying straight at 2 m/s. The second column displays the motion fields occurring in each situation. The third column shows the OF amplitude at 45° azimuth as a function of distance from the wall. OFLeft and OFRight are the OF values measured at $\pm 45^\circ$ azimuth, whereas OFDiv is the sum of OFRight and OFLeft amplitudes, which is a measure of OFDiv, as further explained in the text.

motion field¹ [5]. These fields are based on a fictitious, large FOV, 2-D camera; only portions of these fields are actually covered by the two 1-D cameras present on the robot.

When the plane is flying straight toward a flat surface, the OF field is divergent with an amplitude inversely proportional to the distance, as illustrated in the rightmost graphs of Fig. 5,

¹In general, a difference is made between motion field and OF. The *motion field* results from pure geometrical considerations and is the 2-D projection onto the retina of a moving eye of the relative 3-D motion of scene points. In opposition, the *OF* is defined as the *apparent* motion of the image, and does not necessarily correspond to the motion field, in particular when no contrast is detectable in a given viewing direction or because of the aperture problem [42]. In this paper, however, we assume enough contrast on the surrounding surfaces such that this distinction can be ignored. In practice, we low-pass filter the OF signal over time, so to limit the impact of losing it during a short period of time.

where two OF signals (OFRight and OFLeft) corresponding to the viewing direction of the plane's cameras ($\pm 45^\circ$ azimuth and zero elevation) are plotted. A measurement of OF divergence (OFDiv) can be obtained by subtracting the values of right and left OF amplitudes (as one can read on the last column curves in Fig. 5)²

$$\text{OFDiv} = \text{OFRight} - \text{OFLeft}. \quad (4)$$

This information can thus be used to trigger a saccade when OFDiv exceeds a preset threshold (see, for instance, the dashed

²This way of measuring OFDiv is reminiscent of the minimalist method proposed in [43], using Green's theorem [44].

line at $60^\circ/\text{s}$ corresponding to 2 m from the wall). OFDiv is independent of the angle at which the plane approaches the wall, as can be seen by comparing the curves labeled OFDiv in the first and second rows of Fig. 5. OFDiv is larger when the aircraft happens to fly toward a corner (Fig. 5, third row) because distances from surrounding surfaces are less than the distance from the wall in the heading direction. Therefore, the saccade is triggered farther away than when approaching a flat wall (for instance, the same threshold of $60^\circ/\text{s}$ is reached at more than 3.5 m away), which prevents the plane from going into situations from which it could not easily get out.

The considerations made so far are based on the assumption of constant velocity. It is true that the velocity of the plane will not change abruptly, but one could wonder what would happen if instead of flying at 2 m/s, it slows down to near to the stall speed (1.2 m/s)? It can be shown that TransOF is inversely proportional to the ratio of the distance over the velocity, which is nothing else than the time-to-contact [43], [45], [46]. As a result, even if the forward speed of the flying agent is unknown, OFDiv is proportional to the inverse of the time-to-contact, which means that for a given threshold, the criterion for initiating a saccade will always trigger at the same time before collision. In other words, if the robot moves faster, it will start the saccade farther away from the wall, which is naturally a wise behavior.

In order to decide the direction of the saccade, one can consider another criterion, i.e., the difference between the left and right absolute OF values (OFDiff)

$$\text{OFDiff} = |\text{OFRight}| - |\text{OFLLeft}|. \quad (5)$$

A positive OFDiff means a closer obstacle on the right. Since in this case, only the sign of the criterion is taken into account, the velocity of the plane will have no influence.

B. Rotational OF Correction

One of the problems with this approach is that ensuring that there is absolutely no rotation during straight sequences is nearly impossible, because of air turbulence or small yawing movements due to the gyroscope-based heading control. The resulting RotOF component does not contain any information about surrounding distances, and for all kinds of tasks related to distance, a pure translatory OF field is desirable [47]. This holds for the flying robot just as it does for the fly, which is known to compensate with its head for rotations detected by its halteres [9]. Since the small airplane cannot afford to move its cameras with respect to the fuselage, we propose another means of cancelling residual RotOF, based on the same sensory modality used by the fly.

Since the global OF is a linear combination of translatory and rotatory components [5], if one can measure rotation using another sensory modality, it is, in principle, possible to deduce RotOF from the global flow field by simple vector subtraction. In our case, the situation is even simpler, because the OF detection (see Section IV) is 1-D and the rate gyro axis is oriented perpendicular to the pixel array and the viewing directions, reducing the correction operation to a scalar subtraction. Section IV-B further supports this idea by showing the good match between RotOF and rate gyro output in our robot.

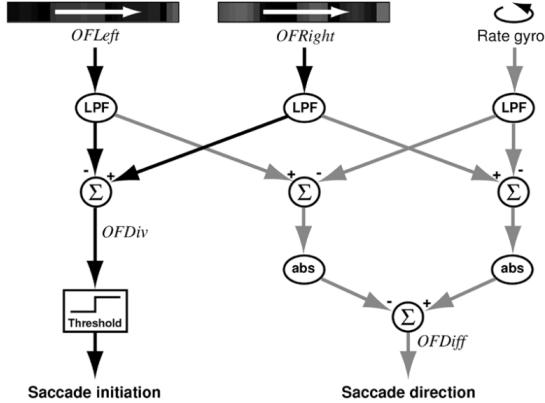


Fig. 6. Signal flow diagram for saccade initiation based on horizontal OFDiv and yaw rotation rate as detected by the gyroscope. "LPF" stands for low-pass filter and "abs" is the absolute value operator.

It is notable, however, that OFDiv, as computed in (4), is not sensitive to yaw rotation. As outlined by Ancona and Poggio [43], this method for computing flow divergence is independent of the location of the focus of expansion (FOE), meaning that even if RotOF due to yaw rotation shifts the FOE, the measured divergence remains unaltered. Unlike OFDiv, OFDiff does suffer from RotOF. As a consequence, it is important, at least for this signal, to discount rotation before computing the criteria in order to decide the saccade direction.

In summary, Fig. 6 gives an overview of the signal flow from sensory input to saccade initiation and direction selection. Temporal leaky integrators are used as a first signal-processing step. Formally equivalent to first-order low-pass filters (LPFs), they also cancel sensory noise. LPFs have equally been proposed as an important component in the model for imminent collision detection in flies [48].

VI. IN-FLIGHT EXPERIMENT

A. Control Implementation

The sensory-motor cycle lasts 80 ms, during which data from onboard sensors are processed, rudder commands are issued, and significant parameters are sent to a laptop for data logging. About 50% of this sensory-motor cycle is used for wireless communication. OF is processed according to (3), taking data from two images grabbed one after the other at the beginning of every sensory-motor cycle. In other words, OF is estimated once every 80 ms.

During a saccade, whose duration is set to 1 s, the rudder deflection follows an experimentally optimized curve up to full deflection in the direction decided by OFDiff. At the end of the saccade, the plane resumes straight flight while it is still in a banked position. Since a banked posture always produces a yawing motion, the proportional controller based on the rate gyro actively compensates, so as to force the plane back to horizontal. This control strategy is summarized in Fig. 7.

As proposed in the model of the fly by Tammero and Dickinson [7], we implemented an inhibition period after each sac-

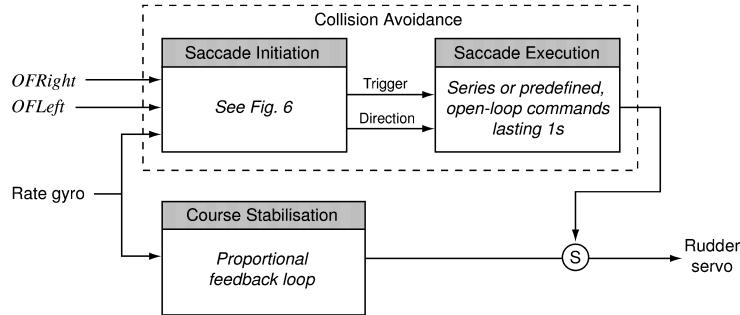


Fig. 7. Overview diagram of the control strategy implementation. On the left are the sensory inputs (OF detectors and gyroscope), and on the right is the system output (the rudder actuator). This diagram is voluntarily inspired from the subsumption architecture proposed by Brooks [49]. The bottom layer accounts for the course stabilization behavior, and the top layer for the collision-avoidance behavior. The encircled “S” represents a suppression node, meaning that, when active, the signal coming from above replaces the signal usually going horizontally through the node.

cade, during which another turning action cannot be triggered. This allows the plane to recover almost straight flight before deciding whether to perform another saccade. In our case, the inhibition lasts as long as the rate gyro indicates an absolute yaw rotational speed larger than $20^{\circ}/\text{s}$. This inhibition is to make sure that when active, the OF detectors are providing reliable data and are not noisy due to excessive rotation rate.

B. Experimental Conditions

Autonomous steering experiments have been carried out in the arena depicted in Fig. 2. In addition to natural light coming from outside, the soft walls were lit from the center of the room with eight projectors sitting on the ground.

The Bluetooth radio connection with a ground-based laptop computer was always established in order to log sensor data in real time while the robot is operating. The plane was started manually from the ground by means of a joystick connected to the laptop. When it reached an altitude of about 2 m, the robot was switched into autonomous steering mode. The human pilot has then no access to the rudder (the vertical control surface, shown in Fig. 1), but could modify the pitch angle by means of the elevator (the horizontal control surface). The operator could switch back to manual mode at any moment, if required.

Before testing the plane in autonomous mode, the OFDiv threshold for saccade trigger has been experimentally determined by flying manually in the arena and recording OF signals when frontally approaching a wall until the last possible moment when the pilot had to start an emergency turn. The recorded data could then be analyzed, and the threshold decided on the basis of the value reached by OFDiv just before steering.

C. Results

The 30-g robot was able to fly collision-free in the $16 \times 16 \text{ m}$ room for more than 4 min without any intervention regarding its steering.³ The plane was engaged in turning actions only 20% of the time, which indicates that it flew always in straight trajectories except when very close to a wall. During those 4 min,

it covered about 300 m in straight motion and generated 50 saccades.

Fig. 8 displays a detailed 18-s sample of the data acquired during typical autonomous flight. Saccade periods are indicated with vertical gray bars spanning all the graphs. In the first row, the rate gyro provides a good indication of the behavior of the plane, i.e., straight trajectories interspersed with turning actions, in which the plane can reach up to $100^{\circ}/\text{s}$. OF is estimated from the 1-D images shown in the second row. The quality of the lightweight imagers and optics does not allow for perfect and noise-free images. As a result, OF estimates (on the two next graphs) are not always very accurate, especially when the plane is close to the walls with a high yaw rotation rate. This situation happens in particular during inhibition periods. Therefore, we set OFDiv and OFDiff (two last rows in Fig. 8) to zero whenever the rate gyro indicates more than $20^{\circ}/\text{s}$. When OFDiv reaches the threshold indicated by the dashed line, a saccade is triggered. The direction of the saccade is given by OFDiff, which is plotted on the bottom graph. The first turning action is leftward, because OFDiff is positive when the saccade is triggered. The other ones are rightward because of the negative value of OFDiff.

VII. DISCUSSION

We presented a bioinspired technique enabling autonomous steering and collision avoidance on an ultralight indoor aircraft. Although the primary purpose of this project is to engineer indoor autonomous microflyers, the size, weight, energy, and computational constraints of the robotic platform encouraged us to look at mechanisms and principles of flight control exploited by insects. Bioinspiration has been used at three different levels in order to meet those constraints.

The first level concerns of the type of sensors we decided to employ. Although the fly also possesses hairs or mechanosensors on its neck or wings [50], eyes and halteres are clearly the most important sensors for flight control. Probably because of their inherent complexity and energy consumption, no active distance sensors, like sonar, are present in the fly. The robot's rate gyro can be seen as a close copy of the fly's halteres, although it measures rotation only about one axis, whereas its biological counterpart is known to be sensitive to rotation around

³Video clips showing those results can be downloaded from <http://phd.zuff.info>.

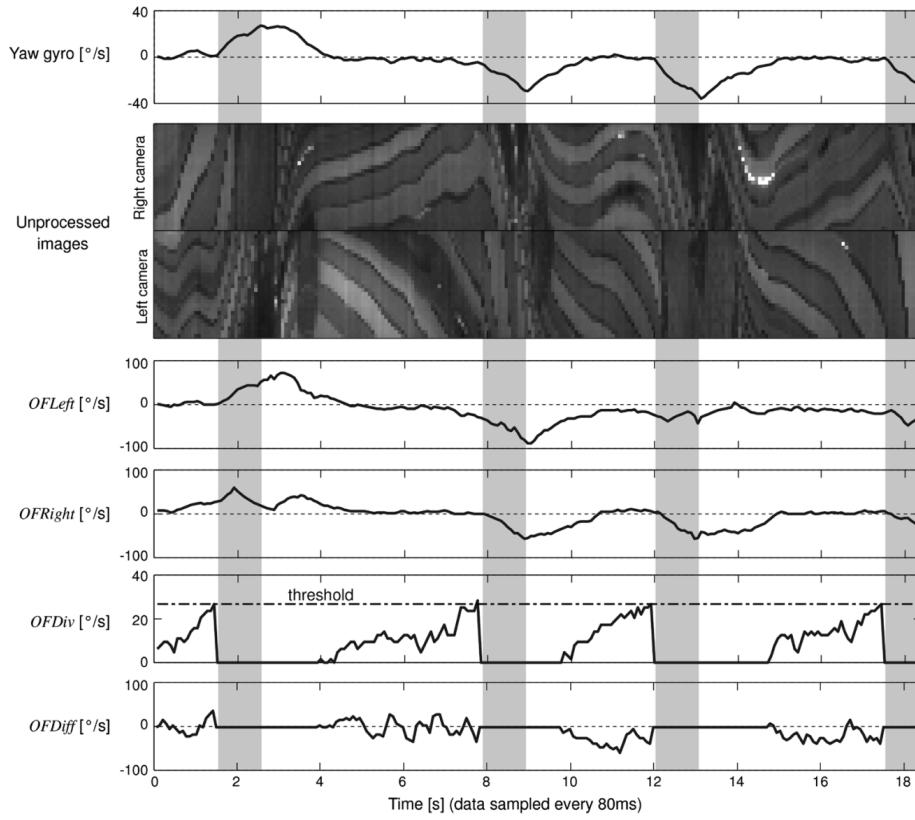


Fig. 8. Sensor and OF data recorded during autonomous flight (about 18 s are displayed). First row is the rate gyro output indicating how much the plane is rotating (leftward positive). The second row displays the raw images (pixels are arranged vertically) grabbed by the two 1-D cameras every sensory-motor cycle. Only the 28 pixels used for OF analysis are displayed for each camera. Third and fourth rows show the OF as estimated from the left and the right cameras, respectively. Fifth and sixth rows show OFDiv and OFDiff when absolute value of the rate gyro is below $20^{\circ}/\text{s}$, i.e., when the plane is roughly flying straight. The dashed horizontal line in the OFDiv graph represents the threshold for triggering a saccade. The gray vertical lines spanning all the graphs indicate the saccades themselves, i.e., when the turning motor program is in action. The first saccade is leftward and the next three are rightward, as indicated by the rate gyro output in the first column.

all three axes [8]. Notice, however, that a future implementation of altitude regulation is likely to require at least one additional rate gyro. The artificial vision system shares with its biological counterpart a rather low resolution, because its interpixel angle (1.4°) is on the same order of magnitude as the interommatidial angle of most flying insects ($1\text{--}3^{\circ}$) [2]. The FOV of our airplane is much smaller than that of most flying insects. Although it is sufficient for the simple environment used here, it will have to be expanded for more complex indoor situations.

The second level of bioinspiration concerns the stage of sensor signal processing. Although the extraction of OF itself is not based on correlation-based elementary motion detectors [51] because of their known dependency on contrast and spatial frequency [52], OF is employed as a primary cue in the behavioral control of the airplane, just as it is in the fly [4]. In particular, using looming cues for detecting approaching objects are believed to play a major role in the initiation of saccades [7]. A model explaining the landing reflex of flies has been proposed [48] where the OFDiv signal is integrated over

time (with some leakage) until it exceeds a threshold, which finally triggers an action. Although the neuroanatomical details are not yet fully understood, the same model is also employed to explain the initiation of saccades [53]. The attractive feature of this simple scheme is that it does not require explicit measurement of distance or time-to-contact, nor does it rely on accurate knowledge of the flight velocity. This idea has also been stressed by Srinivasan and colleagues [54], who pointed out that it is often possible to achieve efficient behaviors like collision avoidance, landing, corridor following, and altitude control by simply relying on OF values rather than trying to estimate accurate distances. This OF-based approach without explicit estimation of distances has also been used in artificial systems for altitude control and automatic landing [28], [55]. Another example of bioinspired signal processing is the fusion of inertial information with vision. Although the simple scalar summation employed in our robot is probably far from what actually happens in the fly's nervous system, it is clear that some important interactions between visual input and haltere

feedback exist [8]. Recently, a model based on a weighted sum between the two sensory modalities has even been proposed to explain the fly's equilibrium reflex [56].

The third level of bioinspiration concerns trajectory control, where the flight along straight lines interspersed with rapid turning actions is patterned after fly behavior. While in flies, some saccades are spontaneously generated in the absence of any visual input, reconstruction of OF patterns based on flies' motion through an artificial visual landscape suggests that image expansion plays a role in triggering saccades [7]. Aside from providing a first stage of RotOF cancellation, straight flight sequences also increase the quality of visual input by maintaining a horizontal attitude. In addition, straight flight is energetically efficient, because when a plane banks, it has to produce additional lift in order to compensate for the centripetal force. In our implementation, the entire saccade is performed without sensory feedback, whereas flies may continue to rely on sensory input. Although during saccades elementary motion detectors [51] are known to operate beyond their linear range, where the signal could even be reversed because of temporal aliasing [52], whether or not visual feedback plays an important role for controlling these fast turning maneuvers is still under investigation [53]. However, haltere feedback is more likely to have a major impact on the duration of the saccade [8], and integration of the rate gyro output over time could provide a good way of ensuring a given rotation angle independently of the aircraft velocity at the beginning of the saccade. Although the precise roles of halteres and vision in course or gaze stabilization of flies is still unclear [39], both sensory modalities are known to have influences. In our implementation, course stabilization and RotOF cancellation, which can be seen as the counterpart of gaze stabilization, rely only on inertial information.

VIII. FUTURE WORK

We are currently integrating an altitude-control system based on an additional OF detector pointing downward and using the same kind of RotOF cancellation as employed for the steering control [1]. Alternative approaches without explicit OF estimation are also being assessed for altitude control [58].

Flying in natural indoor environments is another issue we would like to address. The restricted FOV and constant camera brightness settings employed so far are probably unsuitable for the inhomogeneous lighting and textures of such environments. To tackle this problem, we are developing analog VLSI vision sensors (similar to [57]) with adaptive photoreceptors and built-in OF detection. These chips will provide quick adaptation to ambient light, while being lighter and less energy demanding than the current 1-D cameras. These characteristics are expected to provide a means for reliably detecting OF in more natural environments, and for increasing the FOV covered by the compound eye of our artificial flying insect by using more imaging devices for less weight and power consumption.

ACKNOWLEDGMENT

The authors are grateful to J.-D. Nicoud (<http://www.didel.com>), A. Guignard, and G. Vaucher for their

invaluable support in building the 30-g flying platform. We would like to thank D. Zufferey and C. Ray for their help in constructing the experimental room, and A. Beyeler for his support throughout this project. The authors are equally indebted to the three anonymous reviewers, whose constructive comments have allowed them to significantly improve this paper, and to M. Waibel and C. Cianci who have proofread the manuscript.

REFERENCES

- [1] J. Zufferey and D. Floreano, "Toward 30-gram autonomous indoor aircraft: Vision-based obstacle avoidance and altitude control," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 2605–2610.
- [2] M. Land, "Visual acuity in insects," *Annu. Rev. Entomol.*, vol. 42, pp. 147–177, 1997.
- [3] J. Gibson, *The Perception of the Visual World*. Boston, MA: Houghton Mifflin, 1950.
- [4] M. Egelhaaf and R. Kern, "Vision in flying insects," *Current Opinion Neurobiol.*, vol. 12, no. 6, pp. 699–706, 2002.
- [5] J. Koenderink and A. van Doorn, "Facts on optic flow," *Biol. Cybern.*, vol. 56, pp. 247–254, 1987.
- [6] H. Wagner, "Flight performance and visual control of flight of the free-flying housefly (*musca domestica l.*). I. Organization of the flight motor," *Philos. Trans. Roy. Soc. B*, vol. 312, pp. 527–551, 1986.
- [7] L. Tammero and M. Dickinson, "The influence of visual landscape on the free flight behavior of the fruit fly *drosophila melanogaster*," *J. Exp. Biol.*, vol. 205, pp. 327–343, 2002.
- [8] M. Dickinson, "Haltere-mediated equilibrium reflexes of the fruit fly, *drosophila melanogaster*," *Philos. Trans.: Biol. Sci.*, vol. 354, no. 1385, pp. 903–916, 1999.
- [9] G. Nalbach and R. Hengstenberg, "The halteres of the blowfly caliphora—Three-dimensional organization of compensatory reactions to real and simulated rotations," *J. Comparative Physiol. A*, vol. 175, pp. 695–708, 1994.
- [10] N. Franceschini, J. Pichon, and C. Blanes, "From insect vision to robot vision," *Philos. Trans. Roy. Soc. B*, vol. 337, pp. 283–294, 1992.
- [11] M. Srinivasan, J. Chahl, M. Nagle, and S. Zhang, "Embodying natural vision into machines," in *From Living Eyes to Seeing Machines*, M. Srinivasan and S. Venkatesh, Eds. Oxford, U.K.: Oxford Univ. Press, 1997, pp. 249–265.
- [12] A. Duchon, W. H. Warren, and L. Kaelbling, "Ecological robotics," *Adapt. Behav.*, vol. 6, pp. 473–507, 1998.
- [13] M. Lewis, "Visual navigation in a robot using zig-zag behavior," in *Neural Information Processing Systems*. Cambridge, MA: MIT Press, 1998, vol. 10.
- [14] D. Coombs, M. Herman, T. Hong, and M. Nashman, "Real-time obstacle avoidance using central flow divergence and peripheral flow," in *Proc. 5th Int. Conf. Comput. Vis.*, 1995, pp. 276–283.
- [15] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi, "Divergent stereo for robot navigation: A step forward to a robotic bee," *Int. J. Comput. Vis.*, vol. 14, pp. 159–177, 1995.
- [16] K. Weber, S. Venkatesh, and M. Srinivasan, "Insect inspired behaviors for the autonomous control of mobile robots," in *From Living Eyes to Seeing Machines*, M. V. Srinivasan and S. Venkatesh, Eds. Oxford, U.K.: Oxford Univ. Press, 1997, pp. 226–248.
- [17] M. Srinivasan, J. Chahl, K. Weber, S. Venkatesh, and H. Zhang, "Robot navigation inspired by principles of insect vision," in *Field and Service Robotics*, A. Zelinsky, Ed. New York: Springer-Verlag, 1998, pp. 12–16.
- [18] J. Zufferey, "Bioinspired vision-based flying robots," Ph.D. dissertation, Swiss Federal Inst. Technol. Lausanne (EPFL), Lausanne, Switzerland, 2005.
- [19] I. Kroo and P. Kunz, "Mesoscale flight and miniature rotorcraft development," in *Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Applications*. ser. Progress in Astronautics and Aeronautics, T. J. Mueller, Ed. AIAA, 2001, vol. 195, pp. 503–517.
- [20] T. Pornsin-Sirirak, Y.-C. Tai, and C.-M. Ho, Microbat: A palm-sized electrically powered ornithopter. presented at *Proc. NASA/JPL Workshop Biomimetic Robot*. [Online]. Available: <http://touch.caltech.edu/publications/2001/jpl/jpl2001.pdf>

- [21] K. Jones, C. Bradshaw, J. Papadopoulos, and M. Platzter, "Improved performance and control of flapping-wing propelled micro air vehicles," in *Proc. 42nd Aerosp. Sci. Meeting Exhibit.*, Reno, NV, 2004, Paper 2004-0399.
- [22] R. Fearing, K. Chiang, M. Dickinson, D. Pick, M. Sitti, and J. Yan, "Wing transmission for a micromechanical flying insect," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, pp. 1509–1516.
- [23] S. Avadhanula, R. Wood, E. Steltz, J. Yan, and R. Fearing, "Lift force improvements for the micromechanical flying insect," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2003, pp. 1350–1356.
- [24] W. Wu, L. Schenato, R. Wood, and R. Fearing, "Biomimetic sensor suite for flight control of a micromechanical flight insect: Design and experimental results," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, R.O.C., 2003, pp. 1146–1151.
- [25] L. Schenato, W. Wu, and S. Sastry, "Attitude control for a micromechanical flying insect via sensor output feedback," *IEEE Trans. Robot. Autom.*, vol. 20, no. 1, pp. 93–106, Feb. 2004.
- [26] J. Nicoud and J. Zufferey, "Toward indoor flying robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 787–792.
- [27] F. Mura and N. Franceschini, "Visual control of altitude and speed in a flying agent," in *From Animals to Animats III*. Cambridge, MA: MIT Press, 1994, pp. 91–99.
- [28] F. Ruffier and N. Franceschini, "Visually guided micro-aerial vehicle: Automatic take off, terrain following, landing and wind reaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, T. J. Tarn, T. Fukuda, and K. Valavanis, Eds., New Orleans, LA, Apr. 2004, pp. 2339–2346.
- [29] J. Chahl, M. Srinivasan, and H. Zhang, "Landing strategies in honeybees and applications to uninhabited airborne vehicles," *Int. J. Robot. Res.*, vol. 23, no. 2, pp. 101–1102, 2004.
- [30] T. Neumann and H. Bültthoff, "Behavior-oriented vision for biomimetic flight control," in *Proc. EPSRC/BBSRC Int. Workshop Biol. Inspired Robot.*, 2002, pp. 196–203.
- [31] L. Muratet, S. Doncieux, Y. Brière, and J.-A. Meyer, "A contribution to vision-based autonomous helicopter flight in urban environments," *Robot. Auton. Syst.*, to be published.
- [32] M. Reiser and M. Dickinson, "A test bed for insect-inspired robotic control," *Philos. Trans. Math., Phys., Eng. Sci.*, vol. 361, pp. 2267–2285, 2003.
- [33] G. Barrows and C. Neely, "Mixed-mode VLSI optic flow sensors for in-flight control of a micro air vehicle," *Critical Technol. Future Comput., SPIE*, vol. 4109, pp. 52–63, 2000.
- [34] G. Barrows, C. Neely, and K. Miller, "Optic flow sensors for MAV navigation," in *Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Applications*, see *Progress in Astronautics and Aeronautics*, T. J. Mueller, Ed: AIAA, 2001, vol. 195, pp. 557–574.
- [35] W. Green, P. Oh, K. Sevcik, and G. Barrows, "Autonomous landing for indoor flying robots using optic flow," in *Proc. ASME Int. Mech. Eng. Congr. Expo.*, vol. 2, 2003, pp. 1347–1352.
- [36] P. Oh, W. Green, and G. Barrows, "Closed quarter aerial robot prototype to fly in and around buildings," in *Proc. Int. Conf. Comput., Commun., Control Technol.*, vol. 5, 2003, pp. 302–307.
- [37] W. Green, P. Oh, and G. Barrows, "Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, 2004, pp. 2347–2352.
- [38] R. Nelson and Y. Aloimonos, "Finding motion parameters from spherical flow fields (or the advantages of having eyes in the back of your head)," *Biol. Cybern.*, vol. 58, pp. 261–273, 1988.
- [39] H. Krapp, "Neuronal matched filters for optic flow processing in flying insects," in *Neuronal Processing of Optic Flow*, M. Lappe, Ed. San Diego, CA: Academic, 2000, pp. 93–120.
- [40] J. Zufferey, A. Beyeler, and D. Floreano, "Vision-based navigation from wheels to wings," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, 2003, pp. 2968–2973.
- [41] M. Srinivasan, "An image-interpolation technique for the computation of optic flow and egomotion," *Biol. Cybern.*, vol. 71, pp. 401–416, 1994.
- [42] B. Horn, *Robot Vision*. Cambridge, MA: MIT Press, 1986.
- [43] N. Ancona and T. Poggio, "Optical flow from 1D correlation: Application to a simple time-to-collision detector," in *Proc. 4th Int. Conf. Comput. Vis.*, 1993, pp. 209–214.
- [44] T. Poggio, A. Verri, and V. Torre, "Green theorems and qualitative properties of the optical flow," Mass. Inst. Technol., Cambridge, MA, Tech. Rep. A.I. Memo 1289, 1991.
- [45] D. Lee, "A theory of visual control of braking based on information about time-to-collision," *Perception*, vol. 5, pp. 437–459, 1976.
- [46] T. Camus, "Calculating time-to-contact using real-time quantized optical flow," Nat. Inst. Standards Technol., Tech. Rep. 5609, 1995.
- [47] M. Srinivasan, S. Zhang, M. Lehrer, and T. Collett, "Honeybee navigation en route to the goal: Visual flight control and odometry," *J. Exp. Biol.*, vol. 199, pp. 237–244, 1996.
- [48] A. Borst, "How do flies land? From behavior to neuronal circuits," *BioSci.*, vol. 40, no. 4, pp. 292–299, 1990.
- [49] R. Brooks, *Cambran Intelligence*. Cambridge, MA: The MIT Press, 1999.
- [50] R. Hengstenberg, "Gaze control in the blowfly calliphora: A multisensory two-stage integration process," *Neurosci.*, vol. 3, pp. 19–29, 1991.
- [51] W. Reichardt, "Movement perception in insects," in *Processing of Optical Data by Organisms and by Machines*, W. Reichardt, Ed. New York: Academic, 1969, pp. 465–493.
- [52] M. Srinivasan, M. Poteser, and K. Kral, "Motion detection in insect orientation and navigation," *Vis. Res.*, vol. 39, no. 16, pp. 2749–2766, 1999.
- [53] L. Tammero and M. Dickinson, "Collision-avoidance and landing responses are mediated by separate pathways in the fruit fly," *J. Exp. Biol.*, vol. 205, pp. 2785–2798, 2002.
- [54] M. Srinivasan, S. Zhang, J. Chahl, E. Barth, and S. Venkatesh, "How honeybees make grazing landings on flat surfaces," *Biol. Cybern.*, vol. 83, pp. 171–183, 2000.
- [55] N. Franceschini, "From fly vision to robot vision: Reconstruction as a mode of discovery," in *Sensors and Sensing in Biology and Engineering*, F. G. Barth, J. A. Humphrey, and T. W. Secomb, Eds. New York: Springer, 2003, pp. 223–235.
- [56] A. Sherman and M. Dickinson, "Summation of visual and mechanosensory feedback in *drosophila* flight control," *J. Exp. Biol.*, vol. 207, pp. 133–142, 2004.
- [57] J. Kramer, R. Sarapeshkar, and C. Koch, "An analog VLSI velocity sensor," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1995, pp. 413–416.
- [58] A. Beyeler, C. Mattiusi, J. Zufferey, and D. Floreano, "Vision-based altitude and pitch estimation for ultralight indoor aircraft," in *Proc. IEEE Int. Conf. Robot. Autom.*, to be published.



Jean-Christophe Zufferey (M'05) completed his master project at Carnegie Mellon University, Pittsburgh, PA, as an exchange student, and he received the M.S. degree in microengineering in 2001. In 2005, he received the Ph.D. degree in autonomous robotics.

He is currently a Research Scientist with the Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, in the Laboratory of Intelligent Systems. His research interests are in aerial, bioinspired, and evolutionary robotics.

He has coauthored eight peer-reviewed journal and conference papers. He is also Co-Founder of an EPFL spin-off company, DIDEL, which is involved in educational robotics and ultralight indoor slow-flyers.



Dario Floreano (M'96) is currently an Associate Professor of Intelligent Systems at the Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland, where he is also Director of the Laboratory of Intelligent Systems and of the Institute of Systems Engineering. His research activities span bioinspired and evolutionary robotics, biomimetic electronics, neural computation, self-organizing systems, and biology reverse engineering.

He has published more than 100 peer-reviewed papers, authored two books, and edited three other books. The book *Evolutionary Robotics* was reprinted by MIT Press three times over the last four years. He co-organized eight international conferences, and joined the program committee of more than 70 other conferences. He is on the editorial board of eight international journals: *Neural Networks*, *Genetic Programming and Evolvable Machines*, *Adaptive Behavior*, *Artificial Life*, *Connection Science*, *Evolutionary Computation*, the *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, and *Autonomous Robots*. He is cofounder and member of the Board of Directors of the International Society for Artificial Life, Inc., and a member of the Board of Governors of the International Society for Neural Networks.

8

Scene Reconstruction

In this chapter we will consider the problem of reconstruction a local voxel grid representation of the world based on a video sequence collected by the camera. A high-level architecture for scene reconstruction is shown in Figure 8.1.

The camera image at time $k - 1$ is used to identify good features in the image. Those features are then tracked to the image I_k to produce a set of feature pairs in calibrated normalized homogeneous coordinates, denoted $\mathcal{P}_k = \{(\bar{\epsilon}_i^{k-1}, \bar{\epsilon}_i^k), i = 1, \dots, N\}$. The feature pairs are then used to reconstruct the 3D points corresponding to the feature points. In this chapter we will assume that the camera's rotation and translation between images is known, and therefore the 3D reconstruction algorithm access to these quantities denoted in Figure 8.1 as $(R_{k-1}^k \mathbf{p}_{k-1/k}^k)$. The 3D scene points are used to update the rolling voxel grid described in Chapter 5.

In Section 8.1 we review how IMU integration can be used to obtain the relative pose $(R_{k-1}^k, \mathbf{p}_{k-1/k}^k)$. Section 10.2 derives the essential matrix and discusses some of its properties. Section 10.3 describes the homography matrix and some of its properties. Section 8.3 gives several simple point reconstruction algorithms. Section 8.7 describes how to update the rolling voxel map given point cloud information.

Embedded papers describe more sophisticated reconstruction techniques.

8.1 IMU Integration

The methods used in this chapter will depend on knowledge of the relative pose between camera frames. Let \mathcal{F}_k be the camera frame at time k , then the relative pose is given by $(R_{k-1}^k, \mathbf{p}_{k-1/k}^k)$, where $R_{k-1}^k \in SO(3)$ is the rotation matrix from \mathcal{F}_{k-1} to \mathcal{F}_k , and $\mathbf{p}_{k-1/k}^k \in \mathbb{E}^3$ is the position of \mathcal{F}_{k-1} relative to \mathcal{F}_k , expressed in \mathcal{F}_k .

In this section we show how the relative pose $(R_{k-1}^k, \mathbf{p}_{k-1/k}^k)$ can be obtained from an IMU rigidly attached to the camera. The dis-

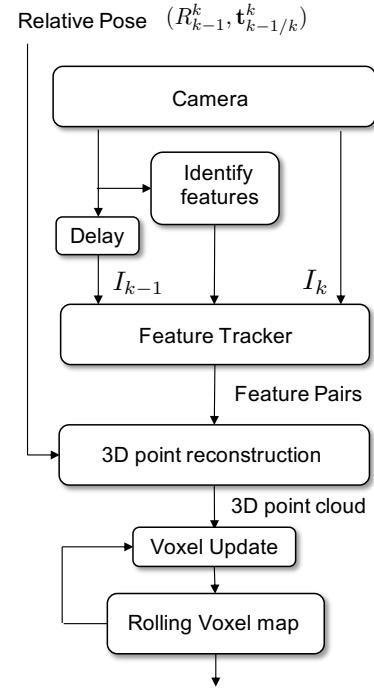


Figure 8.1: Architecture for Scene Reconstruction

cussion in this section follows in some respects, the development in (Forster 2017)¹. We will assume in this section that the IMU is rigidly attached to the camera and that the IMU has been calibrated and the biases have been removed. Then the output of the IMU is the angular velocity $\omega_{c/i}^c$ and the specific acceleration $\mathbf{a}_{c/i}^c$, where \mathbf{F}_c is the camera frame and \mathcal{F}_i is the inertial frame. We will assume that the camera moves according to

$$\dot{\mathbf{p}}_{c/i}^i = \mathbf{v}_{c/i}^i \quad (8.1)$$

$$\dot{\mathbf{v}}_{c/i}^i = g\mathbf{e}_3 + R_c^i \mathbf{a}_{c/i}^c, \quad (8.2)$$

$$\dot{R}_c^i = R_c^i [\omega_{c/i}^c]_\times. \quad (8.3)$$

We begin by developing a sampled-data version of the equations of motion.

Lemma 8.1.1 *Suppose that the camera evolves according to (8.1)–(??) and suppose that over one sample period $(t_\ell, t_{\ell+1})$ of length $T_s = t_{\ell+1} - t_\ell$, that the angular velocity $\omega_{\ell/i}^\ell \triangleq \omega_{c/i}^c(t_\ell)$ and the specific acceleration $\mathbf{a}_{\ell/i}^\ell \triangleq \mathbf{a}_{c/i}^c(t_\ell)$ are constant. Then the discrete evolution equations are given by*

$$R_{\ell+1}^i = R_\ell^i \exp \left([\omega_{\ell/i}^\ell]_\times T_s \right) \quad (8.4)$$

$$\mathbf{v}_{\ell+1/i}^i = \mathbf{v}_{\ell/i}^i + T_s g\mathbf{e}_3 + T_s R_\ell^i \mathbf{a}_{\ell/i}^\ell \quad (8.5)$$

$$\mathbf{p}_{\ell+1/i}^i = \mathbf{p}_{\ell/i}^i + T_s \mathbf{v}_{\ell/i}^i + \frac{T_s^2}{2} g\mathbf{e}_3 + \frac{T_s^2}{2} R_\ell^i \mathbf{a}_{\ell/i}^\ell. \quad (8.6)$$

Proof: We have shown in Lemma ?? that for a constant angular velocity, the solution of Equation (8.3) is (8.4). For a constant input, integrating (8.2) gives

$$\begin{aligned} \mathbf{v}_{c/i}^i(t) &= \mathbf{v}_{c/i}^i(t_0) + \int_{t_0}^t (g\mathbf{e}_3 + R_c^i \mathbf{a}_{c/i}^c) d\tau \\ &= \mathbf{v}_{c/i}^i(t_0) + \int_{t_0}^t d\tau (g\mathbf{e}_3 + R_c^i \mathbf{a}_{c/i}^c) \\ &= \mathbf{v}_{c/i}^i(t_0) + (t - t_0) (g\mathbf{e}_3 + R_c^i \mathbf{a}_{c/i}^c). \end{aligned}$$

Letting the camera frame at time t_ℓ be denoted as \mathcal{F}_ℓ , and letting $t = t_{\ell+1}$ and $t_0 = t_\ell$ gives Equation (8.5). Similarly, integrating Equation (8.1) gives

$$\begin{aligned} \mathbf{p}_{c/i}^c(t) &= \mathbf{p}_{c/i}^c(t_0) + \int_{t_0}^t \mathbf{v}_{c/i}^i(\tau) d\tau \\ &= \mathbf{p}_{c/i}^c(t_0) + \int_{t_0}^t \left(\mathbf{v}_{c/i}^i(t_0) + \int_{t_0}^\tau (g\mathbf{e}_3 + R_c^i \mathbf{a}_{c/i}^c) d\sigma \right) d\tau \\ &= \mathbf{p}_{c/i}^c(t_0) + (t - t_0) \mathbf{v}_{c/i}^i(t_0) + \frac{(t - t_0)^2}{2} (g\mathbf{e}_3 + R_c^i \mathbf{a}_{c/i}^c), \end{aligned}$$

and Equation (8.6) follows by letting $t = t_{\ell+1}$ and $t_0 = t_\ell$. \blacksquare

¹ Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, February 2017

Since the relative pose does not include the inertial frame, we would like to remove references to the inertial frame in Equations (8.6)–(8.4). The next lemma does so.

Lemma 8.1.2 Suppose that the camera evolves according to (8.1)–(8.3) and suppose that over one sample period $(t_\ell, t_{\ell+1})$ of length $T_s = t_{\ell+1} - t_\ell$, that the angular velocity $\omega_{\ell/i}^\ell \triangleq \omega_{c/i}^c(t_\ell)$ and the specific acceleration $\mathbf{a}_{\ell/i}^\ell \triangleq \mathbf{a}_{c/i}^c(t_\ell)$ are constant. Then the discrete evolution from IMU sample ℓ to IMU sample $\ell + 1$ is given by

$$R_\ell^{\ell+1} = \exp\left(-\left[\omega_{\ell/i}^\ell\right]_x T_s\right) \quad (8.7)$$

$$\mathbf{v}_{\ell/\ell+1}^{\ell+1} = R_\ell^{\ell+1} \left(\mathbf{v}_{\ell-1/\ell}^\ell + T_s \left(R_{\ell-1}^\ell \mathbf{a}_{\ell-1/i}^{\ell-1} - \mathbf{a}_{\ell/i}^\ell \right) \right) \quad (8.8)$$

$$\mathbf{p}_{\ell/\ell+1}^{\ell+1} = R_\ell^{\ell+1} \left(\mathbf{p}_{\ell-1/\ell}^\ell + T_s \mathbf{v}_{\ell-1/\ell}^\ell + \frac{T_s^2}{2} \left(R_{\ell-1}^\ell \mathbf{a}_{\ell-1/i}^{\ell-1} - \mathbf{a}_{\ell/i}^\ell \right) \right). \quad (8.9)$$

Proof: The relative pose between IMU measurements is given by

$$R_\ell^{\ell+1} = R_{\ell+1}^{i\top} R_\ell^i = \exp\left(-\left[\omega_{\ell/i}^\ell\right]_x T_s\right)$$

where the second equation follow from Equation (8.4). For the relative velocity, we have from Equation (8.5) that

$$\begin{aligned} \mathbf{v}_{\ell/\ell+1}^{\ell+1} &= R_{\ell+1}^{i\top} (\mathbf{v}_{\ell/i}^i - \mathbf{v}_{\ell+1/i}^i) \\ &= R_{\ell+1}^{i\top} \left(\mathbf{v}_{\ell-1/i}^i + T_s g \mathbf{e}_3 + T_s R_{\ell-1}^i \mathbf{a}_{\ell-1/i}^{\ell-1} \right. \\ &\quad \left. - \mathbf{v}_{\ell/i}^i - T_s g \mathbf{e}_3 - T_s R_\ell^i \mathbf{a}_{\ell/i}^\ell \right) \\ &= R_\ell^{\ell+1} \left(\mathbf{v}_{\ell-1/\ell}^\ell + T_s \left(R_{\ell-1}^\ell \mathbf{a}_{\ell-1/i}^{\ell-1} - \mathbf{a}_{\ell/i}^\ell \right) \right). \end{aligned}$$

Similarly, the relative translation vector is given by

$$\begin{aligned} \mathbf{p}_{\ell/\ell+1}^{\ell+1} &= R_{\ell+1}^{i\top} (\mathbf{p}_{\ell/i}^i - \mathbf{p}_{\ell+1/i}^i) \\ &= R_{\ell+1}^{i\top} \left(\mathbf{p}_{\ell-1/i}^i + T_s \mathbf{v}_{\ell-1/i}^i + \frac{T_s^2}{2} g \mathbf{e}_3 + \frac{T_s^2}{2} R_{\ell-1}^i \mathbf{a}_{\ell-1/i}^{\ell-1} \right. \\ &\quad \left. - \mathbf{p}_{\ell/i}^i - T_s \mathbf{v}_{\ell/i}^i - \frac{T_s^2}{2} g \mathbf{e}_3 - \frac{T_s^2}{2} R_\ell^i \mathbf{a}_{\ell/i}^\ell \right) \\ &= R_\ell^{\ell+1} \left(\mathbf{p}_{\ell-1/\ell}^\ell + T_s \mathbf{v}_{\ell-1/\ell}^\ell + \frac{T_s^2}{2} \left(R_{\ell-1}^\ell \mathbf{a}_{\ell-1/i}^{\ell-1} - \mathbf{a}_{\ell/i}^\ell \right) \right). \end{aligned}$$

■

If there are m IMU samples between camera frame $k - 1$ and camera frame k , then the relative pose can be estimated using the following algorithm.

Initialization. At frame $k - 1$, upon receiving IMU measurement $\ell = 0$ at time t_{k-1} initialize the relative pose as

$$\begin{aligned} R_{k-1}^1 &= \exp\left(-[\omega_{0/i}^0]_x T_s\right) \\ \mathbf{v}_{k-1/1}^1 &= 0 \\ \mathbf{p}_{k-1/1}^1 &= 0. \end{aligned}$$

Iteration. For the ℓ^{th} IMU measurement $(\omega_{\ell/i}^\ell, \mathbf{a}_{\ell/i}^\ell)$, $\ell = 1, \dots, m$ let

$$\begin{aligned} R_{k-1}^{\ell+1} &= R_\ell^{\ell+1} R_{k-1}^\ell \\ \mathbf{v}_{k-1/\ell+1}^{\ell+1} &= R_\ell^{\ell+1} \mathbf{v}_{k-1/\ell}^\ell + \mathbf{v}_{\ell/\ell+1}^{\ell+1} \\ \mathbf{p}_{k-1/\ell+1}^{\ell+1} &= R_\ell^{\ell+1} \mathbf{p}_{k-1/\ell}^\ell + \mathbf{p}_{\ell/\ell+1}^{\ell+1}, \end{aligned}$$

where $R_\ell^{\ell+1}$, $\mathbf{v}_{\ell/\ell+1}^{\ell+1}$, and $\mathbf{p}_{\ell/\ell+1}^{\ell+1}$ are given by Equations (8.7)–(8.9).

When $\ell = m$, then $t_m = t_k$ and the algorithm has determined the relative pose $(R_{k-1}^k, \mathbf{p}_{k-1/k}^k)$.

RWB: Note that this is different than other IMU integration methods in that the method uses the difference of accelerometer measurements. It will be interesting to see if this work!

RWB: Add a description of a VIO approach. The above is just inertial odometry (IO), to correct for biases, need to add the visual correction.

8.2 Essential Matrix

Figure ?? shows the essence of epipolar geometry. Let \mathbf{p}_f be the 3D position of a feature point in the world, and let $\mathbf{p}_{f/a}^a$ be the position vector of the feature point relative to frame \mathcal{F}_a expressed in frame \mathcal{F}_a , and similarly for $\mathbf{p}_{f/b}^b$. The relationship between $\mathbf{p}_{f/a}^a$ and $\mathbf{p}_{f/b}^b$ is

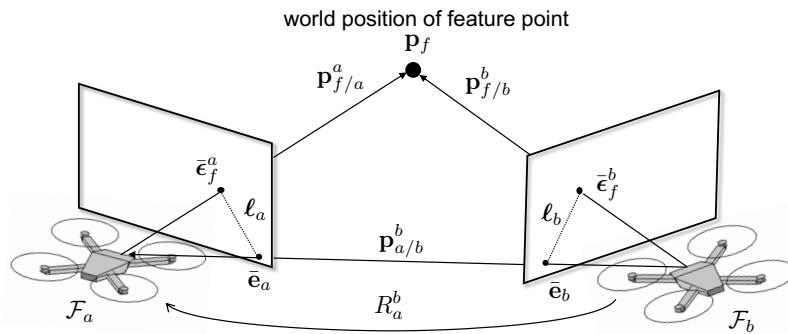


Figure 8.2: Epipolar Geometry

given by

$$\mathbf{p}_{f/b}^b = R_a^b \mathbf{p}_{f/a}^a + \mathbf{p}_{a/b}^b, \quad (8.10)$$

as shown in Figure ???. Multiplying both sides of Equation (8.10) on the left by $\left[\mathbf{p}_{a/b}^b \right]_\times$ gives

$$\left[\mathbf{p}_{a/b}^b \right]_\times \mathbf{p}_{f/b}^b = \left[\mathbf{p}_{a/b}^b \right]_\times R_a^b \mathbf{p}_{f/a}^a,$$

where we have used the fact that $\left[\mathbf{p}_{a/b}^b \right]_\times \mathbf{p}_{a/b}^b = \mathbf{p}_{a/b}^b \times \mathbf{p}_{a/b}^b = 0$.

Since $\left[\mathbf{p}_{a/b}^b \right]_\times \mathbf{p}_{f/b}^b = \mathbf{p}_{a/b}^b \times \mathbf{p}_{f/b}^b$ must be orthogonal to $\mathbf{p}_{f/b}^b$ we have that

$$\mathbf{p}_{f/b}^{b\top} \left[\mathbf{p}_{a/b}^b \right]_\times R_a^b \mathbf{p}_{f/a}^a = 0. \quad (8.11)$$

Dividing Equation (8.11) by the norm of $\mathbf{p}_{a/b}^b$, and defining

$$\mathbf{n}_{a/b}^b \triangleq \frac{\mathbf{p}_{a/b}^b}{\|\mathbf{p}_{a/b}^b\|}$$

gives

$$\mathbf{p}_{f/b}^{b\top} \left[\mathbf{n}_{a/b}^b \right]_\times R_a^b \mathbf{p}_{f/a}^a = 0. \quad (8.12)$$

The matrix

$$E = \left[\mathbf{n}_{a/b}^b \right]_\times R_a^b \quad (8.13)$$

is called the essential matrix, and is completely defined by the relative pose $(R_a^b, \mathbf{p}_{a/b}^b)$.

Theorem 8.2.1 *Given the geometry shown in Figure ?? with relative pose $(R_a^b, \mathbf{p}_{a/b}^b)$, and the associated essential matrix given in Equation (??). Every pair of matching feature points $(\bar{\epsilon}_{f/a}^a, \bar{\epsilon}_{f/b}^b)$ in frames a and b , expressed in normalized (calibrated) coordinates satisfies the so-called epipolar constraint*

$$\bar{\epsilon}_{f/b}^{b\top} E \bar{\epsilon}_{f/a}^a = 0. \quad (8.14)$$

Proof: The normalized pixel coordinates of the feature point projected on to image a and image b satisfy

$$\begin{aligned} \lambda_a \bar{\epsilon}_{f/a}^a &= \mathbf{p}_{f/a}^a \\ \lambda_b \bar{\epsilon}_{f/b}^b &= \mathbf{p}_{f/b}^b, \end{aligned}$$

where λ_a and λ_b are the distances to the feature point along the optical axes in frame a and b , respectively. Substituting into Equation (8.12) gives

$$\bar{\epsilon}_{f/b}^{b\top} E \bar{\epsilon}_{f/a}^a = 0. \quad (8.15)$$

■

There is a related result for uncalibrated coordinates.

Theorem 8.2.2 Define the fundamental matrix as

$$F = K_c^{-\top} E K_c^{-1}, \quad (8.16)$$

where K_c is the camera calibration matrix. Then every set of matching pixels $(\bar{\mathbf{m}}_{f/a}^a, \bar{\mathbf{m}}_{f/b}^b)$ in homogeneous (uncalibrated) coordinates satisfies

$$\bar{\mathbf{m}}_{f/b}^{b\top} F \bar{\mathbf{m}}_{f/a}^a = 0. \quad (8.17)$$

Proof: Follows directly from Equation (8.15) and the fact that

$$\bar{\boldsymbol{\epsilon}} = K_c^{-1} \bar{\mathbf{m}}.$$

The essential matrix invokes several interesting geometrical relationships. For example, consider the relationship

$$\begin{aligned} \boldsymbol{\ell}_a &= E^\top \bar{\boldsymbol{\epsilon}}^b \\ &= R_a^{b\top} \left[\mathbf{n}_{a/b}^b \right]_\times^\top \bar{\boldsymbol{\epsilon}}^b \\ &= R_b^a (\bar{\boldsymbol{\epsilon}}^b \times \mathbf{n}_{a/b}^b), \end{aligned}$$

from which it is clear that $\boldsymbol{\ell}_a$ is a vector (expressed in frame a) that is perpendicular to both $\bar{\boldsymbol{\epsilon}}^b$ and $\mathbf{n}_{a/b}^b$. Therefore $\boldsymbol{\ell}_a$ is perpendicular to the epipolar plane containing $\mathbf{p}_{a/b}$, $\mathbf{p}_{f/a}$ and $\mathbf{p}_{f/b}$. Since $\boldsymbol{\ell}_a$ is in homogenous coordinates, it defines the line in image a where the epipolar plane intersects image a . Similarly, consider the relationship

$$\begin{aligned} \boldsymbol{\ell}_b &= E \bar{\boldsymbol{\epsilon}}^a \\ &= \left[\mathbf{n}_{a/b}^b \right]_\times R_a^b \bar{\boldsymbol{\epsilon}}^a \\ &= R_a^b R_a^{b\top} \left[\mathbf{n}_{a/b}^b \right]_\times R_a^b \bar{\boldsymbol{\epsilon}}^a \\ &= R_a^b \left[R_a^{b\top} \mathbf{n}_{a/b}^b \right]_\times \bar{\boldsymbol{\epsilon}}^a \\ &= R_a^b \left[\mathbf{n}_{a/b}^a \right]_\times \bar{\boldsymbol{\epsilon}}^a \\ &= R_a^b (\mathbf{n}_{a/b}^a \times \bar{\boldsymbol{\epsilon}}^a), \end{aligned}$$

from which it is clear that $\boldsymbol{\ell}_b$ is a vector (expressed in frame b) that is perpendicular to both $\bar{\boldsymbol{\epsilon}}^a$ and $\mathbf{n}_{a/b}^a$. Therefore $\boldsymbol{\ell}_b$ is also perpendicular to the epipolar plane containing $\mathbf{p}_{a/b}$, $\mathbf{p}_{f/a}$ and $\mathbf{p}_{f/b}$. Since $\boldsymbol{\ell}_b$ is in homogenous coordinates, it defines the line in image b where the epipolar plane intersects image b . The lines $\boldsymbol{\ell}_a$ and $\boldsymbol{\ell}_b$ are called the epipolar lines.

As shown in Figure ??, the vector $\mathbf{p}_{a/b}^b$ intersects the two image planes (typically outside the field-of-view) at the points $\bar{\mathbf{e}}_a$ and $\bar{\mathbf{e}}_b$ which are called the epipoles. Since any line in the image can be defined by the cross product of two points on the line, the epipoles are defined by the relationship

$$\begin{aligned} \boldsymbol{\ell}_a &= \bar{\boldsymbol{\epsilon}}^a \times \bar{\mathbf{e}}^a \\ \boldsymbol{\ell}_b &= \bar{\boldsymbol{\epsilon}}^b \times \bar{\mathbf{e}}^b. \end{aligned}$$

RWB: Add: (1) How to compute E from matching image points.
(2) Given E , how to find R and \mathbf{n} . (3) Other interesting facts about epipolar geometry.

8.2.1 Point Triangulation

We desire to recover point locations in 3-space given two camera views of the same point. In the ideal case where there is no noise or discretization in pixel coordinates of the imaged point, this can be done using a simple triangulation method. However, consider the case shown in Fig. ???. If there is any noise in homogeneous image coordinates in either of the images, the rays from camera centers through the noisy image coordinates will never intersect, and the epipolar constraint will not be satisfied. A simple way to overcome this issue is to pick the midpoint of the line segment of minimum length that connects the two rays as the 3D point. However, this method of error minimization is not projective-invariant². A more suitable method is presented in Section 8.2.3.

2

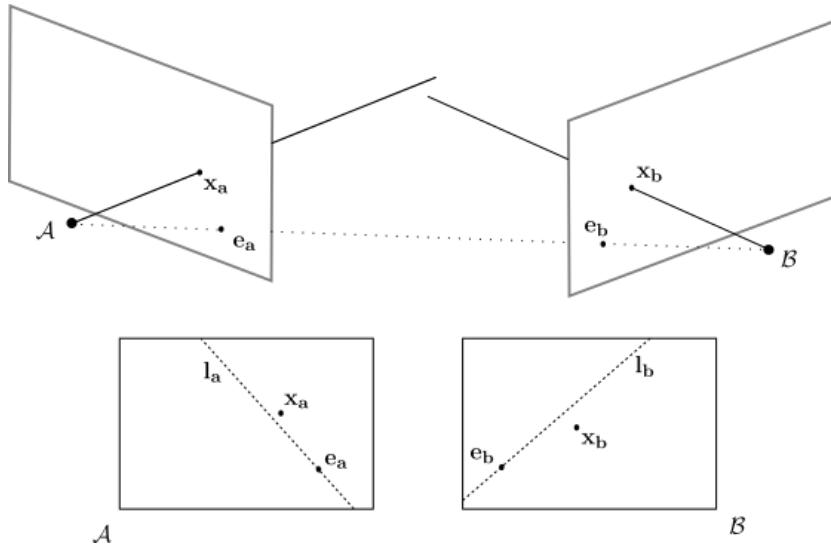


Figure 8.3: Effect of noise on epipolar geometry.

8.2.2 Naive Point Triangulation

In this section we will describe a simple method for point triangulation when the relative pose $(R_a^b, \mathbf{p}_{a/b}^b)$ are known.

Theorem 8.2.3 Suppose that the relative pose $(R_a^b, \mathbf{p}_{a/b}^b)$ is known, and that the normalized (calibrated) pixel coordinates of a feature are

known, i.e.,

$$\begin{aligned}\bar{\epsilon}_{f/a}^a &= \lambda_a \mathbf{p}_{f/a}^a \\ \bar{\epsilon}_{f/b}^b &= \lambda_b \mathbf{p}_{f/b}^b.\end{aligned}$$

Then the unknown scales are given by

$$\begin{aligned}\lambda_a &= -\left\| \mathbf{p}_{a/b}^b \right\| \frac{(R_a^b \bar{\epsilon}_{f/a}^a)^\top \left[\bar{\epsilon}_{f/b}^b \right]_\times E R_a^{b\top} \bar{\epsilon}_{f/b}^b}{\left\| \left[\bar{\epsilon}_{f/b}^b \right]_\times R_a^b \bar{\epsilon}_{f/a}^a \right\|^2}, \\ \lambda_b &= \left\| \mathbf{p}_{a/b}^b \right\| \frac{\bar{\epsilon}_{f/b}^{b\top} \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times E \bar{\epsilon}_{f/a}^a}{\left\| \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \bar{\epsilon}_{f/b}^b \right\|^2},\end{aligned}$$

where $E = \begin{bmatrix} \mathbf{p}_{a/b}^b \\ \left\| \mathbf{p}_{a/b}^b \right\| \end{bmatrix}_\times R_a^b$ is the essential matrix.

Proof: From Equation (8.10) we have

$$\mathbf{p}_{f/b}^b = R_a^b \mathbf{p}_{f/a}^a + \mathbf{p}_{a/b}^b.$$

In terms of calibrated pixel coordinates we have

$$\lambda_b \bar{\epsilon}_{f/b}^b = \lambda_a R_a^b \bar{\epsilon}_{f/a}^a + \mathbf{p}_{a/b}^b,$$

where λ_a and λ_b are unknown scales. Multiplying both sides by $\left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times$ gives

$$\lambda_b \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \bar{\epsilon}_{f/b}^b = \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \mathbf{p}_{a/b}^b.$$

Multiplying both sides by $(\left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \bar{\epsilon}_{f/b}^b)^\top$, and solving for the scale λ_b gives

$$\lambda_b = \frac{(\left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \bar{\epsilon}_{f/b}^b)^\top \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \mathbf{p}_{a/b}^b}{\left\| \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \bar{\epsilon}_{f/b}^b \right\|^2}.$$

The numerator can be simplified as

$$\begin{aligned}& (\left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \bar{\epsilon}_{f/b}^b)^\top \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \mathbf{p}_{a/b}^b \\ &= -\bar{\epsilon}_{f/b}^{b\top} \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \mathbf{p}_{a/b}^b \\ &= -\bar{\epsilon}_{f/b}^{b\top} \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times \left[\mathbf{p}_{a/b}^b \right]_\times R_a^b \bar{\epsilon}_{f/a}^a \\ &= \left\| \mathbf{p}_{a/b}^b \right\| \bar{\epsilon}_{f/b}^{b\top} \left[R_a^b \bar{\epsilon}_{f/a}^a \right]_\times E \bar{\epsilon}_{f/a}^a.\end{aligned}$$

The formula for λ_a is derived similarly. ■

RWB: Revise stuff below to fit notation.

Let $\mathbf{p}_{f/a}^a$ and $\mathbf{p}_{f/b}^b$ be the position vector of a feature in frames a and b respectively. Then

$$\mathbf{p}_{f/b}^b = R_a^b \mathbf{p}_{f/a}^a + \mathbf{p}_{a/b}^b.$$

In terms of calibrated pixel coordinates we have

$$\lambda_b \epsilon_{f/b}^b = \lambda_a R_a^b \epsilon_{f/a}^a + \gamma \mathbf{n}_{a/b}^b,$$

where λ_a , λ_b , and γ are unknown scales. Multiplying both sides by $\begin{bmatrix} \epsilon_{f/b}^b \\ \epsilon_{f/b}^b \end{bmatrix}_\times$ gives

$$\begin{aligned} & \lambda_a \begin{bmatrix} \epsilon_{f/b}^b \\ \epsilon_{f/b}^b \end{bmatrix}_\times R_a^b \epsilon_{f/a}^a + \gamma \begin{bmatrix} \epsilon_{f/b}^b \\ \epsilon_{f/b}^b \end{bmatrix}_\times \mathbf{n}_{a/b}^b = 0 \\ \implies & \left(\begin{bmatrix} \epsilon_{f/b}^b \\ \epsilon_{f/b}^b \end{bmatrix}_\times R_a^b \epsilon_{f/a}^a \quad \begin{bmatrix} \epsilon_{f/b}^b \\ \epsilon_{f/b}^b \end{bmatrix}_\times \mathbf{n}_{a/b}^b \right) \begin{pmatrix} \lambda_a \\ \gamma \end{pmatrix} = 0. \end{aligned}$$

Suppose now that there are n features tracked between frames a and b , and let $\epsilon_{f_i/b}^b$ and $\epsilon_{f_i/a}^a$ be the corresponding feature locations in normalized calibrated pixels. Then the depth factors $\lambda_{i/a}$ relative to frame a and the depth factor γ satisfy

$$M\boldsymbol{\lambda} \triangleq \begin{pmatrix} \begin{bmatrix} \epsilon_{f_1/b}^b \\ \epsilon_{f_1/b}^b \end{bmatrix}_\times R_a^b \epsilon_{f_1/a}^a & \mathbf{0} & \cdots & \mathbf{0} & \begin{bmatrix} \epsilon_{f_1/b}^b \\ \epsilon_{f_1/b}^b \end{bmatrix}_\times \mathbf{n}_{a/b}^b \\ \mathbf{0} & \begin{bmatrix} \epsilon_{f_2/b}^b \\ \epsilon_{f_2/b}^b \end{bmatrix}_\times R_a^b \epsilon_{f_2/a}^a & \cdots & \mathbf{0} & \begin{bmatrix} \epsilon_{f_2/b}^b \\ \epsilon_{f_2/b}^b \end{bmatrix}_\times \mathbf{n}_{a/b}^b \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \begin{bmatrix} \epsilon_{f_n/b}^b \\ \epsilon_{f_n/b}^b \end{bmatrix}_\times R_a^b \epsilon_{f_n/a}^a & \begin{bmatrix} \epsilon_{f_n/b}^b \\ \epsilon_{f_n/b}^b \end{bmatrix}_\times \mathbf{n}_{a/b}^b \end{pmatrix} \begin{pmatrix} \lambda_{1/a} \\ \lambda_{2/a} \\ \vdots \\ \lambda_{n/a} \\ \gamma \end{pmatrix} = \mathbf{0}.$$

Therefore, the unknown scale factors $\boldsymbol{\lambda}$ are in the null space of the $3n \times (n+1)$ matrix M . Let

$$M = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^\top \\ V_2^\top \end{pmatrix}$$

be the singular value decomposition of M , then $\boldsymbol{\lambda} = V_2 \in \mathbb{R}^{(n+1) \times 1}$. Note that any scaled version of V_2 is also a solution. Therefore, to use this method we require one of the parameters $\lambda_{1/a}, \dots, \lambda_{n/a}, \gamma$ to be known. If for example, the translation γ is known via, e.g., IMU integration, then V_2 is scaled so that its last element equals γ , and the rest of the scale factors are recovered.

Theorem 8.2.4 *The matrix M defined above has a null space with dimension one if and only iff the pixel pairs $(\epsilon_{f_i/b}^b, \epsilon_{f_i/a}^a)$ satisfy the epipolar constraint, and the originating point \mathbf{p}_f does not lie along the epipolar baseline.*

Proof: If $(\epsilon_{f_i/b}^b, \epsilon_{f_i/a}^a)$ do not satisfy the epipolar constraint, i.e.

$$\epsilon_{f_i/b}^{b\top} E_a^b \epsilon_{f_i/a}^a = \mu,$$

where $\mu \neq 0$. It is clear that the first n columns of M are linearly independent. Therefore M is at least rank n , and will be rank $n+1$ iff the last column can be written as linear combination of the first n columns. Considering only rows associated with the i^{th} pixel pair, there must exist a non-zero constant c such that

$$c \begin{bmatrix} \epsilon_{f_i/b}^b \\ \vdots \end{bmatrix}_\times R_a^b \epsilon_{f_i/a}^a = \begin{bmatrix} \epsilon_{f_i/b}^b \\ \vdots \end{bmatrix}_\times \mathbf{n}_{a/b}^b.$$

Multiplying both sides by $\mathbf{n}_{a/b}^{b\top}$ gives

$$\begin{aligned} c \mathbf{n}_{a/b}^{b\top} \begin{bmatrix} \epsilon_{f_i/b}^b \\ \vdots \end{bmatrix}_\times R_a^b \epsilon_{f_i/a}^a &= \mathbf{n}_{a/b}^{b\top} \begin{bmatrix} \epsilon_{f_i/b}^b \\ \vdots \end{bmatrix}_\times \mathbf{n}_{a/b}^b \\ \implies -c \epsilon_{f_i/b}^{b\top} \begin{bmatrix} \mathbf{n}_{a/b}^b \\ \vdots \end{bmatrix}_\times R_a^b \epsilon_{f_i/a}^a &= 0 \\ -c \epsilon_{f_i/b}^{b\top} E_a^b \epsilon_{f_i/a}^a &= 0 \\ -c \mu &= 0. \end{aligned}$$

Therefore, if the epipolar constraint is not satisfied for the i^{th} , then c must be 0 to satisfy this relationship. If $c = 0$, then we must have that

$$\begin{bmatrix} \epsilon_{f_i/b}^b \\ \vdots \end{bmatrix}_\times \mathbf{n}_{a/b}^b = 0$$

which is true only when $\epsilon_{f_i/b}^b$ is imaged at the epipole in frame b . ■

8.2.3 Point Triangulation Using Maximum Likelihood Estimation

RWB: Revise to fit notation.

The solution obtained in section 8.2.2 assumed that the normalized image points $\bar{\mathbf{x}}_a$ and $\bar{\mathbf{x}}_b$ satisfy the epipolar constraint. However, image coordinates are likely to be subject to noise caused by imperfect feature matching, discretization, imperfect camera models, inaccurate distortion correction, etc. This noise causes the epipolar constraint to not be satisfied.

We seek to obtain a new point correspondence $\hat{\mathbf{x}}_a, \hat{\mathbf{x}}_b$ that minimizes the distance between measured and new points such that the epipolar constraint is satisfied, as shown in Fig. ???. This results in the problem

$$\min_{\hat{\mathbf{x}}_a, \hat{\mathbf{x}}_b} d(\bar{\mathbf{x}}_a, \hat{\mathbf{x}}_a)^2 + d(\bar{\mathbf{x}}_b, \hat{\mathbf{x}}_b)^2 \quad \text{s.t. } \hat{\mathbf{x}}_b^\top E_{ab} \hat{\mathbf{x}}_a = 0, \quad (8.18)$$

where $d(*, *)$ is the 2-norm between the two points.

This problem can be solved using any iterative minimization strategy. However, section 12.5 of ³ presents an analytical solution that

involves finding the roots of a six-degree polynomial. Additionally, section 12.4 presents the solution to the first order approximation of Eq. 8.18 that can be used so long as the error in image coordinates is around 1 pixel.

After a solution to Eq. 8.18 is obtained, the method presented in section 8.2.2 can be used to obtain the triangulation for \mathbf{X}_a .

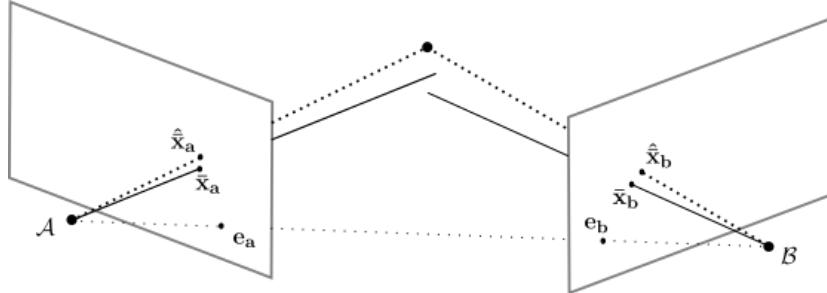


Figure 8.4: Maximum likelihood estimate of image correspondence that satisfies the epipolar constraint.

8.3 Simple point reconstruction methods

The point reconstruction problem is depicted graphically in Figure ??, where the airc

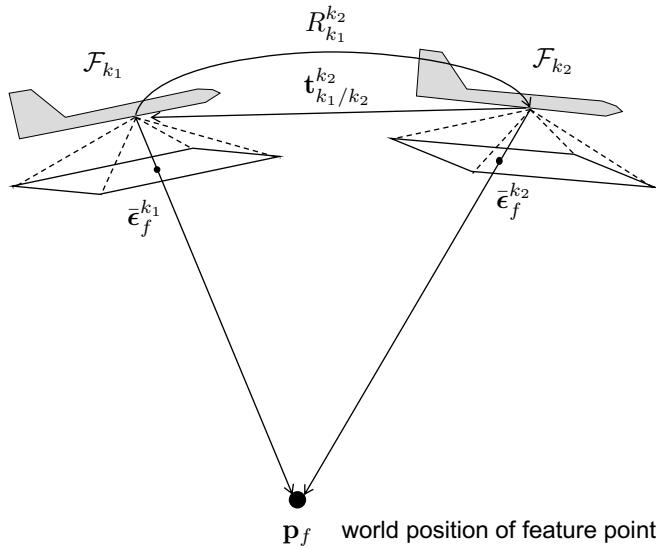


Figure 8.5: The point reconstruction problem.

8.4 Visual Inertial Odometry

8.4.1 Error State EKF

Sometimes called indirect Kalman filter, or multiplicative Kalman filter.

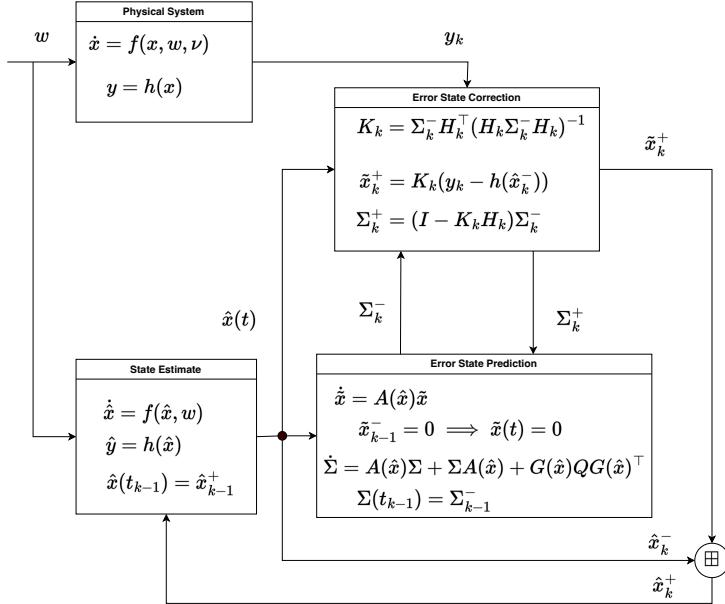


Figure 8.6: Block diagram for the error state extended Kalman filter (ESEKF).

The algorithm is implemented as follows:

Between Measurements

- Propagate the state estimate for $t \in [t_{k-1}, t_k]$:

$$\dot{\hat{x}} = f(\hat{x}, w),$$

with initial conditions $\hat{x}(t_{k-1}) = \hat{x}_{k-1}^+$. The solution at time $t = t_k$ is \hat{x}_k^- .

- The error state is propagated as $\tilde{x}(t) = 0$. ($\dot{\tilde{x}} = A\tilde{x}$ with initial conditions $\tilde{x}(t_{k-1}) = 0$)
- Propagate the error covariance for $t \in [t_{k-1}, t_k]$:

$$\dot{\Sigma} = A(\hat{x}(t), w(t)) \Sigma + \Sigma A(\hat{x}(t), w(t)) + G(\hat{x}(t), w(t)) Q G(\hat{x}(t), w(t))^\top$$

with initial condition $\Sigma(t_{k-1}) = \Sigma_{k-1}^+$. The solution at time $t = t_k$ is Σ_k^- .

At the Measurement of sensor s :

- Compute the Kalman gain:

$$K_s = \Sigma_k^- H_s^\top (\hat{x}_k^-) \left(H_s(\hat{x}_k^-) \Sigma_k^- H_s^\top (\hat{x}_k^-) + R_k \right)^{-1}$$

- Compute the predicted measurement $\hat{y}_s = h_s(\hat{x}_k^-)$
- Update the error state:

$$\tilde{x}_k^+ = K_x (y_s(t_k) - \hat{y}_s)$$

- Update the error covariance:

$$\Sigma_k^+ = (I - K_s H_s(\hat{x}_k^-)) \Sigma_k^-$$

- Update the state estimate:

$$\hat{x}_k^+ = \hat{x}_k^- \boxplus \tilde{x}_k^+$$

8.4.2 Almost-Invariant EKF for VIO

In this section we outline a method for combining visual and IMU measurements to estimate the position of the vehicle as well as the position of features in the environment, assuming no IMU biases. We will assume that the camera moves according to (8.1)–(8.3). We also assume that the gyro measurements are corrupted by unknown biases and zero mean Gaussian noise to get

$$\begin{aligned} \dot{\mathbf{p}}_{b/i}^i &= \mathbf{v}_{b/i}^i \\ \dot{\mathbf{v}}_{b/i}^i &= \mathbf{g}^i \mathbf{e}_3 + R_b^i (\mathbf{a}_{b/i}^b - \mathbf{b}_a - \boldsymbol{\nu}_a), \\ \dot{R}_b^i &= R_b^i [\boldsymbol{\omega}_{b/i}^b - \mathbf{b}_\omega - \boldsymbol{\nu}_\omega]_\times \\ \dot{\mathbf{b}}_a &= 0 \\ \dot{\mathbf{b}}_\omega &= 0 \\ \dot{\ell}_1 &= 0 \\ &\vdots \\ \dot{\ell}_N &= 0, \end{aligned}$$

where $\boldsymbol{\nu}_a \sim \mathcal{N}(0, \Sigma_a)$, $\boldsymbol{\nu}_\omega \sim \mathcal{N}(0, \Sigma_\omega)$, and where we assumed that the body and camera frames are aligned, and where we have assumed a set of N stationary landmarks in the environment $\{\ell_i\}_{i=1}^N$, where $\ell_i \in \mathbb{R}^3$.

In this section we derive an error state extended Kalman filter that uses the state prediction equations

$$\begin{aligned} \dot{\hat{\mathbf{p}}}_{b/i}^i &= \hat{\mathbf{v}}_{b/i}^i \\ \dot{\hat{\mathbf{v}}}_{b/i}^i &= \mathbf{g}^i \mathbf{e}_3 + \hat{R}_b^i (\mathbf{a}_{b/i}^b - \hat{\mathbf{b}}_a), \\ \dot{\hat{R}}_b^i &= \hat{R}_b^i [\boldsymbol{\omega}_{b/i}^b - \hat{\mathbf{b}}_\omega]_\times \\ \dot{\hat{\mathbf{b}}}_a &= 0 \\ \dot{\hat{\mathbf{b}}}_\omega &= 0 \\ \dot{\hat{\ell}}_i &= 0. \end{aligned}$$

Following the development of the Invariant Kalman Filter⁴, we define the errors in the inertial frame as

$$\begin{aligned}\tilde{\mathbf{p}} &= \hat{\mathbf{p}}^i - \hat{R}_b^i R_b^{i\top} \mathbf{p}^i \\ \tilde{\mathbf{v}} &= \hat{\mathbf{v}}^i - \hat{R}_b^i R_b^{i\top} \mathbf{v}^i \\ \tilde{\mathbf{r}} &= \log(\hat{R}_b^i R_b^{i\top})^\vee \\ \tilde{\mathbf{b}}_a &= \hat{\mathbf{b}}_a - \mathbf{b}_a \\ \tilde{\mathbf{b}}_\omega &= \hat{\mathbf{b}}_\omega - \mathbf{b}_\omega \\ \tilde{\boldsymbol{\ell}}_i &= \hat{\boldsymbol{\ell}}_i^i - \hat{R}_b^i R_b^{i\top} \boldsymbol{\ell}_i^i\end{aligned}$$

Differentiating the position errors gives

$$\begin{aligned}\dot{\tilde{\mathbf{p}}} &= \dot{\hat{\mathbf{p}}}^i - \dot{\hat{R}}_b^i R_b^{i\top} \mathbf{p}^i - \hat{R}_b^i \frac{d}{dt}(R_b^{i\top}) \mathbf{p}^i - \hat{R}_b^i R_b^{i\top} \dot{\tilde{\mathbf{p}}}^i \\ &= \dot{\hat{\mathbf{v}}}^i - \hat{R}_b^i [\omega^b - \hat{\mathbf{b}}_\omega]_\times R_b^{i\top} \mathbf{p}^i + \hat{R}_b^i R_b^{i\top} R_b^i [\omega^b - \mathbf{b}_\omega - \boldsymbol{\nu}_\omega]_\times R_b^{i\top} \mathbf{p}^i - \hat{R}_b^i R_b^{i\top} \mathbf{v}^i \\ &= \dot{\hat{\mathbf{v}}}^i - \hat{R}_b^i R^{i\top} \mathbf{v}^i - \hat{R}_b^i [\tilde{\mathbf{b}}_\omega]_\times R_b^{i\top} \mathbf{p}^i + \hat{R}_b^i [\boldsymbol{\nu}_\omega]_\times R_b^{i\top} \mathbf{p}^i \\ &= \dot{\hat{\mathbf{v}}}^i - [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \hat{R}_b^i R^{i\top} (\hat{R}_b^i R^{i\top})^\top (\hat{\mathbf{p}} - \tilde{\mathbf{p}}) + [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \hat{R}_b^i R^{i\top} (\hat{R}_b^i R^{i\top})^\top (\hat{\mathbf{p}} - \tilde{\mathbf{p}}) \\ &= \dot{\hat{\mathbf{v}}}^i - [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \hat{\mathbf{p}} + [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \tilde{\mathbf{p}} + [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \hat{\mathbf{p}} - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \tilde{\mathbf{p}} \\ &\approx \dot{\hat{\mathbf{v}}}^i + [\hat{\mathbf{p}}]_\times \hat{R}_b^i \tilde{\mathbf{b}}_\omega - [\hat{\mathbf{p}}]_\times \hat{R}_b^i \boldsymbol{\nu}_\omega\end{aligned}$$

Differentiating the velocity errors gives

$$\begin{aligned}\dot{\hat{\mathbf{v}}}^i &= \dot{\hat{\mathbf{v}}}^i - \dot{\hat{R}}_b^i R_b^{i\top} \mathbf{v}^i - \dot{\hat{R}}_b^i \frac{d}{dt}(R_b^{i\top}) \mathbf{v}^i - \dot{\hat{R}}_b^i R_b^{i\top} \dot{\tilde{\mathbf{v}}}^i \\ &= (\mathbf{g}^i + \hat{R}_b^i (\mathbf{a}^b + \hat{\mathbf{b}}_a)) - \hat{R}_b^i [\omega^b - \hat{\mathbf{b}}_\omega]_\times R_b^{i\top} \mathbf{v}^i - \hat{R}_b^i R_b^{i\top} R_b^i [\omega^b - \mathbf{b}_\omega - \boldsymbol{\nu}_\omega]_\times R_b^{i\top} \mathbf{v}^i - \hat{R}_b^i R_b^{i\top} (\mathbf{g}^i + R_b^i (\mathbf{a}^b - \mathbf{b}_a - \boldsymbol{\nu}_a)) \\ &= (I - \hat{R}_b^i R_b^{i\top}) \mathbf{g}^i + \hat{R}_b^i \tilde{\mathbf{b}}_a - \hat{R}_b^i [\tilde{\mathbf{b}}_\omega]_\times R_b^{i\top} \mathbf{v}^i - \hat{R}_b^i \boldsymbol{\nu}_a + \hat{R}_b^i [\boldsymbol{\nu}_\omega]_\times R_b^{i\top} \mathbf{v}^i \\ &\approx (I - \exp([\tilde{\mathbf{r}}]_\times)) \mathbf{g}^i + \hat{R}_b^i \tilde{\mathbf{b}}_a - [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \hat{R}_b^i R_b^{i\top} (\hat{R}_b^i R_b^{i\top})^\top (\hat{\mathbf{v}}^i - \tilde{\mathbf{v}}) - \hat{R}_b^i \boldsymbol{\nu}_a - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \hat{R}_b^i R_b^{i\top} (\hat{R}_b^i R_b^{i\top})^\top (\hat{\mathbf{v}}^i - \tilde{\mathbf{v}}) \\ &\approx -[\tilde{\mathbf{r}}]_\times \mathbf{g}^i + \hat{R}_b^i \tilde{\mathbf{b}}_a - [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \hat{\mathbf{v}}^i - \hat{R}_b^i \boldsymbol{\nu}_a + [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \hat{\mathbf{v}}^i \\ &= [\mathbf{g}^i]_\times \tilde{\mathbf{r}} + \hat{R}_b^i \tilde{\mathbf{b}}_a + [\hat{\mathbf{v}}^i]_\times \hat{R}_b^i \tilde{\mathbf{b}}_\omega - \hat{R}_b^i \boldsymbol{\nu}_a - [\hat{\mathbf{v}}^i]_\times \hat{R}_b^i \boldsymbol{\nu}_\omega.\end{aligned}$$

For the evolution of the attitude error we use the fact that

$$\frac{d}{dt} \exp([\tilde{\mathbf{r}}]_\times) \approx \frac{d}{dt} (I + [\tilde{\mathbf{r}}]_\times) = [\dot{\tilde{\mathbf{r}}}]_\times,$$

to get that

$$\begin{aligned}
\dot{\tilde{\mathbf{r}}} &= \left(\hat{R}_b^i R_b^{i\top} + \hat{R}_b^i \frac{d}{dt} (R_b^{i\top}) \right)^\vee \\
&= \left(\hat{R}_b^i [\omega^b - \hat{\mathbf{b}}_\omega]_\times R_b^{i\top} - \hat{R}_b^i R_b^{i\top} R_b^i [\omega^b - \mathbf{b}_\omega - \boldsymbol{\nu}_\omega]_\times R_b^{i\top} \right)^\vee \\
&= - \left(\hat{R}_b^i [\hat{\mathbf{b}}_\omega]_\times R_b^{i\top} + \hat{R}_b^i [\mathbf{b}_\omega]_\times R_b^{i\top} + \hat{R}_b^i [\boldsymbol{\nu}_\omega]_\times R_b^{i\top} \right)^\vee \\
&= - \left([\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \hat{R}_b^i R_b^{i\top} - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \hat{R}_b^i R_b^{i\top} \right)^\vee \\
&\approx - \left([\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times (I + [\tilde{\mathbf{r}}]_\times) - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times (I + [\tilde{\mathbf{r}}]_\times) \right)^\vee \\
&\approx - \left([\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \right)^\vee \\
&= \hat{R}_b^i \tilde{\mathbf{b}}_\omega - \hat{R}_b^i \boldsymbol{\nu}_\omega
\end{aligned}$$

The evolution of a landmark error is given by

$$\begin{aligned}
\dot{\tilde{\ell}} &= \dot{\hat{\ell}}^i - \hat{R}_b^i R_b^{i\top} \ell^i - \hat{R}_b^i \frac{d}{dt} (R_b^{i\top}) \ell^i - \hat{R}_b^i R_b^{i\top} \dot{\ell}^i \\
&= - \hat{R}_b^i [\omega^b - \hat{\mathbf{b}}_\omega]_\times R_b^{i\top} \ell^i + \hat{R}_b^i R_b^{i\top} R_b^i [\omega^b - \mathbf{b}_\omega - \boldsymbol{\nu}_\omega]_\times R_b^{i\top} \ell^i \\
&= \hat{R}_b^i [\tilde{\mathbf{b}}_\omega]_\times R_b^{i\top} \ell^i - \hat{R}_b^i [\boldsymbol{\nu}_\omega]_\times R_b^{i\top} \ell^i \\
&= [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \hat{R}_b^i R_b^{i\top} (\hat{R}_b^i R_b^{i\top})^\top (\dot{\ell}^i - \tilde{\ell}) - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \hat{R}_b^i R_b^{i\top} (\hat{R}_b^i R_b^{i\top})^\top (\dot{\ell}^i - \tilde{\ell}) \\
&\approx [\hat{R}_b^i \tilde{\mathbf{b}}_\omega]_\times \dot{\ell}^i - [\hat{R}_b^i \boldsymbol{\nu}_\omega]_\times \dot{\ell}^i \\
&\approx - [\dot{\ell}^i]_\times \hat{R}_b^i \tilde{\mathbf{b}}_\omega + [\dot{\ell}^i]_\times \hat{R}_b^i \boldsymbol{\nu}_\omega.
\end{aligned}$$

Therefore, the error state satisfies

$$\begin{pmatrix} \dot{\tilde{\mathbf{p}}}^i \\ \dot{\tilde{\mathbf{v}}}^i \\ \dot{\tilde{\mathbf{r}}} \\ \dot{\tilde{\mathbf{b}}}_a \\ \dot{\tilde{\mathbf{b}}}_\omega \\ \dot{\tilde{\ell}}_1 \\ \vdots \\ \dot{\tilde{\ell}}_N \end{pmatrix} = \begin{pmatrix} 0 & I & 0 & 0 & [\hat{\mathbf{p}}]_\times \hat{R}_b^i & 0 & \dots & 0 \\ 0 & 0 & -[\mathbf{g}^i]_\times & \hat{R}_b^i & [\hat{\mathbf{v}}^i]_\times \hat{R}_b^i & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \hat{R}_b^i & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & -[\dot{\ell}_1^i]_\times \hat{R}_b^i & 0 & \dots & 0 \\ \vdots & & & & & \dots & \vdots & \\ 0 & 0 & 0 & 0 & -[\dot{\ell}_N^i]_\times \hat{R}_b^i & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{p}}^i \\ \tilde{\mathbf{v}}^i \\ \tilde{\mathbf{r}} \\ \tilde{\mathbf{b}}_a \\ \tilde{\mathbf{b}}_\omega \\ \tilde{\ell}_1 \\ \vdots \\ \tilde{\ell}_N \end{pmatrix} + \begin{pmatrix} 0 & -[\hat{\mathbf{p}}]_\times \hat{R}_b^i \\ -\hat{R}_b^i & -[\hat{\mathbf{v}}^i]_\times \hat{R}_b^i \\ 0 & -\hat{R}_b^i \\ 0 & 0 \\ 0 & 0 \\ 0 & [\dot{\ell}_1^i]_\times \hat{R}_b^i \\ \vdots & \vdots \\ 0 & [\dot{\ell}_N^i]_\times \hat{R}_b^i \end{pmatrix} \begin{pmatrix} \boldsymbol{\nu}_a \\ \boldsymbol{\nu}_\omega \end{pmatrix}.$$

These equations can be summarized as

$$\dot{\tilde{x}} = A(\hat{x}(t)) \tilde{x} + G(\hat{x}(t)) \boldsymbol{\nu},$$

where $A(\cdot)$ and $G(\cdot)$ are appropriately defined.

For the measurement update, we assume that at each sample we observe a subset of the landmarks through the camera. Therefore, if

the j^{th} landmark is observed, then the associated measurement model is

$$\epsilon_j = h(x) \triangleq \pi \left(R_b^{i\top} (\ell_j^i - \mathbf{p}^i) \right),$$

where $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the projection operator from an inertial vector relative to the camera frame to calibrated pixel locations via

$$\pi((x, y, z)^\top) \triangleq \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}.$$

Expressing the measurement in error states gives

$$\begin{aligned} \epsilon_j &= \pi \left(R_b^{i\top} (\ell^i - \mathbf{p}^i) \right) \\ &= \pi \left(R_b^{i\top} \left((\hat{R}_b^i R_b^{i\top})^\top (\hat{\ell}_j^i - \tilde{\ell}_j) - (\hat{R}_b^i R_b^{i\top})^\top (\hat{\mathbf{p}}^i - \tilde{\mathbf{p}}^i) \right) \right) \\ &= \pi \left(\hat{R}_b^{i\top} (\hat{\ell}_j^i - \hat{\mathbf{p}}^i) + \hat{R}_b^{i\top} (\tilde{\mathbf{p}}^i - \tilde{\ell}_j^i) \right) \\ &\approx \pi(\hat{\mathbf{z}}_j) + \frac{\partial \pi}{\partial \mathbf{z}} \Big|_{\mathbf{z}=\hat{\mathbf{z}}_j} \hat{R}_b^{i\top} (\tilde{\mathbf{p}}^i - \tilde{\ell}_j^i), \end{aligned}$$

where $\hat{\mathbf{z}}_j \triangleq \hat{R}_b^{i\top} (\hat{\ell}_j^i - \hat{\mathbf{p}}^i)$, and

$$\frac{\partial \pi}{\partial \mathbf{z}} ((p_x, p_y, p_z)^\top) = \begin{pmatrix} \frac{1}{p_z} & 0 & -\frac{p_x}{p_z^2} \\ 0 & \frac{1}{p_y} & -\frac{p_y}{p_z^2} \end{pmatrix}.$$

Defining the matrix

$$\Pi_j \triangleq \frac{\partial \pi}{\partial \mathbf{z}} \Big|_{\mathbf{z}=\hat{\mathbf{z}}_j},$$

and letting $\tilde{\epsilon}_j = \hat{\epsilon}_j - \epsilon_j$, the output equation can be expressed as

$$\tilde{\epsilon}_j = \begin{pmatrix} -\Pi_j \hat{R}_b^{i\top} & 0 & 0 & 0 & 0 & \dots & \Pi_j \hat{R}_b^{i\top} & \dots \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}}^i \\ \hat{\mathbf{v}}^i \\ \tilde{\mathbf{r}}_r \\ \hat{\mathbf{b}}_a \\ \tilde{\mathbf{b}}_\omega \\ \vdots \\ \tilde{\ell}_j \\ \vdots \end{pmatrix}.$$

8.5 *Inverse Depth Parametrization for Monocular SLAM, TRO,
2008*

Inverse Depth Parametrization for Monocular SLAM

Javier Civera, Andrew J. Davison, and J. M. Martínez Montiel

Abstract—We present a new parametrization for point features within monocular simultaneous localization and mapping (SLAM) that permits efficient and accurate representation of uncertainty during undelayed initialization and beyond, all within the standard extended Kalman filter (EKF). The key concept is direct parametrization of the inverse depth of features relative to the camera locations from which they were first viewed, which produces measurement equations with a high degree of

Manuscript received February 27, 2007; revised September 28, 2007. First published October 3, 2008; current version published October 31, 2008. This work was supported in part by the Spanish Grant PR2007-0427, Grant DPI2006-13578, and Grant DGA(CONSI+D)-CAI IT12-06, in part by the Engineering and Physical Sciences Research Council (EPSRC) Grant GR/T24685, in part by the Royal Society International Joint Project grant between the University of Oxford, University of Zaragoza, and Imperial College London, and in part by the Robotics Advancement through Web-Publishing of Sensorial and Elaborated Extensive Datasets (RAWSEEDS) under Grant FP6-IST-045144. The work of A. J. Davison was supported the EPSRC Advanced Research Fellowship Grant. This paper was recommended for publication by Associate Editor P. Rives and Editor L. Parker upon evaluation of the reviewers' comments.

J. Civera and J. M. Martínez Montiel are with the Departamento de Informática, University of Zaragoza, 50018 Zaragoza, Spain (e-mail: josemari@unizar.es; jcivera@unizar.es).

A. J. Davison is with the Department of Computing, Imperial College London, SW7 2AZ London, U.K. (e-mail: ajd@doc.ic.ac.uk).

This paper has supplementary downloadable multimedia material available at <http://ieeexplore.ieee.org> provided by the author. This material includes the following video files. *inverseDepth_indoor.avi* (11.7 MB) shows simultaneous localization and mapping, from a hand-held camera observing an indoor scene. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. *Player information XviD MPEG-4, inverseDepth_outdooravi* (12.4 MB) shows real-time simultaneous localization and mapping, from a hand-held camera observing an outdoor scene, including rather distant features. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. The processing is done with a standard laptop. *Player information XviD MPEG-4, inverseDepth_loopClosing.avi* (10.2MB) shows simultaneous localization and mapping, from a hand-held camera observing a loop-closing indoor scene. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. *Player information XviD MPEG-4, inverseDepth_loopClosing_ID_to_XYZ_conversion.avi* (10.1 MB) shows simultaneous localization and mapping, from a hand-held camera observing the same loop-closing indoor sequence as in *inverseDepth loopClosing.avi*, but switching from inverse depth to XYZ parameterization when necessary. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. *Player information XviD MPEG-4, inverseDepth_indoorRawImages.tar.gz* (44 MB) shows indoor sequences raw images in .pgm format. Camera calibration in an ASCII file. *inverseDepth_outdoorRawImages.tar.gz* (29 MB) shows outdoor sequence raw images in .pgm format. Camera calibration in an ASCII file. *inverseDepth_loopClosingRawImages.tar.gz* (33 MB) shows loop-closing sequence raw images in .pgm format. Camera calibration in an ASCII file. Contact information jcivera@unizar.es; josemari@unizar.es.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TRO.2008.2003276

linearity. Importantly, our parametrization can cope with features over a huge range of depths, even those that are so far from the camera that they present little parallax during motion—maintaining sufficient representative uncertainty that these points retain the opportunity to “come in” smoothly from infinity if the camera makes larger movements. Feature initialization is undelayed in the sense that even distant features are immediately used to improve camera motion estimates, acting initially as bearing references but not permanently labeled as such. The inverse depth parametrization remains well behaved for features at all stages of SLAM processing, but has the drawback in computational terms that each point is represented by a 6-D state vector as opposed to the standard three of a Euclidean XYZ representation. We show that once the depth estimate of a feature is sufficiently accurate, its representation can safely be converted to the Euclidean XYZ form, and propose a linearity index that allows automatic detection and conversion to maintain maximum efficiency—only low parallax features need be maintained in inverse depth form for long periods. We present a real-time implementation at 30 Hz, where the parametrization is validated in a fully automatic 3-D SLAM system featuring a hand-held single camera with no additional sensing. Experiments show robust operation in challenging indoor and outdoor environments with a very large ranges of scene depth, varied motion, and also real time 360° loop closing.

Index Terms—Monocular simultaneous localization and mapping (SLAM), real-time vision.

I. INTRODUCTION

A MONOCULAR camera is a projective sensor that measures the bearing of image features. Given an image sequence of a rigid 3-D scene taken from a moving camera, it is now well known that it is possible to compute both a scene structure and a camera motion up to a scale factor. To infer the 3-D position of each feature, the moving camera must observe it repeatedly each time, capturing a ray of light from the feature to its optic center. The measured angle between the captured rays from different viewpoints is the feature's *parallax*—this is what allows its depth to be estimated.

In offline “structure from motion (SFM)” solutions from the computer vision literature (e.g., [11] and [23]), motion and structure are estimated from an image sequence by first applying a robust feature matching between pairs or other short overlapping sets of images to estimate relative motion. An optimization procedure then iteratively refines global camera location and scene feature position estimates such that features project as closely as possible to their measured image positions (bundle adjustment). Recently, work in the spirit of these methods, but with “sliding window” processing and refinement rather than global optimization, has produced impressive real-time “visual odometry” results when applied to stereo sequences in [21] and for monocular sequences in [20].

An alternative approach to achieving real-time motion and structure estimation are online visual simultaneous localization and mapping (SLAM) approaches that use a probabilistic

filtering approach to sequentially update estimates of the positions of features (the map) and the current location of the camera. These SLAM methods have different strengths and weaknesses to visual odometry, being able to build consistent and drift-free global maps, but with a bounded number of mapped features. The core single extended Kalman filter (EKF) SLAM technique, previously proven in multisensor robotic applications, was first applied successfully to real-time monocular camera tracking by Davison *et al.* [8], [9] in a system that built sparse room-sized maps at 30 Hz.

A significant limitation of Davison's and similar approaches, however, was that they could only make use of features that were close to the camera relative to its distance of translation, and therefore exhibited significant parallax during motion. The problem was in initializing uncertain depth estimates for distant features: in the straightforward Euclidean XYZ feature parametrization adopted, position uncertainties for low parallax features are not well represented by the Gaussian distributions implicit in the EKF. The depth coordinate of such features has a probability density that rises sharply at a well-defined minimum depth to a peak, but then, tails off very slowly toward infinity—from low parallax measurements, it is very difficult to tell whether a feature has a depth of 10 units rather than 100, 1000, or more. For the rest of the paper, we refer to Euclidean XYZ parametrization simply as *XYZ*.

There have been several recent methods proposed for coping with this problem, relying on generally undesirable special treatment of newly initialized features. In this paper, we describe a new feature parametrization that is able to smoothly cope with initialization of features at all depths—even up to “infinity”—within the standard EKF framework. The key concept is direct parametrization of inverse depth relative to the camera position from which a feature was first observed.

A. Delayed and Undelayed Initialization

The most obvious approach to coping with feature initialization within a monocular SLAM system is to treat newly detected features separately from the main map, accumulating information in a special processing over several frames to reduce depth uncertainty before insertion into the full filter with a standard *XYZ* representation. Such *delayed initialization* schemes (e.g., [3], [8], and [14]) have the drawback that new features, held outside the main probabilistic state, are not able to contribute to the estimation of the camera position until finally included in the map. Further, features that retain low parallax over many frames (those very far from the camera or close to the motion epipole) are usually rejected completely because they never pass the test for inclusion.

In the delayed approach of Bailey [2], initialization is delayed until the measurement equation is approximately Gaussian and the point can be safely triangulated; here, the problem was posed in 2-D and validated in simulation. A similar approach for a 3-D monocular vision with inertial sensing was proposed in [3]. Davison [8] reacted to the detection of a new feature by inserting a 3-D semiinfinite ray into the main map representing everything about the feature except its depth, and then, used an auxiliary

particle filter to explicitly refine the depth estimate over several frames, taking advantage of all the measurements in a high frame rate sequence, but again with new features held outside the main state vector until inclusion.

More recently, several *undelayed initialization* schemes have been proposed, which still treat new features in a special way but are able to benefit immediately from them to improve camera motion estimates—the key insight being that while features with highly uncertain depths provide little information on camera translation, they are extremely useful as bearing references for orientation estimation. The undelayed method proposed by Kwok and Dissanayake [15] was a multiple hypothesis scheme, initializing features at various depths and pruning those not reobserved in subsequent images.

Sola *et al.* [24], [25] described a more rigorous undelayed approach using a Gaussian sum filter approximated by a federated information sharing method to keep the computational overhead low. An important insight was to spread the Gaussian depth hypotheses along the ray according to inverse depth, achieving much better representational efficiency in this way. This method can perhaps be seen as the direct stepping stone between Davison's particle method and our new inverse depth scheme; a Gaussian sum is a more efficient representation than particles (efficient enough that the separate Gaussians can all be put into the main state vector), but not as efficient as the single Gaussian representation that the inverse depth parametrization allows. Note that neither [15] nor [25] considers features at very large “infinite” depths.

B. Points at Infinity

A major motivation of the approach in this paper is not only the efficient undelayed initialization, but also the desire to cope with features at *all* depths, particularly in outdoor scenes. In SFM, the well-known concept of a point at infinity is a feature that exhibits no parallax during camera motion due to its extreme depth. A star for instance would be observed at the same image location by a camera that translated through many kilometers pointed up at the sky without rotating. Such a feature cannot be used for estimating camera translation but is a perfect bearing reference for estimating rotation. The homogeneous coordinate systems of visual projective geometry used normally in SFM allow explicit representation of points at infinity, and they have proven to play an important role during offline structure and motion estimation.

In a sequential SLAM system, the difficulty is that we do not know in advance which features are infinite and which are not. Montiel and Davison [19] showed that in the special case where *all features are known to be infinite*—in very-large-scale outdoor scenes or when the camera rotates on a tripod—SLAM in pure angular coordinates turns the camera into a real-time visual compass. In the more general case, let us imagine a camera moving through a 3-D scene with observable features at a range of depths. From the estimation point of view, we can think of all features starting at infinity and “coming in” as the camera moves far enough to measure sufficient parallax. For nearby indoor features, only a few centimeters of movement will be

sufficient. Distant features may require many meters or even kilometers of motion before parallax is observed. It is important that these features are not permanently labeled as infinite—a feature that seems to be at infinity should always have the chance to prove its finite depth given enough motion, or there will be the serious risk of systematic errors in the scene map. Our probabilistic SLAM algorithm must be able to represent the uncertainty in depth of seemingly infinite features. Observing no parallax for a feature after 10 units of camera translation does tell us something about its depth—it gives a reliable lower bound, which depends on the amount of motion made by the camera (if the feature had been closer than this, we *would* have observed parallax). This explicit consideration of uncertainty in the locations of points has not been previously required in offline computer vision algorithms, but is very important in a more difficult online case.

C. Inverse Depth Representation

Our contribution is to show that, in fact, there is a unified and straightforward parametrization for feature locations that can handle both initialization and standard tracking of both close and very distant features within the standard EKF framework. An explicit parametrization of the *inverse depth* of a feature along a semiinfinite ray from the position from which it was first viewed allows a Gaussian distribution to cover uncertainty in depth that spans a depth range from nearby to infinity, and permits seamless crossing over to finite depth estimates of features that have been apparently infinite for long periods of time. The unified representation means that our algorithm requires no special initialization process for features. They are simply tracked right from the start, immediately contribute to improved camera estimates, and have their correlations with all other features in the map correctly modeled. Note that our parameterization would be equally compatible with other variants of Gaussian filtering such as sparse information filters.

We introduce a linearity index and use it to analyze and prove the representational capability of the inverse depth parametrization for both low and high parallax features. The only drawback of the inverse depth scheme is the computational issue of increased state vector size since an inverse depth point needs six parameters rather than the three of *XYZ* coding. As a solution to this, we show that our linearity index can also be applied to the *XYZ* parametrization to signal when a feature can be safely switched from inverse depth to *XYZ*; the usage of the inverse depth representation can, in this way, be restricted to low parallax feature cases where the *XYZ* encoding departs from Gaussianity. Note that this “switching,” unlike in delayed initialization methods, is purely to reduce computational load; SLAM accuracy with or without switching is almost the same.

The fact is that the projective nature of a camera means that the image measurement process is nearly linear in this inverse depth coordinate. Inverse depth is a concept used widely in computer vision: it appears in the relation between image disparity and point depth in a stereo vision; it is interpreted as the parallax with respect to the plane at infinity in [12]. Inverse depth is also used to relate the motion field induced by scene points with

the camera velocity in optical flow analysis [13]. In the tracking community, “modified polar coordinates” [1] also exploit the linearity properties of the inverse depth representation in a slightly different, but closely related, problem of a target motion analysis (TMA) from measurements gathered by a bearing-only sensor with known motion.

However, the inverse depth idea has not previously been properly integrated in sequential, probabilistic estimation of motion, and structure. It has been used in EKF-based sequential depth estimation from camera-known motion [16], and in a multibaseline stereo, Okutomi and Kanade [22] used the inverse depth to increase matching robustness for scene symmetries; matching scores coming from multiple stereo pairs with different baselines were accumulated in a common reference coded in inverse depth, this paper focusing on matching robustness and not on probabilistic uncertainty propagation. Chowdhury and Chellappa [5] proposed a sequential EKF process using inverse depth, but this was in some way short of full SLAM in its details. Images are first processed pairwise to obtain a sequence of 3-D motions that are then fused with an individual EKF per feature.

It is our parametrization of inverse depth *relative to the positions from which features were first observed*, which means that a Gaussian representation is uniquely well behaved, this is the reason why a straightforward parametrization of monocular SLAM in the homogeneous coordinates of SFM will not give a good result—that representation only meaningfully represents points that appear to be infinite relative to the coordinate origin. It could be said in projective terms that our method defines separate but correlated projective frames for each feature. Another interesting comparison is between our method, where the representation for each feature includes the camera position from which it was first observed, and smoothing/full SLAM schemes, where all historical sensor pose estimates are maintained in a filter.

Two recently published papers from other authors have developed methods that are quite similar to ours. Trawny and Roumeliotis [26] proposed an undelayed initialization for 2-D monocular SLAM that encodes a map point as the intersection of two projection rays. This representation is overparametrized but allows undelayed initialization and encoding of both close and distant features, the approach validated with simulation results.

Eade and Drummond presented an inverse depth initialization scheme within the context of their FastSLAM-based system for monocular SLAM [10], offering some of the same arguments about advantages in linearity as in our paper. The position of each new partially initialized feature added to the map is parametrized with three coordinates representing its direction and inverse depth relative to the camera pose at the first observation, and estimates of these coordinates are refined within a set of Kalman filters for each particle of the map. Once the inverse depth estimation has collapsed, the feature is converted to a fully initialized standard *XYZ* representation. While retaining the differentiation between partially and fully initialized features, they go further and are able to use measurements of partially initialized features with unknown depth to improve estimates of camera orientation and translation via a special epipolar update step. Their approach certainly appears appropriate within

a FastSLAM implementation. However, it lacks the satisfying unified quality of the parametrization we present in this paper, where the transition from partially to fully initialized need not be explicitly tackled and full use is automatically made of all of the information available in measurements.

This paper offers a comprehensive and extended version of our work previously published as two conference papers [7], [18]. We now present a full real-time implementation of the inverse depth parameterization that can map up to 50–70 features in real time on a standard laptop computer. Experimental validation has shown the important role of an accurate camera calibration to improve the system performance, especially with wide-angle cameras. Our results section includes new real-time experiments, including the key result of vision-only loop closing. Input test image sequences and movies showing the computed solution are included in the paper as multimedia material.

Section II is devoted to defining the state vector, including the camera motion model, *XYZ* point coding, and inverse depth point parametrization. The measurement equation is described in Section III. Section IV presents a discussion about measurement equation linearization errors. Next, feature initialization from a single-feature observation is detailed in Section V. In Section VI, the switch from inverse depth to *XYZ* coding is presented, and in Section VII, we present experimental validations over real-image sequences captured at 30 Hz in large-scale environments, indoors and outdoors, including real-time performance, and a loop closing experiment; links to movies showing the system performance are provided. Finally, Section VIII is devoted to conclusions.

II. STATE VECTOR DEFINITION

A. Camera Motion

A constant angular and linear velocity model is used to model handheld camera motion. The camera state \mathbf{x}_v is composed of pose terms: \mathbf{r}^{WC} camera optical center position and \mathbf{q}^{WC} quaternion defining orientation, and linear and angular velocity \mathbf{v}^W and ω^C relative to world frame W and camera frame C , respectively.

We assume that linear and angular accelerations \mathbf{a}^W and α^C affect the camera, producing at each step, an impulse of linear velocity $\mathbf{V}^W = \mathbf{a}^W \Delta t$ and angular velocity $\Omega^C = \alpha^C \Delta t$, with zero mean and known Gaussian distribution. We currently assume a diagonal covariance matrix for the unknown input linear and angular accelerations.

The state update equation for the camera is

$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}_{k+1}^{WC} \\ \mathbf{q}_{k+1}^{WC} \\ \mathbf{v}_{k+1}^W \\ \omega_{k+1}^C \end{pmatrix} = \begin{pmatrix} \mathbf{r}_k^{WC} + (\mathbf{v}_k^W + \mathbf{V}_k^W) \Delta t \\ \mathbf{q}_k^{WC} \times \mathbf{q}((\omega_k^C + \Omega^C) \Delta t) \\ \mathbf{v}_k^W + \mathbf{V}_k^W \\ \omega_k^C + \Omega^C \end{pmatrix} \quad (1)$$

where $\mathbf{q}((\omega_k^C + \Omega^C) \Delta t)$ is the quaternion defined by the rotation vector $(\omega_k^C + \Omega^C) \Delta t$.

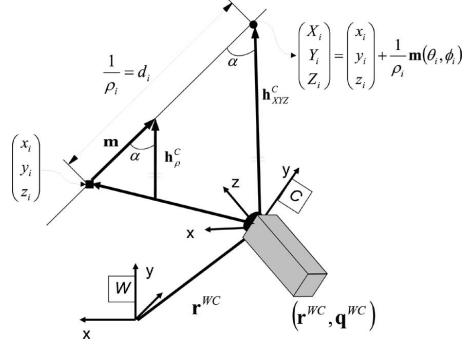


Fig. 1. Feature parametrization and measurement equation.

B. Euclidean XYZ Point Parametrization

The standard representation for scene points i in terms of Euclidean *XYZ* coordinates (see Fig. 1) is

$$\mathbf{x}_i = (X_i \ Y_i \ Z_i)^\top. \quad (2)$$

In this paper, we refer to the Euclidean *XYZ* coding simply as *XYZ* coding.

C. Inverse Depth Point Parametrization

In our new scheme, a scene 3-D point i can be defined by the 6-D state vector:

$$\mathbf{y}_i = (x_i \ y_i \ z_i \ \theta_i \ \phi_i \ \rho_i)^\top \quad (3)$$

which models a 3-D point located at (see Fig. 1)

$$\mathbf{x}_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i} \mathbf{m}(\theta_i, \phi_i) \quad (4)$$

$$\mathbf{m} = (\cos \phi_i \sin \theta_i, -\sin \phi_i, \cos \phi_i \cos \theta_i)^\top. \quad (5)$$

The \mathbf{y}_i vector encodes the *ray from the first camera position from which the feature was observed* by x_i, y_i, z_i , the camera optical center, and θ_i, ϕ_i azimuth and elevation (coded in the world frame) defining unit directional vector $\mathbf{m}(\theta_i, \phi_i)$. The point's depth along the ray d_i is encoded by its inverse $\rho_i = 1/d_i$.

D. Full State Vector

As in standard EKF SLAM, we use a single-joint state vector containing camera pose and feature estimates, with the assumption that the camera moves with respect to a static scene. The whole state vector \mathbf{x} is composed of the camera and all the map features

$$\mathbf{x} = (\mathbf{x}_v^\top, \mathbf{y}_1^\top, \mathbf{y}_2^\top, \dots, \mathbf{y}_n^\top)^\top. \quad (6)$$

III. MEASUREMENT EQUATION

Each observed feature imposes a constraint between the camera location and the corresponding map feature (see Fig. 1).

Observation of a point $\mathbf{y}_i(\mathbf{x}_i)$ defines a ray coded by a directional vector in the camera frame $\mathbf{h}^C = (h_x \ h_y \ h_z)^\top$. For points in *XYZ*

$$\mathbf{h}^C = \mathbf{h}_{XYZ}^C = \mathbf{R}^{CW} \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} - \mathbf{r}^{WC}. \quad (7)$$

For points in inverse depth

$$\mathbf{h}^C = \mathbf{h}_\rho^C = \mathbf{R}^{CW} \left(\rho_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - \mathbf{r}^{WC} \right) + \mathbf{m}(\theta_i, \phi_i) \quad (8)$$

where the directional vector has been normalized using the inverse depth. It is worth noting that (8) can be safely used even for points at infinity, i.e., $\rho_i = 0$.

The camera does not directly observe \mathbf{h}^C but its projection in the image according to the pinhole model. Projection to a normalized retina, and then, camera calibration is applied:

$$\mathbf{h} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 - \frac{f}{d_x} h_x \\ v_0 - \frac{f}{d_y} h_y \end{pmatrix} \quad (9)$$

where u_0, v_0 is the camera's principal point, f is the focal length, and d_x, d_y is the pixel size. Finally, a distortion model has to be applied to deal with real camera lenses. In this paper, we have used the standard two parameters distortion model from photogrammetry [17] (see the Appendix for details).

It is worth noting that the measurement equation in inverse depth has a sensitive dependency on the parallax angle α (see Fig. 1). At low parallax, (8) can be approximated by $\mathbf{h}^C \approx \mathbf{R}^{CW}(\mathbf{m}(\theta_i, \phi_i))$, and hence, the measurement equation only provides information about the camera orientation and the directional vector $\mathbf{m}(\theta_i, \phi_i)$.

IV. MEASUREMENT EQUATION LINEARITY

The more linear the measurement equation is, the better a Kalman filter performs. This section is devoted to presenting an analysis of measurement equation linearity for both *XYZ* and inverse depth codings. These linearity analyses theoretically support the superiority of the inverse depth coding.

A. Linearized Propagation of a Gaussian

Let x be an uncertain variable with Gaussian distribution $x \sim N(\mu_x, \sigma_x^2)$. The transformation of x through the function f is a variable y that can be approximated with Gaussian distribution:

$$y \sim N(\mu_y, \sigma_y^2), \quad \mu_y = f(\mu_x), \quad \sigma_y^2 = \left. \frac{\partial f}{\partial x} \right|_{\mu_x} \sigma_x^2 \left. \frac{\partial f}{\partial x} \right|_{\mu_x}^\top \quad (10)$$

if the function f is linear in an interval around μ_x (Fig. 2). The interval size in which the function has to be linear depends on σ_x ; the bigger σ_x the wider the interval has to be to cover a significant fraction of the random variable x values. In this paper, we fix the linearity interval to the 95% confidence region defined by $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$.

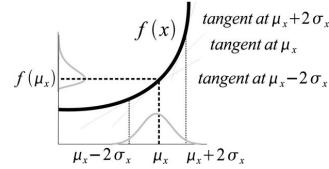


Fig. 2. First derivative variation in $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$ codes the departure from Gaussianity in the propagation of the uncertain variable through a function.

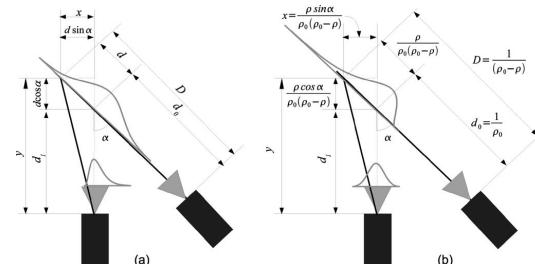


Fig. 3. Uncertainty propagation from the scene point to the image. (a) *XYZ* coding. (b) Inverse depth coding.

If a function is linear in an interval, the first derivative is constant in that interval. To analyze the first derivative variation around the interval $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$, consider the Taylor expansion for the first derivative:

$$\frac{\partial f}{\partial x}(\mu_x + \Delta x) \approx \left. \frac{\partial f}{\partial x} \right|_{\mu_x} + \left. \frac{\partial^2 f}{\partial x^2} \right|_{\mu_x} \Delta x. \quad (11)$$

We propose to compare the value of the derivative at the interval center μ_x with the value at the extremes $\mu_x \pm 2\sigma_x$, where the deviation from linearity will be maximal, using the following dimensionless linearity index:

$$L = \left| \frac{\left. \frac{\partial^2 f}{\partial x^2} \right|_{\mu_x} 2\sigma_x}{\left. \frac{\partial f}{\partial x} \right|_{\mu_x}} \right|. \quad (12)$$

When $L \approx 0$, the function can be considered linear in the interval, and hence, Gaussianity is preserved during transformation.

B. Linearity of *XYZ* Parametrization

The linearity of the *XYZ* representation is analyzed by means of a simplified model that only estimates the depth of a point with respect to the camera. In our analysis, a scene point is observed by two cameras [Fig. 3(a)], both of which are oriented toward the point. The first camera detects the ray on which the point lies. The second camera observes the same point from a distance d_1 ; the parallax angle α is approximated by the angle between the cameras' optic axes.

The point's location error d is encoded as Gaussian in depth

$$D = d_0 + d, \quad d \sim N(0, \sigma_d^2). \quad (13)$$

This error d is propagated to the image of the point in the second camera u as

$$u = \frac{x}{y} = \frac{d \sin \alpha}{d_1 + d \cos \alpha}. \quad (14)$$

The Gaussianity of u is analyzed by means of (12), giving the following linearity index:

$$L_d = \left| \frac{\partial^2 u / \partial d^2}{\partial u / \partial d} \right| = \frac{4\sigma_d}{d_1} |\cos \alpha|. \quad (15)$$

C. Linearity of Inverse Depth Parametrization

The inverse depth parametrization is based on the same scene geometry as the direct depth coding, but the depth error is encoded as Gaussian in inverse depth [Fig. 3(b)]:

$$D = \frac{1}{\rho_0 - \rho}, \quad \rho \sim N(0, \sigma_\rho^2) \quad (16)$$

$$d = D - d_0 = \frac{\rho}{\rho_0(\rho_0 - \rho)} \quad d_0 = \frac{1}{\rho_0}. \quad (17)$$

So, the image of the scene point is computed as

$$u = \frac{x}{y} = \frac{d \sin \alpha}{d_1 + d \cos \alpha} = \frac{\rho \sin \alpha}{\rho_0 d_1 (\rho_0 - \rho) + \rho \cos \alpha} \quad (18)$$

and the linearity index L_ρ is now

$$L_\rho = \left| \frac{\partial^2 u / \partial \rho^2}{\partial u / \partial \rho} \right| = \frac{4\sigma_\rho}{\rho_0} \left| 1 - \frac{d_0}{d_1} \cos \alpha \right|. \quad (19)$$

D. Depth Versus Inverse Depth Comparison

When a feature is initialized, the depth prior has to cover a vast region in front of the camera. With the inverse depth representation, the 95% confidence region with parameters ρ_0 , σ_ρ is

$$\left[\frac{1}{\rho_0 + 2\sigma_\rho}, \frac{1}{\rho_0 - 2\sigma_\rho} \right]. \quad (20)$$

This region cannot include zero depth but can easily extend to infinity.

Conversely, with the depth representation, the 95% region with parameters d_0 , σ_d is $[d_0 - 2\sigma_d, d_0 + 2\sigma_d]$. This region can include zero depth but cannot extend to infinity.

In the first few frames, after a new feature has been initialized, little parallax is likely to have been observed. Therefore, $d_0/d_1 \approx 1$ and $\alpha \approx 0 \implies \cos \alpha \approx 1$. In this case, the L_d linearity index for depth is high (bad), while the L_ρ linearity index for inverse depth is low (good): during initialization, the inverse depth measurement equation linearity is superior to the *XYZ* coding.

As estimation proceeds and α increases, leading to more accurate depth estimates, the inverse depth representation continues to have a high degree of linearity. This is because in the expression for L_ρ , the increase in the term $|1 - (d_0/d_1)\cos \alpha|$ is compensated by the decrease in $4\sigma_\rho/\rho_0$. For inverse depth features, a good linearity index is achieved along the whole estimation history. So, the inverse depth coding is suitable for both low and high parallax cases if the feature is continuously observed.

The *XYZ* encoding has low computational cost, but achieves linearity only at low depth uncertainty and high parallax. In Section VI, we explain how the representation of a feature can be switched over such that the inverse depth parametrization is only used when needed—for features that are either just initialized or at extreme depths.

V. FEATURE INITIALIZATION

From just a single observation, no feature depth can be estimated (although it would be possible in principle to impose a very weak depth prior by knowledge of the type of scene observed). What we do is to assign a general Gaussian prior in inverse depth that encodes probabilistically the fact that the point has to be in front of the camera. Hence, due to the linearity of inverse depth at low parallax, the filter can be initialized from just one observation. Experimental tuning has shown that infinity should be included with reasonable probability within the initialization prior, despite the fact that this means that depth estimates can become negative. Once initialized, features are processed with the standard EKF prediction-update loop—even in the case of negative inverse depth estimates—and immediately contribute to camera location estimation within SLAM.

It is worth noting that while a feature retains low parallax, it will automatically be used mainly to determine the camera orientation. The feature's depth will remain uncertain with the hypothesis of infinity still under consideration (represented by the probability mass corresponding to negative inverse depths). If the camera translates to produce enough parallax, then the feature's depth estimation will be improved and it will begin to contribute more to the camera location estimation.

The initial location for a newly observed feature inserted into the state vector is

$$\hat{\mathbf{y}}(\hat{\mathbf{r}}^{WC}, \hat{\mathbf{q}}^{WC}, \mathbf{h}, \rho_0) = (\hat{x}_i \quad \hat{y}_i \quad \hat{z}_i \quad \hat{\theta}_i \quad \hat{\phi}_i \quad \hat{\rho}_i)^\top \quad (21)$$

a function of the current camera pose estimate $\hat{\mathbf{r}}^{WC}$, $\hat{\mathbf{q}}^{WC}$, the image observation $\mathbf{h} = (u \quad v)^\top$, and the parameters determining the depth prior ρ_0 , σ_ρ .

The endpoint of the initialization ray (see Fig. 1) is taken from the current camera location estimate

$$(\hat{x}_i \quad \hat{y}_i \quad \hat{z}_i)^\top = \hat{\mathbf{r}}^{WC} \quad (22)$$

and the direction of the ray is computed from the observed point, expressed in the world coordinate frame

$$\mathbf{h}^W = \mathbf{R}_{WC}(\hat{\mathbf{q}}^{WC})(v \quad \nu \quad 1)^\top \quad (23)$$

where v and ν are normalized retina image coordinates. Despite \mathbf{h}^W being a nonunit directional vector, the angles by which we parametrize its direction can be calculated as

$$\begin{pmatrix} \theta_i \\ \phi_i \end{pmatrix} = \begin{pmatrix} \arctan(\mathbf{h}_x^W, \mathbf{h}_z^W) \\ \arctan(-\mathbf{h}_y^W, \sqrt{\mathbf{h}_x^W \cdot \mathbf{h}_z^W}) \end{pmatrix}. \quad (24)$$

The covariance of \hat{x}_i , \hat{y}_i , \hat{z}_i , $\hat{\theta}_i$, and $\hat{\phi}_i$ is derived from the image measurement error covariance \mathbf{R}_i and the state covariance estimate $\hat{\mathbf{P}}_{k|k}$.

The initial value for ρ_0 and its standard deviation are set empirically such that the 95% confidence region spans a range of

depths from close to the camera up to infinity. In our experiments, we set $\hat{\rho}_0 = 0.1$, $\sigma_\rho = 0.5$, which gives an inverse depth confidence region $[1.1, -0.9]$. Notice that infinity is included in this range. Experimental validation has shown that the precise values of these parameters are relatively unimportant to the accurate operation of the filter as long as infinity is clearly included in the confidence interval.

The state covariance after feature initialization is

$$\hat{\mathbf{P}}_{k|k}^{\text{new}} = \mathbf{J} \begin{pmatrix} \hat{\mathbf{P}}_{k|k} & 0 & 0 \\ 0 & \mathbf{R}_i & 0 \\ 0 & 0 & \sigma_\rho^2 \end{pmatrix} \mathbf{J}^\top \quad (25)$$

$$\mathbf{J} = \left(\begin{array}{c|c} I & 0 \\ \hline \frac{\partial \mathbf{y}}{\partial \mathbf{r}^{WC}}, \frac{\partial \mathbf{y}}{\partial \mathbf{q}^{WC}}, 0, \dots, 0, & \frac{\partial \mathbf{y}}{\partial \mathbf{h}}, \frac{\partial \mathbf{y}}{\partial \rho} \end{array} \right). \quad (26)$$

The inherent scale ambiguity in a monocular SLAM has usually been fixed by observing some known initial features that fix the scale (e.g., [8]). A very interesting experimental observation we have made using the inverse depth scheme is that sequential monocular SLAM can operate successfully without *any* known features in the scene, and in fact, the experiments we present in this paper do not use an initialization target. In this case, of course, the overall scale of the reconstruction and camera motion is undetermined, although with the formulation of the current paper, the estimation will settle on a (meaningless) scale of some value. In a very recent work [6], we have investigated this issue with a new dimensionless formulation of monocular SLAM.

VI. SWITCHING FROM INVERSE DEPTH TO XYZ

While the inverse depth encoding can be used at both low and high parallax, it is advantageous for reasons of computational efficiency to restrict inverse depth to cases where the XYZ encoding exhibits nonlinearity according to the L_d index. This section details switching from inverse depth to XYZ for high parallax features.

A. Conversion From Inverse Depth to XYZ Coding

After each estimation step, the linearity index L_d (15) is computed for every map feature coded in inverse depth

$$\begin{aligned} \mathbf{h}_{XYZ}^W &= \hat{\mathbf{x}}_i - \hat{\mathbf{r}}^{WC} & \sigma_d &= \frac{\sigma_\rho}{\rho_i^2} & \sigma_\rho &= \sqrt{\mathbf{P}_{\mathbf{y}_i \mathbf{y}_i}} (6, 6) \\ d_i &= \|\mathbf{h}_{XYZ}^W\| & \cos \alpha &= \mathbf{m}^\top \mathbf{h}_{XYZ}^W \|\mathbf{h}_{XYZ}^W\|^{-1} \end{aligned} \quad (27)$$

where $\hat{\mathbf{x}}_i$ is computed using (4) and $\mathbf{P}_{\mathbf{y}_i \mathbf{y}_i}$ is the submatrix 6×6 covariance matrix corresponding to the considered feature.

If L_d is below a switching threshold, the feature is transformed using (4) and the *full state* covariance matrix \mathbf{P} is transformed with the corresponding Jacobian:

$$\mathbf{P}_{\text{new}} = \mathbf{J} \mathbf{P} \mathbf{J}^\top \quad \mathbf{J} = \text{diag} \left(\mathbf{I}, \frac{\partial \mathbf{x}_i}{\partial \mathbf{y}_i}, \mathbf{I} \right). \quad (28)$$

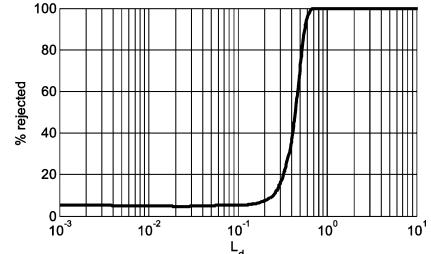


Fig. 4. Percentage of test rejections as a function of the linearity index L_d .

B. Linearity Index Threshold

We propose to use index L_d (15) to define a threshold for switching from inverse depth to XYZ encoding at the point when the latter can be considered linear. If the XYZ representation is linear, then the measurement u is Gaussian distributed (10), i.e.,

$$u \sim N(\mu_u, \sigma_u^2) \quad \mu_u = 0 \quad \sigma_u^2 = \left(\frac{\sin \alpha}{d_1} \right)^2 \sigma_d^2. \quad (29)$$

To determine the threshold in L_d that signals a lack of linearity in the measurement equation, a simulation experiment has been performed. The goal was to generate samples from the uncertain distribution for variable u , and then, apply a standard Kolmogorov-Smirnov Gaussianity [4] test to these samples, counting the percentage of rejected hypotheses h . When u is effectively Gaussian, the percentage should match the test significance level α_{sl} (5% in our experiments); as the number of rejected hypotheses increases, the measurement equation departs from linearity. A plot of the percentage of rejected hypotheses h with respect to the linearity index L_d is shown in Fig. 4. It can be clearly seen that when $L_d > 0.2$, h sharply departs from 5%. So, we propose the $L_d < 10\%$ threshold for switching from inverse depth to XYZ encoding.

Notice that the plot in Fig. 4 is smooth (log scale in L_d), which indicates that the linearity index effectively represents the departure from linearity.

The simulation has been performed for a variety of values of α , d_1 , and σ_d ; more precisely, all triplets resulting from the following parameter values:

$$\begin{aligned} \alpha(\text{deg}) &\in \{0.1, 1, 3, 5, 7, 10, 20, 30, 40, 50, 60, 70\} \\ d_1(\text{m}) &\in \{1, 3, 5, 7, 10, 20, 50, 100\} \\ \sigma_d(\text{m}) &\in \{0.05, 0.1, 0.25, 0.5, 0.75, 1, 2, 5\}. \end{aligned}$$

The simulation algorithm detailed in Fig. 5 is applied to every triplet $\{\alpha, d_1, \sigma_d\}$ to count the percentage of rejected hypotheses h and the corresponding linearity index L_d .

VII. EXPERIMENTAL RESULTS

The performance of the new parametrization has been tested on real-image sequences acquired with a handheld-low-cost Unibrain IEEE1394 camera, with a 90° field of view and

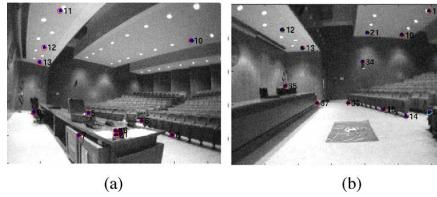
```

input:  $\alpha, d_1, \sigma_d$ 
output:  $h, L_d$ 
 $\sigma_u = \left| \frac{\sin \alpha}{d_1} \right| \sigma_d; \mu_u = 0; // (29)$ 
 $\alpha_{st} = 0.05; // \text{Kolm. test sign. level}$ 
 $L_d = \frac{4\sigma_d}{d_1} |\cos \alpha|$ 
n_rejected=0 ;
N_GENERATED_SAMPLES=1000;
SAMPLE_SIZE=1000;

for j=1 to N_GENERATED_SAMPLES repeat
     $\{d_i\}_j = \text{random\_normal}(0, \sigma_d^2, \text{SAMPLE\_SIZE});$ 
    //generate a normal sample from  $N(0, \sigma_d^2)$ ;
     $\{u_i\}_j = \text{propagate\_from\_dept\_to\_image}(\{d_i\}_j, \alpha, d_1); // (14)$ 
    if rejected==Kolmogorov_Smirnov( $\{u_i\}_j, \mu_u, \sigma_u, \alpha_{st}$ )
        n_rejected=n_rejected+1;
endfor
 $h=100 \frac{[n\_rejected]}{[N\_GENERATED\_SAMPLES]}$ ;

```

Fig. 5. Simulation algorithm to test the linearity of the measurement equation.



(a)

(b)

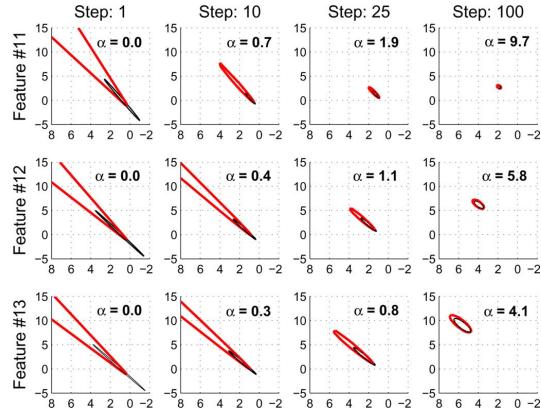
Fig. 6. First (a) and last (b) frame in the sequence of the indoor experiment of Section VII-A. Features 11, 12, and 13 are analyzed. These features are initialized in the same frame but are located at different distances from the camera.

320 × 240 resolution, capturing monochrome image sequences at 30 fps.

Five experiments were performed. The first was an indoor sequence processed offline with a Matlab implementation, the goal being to analyze initialization of scene features located at different depths. The second experiment shows an outdoor sequence processed in real time with a C++ implementation. The focus was on distant features observed under low parallax along the whole sequence. The third experiment was a loop closing sequence, concentrating on camera covariance evolution. Fourth was a simulation experiment to analyze the effect of switching from inverse depth to XYZ representations. In the last experiment, the switching performance was verified on the real loop closing sequence. This section ends with a computing time analysis. It is worth noting that no initial pattern to fix the scale was used in any of the last three experiments.

A. Indoor Sequence

This experiment analyzes the performance of the inverse depth scheme as several features at a range of depths are tracked within SLAM. We discuss three features, which are all detected in the same frame but have very different depths. Fig. 6 shows the image where the analyzed features are initialized (frame 18 in the sequence) and the last image in the sequence. Fig. 7 focuses on the evolution of the estimates corresponding to the features, with labels 11, 12, and 13, at frames 1, 10, 25, and 100. Confidence regions derived from the inverse depth representa-

Fig. 7. Feature initialization. Each column shows the estimation history for a feature horizontal components. For each feature, the estimates after 1, 10, 25, and 100 frames since initialization are plotted; the parallax angle α in degrees between the initial observation and the current frame is displayed. The thick (red) lines show (calculated by a Monte Carlo numerical simulation) the 95% confidence region when coded as Gaussian in inverse depth. The thin (black) ellipsoids show the uncertainty as a Gaussian in the XYZ space propagated according to (28). Notice how at low parallax, the inverse depth confidence region is very different from the elliptical Gaussian. However, as the parallax increases, the uncertainty reduces and collapses to the Gaussian ellipse.

tion (thick red line) are plotted in the XYZ space by numerical Monte Carlo propagation from the 6-D multivariate Gaussians representing these features in the SLAM EKF. For comparison, standard Gaussian XYZ acceptance ellipsoids (thin black line) are linearly propagated from the 6-D representation by means of the Jacobian of (28). The parallax α in degrees for each feature at every step is also displayed.

When initialized, the 95% acceptance region of all the features includes $\rho = 0$, so infinite depth is considered as a possibility. The corresponding confidence region in depth is highly asymmetric, excluding low depths but extending to infinity. It is clear that Gaussianity in inverse depth is not mapped to Gaussianity in XYZ, so the black ellipsoids produced by Jacobian transformation are far from representing the true depth uncertainty. As stated in Section IV-D, it is at low parallax that the inverse depth parametrization plays a key role.

As rays producing bigger parallax are gathered, the uncertainty in ρ becomes smaller but still maps to a nonGaussian distribution in XYZ. Eventually, at high parallax, for all of the features, the red confidence regions become closely Gaussian and well approximated by the linearly propagated black ellipses—but this happens much sooner for nearby feature 11 than distant feature 13.

A movie showing the input sequence and estimation history of this experiment is available as multimedia data `inverseDepth.indoor.avi`. The raw input image sequence is also available at `inverseDepth.indoorRaw-Images.tar.gz`.

B. Real-Time Outdoor Sequence

This 860 frame experiment was performed with a C++ implementation that achieves real-time performance

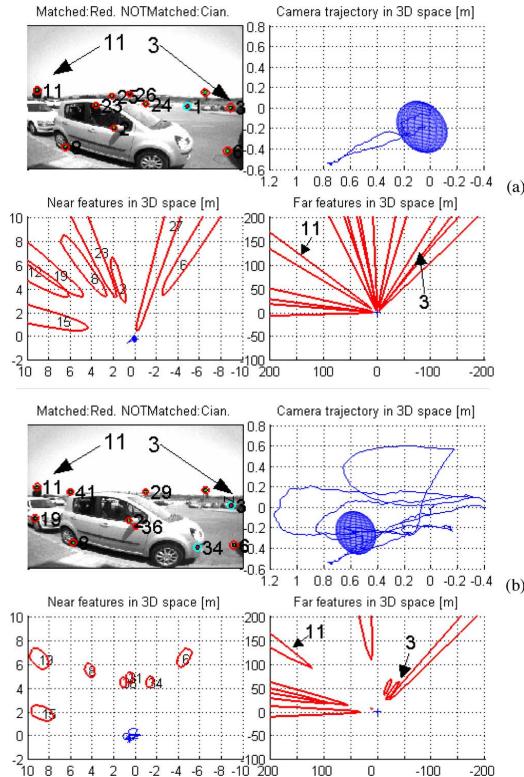


Fig. 8. (a) and (b) show frames #163 and #807 from the outdoor experiment of Section VII-B. This experiment was processed in real time. The focus was two features: 11 (tree on the left) and 3 (car on the right) at low parallax. Each of the two figures shows the current images and top-down views illustrating the horizontal components of the estimation of camera and feature locations at three different zoom scales for clarity: the top-right plots (maximum zoom) highlight the estimation of the camera motion; bottom-left (medium zoom) views highlight nearby features; and bottom-right (minimum zoom) emphasizes distant features.

at 30 fps with the handheld camera. Here, we highlight the ability of our parametrization to deal with both close and distant features in an outdoor setting. The input image sequence is available at multimedia material `inverseDepth_outdoorRawImages.tar.gz`. A movie showing the estimation process is also available at `inverseDepth_outdoor.avi`.

Fig. 8 shows two frames of the movie illustrating the performance. For most of the features, the camera ended up gathering enough parallax to accurately estimate their depths. However, being outdoors, there were distant features producing low parallax during the whole camera motion.

The inverse depth estimation history for two features is highlighted in Fig. 9. It is shown that distant, low parallax features are persistently tracked through the sequence, despite the fact that their depths cannot be precisely estimated. The large depth uncertainty, represented with the inverse depth scheme, is

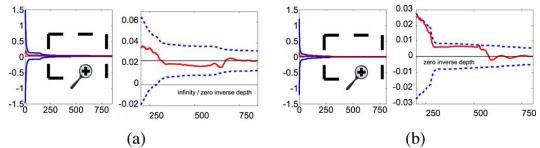


Fig. 9. Analysis of outdoor experiment of Section VII-B. (a) Inverse depth estimation history for feature 3, on the car, and (b) for feature 11, on a distant tree. Due to the uncertainty reduction during estimation, two plots at different scales are shown for each feature. It shows the 95% confidence region, and with a thick line, the estimated inverse depth. The thin solid line is the inverse depth estimated after processing the whole sequence. In (a), for the first 250 steps, zero inverse depth is included in the confidence region, meaning that the feature might be at infinity. After this, more distant but finite locations are gradually eliminated, and eventually, the feature's depth is accurately estimated. In (b), the tree is so distant that the confidence region always includes zero since little parallax is gathered for that feature.

successfully managed by the SLAM EKF, allowing the orientation information supplied by these features to be exploited.

Feature 3, on a nearby car, eventually gathers enough parallax enough to have an accurate depth estimate after 250 images, where infinite depth is considered as a possibility. Meanwhile, the estimate of feature 11, on a distant tree and never displaying significant parallax, never collapses in this way and zero inverse depth remains within its confidence region. Delayed initialization schemes would have discarded this feature without obtaining any information from it, while in our system, it behaves like a bearing reference. This ability to deal with distant points in real time is a highly advantageous quality of our parametrization. Note that what does happen to the estimate of feature 11 as translation occurs is that hypotheses of nearby depths are ruled out—the inverse depth scheme correctly recognizes that measuring little parallax while the camera has translated some distance allows a minimum depth for the feature to be set.

C. Loop Closing Sequence

A loop closing sequence offers a challenging benchmark for any SLAM algorithm. In this experiment, a handheld camera was carried by a person walking in small circles within a very large student laboratory, carrying out two complete laps. The raw input image sequence is available at `inverseDepth_loopClosingRawImages.tar.gz`, and a movie showing the mapping process at `inverseDepth_loopClosing.avi`.

Fig. 10 shows a selection of the 737 frames from the sequence, concentrating on the beginning, first loop closure, and end of the sequence. Fig. 11 shows the camera location estimate covariance history, represented by the 95% confidence regions for the six camera DOF and expressed in a reference local to the camera.

We observe the following properties of the evolution of the estimation, focussing, in particular, on the uncertainty in the camera location.

- 1) After processing the first few images, the uncertainty in the depth of features is huge, with highly nonelliptical confidence regions in the XYZ space [Fig. 10(a)].
- 2) In Fig. 11, the first peak in the X and Z translation uncertainty corresponds to a camera motion backward along

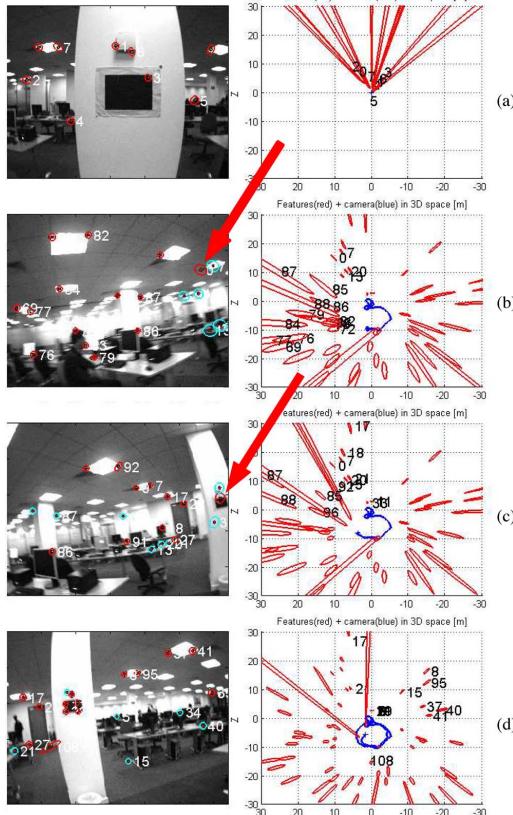


Fig. 10. Selection of frames from the loop closing experiment of Section VII-C. For each frame, we show the current image and reprojected map (left), and a top-down view of the map with 95% confidence regions and camera trajectory (right). Notice that confidence regions for the map features are far from being Gaussian ellipses, especially for newly initialized or distant features. The selected frames are: (a) #11, close to the start of the sequence; (b) #417, where the first loop closing match, corresponding to a distant feature, is detected; the loop closing match is signaled with an arrow; (c) #441, where the first loop closing match corresponding to a close feature is detected; the match is signaled with an arrow; and (d) #737, the last image, in the sequence, after reobserving most of the map features during the second lap around the loop.

- the optical axis; this motion produces poor parallax for newly initialized features, and we, therefore, see a reduction in the orientation uncertainty and an increase in the translation uncertainty. After frame #50, the camera again translates in the X -direction, parallax is gathered, and the translation uncertainty is reduced.
- 3) From frame #240, the camera starts a 360° circular motion in the XZ plane. The camera explores new scene regions, and the covariance increases steadily as expected (Fig. 11).
 - 4) In frame #417, the first loop closing feature is reobserved. This is a feature that is distant from the camera, and causes an abrupt reduction in the orientation and translation uncertainty (Fig. 11), though a medium level of uncertainty remains.

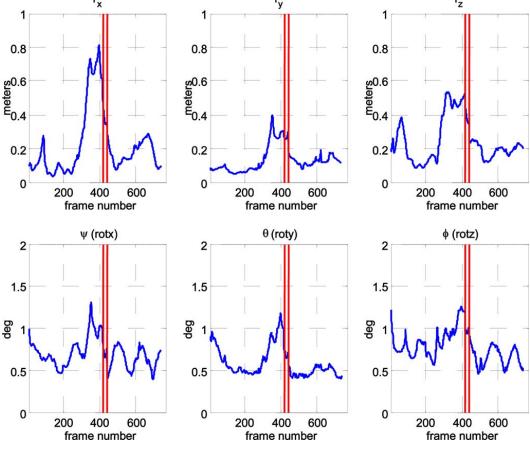


Fig. 11. Camera location estimate covariance along the sequence. The 95% confidence regions for each of the 6 DOF of camera motion are plotted. Note that errors are expressed in a reference local to the camera. The vertical solid lines indicate the loop closing frames #417 and #441.

- 5) In frame #441, a much closer loop closing feature (mapped with high parallax) is matched. Another abrupt covariance reduction takes place (Fig. 11) with the extra information this provides.
- 6) After frame #441, as the camera goes on a second lap around the loop, most of the map features are revisited, almost no new features are initialized, and hence, the uncertainty in the map is further reduced. Comparing the map at frame #441 (the beginning of the second lap) and #737, (the end of the second lap), we see a significant reduction in uncertainty. During the second lap, the camera uncertainty is low, and as features are reobserved, their uncertainties are noticeably reduced [Fig. 10(c) and (d)].

Note that these loop closing results with the inverse depth representation show a marked improvement on the experiments on monocular SLAM with a humanoid robot presented in [9], where a gyro was needed in order to reduce angular uncertainty enough to close loops with very similar camera motions.

D. Simulation Analysis for Inverse Depth to XYZ Switching

In order to analyze the effect of the parametrization switching proposed in Section VI on the consistency of SLAM estimation, simulation experiments with different switching thresholds were run. In the simulations, a camera completed two laps of a circular trajectory of radius 3 m in the XZ plane, looking out radially at a scene composed of points lying on three concentric spheres of radius 4.3, 10, and 20 m. These points at different depths were intended to produce observations with a range of parallax angles (Fig. 12).

The camera parameters of the simulation correspond with our real image acquisition system: camera 320×240 pixels, frame rate 30 frames/s, image field of view 90° , measurement uncertainty for a point feature in the image, and Gaussian

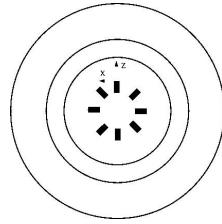


Fig. 12. Simulation configuration for analysis of parametrization switching in Section VII-D, sketching the circular camera trajectory and 3-D scene, composed of three concentric spheres of radius 4.3, 10, and 20 m. The camera completes two circular laps in the (X - Z)-plane with radius 3 m, and is oriented radially.

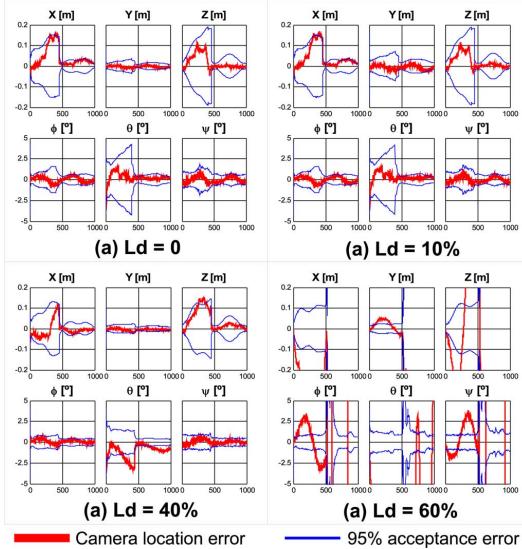


Fig. 13. Details from the parametrization switching experiment. Camera location estimation error history in 6 DOF (translation in XYZ , and three orientation angles $\psi\theta\phi$) for four switching thresholds: With $L_d = 0\%$, no switching occurs and the features all remain in the inverse depth parametrization. At $L_d = 10\%$, although features from the spheres at 4.3 and 10 m are eventually converted, no degradation with respect to the non-switching case is observed. At $L_d = 40\%$, some features are switched before achieving true Gaussianity, and there is noticeable degradation, especially in θ rotation around the Y axis. At $L_d = 60\%$, the map becomes totally inconsistent and loop closing fails.

$N(0, 1 \text{ pixel}^2)$. The simulated image sequence contained 1000 frames. Features were selected following the randomized map management algorithm proposed in [8] in order to have 15 features visible in the image at all times. All our simulation experiments work using the same scene features in order to homogenize the comparison.

Four simulation experiments for different thresholds for switching each feature from inverse depth to XYZ parametrization were run with $L_d \in \{0\%, 10\%, 40\%, 60\%\}$. Fig. 13 shows the camera trajectory estimation history in 6 DOF (translation in XYZ , and three orientation angles

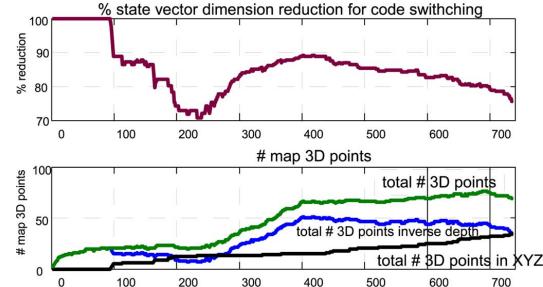


Fig. 14. Parametrization switching on a real sequence (Section VII-E): state vector size history. Top: percentage reduction in state dimension when using switching compared with keeping all points in inverse depth. Bottom: total number of points in the map, showing the number of points in inverse depth and the number of points in XYZ .

$\psi(\text{Rot}_x), \theta(\text{Rot}_y), \phi(\text{Rot}_z)$, cyclotorsion). The following conclusions can be made.

- 1) Almost the same performance is achieved with no switching (0%) and with 10% switching. So, it is clearly advantageous to perform 10% switching because there is no penalization in accuracy and the state vector size of each converted feature is halved.
- 2) Switching too early degrades accuracy, especially in the orientation estimate. Notice how for 40% the orientation estimate is worse and the orientation error covariance is smaller, showing filter inconsistency. For 60%, the estimation is totally inconsistent and the loop closing fails.
- 3) Since early switching degrades performance, the inverse depth parametrization is mandatory for initialization of every feature and over the long term for low parallax features.

E. Parametrization Switching With Real Images

The loop closing sequence of Section VII-C was processed without any parametrization switching, and with switching at $L_d = 10\%$. A movie showing the results is available at `inverseDepth_loopClosing_ID_to_XYZ_conversion.avi`. As in the simulation experiments, no significant change was noticed in the estimated trajectory or map.

Fig. 14 shows the history of the state size, the number of map features, and how their parametrization evolves. At the last estimation step, about half of the features had been switched; at this step, the state size had reduced from 427 to 322 (34 inverse depth features and 35 XYZ), i.e., 75% of the original vector size. Fig. 15 shows four frames from the sequence illustrating feature switching. Up to step 100, the camera has low translation and all the features are in inverse depth form. As the camera translates, nearby features switch to XYZ . Around step 420, the loop is closed and features are reobserved, producing a significant reduction in uncertainty that allows switching of more reobserved close features. Our method automatically determines which features should be represented in the inverse depth or XYZ forms, optimizing computational efficiency without sacrificing accuracy.

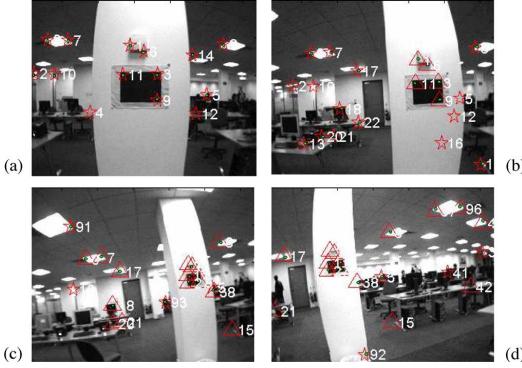


Fig. 15. Parametrization switching seen in image space: points coded in inverse depth (+) and coded in XYZ (\triangle). (a) First frame, with all features in inverse depth. (b) Frame #100; nearby features start switching. (c) Frame #470, loop closing; most features in XYZ . (d) Last image of the sequence.

F. Processing Time

We give some details of the real-time operation of our monocular SLAM system, running on a 1.8 GHz Pentium M processor laptop. A typical EKF iteration would imply the following.

- 1) A state vector dimension of 300.
- 2) 12 features observed in the image, a measurement dimension of 24.
- 3) 30 fps, so 33.3 ms available for processing.

Typical computing time breaks down as follows: image acquisition, 1 ms.; EKF prediction, 2 ms; image matching, 1 ms.; and EKF update, 17 ms. This adds up to a total of 21 ms. The remaining time is used for graphics functions using OpenGL on an NVidia card and scheduled at a low priority.

The quoted state vector size 300 corresponds to a map size of 50 if all features are encoded using inverse depth. In indoor scenes, due to switching maps of up to 60–70, features can be computed in real time. This size is enough to map many typical scenes robustly.

VIII. CONCLUSION

We have presented a parametrization for monocular SLAM that permits operation based uniquely on the standard EKF prediction-update procedure at every step, unifying initialization with the tracking of mapped features. Our inverse depth parametrization for 3-D points allows unified modeling and processing for any point in the scene, close or distant, or even at “infinity.” In fact, close, distant, or just-initialized features are processed within the routine EKF prediction-update loop without making any binary decisions. Due to the undelayed initialization and immediate full use of infinite points, estimates of camera orientation are significantly improved, reducing the camera estimation jitter often reported in previous work. The jitter reduction, in turn, leads to computational benefits in terms of smaller search regions and improved image processing speed.

The key factor is that due to our parametrization of the direction and inverse depth of a point relative to the location

from which it was first seen, our measurement equation has low linearization errors at low parallax, and hence, the estimation uncertainty is accurately modeled with a multivariate Gaussian. In Section IV, we presented a model that quantifies linearization error. This provides a theoretical understanding of the impressive outdoor, real-time performance of the EKF with our parametrization.

The inverse depth representation requires a 6-D state vector per feature, compared to three for XYZ coding. This doubles the map state vector size, and hence produces a fourfold increase in the computational cost of the EKF. Our experiments show that it is essential to retain the inverse depth parametrization for initialization and distant features, but nearby features can be safely converted to the cheaper XYZ representation, meaning that the long-term computational cost need not increase significantly. We have given details on when this conversion should be carried out for each feature to optimize computational efficiency without sacrificing accuracy.

The experiments presented have validated the method with real imagery using a handheld camera as the only sensor, both indoors and outdoors. We have experimentally verified the following the key contributions of our study:

- 1) real-time performance achieving 30 fps real-time processing for maps up to 60–70 features;
- 2) real-time loop closing;
- 3) dealing simultaneously with low and high parallax features;
- 4) nondelayed initialization;
- 5) low jitter, full 6-DOF monocular SLAM.

In the experiments, we have focused on a map size around 60–100 features because these map sizes can be dealt with in real time at 30 Hz, and we have focused on the challenging loop closing issue. Useful future work would be a thorough analysis of the limiting factors in EKF inverse depth monocular SLAM in terms of linearity, data association errors, accuracy, map size, and ability to deal with degenerate motion such as pure rotations or a static camera for long-time periods.

Finally, our simulations and experiments have shown that inverse depth monocular SLAM operates well without known patterns in the scene to fix scale. This result points toward further work in understanding the role of scale in monocular SLAM (an avenue that we have begun to investigate in a dimensionless formulation in [6]) and further bridging the gap between sequential SLAM techniques and structure from motion methods from the computer vision literature.

APPENDIX

To recover the ideal projective undistorted coordinates $\mathbf{h}_u = (u_u, v_u)^\top$ from the actually distorted ones gathered by the camera $\mathbf{h}_d = (u_d, v_d)^\top$, the classical two parameters radial distortion model [17] is applied:

$$\begin{pmatrix} u_u \\ v_u \end{pmatrix} = \mathbf{h}_u \begin{pmatrix} u_d \\ v_d \end{pmatrix} = \begin{pmatrix} u_0 + (u_d - u_0)(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) \\ v_0 + (v_d - v_0)(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) \end{pmatrix}$$

$$r_d = \sqrt{(d_x(u_d - u_0))^2 + (d_y(v_d - v_0))^2} \quad (30)$$

where u_0, v_0 are the image centers and κ_1, κ_2 are the radial distortion coefficients.

To compute the distorted coordinates from the undistorted

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \mathbf{h}_d \begin{pmatrix} u_u \\ v_u \end{pmatrix} = \begin{pmatrix} u_0 + \frac{(u_u - u_0)}{(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4)} \\ v_0 + \frac{(v_u - v_0)}{(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4)} \end{pmatrix} \quad (31)$$

$$r_u = r_d (1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) \quad (32)$$

$$r_u = \sqrt{(d_x(u_u - u_0))^2 + (d_y(v_u - v_0))^2} \quad (33)$$

where r_u is readily computed from (33), but r_d has to be numerically solved from (32), e.g., using Newton–Raphson method; hence, (31) can be used to compute the distorted point.

Undistortion Jacobian $\partial \mathbf{h}_u / \partial (u_d, v_d)$ has the following analytical expression:

$$\left(\begin{array}{c|c} (1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) + 2((u_d - u_0) d_x)^2 \times (\kappa_1 + 2\kappa_2 r_d^2) & 2d_y^2 (u_d - u_0) (v_d - v_0) \times (\kappa_1 + 2\kappa_2 r_d^2) \\ \hline 2d_x^2 (v_d - v_0) (u_d - u_0) \times (\kappa_1 + 2\kappa_2 r_d^2) & (1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) + 2((v_d - v_0) d_y)^2 \times (\kappa_1 + 2\kappa_2 r_d^2) \end{array} \right) \quad (34)$$

The Jacobian for the distortion is computed by inverting expression (34)

$$\frac{\partial \mathbf{h}_d}{\partial (u_u, v_u)} \Big|_{(u_d, v_d)} = \left(\frac{\partial \mathbf{h}_u}{\partial (u_d, v_d)} \Big|_{\mathbf{h}_d(u_u, v_u)} \right)^{-1}. \quad (35)$$

ACKNOWLEDGMENT

The authors are very grateful to D. Murray, I. Reid, and other members of Oxford's Active Vision Laboratory for discussions and software collaboration. They also thank the anonymous reviewers for their useful comments.

REFERENCES

- [1] V. J. Aidala and S. E. Hammel, "Utilization of modified polar coordinates for bearing-only tracking," *IEEE Trans. Autom. Control*, vol. 28, no. 3, pp. 283–294, Mar. 1983.
- [2] T. Bailey, "Constrained initialisation for bearing-only SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, R.O.C., Sep. 14–19, 2003, vol. 2, pp. 1966–1971.
- [3] M. Bryson and S. Sukkarieh, "Bearing-only SLAM for an airborne vehicle," presented at the Australasian Conf. Robot. Autom. (ACRA 2005), Sydney, Australia.
- [4] G. C. Canavos, *Applied Probability and Statistical Methods*. Boston, MA: Little, Brow, 1984.
- [5] A. Chowdhury and R. Chellappa, "Stochastic approximation and rate-distortion analysis for robust structure and motion estimation," *Int. J. Comput. Vis.*, vol. 55, no. 1, pp. 27–53, 2003.
- [6] J. Civera, A. J. Davison, and J. M. M. Montiel, "Dimensionless monocular SLAM," in *Proc. 3rd Iberian Conf. Pattern Recogn. Image Anal.*, 2007, pp. 412–419.
- [7] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth to depth conversion for monocular SLAM," in *Proc. Int. Conf. Robot. Autom.*, 2007, pp. 2778–2783.
- [8] A. Davison, "Real-time simultaneous localization and mapping with a single camera," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2003, pp. 1403–1410.
- [9] A. J. Davison, I. Reid, N. Molton, and O. Stasse, "Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.
- [10] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.* Jun. 17–22, 2006, vol. 1, pp. 469–476.
- [11] A. W. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *Proc. Eur. Conf. Comput. Vis.*, Jun. 1998, pp. 311–326.
- [12] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.
- [13] D. Heeger and A. Jepson, "Subspace methods for recovering rigid motion I: Algorithm and implementation," *Int. J. Comput. Vis.*, vol. 7, no. 2, pp. 95–117, Jan. 1992.
- [14] J. H. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *Proc. IEEE Int. Conf. Robot. Autom.* Sep. 14–19, 2003, vol. 1, pp. 406–411.
- [15] N. Kwok and G. Dissanayake, "An efficient multiple hypothesis filter for bearing-only SLAM," in *Proc. IROS*, 28 Sep.–2 Oct. 2004, vol. 1, pp. 736–741.
- [16] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *Int. J. Comput. Vis.*, vol. 3, no. 3, pp. 209–238, 1989.
- [17] E. Mikhail, J. Bethel, and J. C. McGlone, *Introduction to Modern Photogrammetry*. New York: Wiley, 2001.
- [18] J. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular SLAM," presented at the Robot. Sci. Syst. Conf., Philadelphia, PA, Aug. 2006.
- [19] J. Montiel and A. J. Davison, "A visual compass based on SLAM," in *Proc. Int. Conf. Robot. Autom.*, Orlando, FL, May 15–19, 2006, pp. 1917–1922.
- [20] E. Mouragnon, M. Lhuillier, M. Dhorne, F. Dekeyser, and P. Sayd, "Real-time localization and 3-D reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2006, vol. 3, pp. 1027–1031.
- [21] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *J. Field Robot.*, vol. 23, no. 1, pp. 3–26, 2006.
- [22] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 353–363, Apr. 1993.
- [23] M. Pollefeys, R. Koch, and L. Van Gool, "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters," *Int. J. Comput. Vis.*, vol. 32, no. 1, pp. 7–25, 1999.
- [24] J. Sola, "Towards visual localization, mapping and moving objects tracking by a mobile robot: A geometric and probabilistic approach," Ph.D. dissertation, LAAS-CNRS, Toulouse, France, 2007.
- [25] J. Sola, A. Monin, M. Devy, and T. Lemaire, "Undelayed initialization in bearing only SLAM," in *Proc. 2005 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2–6 Aug., 2005, pp. 2499–2504.
- [26] N. Trawny and S. I. Roumeliotis, "A unified framework for nearby and distant landmarks in bearing-only SLAM," in *Proc. Int. Conf. Robot. Autom.*, May 15–19, 2006, pp. 1923–1929.



Javier Civera was born in Barcelona, Spain, in 1980. He received the M.S. degree in industrial-electrical engineering in 2004 from the University in Zaragoza, where he is currently working toward the Ph.D. degree with the Robotics, Perception, and Real-Time Group.

He is currently an Assistant Lecturer with the Departamento de Informática, University of Zaragoza, where he teaches courses in automatic control theory. His current research interests include computer vision and mobile robotics.



Andrew J. Davison received the B.A. degree in physics and the D.Phil. degree in computer vision from the University of Oxford, Oxford, U.K., in 1994 and 1998, respectively.

He was with Oxford's Robotics Research Group, where he developed one of the first robot simultaneous localization and mapping (SLAM) systems using vision. He was a European Union (EU) Science and Technology Fellow at the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan, for two years, where he was engaged in visual robot navigation. In 2000, he returned to the University of Oxford as a Postdoctoral Researcher. In 2005, he joined Imperial College London, London, U.K., where he currently holds the position of Reader with the Department of Computing. His current research interests include advancing the basic technology of real-time localization and mapping using vision while collaborating to apply these techniques in robotics and related areas.

Dr. Davison was awarded a five-year Engineering and Physical Sciences Research Council (EPSRC) Advanced Research Fellowship in 2002. In 2008, he was awarded a European Research Council (ERC) Starting Grant.



J. M. Martínez Montiel was born in Arnedo, Spain. He received the M.S. and Ph.D. degrees in electrical engineering from the University of Zaragoza, Zaragoza, Spain, in 1991 and 1996, respectively.

He is currently an Associate Professor with the Departamento de Informática, University of Zaragoza, where he is in charge of Perception and Computer Vision courses. His current interests include computer vision, real-time vision localization and mapping research, and the transference of this technology to robotic and nonrobotic application domains.

Dr. Montiel is member of the the Robotics, Perception, and Real-Time Group. He has been awarded the Spanish Merrimack Education Center (MEC) grants to fund research at the University of Oxford, Oxford, U.K., and Imperial College London, London, U.K.

8.6 Visual Inertial Odometry: Robust Visual Inertial Odometry Using a Direct EKF-Based Approach, IROS, 2015

2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
 Congress Center Hamburg
 Sept 28 - Oct 2, 2015. Hamburg, Germany

Robust Visual Inertial Odometry Using a Direct EKF-Based Approach

Michael Bloesch, Sammy Omari, Marco Hutter, Roland Siegwart
 Autonomous Systems Lab, ETH Zürich, Switzerland, bloeschm@ethz.ch

Abstract—In this paper, we present a monocular visual-inertial odometry algorithm which, by directly using pixel intensity errors of image patches, achieves accurate tracking performance while exhibiting a very high level of robustness. After detection, the tracking of the multilevel patch features is closely coupled to the underlying extended Kalman filter (EKF) by directly using the intensity errors as innovation term during the update step. We follow a purely robocentric approach where the location of 3D landmarks are always estimated with respect to the current camera pose. Furthermore, we decompose landmark positions into a bearing vector and a distance parametrization whereby we employ a minimal representation of differences on a corresponding σ -Algebra in order to achieve better consistency and to improve the computational performance. Due to the robocentric, inverse-distance landmark parametrization, the framework does not require any initialization procedure, leading to a truly power-up-and-go state estimation system. The presented approach is successfully evaluated in a set of highly dynamic hand-held experiments as well as directly employed in the control loop of a multirotor unmanned aerial vehicle (UAV).

I. INTRODUCTION

Navigation and control of autonomous robots in rough and highly unstructured environments requires high-bandwidth and precise knowledge of position and orientation. Especially in dynamic operation of robots, the underlying state estimation can quickly become the bottleneck in terms of achievable bandwidth, robustness and speed. To enable the required performance for highly dynamic operation of robots, we combine complementary information from vision- and inertial sensors. This approach has a long history and has been successfully applied to navigate unmanned aerial robots [24], [21], walking robots [23], [25] or cars [8].

Within the field of computer vision, Davison et al. [5] proposed one of the first real-time 3D monocular localization and mapping frameworks. Since then, a lot of improvements have been contributed from various research groups and further approaches have been proposed. A key issue is to improve the consistency of the estimation framework which is affected by its inherent nonlinearity [13], [3]. One approach is to make use of a robocentric representation for the tracked features and thereby significantly reduce the effect of nonlinearities [3], [4]. As alternative, Huang et al. [11] propose the use of a so-called observability constrained extended Kalman filter, whereby the inconsistencies can be avoided by using special linearization points while evaluating the system Jacobians.

This research was supported in part by the Swiss National Science Foundation (SNF) through project 200021_149427/1 and the National Centre of Competence in Research Robotics.

A somewhat related problem is the choice of the specific representation of the features. Since for monocular setups, the depth of a newly detected feature is unknown the initial 3D location estimate of the feature exhibits a high (infinite) uncertainty along the corresponding axis. In order to integrate this feature from the beginning into the estimation framework, Montiel et al. [18] proposed the use of an inverse-depth parametrization (IDP). With this parametrization, each feature location is represented by the camera position where the feature was initially detected, by a bearing vector (parametrized with azimuth and elevation angle), as well as the inverse depth of the feature. The resulting increase in consistency was analyzed in more detail for the IDP and other feature parametrization in [22].

While most standard visual odometry approaches are based on detected and tracked point features as source of visual information, so-called *direct* approaches directly use the image intensities in their estimation framework. Especially with the recent advent of RGBD cameras, so called *dense* approaches, where the intensity error over the full image is considered, have gained a lot of attention [1], [15]. In comparison to traditional vision-based state estimators, dense approaches have a significantly larger error term count and require appropriate methods in order to tackle the additional computational load. By employing highly optimized SSE-vectorized implementations, first real-time, CPU-based approaches for dense- or semi-dense motion estimation using a RGBD [15] or a monocular RGB camera [6], [7] have recently been proposed.

Incorporating inertial measurements in the estimation can significantly improve the robustness of the system, provides the estimation process with the notion of gravity, and allows for a more accurate and high bandwidth estimation of the velocities and rotational rates. By adapting the original EKF proposed by Davison et al. [5], additional IMU measurements can be relatively simply integrated into the ego-motion estimation, whereby calibration parameters can be co-estimated online [14], [12]. Leutenegger et al. [16] describe a *tightly* coupled approach in which the robot trajectory and sparse 3D landmarks are estimated in a joint optimization problem using inertial error terms as well as the reprojection error of the landmarks in the camera image. This is done in a windowed bundle adjustment approach over a set of keyframe images and a temporal inertial measurement window. Similarly, in [19] the authors estimate the trajectory in an IMU-driven filtering framework using the reprojection error of 3D landmarks as measurement updates. Instead of adding the landmarks to the filter state, they immediately

marginalize them out using a nullspace decomposition, thus leading to a small filter state size.

In the present paper we propose a visual-inertial odometry framework which combines and extends several of the above mentioned approaches. While targeting a simple and consistent approach and avoiding ad-hoc solutions, we adapt the structure of the standard visual-inertial EKF-SLAM formulation [14], [12]. The following keypoints are integrated into the proposed framework:

- Point features are parametrized by a bearing vector and a distance parameter with respect to the current frame. A suitable σ -Algebra is used for deriving the corresponding dynamics and performing filtering operations.
- Multilevel patch features are directly tracked within the EKF, whereby the intensity errors are used as innovation terms during the update step.
- A QR-decomposition is employed in order to reduce the high dimensional error terms and thus keep the Kalman update computationally tractable.
- A purely robocentric representation of the *full* filter state is employed. The camera extrinsics as well as the additive IMU biases are also co-estimated.

Together this yields a *fully robocentric* and *direct* monocular visual-inertial odometry framework which can be run real-time on a single standard CPU core. In several experiments on real data we show its reliable and accurate tracking performance while exhibiting a high robustness against fast motions and various disturbances. The framework is implemented in c++ and is available as open-source software [2].

II. FILTER SETUP

A. Overall Filter Structure and State Parametrization

The overall structure of the filter is derived from the one employed in [14], [12]: The inertial measurements are used to propagate the state of the filter, while the visual information is taken into account during the filter update steps. As a fundamental difference we make use of a fully robocentric representation of the filter state which can be seen as an adaptation of [4] (which is vision-only). One advantage of this formulation is that problems with unobservable states can inherently be avoided and thus the consistency of the estimates can be improved. On the other hand noise from the gyroscope will affect all states that need to be rotated during the state propagation (see section II-B). However, since the gyroscope noise is relatively small and because most states are observable this does not represent a significant issue.

Three different coordinate frames are used throughout the paper: the inertial world coordinate frame, \mathcal{I} , the IMU fixed coordinate frame, \mathcal{B} , as well as the camera fixed coordinate frame, \mathcal{V} . For tracking N visual features, we use the following filter state:

$$\mathbf{x} := (\mathbf{r}, \mathbf{v}, \mathbf{q}, \mathbf{b}_f, \mathbf{b}_\omega, \mathbf{c}, \mathbf{z}, \boldsymbol{\mu}_0, \dots, \boldsymbol{\mu}_N, \rho_0, \dots, \rho_N), \quad (1)$$

with:

- \mathbf{r} : robocentric position of IMU (expressed in \mathcal{B}),
- \mathbf{v} : robocentric velocity of IMU (expressed in \mathcal{B}),

- \mathbf{q} : attitude of IMU (map from \mathcal{B} to \mathcal{I}),
- \mathbf{b}_f : additive bias on accelerometer (expressed in \mathcal{B}),
- \mathbf{b}_ω : additive bias on gyroscope (expressed in \mathcal{B}),
- \mathbf{c} : translational part of IMU-camera extrinsics (expressed in \mathcal{B}),
- \mathbf{z} : rotational part of IMU-camera extrinsics (map from \mathcal{B} to \mathcal{V}),
- $\boldsymbol{\mu}_i$: bearing vector to feature i (expressed in \mathcal{V}),
- ρ_i : distance parameter of feature i .

The generic parametrization for the distance d_i of a feature i is given by the mapping $d_i = d(\rho_i)$ (with derivative $d'(\rho_i)$). In the context of this work we mainly tested the inverse distance parametrization, $d(\rho_i) = 1/\rho_i$. The investigation of further parametrization will be part of future work.

Rotations ($\mathbf{q}, \mathbf{z} \in SO(3)$) and unit vectors ($\boldsymbol{\mu}_i \in S^2$) are parametrized by following the approach of Hertzberg et al. [10]. This is required in order to perform operations like computing differences or derivatives as well as representing the uncertainty of the state in a minimal manner. For parametrizing unit vectors we employ rotations as underlying representation, whereby we define a \boxminus -operator which returns a difference between two unit vectors within a 2D linear subspace. The advantage of this parametrization is that the tangent space can be easily computed (which is used for defining the \boxminus -operator).

By using the combined bearing vector and distance parameterization, features can be initialized in an *undelayed* manner, i.e., the features are integrated into the filter at detection. The distance of a feature is initialized with a fixed value or, if sufficiently converged, with an estimate of the current average scene distance. The corresponding covariance is set to a very large value. In comparison to other parameterizations we do not over-parametrize the 3D feature location estimates, whereby each feature corresponds to 3 columns in the covariance matrix of the state (2 for the bearing vector and 1 for the distance parameter). This also avoids the need for re-parameterization [22].

B. State Propagation

Based on the proper acceleration measurement, $\tilde{\mathbf{f}}$, and the rotational rate measurement, $\tilde{\boldsymbol{\omega}}$, the evaluation of the IMU driven state propagation results in the following set of continuous differential equations (the superscript \times denotes the skew symmetric matrix of a vector):

$$\dot{\mathbf{r}} = -\hat{\boldsymbol{\omega}}^\times \mathbf{r} + \mathbf{v} + \mathbf{w}_r, \quad (2)$$

$$\dot{\mathbf{v}} = -\hat{\boldsymbol{\omega}}^\times \mathbf{v} + \hat{\mathbf{f}} + \mathbf{q}^{-1}(\mathbf{g}), \quad (3)$$

$$\dot{\mathbf{q}} = -\mathbf{q}(\hat{\boldsymbol{\omega}}), \quad (4)$$

$$\dot{\mathbf{b}}_f = \mathbf{w}_{bf}, \quad (5)$$

$$\dot{\mathbf{b}}_\omega = \mathbf{w}_{b\omega}, \quad (6)$$

$$\dot{\mathbf{c}} = \mathbf{w}_c, \quad (7)$$

$$\dot{\mathbf{z}} = \mathbf{w}_z, \quad (8)$$

$$\dot{\boldsymbol{\mu}}_i = \mathbf{N}^T(\boldsymbol{\mu}_i) \hat{\boldsymbol{\omega}}_{\mathcal{V}} - \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{N}^T(\boldsymbol{\mu}_i) \frac{\dot{\mathbf{v}}_{\mathcal{V}}}{d(\rho_i)} + \mathbf{w}_{\mu,i}, \quad (9)$$

$$\dot{\rho}_i = -\boldsymbol{\mu}_i^T \dot{\mathbf{v}}_{\mathcal{V}} / d'(\rho_i) + w_{\rho,i}, \quad (10)$$

where $N^T(\mu)$ linearly projects a 3D vector onto the 2D tangent space around the bearing vector μ , with the bias corrected and noise affected IMU measurements:

$$\hat{\mathbf{f}} = \tilde{\mathbf{f}} - \mathbf{b}_f - \mathbf{w}_f, \quad (11)$$

$$\hat{\boldsymbol{\omega}} = \tilde{\boldsymbol{\omega}} - \mathbf{b}_\omega - \mathbf{w}_\omega, \quad (12)$$

and with the camera linear velocity and rotational rate:

$$\hat{\mathbf{v}}_V = \mathbf{z}(\mathbf{v} + \boldsymbol{\omega}^\times \mathbf{c}), \quad (13)$$

$$\hat{\boldsymbol{\omega}}_V = \mathbf{z}(\hat{\boldsymbol{\omega}}). \quad (14)$$

Furthermore, \mathbf{g} is the gravity vector expressed in the world coordinate frame, and the terms of the form \mathbf{w}_* are white Gaussian noise processes. The corresponding covariance parameters can either be taken from the IMU specifications or have to be tuned manually. Using an appropriate Euler forward integration scheme, i.e., using the \boxplus -operator where appropriate, the above time continuous equation can be transformed into a set of discrete prediction equations which are used during the prediction of the filter state [10].

Please note that the derivatives of bearing vectors and rotations lie within 2D and 3D vector spaces, respectively. This is required for achieving a minimal and consistent representation of the filter state and covariance.

C. Filter Update

For every captured image we perform a state update. We assume that we know the intrinsic calibration of the camera and can therefore compute the projection of a bearing μ to the corresponding pixel coordinate $\mathbf{p} = \pi(\mu)$. As will be described in section III-B, we derive a 2D linear constraint, $b_i(\pi(\hat{\mu}_i))$, for each feature i which is predicted to be visible in the current frame with bearing vector $\hat{\mu}_i$. This linear constraint encompasses the intensity errors associated with a specific feature and can be directly employed as innovation term within the Kalman update (affected by additive discrete Gaussian pixel intensity noise \mathbf{n}_i):

$$\mathbf{y}_i = b_i(\pi(\hat{\mu}_i)) + \mathbf{n}_i, \quad (15)$$

together with the Jacobian:

$$\mathbf{H}_i = \mathbf{A}_i(\pi(\hat{\mu}_i)) \frac{d\pi}{d\mu}(\hat{\mu}_i). \quad (16)$$

By stacking the above terms for all visible features we can directly perform a standard EKF update. However, if the initial guess for a certain bearing vector $\hat{\mu}_i$ has a large uncertainty the update will potentially fail. This typically occurs if features get newly initialized and exhibit a large distance uncertainty. In order to avoid this issue we improve the initial guess for a bearing vector with large uncertainty by performing a patch based search of the feature (section III-B). This basically improves the linearization point of the EKF by using the bearing vector obtained from the patch search $\bar{\mu}_i$ for evaluating the terms in eqs. (15) and (16). Please note that the EKF update equations have to be slightly adapted in order to account for the altered linearization point. A similar alternative would be to directly employ an iterative EKF.

In order to account for moving objects or other disturbances, a simple Mahalanobis based outlier detection is implemented within the update step. It compares the obtained innovation with the predicted innovation covariance and rejects the measurement whenever the weighted norm exceeds a certain threshold. This method inherently takes into account the covariance of the state and measurements. For instance it also considers the image gradients and thereby tends to reject gradient-less image patches easier.

III. MULTILEVEL PATCH FEATURE HANDLING

Along the lines of other visual-inertial EKF approaches ([14], [12]) we fully integrate visual features into the state of the Kalman filter (see also section II-A). Within the prediction step the new locations of the multilevel patch features are estimated by considering the IMU-driven motion model (eq. (9)). Especially if the calibration of the extrinsics and the feature distance parameters have converged, this yields high quality predictions for the feature locations. Additionally, the covariance of the predicted pixel location can be easily computed and the computational effort of a possible pre-alignment strategy can be adapted accordingly. The subsequent update step computes an innovation term by evaluating the discrepancy between the projection of the multilevel patch into the image frame and the image itself. Considering the cross-correlation between the states the EKF spreads the resulting corrections throughout the filter state. In the following the different steps and algorithms involving feature handling are discussed in more details. The overall workflow for a single feature is depicted in fig. 1.

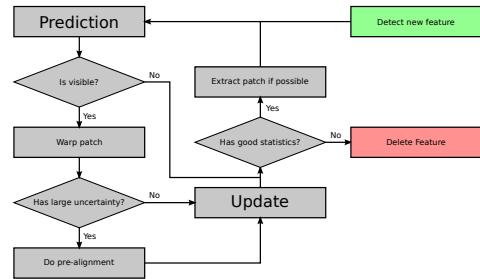


Fig. 1. Overview on the workflow of a feature in the filter state. The heuristics for adding and removing features are adapted to the total number of possible features.

A. Structure and Warping

For a given image pyramid (factor 2 down-sampling) and a given bearing vector μ a multilevel patch is obtained by extracting constant size (here 8x8 pixels) patches, P_l , for each image level l at the corresponding pixel coordinate $\mathbf{p} = \pi(\mu)$. The advantage is that tracking such features is robust against bad initial guesses and image blur. Furthermore such patch features allow a *direct* intensity error feedback into the filter. In comparison to reprojection error based algorithms this allows to formulate a more accurate error model which

inherently takes into account the texture of the tracked image patch. For instance it also enables the use of edge features, whereby the gained information would be along the perpendicular to the edge.

By tracking two additional bearing vectors within the patch, we can compute an affine warping matrix $\mathbf{W} \in \mathbb{R}^{2 \times 2}$ in order to account for the local distortion of the patches between subsequent images. We assume that the distance of the feature is large w.r.t. the size of the patch and can thus choose the normal of the patches to point towards the center of the camera. Also, when a feature was successfully tracked within a frame, the multilevel patch is re-extracted in order to avoid the accumulation of errors.

B. Alignment Equations and QR-decomposition

Throughout the framework we make use of intensity errors in order to pre-align features or update the filter state. For a given image pyramid with images I_l and a given multilevel patch feature (with coordinates \mathbf{p} and patches P_l) the following intensity errors can be evaluated for image level l and patch pixel p_j :

$$e_{l,j} = P_l(\mathbf{p}_j) - I_l(\mathbf{p}_j s_l + \mathbf{W}\mathbf{p}_j) - m, \quad (17)$$

where the scalar $s_l = 0.5^l$ accounts for the down-sampling between the images of the image pyramid. Furthermore, by subtracting the mean intensity error m we can account for inter-frame illumination changes.

For regular patch alignment, the squared error terms of eq. (17) can be summed over all image levels and patch pixels and combined into a single Gauss-Newton optimization in order to find the optimal patch coordinates. However, the direct use of such a large number of error terms within an EKF would make it computationally intractable. In order to tackle this issue we apply a QR-decomposition on the linear equation system resulting from stacking all error terms in eq. (17) together for given estimated coordinates $\hat{\mathbf{p}}$:

$$\bar{\mathbf{b}}(\hat{\mathbf{p}}) = \bar{\mathbf{A}}(\hat{\mathbf{p}})\delta\mathbf{p}, \quad (18)$$

where $\bar{\mathbf{A}}(\hat{\mathbf{p}})$ can be computed based on the patch intensity gradients. Independent of the rank of the matrix $\bar{\mathbf{A}}(\hat{\mathbf{p}})$, the QR-decomposition of $\bar{\mathbf{A}}(\hat{\mathbf{p}})$ can be used to obtain an equivalent reduced linear equation system:

$$\mathbf{b}(\hat{\mathbf{p}}) = \mathbf{A}(\hat{\mathbf{p}})\delta\mathbf{p}, \quad (19)$$

with $\mathbf{A}(\hat{\mathbf{p}}) \in \mathbb{R}^{2 \times 2}$ and $\mathbf{b}(\hat{\mathbf{p}}) \in \mathbb{R}^2$. Since we assume that the additive noise magnitude on the intensities is equal for every patch pixel we can leave it out of the above derivations (it will remain constant for every entry).

One interesting remark is, that due to the scaling factor s_l in eq. (17), error terms for higher image levels will have a weaker corrective influence on the filter state or the patch alignment. On the other hand, their increased robustness w.r.t. image blur or bad initial alignment strongly increases the robustness of the overall alignment method for multilevel patch features.

C. Feature Detection and Removal

The detection of new features is based on a standard fast corner detector which provides a large amount of candidate feature locations. After removing candidates which are close to current tracked features, we compute an adapted Shi-Tomasi score for selecting new features which will be added to the state. The adapted Shi-Tomasi score basically considers the combined Hessian on multiple image levels, instead of only a single level. It directly approximates the Hessian of the above gradient matrix with $\mathbf{H} = \bar{\mathbf{A}}^T(\hat{\mathbf{p}})\bar{\mathbf{A}}(\hat{\mathbf{p}})$ and extracts the minimal eigenvalue. The advantage is that a high score is directly correlated with the alignment accuracy of the corresponding multilevel patch feature. Instead of returning the minimal eigenvalue, the method can return other eigenvalue based scores like the 1- or 2-norm. This could be useful in environments with scarce corner data, whereby the presented filter could be complemented by available edge-shaped features. Finally, the detection process is also coupled to a bucketing technique in order to achieve a good distribution of the features within the image frame.

Due to the fact that we can only track a limited number of features in the EKF, we have to implement a landmark management system to ensure that only reliable landmarks are inserted and kept in the filter state. Here, we fall back to heuristic methods, where we compute quality scores in order to decide whether a feature should be kept or not. The overall idea is to evaluate a local (only last few frames) and a global (how good was the feature tracked since it has been detected) quality score and remove the features below a certain threshold. Using an adaptive threshold we can control the total amount of features which are currently in the frame.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

The data for the experiments were recorded with the VI-Sensor [20], equipped with two time-synchronized, global-shutter, wide-VGA 1/3 inch imagers in a fronto-parallel stereo configuration. The cameras are equipped with lenses with a diagonal field of view of 120 degrees and are factory-calibrated by the manufacturer for a standard pinhole projection model and a radial-tangential distortion model. The imagers are hardware time-synchronized to the IMU to ensure mid-exposure IMU triggering. In the context of this work only the image stream from one camera is required.

Ground truth is provided through an external motion capture system for the pose of the sensor. The rate of the IMU measurements is 200 Hz and the image frame rate is 20 Hz. The employed IMU is an industrial-grade ADIS 16448, with an angular random walk of 0.66 deg/√Hz and a velocity random walk of 0.11 m/s/√Hz. The maximal number of features in the state is set to 50 and the algorithm is run using image pyramids with 4 levels. Whenever possible, covariance parameters are selected based on hardware specifications. Strong tuning was not necessary, and the framework works well for a large range of parameters. The initial IMU-camera extrinsics are only roughly guessed (the translation is set to

zero), and the initial inverse distance parameter for a feature is set to 0.5 m^{-1} with a standard deviation of 1 m^{-1} . A screenshot of the running framework is depicted in fig. 2.



Fig. 2. Screenshot of the running visual-inertial odometry framework. The 2σ uncertainty ellipses of the predicted feature locations are in yellow, whereby only features which are newly initialized (stretched ellipses) and features which re-enter the frame have a significant uncertainty. Green points are the locations after the update step. Green numbers are the tracking counts (1 for newly initialized features). In the top left a virtual horizon is depicted.

B. Experiment with Slow Motions

An experiment with slow to medium fast hand-held motions of about 1 min was carried out to evaluate the performance of the framework with different numbers of total features (from 10 to 50 in steps of 10). The performance was assessed by computing the relative position error w.r.t. the traveled distance [9]. Furthermore we compared the obtained results to a batch optimization framework along the lines of [16]. Figure 3 depicts the extracted relative error values. The achieved performance tends to be similar to the one of the batch optimization framework and often achieves slightly higher accuracy. While these results depend on the specific dataset and parameter tuning, we also have to mention that the relatively high rotational motion (average of around 1.5 rad/s) favors approaches which can handle arbitrarily short feature tracks. Given the *undelayed* initialization of feature within our approach, the resulting filter is able to extract visual information from a feature's second observation onwards.

Surprisingly, the performance was relatively independent of the total amount of tracked features. A significant drop in accuracy could only be observed with feature counts below 20. This observation can have different reasons. One could be the type of sensor motions with relatively high rotational rates, which can lead to more bad features or outliers. Another point is also that our approach considers $256 = 4 \times 8 \times 8$ intensity errors per tracked features and thus we cannot directly compare to standard feature tracking based visual odometry frameworks, which typically require much higher feature counts. More in-depth evaluation of this effect will be part of future work. The timings of the proposed framework are listed in table I for a single core

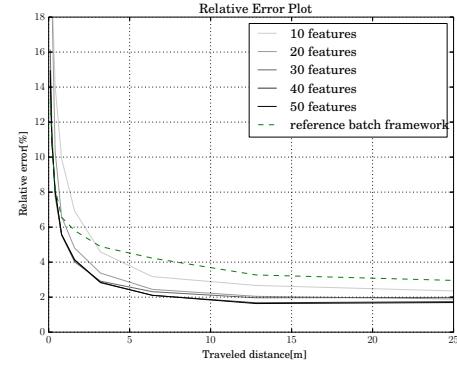


Fig. 3. Gray lines are the relative errors of the presented approach, where the darkest lines corresponds to 50 features and the brightest line to 10 features respectively. The dashed green line represents the performance of the reference batch optimization framework.

TABLE I
TIMINGS OF PRESENTED APPROACH PER PROCESSED IMAGE

Tot. Features	10	20	30	40	50
Timing [ms]	6.65	10.50	14.87	21.48	29.72

of an Intel i7-2760QM. The setup with 50 features uses an average processing time of 29.72 ms per processed image and can thus easily be run at 20 Hz .

C. Experiment with Fast Motions

Here, we evaluate the robustness of the proposed approach w.r.t. very fast motions. We recorded a hand-held dataset with mean rotational rate of around 3.5 rad/s and with peaks of up to 8 rad/s . The motion capture system exhibited a relative high number of bad tracking, whereby we filtered them out as good as possible. We investigate the tracking performance of the attitude and of the robocentric velocities, where the corresponding estimates with 3σ -bounds are plotted in figs. 4 and 5 respectively. It can clearly be seen that the estimates nicely fit the ground truth data from the motion capture. As known from previous work the inclination angles and the robocentric velocities of visual-inertial setups are fully observable [17], and we can nicely observe the initial decrease of the corresponding covariance (especially when the system gets excited). On the other hand the yaw angle is unobservable and drifts slowly with time.

Figures 6 and 7 depict the estimation of the calibration parameters. Again, the estimates together with their 3σ -bounds are plotted. Depending on the excitation of the system the estimated values converge relatively quickly. It can be observed, that the translational term of the IMU-camera calibration requires a lot of rotational motion in order to converge appropriately. For the presented experiment, the accelerometer bias exhibits the worse convergence rate but is still within a reasonable range.

Furthermore, we also observed a divergence mode for the presented approach. It can occur when the velocity estimate diverge, e.g., due to missing motion or too many outliers. The problem is then, that the filter attempts to minimize the effect of the erroneous velocity on the bearing vectors by setting the distance of the features to infinity. This again lowers any corrective effect on the diverging velocity resulting in further divergence. All in all this was very rarely observed for regular usage, especially if the system was properly excited at the start.

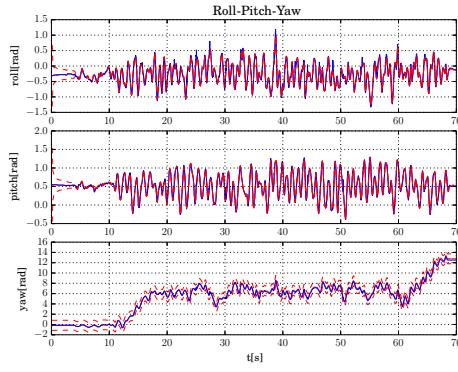


Fig. 4. Euler angle estimates. Red: estimate, blue: motion capture, red dashed: 3σ -bound. Only the yaw angle is not observable and exhibits a growing covariance. The inclination angles (roll and pitch) exhibit a high quality tracking accuracy.

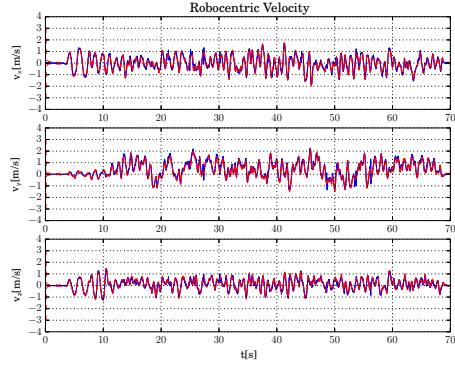


Fig. 5. Velocity estimates. Red: estimate, blue: motion capture, red dashed: 3σ -bound. The robo-centric velocity is fully observable and thus exhibits a bounded uncertainty. It very nicely tracks the reference from the motion capture system (and probably also exhibits a higher precision).

D. Flying Experiments

Implementing the framework on-board a UAV with a forward oriented visual-inertial sensor, we also performed

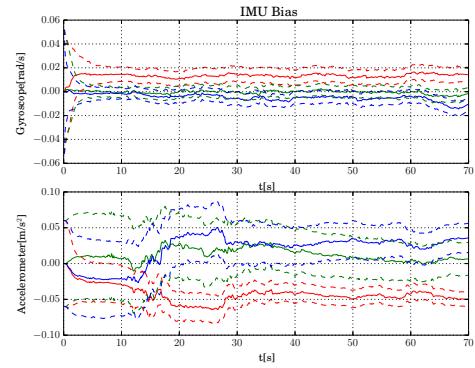


Fig. 6. Estimated IMU biases. Top: gyroscope bias (red: x, blue: y, green: z), bottom: accelerometer bias (red: x, blue: y, green: z). The gyroscope biases exhibit a better convergence than the accelerometer biases, probably due to the more direct link of rotational rates to visual errors.

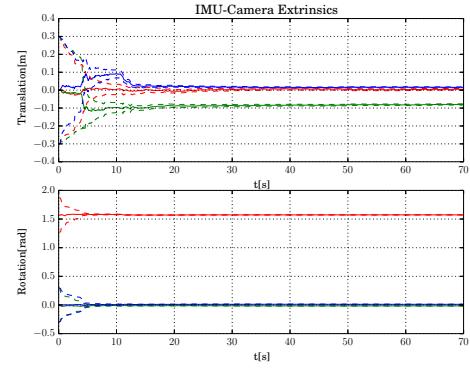


Fig. 7. Estimated IMU-camera extrinsics. Top: translation (red: x, blue: y, green: z), bottom: orientation (red: yaw, blue: pitch, green: roll). Especially when sufficiently excited, the estimates converge quickly. The reached values correspond approximately to the ones obtained from an offline calibration.

preliminary experiments on a real robot. The special aspect here is that the visual-inertial odometry framework was initialized on the ground without any previous calibration motions, i.e. the calibration parameters had to converge during take-off. The output of the filter was directly used for feedback control of the UAV. Figure 8 depicts the estimated position output of the framework during take-off, flying and landing. If compared to the motion capture system the filter exhibits a certain offset which can be mainly attributed to the online calibration of the filter.

V. CONCLUSION

In this paper we presented a visual-inertial filtering framework which uses direct intensity errors as visual measurements within the extended Kalman filter update. By choosing

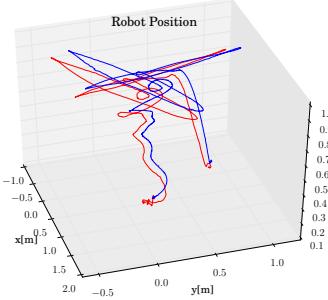


Fig. 8. Estimated trajectory (red) on-board a UAV compared to groundtruth (blue) from the motion capture system. During take-off, flying, and landing the output of the filter is used to stabilize and control the UAV. Calibration is performed online.

a fully robocentric representation of the filter state together with a numerically minimal bearing/distance representation of features, we avoid major consistency problems while exhibiting accurate tracking performance and high robustness. Especially in difficult situations with very fast motions or outliers the presented approach manages to keep track of the state with only minor drift of the yaw and position estimates. The framework can be run on-board a UAV with a feature count of 50 at a framerate of 20 Hz and was used to stabilize the flight of a UAV from take-off to landing.

Future work will include more extensive evaluation of the multilevel patch features in context of intensity error based visual-inertial odometry frameworks. Furthermore we would also like to try to extend the online calibration in order to include the camera intrinsics. Also, the framework could be relatively easily adapted in order to handle multiple cameras. This could improve the filter performance, especially for cases with lack of translational motion. Another option to avoid divergence would be to use some heuristics based methods in order to detect such modes and to add zero-velocity pseudo-measurements in order to stabilize the filter. A detailed observability analysis could also be performed, where the dependency of unobservable modes w.r.t. sensor motions would be of high interest.

REFERENCES

- [1] C. Audras, A. I. Comport, M. Meillard, and P. Rives, "Real-time dense appearance-based SLAM for RGB-D sensors," in *Australasian Conf. on Robotics and Automation*, 2011.
- [2] M. Bloesch, S. Omari, and A. Jaeger, "ROVIO," 2015. [Online]. Available: <https://github.com/ethz-asl/rovio>
- [3] J. A. Castellanos, J. Neira, and J. D. Tardos, "Limits to the consistency of EKF-based SLAM," in *5th IFAC Symp. on Intelligent Autonomous Vehicles*, 2004.
- [4] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point RANSAC for EKF-based Structure from Motion," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.
- [5] A. J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," in *IEEE Int. Conference on Computer Vision*, 2003.
- [6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conf. on Computer Vision*, 2014.
- [7] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO : Fast Semi-Direct Monocular Visual Odometry," in *IEEE Int. Conf. on Robotics and Automation*, 2014.
- [8] S. Gehrig, F. Eberli, and T. Meyer, "A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching," *Computer Vision Systems*, vol. 5815, pp. 134–143, 2009.
- [9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. October, pp. 1231–1237, 2013.
- [10] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2011.
- [11] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and improvement of the consistency of extended Kalman filter based SLAM," in *IEEE Int. Conf. on Robotics and Automation*, May 2008.
- [12] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.
- [13] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *IEEE Int. Conf. on Robotics and Automation*, May 2001.
- [14] J. Kelly and G. S. Sukhatme, "Visual-Inertial Sensor Fusion: Localization, Mapping, and Sensor-to-Sensor Self-calibration," *Int. Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, Nov. 2011.
- [15] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE Int. Conf. on Robotics and Automation*, IEEE, May 2013.
- [16] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.
- [17] M. Li and a. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.
- [18] J. Montiel, J. Civera, and A. Davison, "Unified Inverse Depth Parametrization for Monocular SLAM," in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, Aug. 2006.
- [19] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *IEEE Int. Conf. on Robotics and Automation*, 2007.
- [20] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM," in *IEEE Int. Conf. on Robotics and Automation*, 2014.
- [21] S. Shen, M. N. Mulgaonkar, Y., and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *International Symposium on Experimental Robotics*, 2014.
- [22] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *International Journal of Computer Vision*, vol. 97, pp. 339–368, 2012.
- [23] A. Stelzer, H. Hirschmüller, and M. Gorner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 381–402, Feb. 2012.
- [24] S. Weiss, M. Achterlik, S. Lynen, L. Kneip, M. Chli, and R. Siegwart, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.
- [25] D. Wooden, M. Malchano, K. Blankepoor, A. Howard, A. A. Rizzi, and M. Raibert, "Autonomous navigation for BigDog," in *IEEE Int. Conf. on Robotics and Automation*, 2010.

8.7 Building a Voxel Map

RWB: Need to rewrite this section.

Associated with every voxel in the rolling voxel world is a probability that the voxel is occupied by an object. The probability that a voxel is occupied at time k will be a function of the output of the vision sensor at times $k, k-1, \dots, 0$. Let $X_\ell[k]$ be a binary random variable representing the state of the ℓ^{th} voxel at time k , with the following possible values:

$$X_\ell[k] = \begin{cases} O & \text{meaning voxel } \ell \text{ is occupied with an object at time } k \\ E & \text{meaning voxel } \ell \text{ is empty at time } k \end{cases}.$$

Let $Z_\ell[k]$ be a binary random variable representing the measurement of the ℓ^{th} voxel at time k with the following possible value:

$$Z_\ell[k] = \begin{cases} O & \text{meaning the sensor measures an object in voxel } \ell \text{ at time } k \\ E & \text{meaning the sensor does not measure an object in voxel } \ell \text{ at time } k \end{cases}.$$

Let

$$\pi_\ell[k] = P(X_\ell[k] = O | Z_\ell[k : 0]).$$

Using Bayes law $P(A|B, C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}$ we get that

$$\begin{aligned} P(X_\ell[k] = O | Z_\ell[k : 0]) &= \frac{P(Z_\ell[k] | X_\ell[k] = O, Z_\ell[k-1 : 0])P(X_\ell[k] = O | Z_\ell[k-1 : 0])}{P(Z_\ell[k] | Z_\ell[k-1 : 0])} \\ &= \frac{P(Z_\ell[k] | X_\ell = O)P(X_\ell[k] = O | Z_\ell[k-1 : 0])}{P(Z_\ell[k] | Z_\ell[k-1 : 0])}, \end{aligned}$$

where

$$\begin{aligned} P(Z_\ell[k] | Z_\ell[k-1 : 0]) &= P(Z_\ell[k] | X_\ell[k] = O, Z_\ell[k-1 : 0])P(X_\ell[k] = O | Z_\ell[k-1 : 0]) \\ &\quad + P(Z_\ell[k] | X_\ell = E, Z_\ell[k-1 : 0])P(X_\ell[k] = E | Z_\ell[k-1 : 0]) \\ &= P(Z_\ell[k] | X_\ell[k] = O)P(X_\ell[k] = O | Z_\ell[k-1 : 0]) \\ &\quad + P(Z_\ell[k] | X_\ell[k] = E)P(X_\ell[k] = E | Z_\ell[k-1 : 0]), \end{aligned}$$

and where we have used the assumption that the measurement at time k only depends on the state at time k . Define the *a priori* probability as

$$\bar{\pi}_\ell[k] \stackrel{\Delta}{=} P(X_\ell[k] = O | Z_\ell[k-1 : 0]),$$

and also define the probability of detection and probability of false alarm as

$$\begin{aligned} p_D &\stackrel{\Delta}{=} P(Z_\ell[k] = O | X_\ell[k] = O) \\ p_{FA} &\stackrel{\Delta}{=} P(Z_\ell[k] = O | X_\ell[k] = E), \end{aligned}$$

This section is inspired by a similar discussion in (Thrun, 2006).

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2006

then the probability update law becomes

$$\pi_\ell[k] = \begin{cases} \frac{p_D \bar{\pi}_\ell[k]}{p_D \bar{\pi}_\ell[k] + p_{FA}(1 - \bar{\pi}_\ell[k])} & Z_k = O \\ \frac{(1 - p_D) \bar{\pi}_\ell[k]}{(1 - p_D) \bar{\pi}_\ell[k] + (1 - p_{FA})(1 - \bar{\pi}_\ell[k])} & Z_k = E \end{cases}.$$

The *a priori* probability of occupancy is given by

$$\begin{aligned} \bar{\pi}_\ell[k] &= P(X_\ell[k] = O | Z_\ell[k-1 : 0]) \\ &= P(X_\ell[k] = O | X_\ell[k-1] = O, Z_\ell[k-1 : 0])P(X_\ell[k-1] = O | Z_\ell[k-1 : 0]) \\ &\quad + P(X_\ell[k] = O | X_\ell[k-1] = E, Z_\ell[k-1 : 0])P(X_\ell[k-1] = E | Z_\ell[k-1 : 0]) \\ &= P(X_\ell[k] = O | X_\ell[k-1] = O)\pi_\ell[k-1] \\ &\quad + P(X_\ell[k] = O | X_\ell[k-1] = E)(1 - \pi_\ell[k-1]), \end{aligned}$$

where we have used the assumption that state transitions do not depend on the measurements. Define the static and transition probabilities as

$$\begin{aligned} p_s &= P(X_\ell[k] = O | X_\ell[k-1] = O) \\ p_t &= P(X_\ell[k] = O | X_\ell[k-1] = E), \end{aligned}$$

then the *a priori* probability is given as

$$\bar{\pi}_\ell[k] = p_s \pi_\ell[k-1] + p_t(1 - \pi_\ell[k-1]).$$

If voxel ℓ is not measured at time k , then

$$P(X_\ell[k] = O | Z_\ell[0 : k]) = P(X_\ell[k] = O | Z_\ell[0 : k-1]),$$

which implies that

$$\pi_\ell[k] = \bar{\pi}_\ell[k].$$

In summary we have the following result.

Lemma 8.7.1 Define the probability of detection p_D , the probability of false alarm p_{FA} , the probability of a static voxel p_s , and the probability of a transitioning voxel p_t as

$$\begin{aligned} p_D &= P(Z_\ell[k] = O | X_\ell[k] = O) \\ p_{FA} &= P(Z_\ell[k] = O | X_\ell[k] = E), \\ p_s &= P(X_\ell[k] = O | X_\ell[k-1] = O) \\ p_t &= P(X_\ell[k] = O | X_\ell[k-1] = E). \end{aligned}$$

Then the optimal Bayes filter for the ℓ^{th} voxel cell is given by

$$\begin{aligned} \bar{\pi}_\ell[k] &= p_s \pi_\ell[k-1] + p_t(1 - \pi_\ell[k-1]) \\ \pi_\ell[k] &= \begin{cases} \frac{p_D \bar{\pi}_\ell[k]}{p_D \bar{\pi}_\ell[k] + p_{FA}(1 - \bar{\pi}_\ell[k])} & Z_\ell[k] = O \\ \frac{(1 - p_D) \bar{\pi}_\ell[k]}{(1 - p_D) \bar{\pi}_\ell[k] + (1 - p_{FA})(1 - \bar{\pi}_\ell[k])} & Z_\ell[k] = E \\ \bar{\pi}_\ell[k] & \text{no measurement} \end{cases}. \end{aligned}$$

9

Flight Control using Visual Servoing

This chapter will describe vision-based flight control based on visual servoing concepts. The basic idea is that the motion of the multi-rotor is commanded based on the desired motion of features in the image plane.

9.1 Feature motion related to vehicle motion

In this section we derive an expression that relates the velocity and angular velocity of the multirotor, and the motion of features on the image plane. We assume calibrated normalized pixel coordinates

$$\lambda_f \boldsymbol{\epsilon}_f = \lambda_f \begin{pmatrix} \epsilon_x \\ \epsilon_y \end{pmatrix} = p_z \begin{pmatrix} \frac{p_x}{p_z} \\ \frac{p_y}{p_z} \end{pmatrix},$$

where

$$\mathbf{p}_{f/c}^c = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \mathbf{p}_{f/i}^c - \mathbf{p}_{c/i}^c$$

is the location of the feature relative to the camera, expressed in the camera frame. The following lemma relates the translational and rotational motion of the camera to the pixel motion.

Lemma 9.1.1 *If the feature f is moving relative to the camera with velocity $\mathbf{v}_{f/c}^c$, and suppose that the camera is rotating with angular velocity $\boldsymbol{\omega}_{c/i}^c$, then the pixel motion of the feature is given by*

$$\dot{\boldsymbol{\epsilon}}_f = \lambda_f V(\boldsymbol{\epsilon}_f) \mathbf{v}_{c/f}^c + \boldsymbol{\Omega}(\boldsymbol{\epsilon}_f) \boldsymbol{\omega}_{c/i}^c, \quad (9.1)$$

where

$$V(\boldsymbol{\epsilon}_f) \triangleq \begin{pmatrix} -1 & 0 & \epsilon_x \\ 0 & -1 & \epsilon_y \end{pmatrix} \quad (9.2)$$

$$\boldsymbol{\Omega}(\boldsymbol{\epsilon}_f) \triangleq \begin{pmatrix} \epsilon_x \epsilon_y & -(1 + \epsilon_x^2) & \epsilon_y \\ (1 + \epsilon_y^2) & -\epsilon_x \epsilon_y & -\epsilon_x \end{pmatrix}. \quad (9.3)$$

Proof: The position of the feature relative to the camera, expressed in the camera frame is

$$\mathbf{p}_{f/c}^c = R_i^c(\mathbf{p}_{f/i}^i - \mathbf{p}_{c/i}^i).$$

Differentiating with respect to time gives

$$\begin{aligned}\dot{\mathbf{p}}_{f/c}^c &= \dot{R}_i^c(\mathbf{p}_{f/i}^i - \mathbf{p}_{c/i}^i) + R_i^c(\dot{\mathbf{p}}_{f/i}^i - \dot{\mathbf{p}}_{c/i}^i) \\ &= -[\boldsymbol{\omega}_{i/c}^c]_x R_i^c(\mathbf{p}_{f/i}^i - \mathbf{p}_{c/i}^i) + R_i^c(\mathbf{v}_{f/i}^i - \mathbf{v}_{c/i}^i) \\ &= [\boldsymbol{\omega}_{c/i}^c]_x \mathbf{p}_{f/c}^c + \mathbf{v}_{f/i}^c - \mathbf{v}_{c/i}^c \\ &= \begin{pmatrix} v_{fx} - \omega_{cy} p_z + \omega_{cz} p_y \\ v_{fy} - \omega_{cz} p_x + \omega_{cx} p_z \\ v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x \end{pmatrix},\end{aligned}$$

where $\mathbf{v}_{f/c}^i = (v_{fx}, v_{fy}, v_{ fz})^\top$ and $\boldsymbol{\omega}_{c/i}^i = (\omega_{cx}, \omega_{cy}, \omega_{cz})^\top$.

The optical flow, or pixel motion of the feature is therefore given by

$$\begin{aligned}\dot{\epsilon}_f &= \frac{d}{dt} \begin{pmatrix} \frac{p_x}{p_z} \\ \frac{p_y}{p_z} \\ \frac{p_z}{p_z} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\dot{p}_x}{p_z} - \frac{p_x}{p_z} \frac{\dot{p}_z}{p_z} \\ \frac{\dot{p}_y}{p_z} - \frac{p_y}{p_z} \frac{\dot{p}_z}{p_z} \\ \frac{\dot{p}_z}{p_z} \end{pmatrix} \\ &= \begin{pmatrix} \left(\frac{-v_{fx} - \omega_{cy} p_z + \omega_{cz} p_y}{p_z} - \frac{p_x}{p_z} \frac{-v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x}{p_z} \right) \\ \left(\frac{-v_{fy} - \omega_{cz} p_x + \omega_{cx} p_z}{p_z} - \frac{p_y}{p_z} \frac{-v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x}{p_z} \right) \\ \left(\frac{1}{p_z}(-v_{fx} - \omega_{cy} p_z + \omega_{cz} p_y) - \epsilon_x \frac{-v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x}{p_z} \right) \\ \left(\frac{1}{p_z}(-v_{fy} - \omega_{cz} p_x + \omega_{cx} p_z) - \epsilon_y \frac{-v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x}{p_z} \right) \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{p_z}(-v_{fx} - \omega_{cy} p_z + \omega_{cz} p_y) - \epsilon_x \frac{-v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x}{p_z} \\ \frac{1}{p_z}(-v_{fy} - \omega_{cz} p_x + \omega_{cx} p_z) - \epsilon_y \frac{-v_{ fz} - \omega_{cx} p_y + \omega_{cy} p_x}{p_z} \\ \frac{1}{p_z}(-v_{fx}) - \omega_{cy} + \omega_{cz} \epsilon_y - \epsilon_x \frac{-v_{ fz}}{p_z} + \omega_{cx} \epsilon_x \epsilon_y - \omega_{cy} \epsilon_x^2 \\ \frac{1}{p_z}(-v_{fy}) - \omega_{cz} \epsilon_x + \omega_{cx} - \epsilon_y \frac{-v_{ fz}}{p_z} + \omega_{cx} \epsilon_y^2 - \omega_{cy} \epsilon_x \epsilon_y \end{pmatrix} \\ &= \lambda_f \begin{pmatrix} -1 & 0 & \epsilon_x \\ 0 & -1 & \epsilon_y \end{pmatrix} \begin{pmatrix} v_{fx} \\ v_{fy} \\ v_{ fz} \end{pmatrix} + \begin{pmatrix} \epsilon_x \epsilon_y & -(1 + \epsilon_x^2) & \epsilon_y \\ (1 + \epsilon_y^2) & -\epsilon_x \epsilon_y & -\epsilon_x \end{pmatrix} \begin{pmatrix} \omega_{cx} \\ \omega_{cy} \\ \omega_{cz} \end{pmatrix} \\ &= \lambda_f V(\epsilon_f) \mathbf{v}_{c/f}^c + \boldsymbol{\Omega}(\epsilon_f) \boldsymbol{\omega}_{c/i}^c.\end{aligned}$$

■

Now consider the case where a gimbaled camera is mounted to a multirotor. The relationship between the camera frame, the gimbal frame, and the body frame is illustrated in Figure 9.1, where $\mathbf{d}_{g/b}$ is the position vector of the gimbal frame \mathcal{F}_g with respect to the body frame \mathcal{F}_b , and R_b^g is the rotation matrix that transforms coordinates in the body frame into the gimbal frame. Also, $\mathbf{d}_{c/g}^g$ is the position of the camera relative to the gimbal, expressed in the gimbal frame, and R_g^c is the rotation matrix from the gimbal to the camera frame.

The camera frame $\mathcal{F}_c = \{\mathbf{i}_c, \mathbf{j}_c, \mathbf{k}_c\}$ is defined so that \mathbf{i}_c points to the right in the image, \mathbf{j}_c points down, and \mathbf{k}_c points along the optical

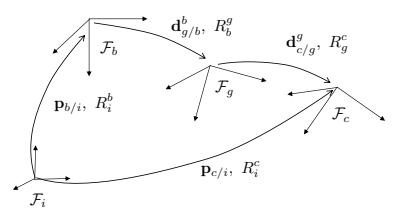


Figure 9.1: The relationship between the body frame, the gimbal frame, and the camera frame. [TOC](#)

axis. The gimbal frame $\mathcal{F}_g = \{\mathbf{i}_g, \mathbf{j}_g, \mathbf{k}_g\}$ is defined so that \mathbf{i}_g points along the optical axis, \mathbf{j}_g points to the right in the image, and \mathbf{k}_g points down in the image. Therefore

$$R_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

The gimbal frame \mathcal{F}_g is obtained by first rotating about the body z -axis by the azimuth angle α to obtain the gimbal-1 frame, and then rotating about the gimbal-1 frame by the elevation angle β to obtain the gimbal frame. Therefore

$$\begin{aligned} R_b^g &= R_{g1}^g R_b^{g1} \\ &= \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} \cos \beta \cos \alpha & -\cos \beta \sin \alpha & \sin \beta \\ \sin \alpha & \cos \alpha & 0 \\ -\sin \beta \cos \alpha & \sin \beta \sin \alpha & \cos \beta \end{pmatrix}. \end{aligned}$$

The follow theorem gives the most general result.

Theorem 9.1.2 *If the inertial position of the feature is moving with velocity $\mathbf{v}_{f/i}^i$, and suppose that the multirotor is moving with velocity $\mathbf{v}_{b/i}^i$ and angular velocity $\boldsymbol{\omega}_{b/i}^b$, and that the gimbal is rotating with angular velocity $\boldsymbol{\omega}_{g/b}^g$, then the pixel motion of the feature satisfies*

$$\dot{\epsilon}_f = G_1(\epsilon_f, \lambda_f, t)(\mathbf{v}_{b/i}^i - \mathbf{v}_{f/i}^i) + G_2(\epsilon_f, \lambda_f, t)\boldsymbol{\omega}_{b/i}^b + G_3(\epsilon_f, \lambda_f)\boldsymbol{\omega}_{g/b}^g. \quad (9.4)$$

where

$$\begin{aligned} G_1(\epsilon_f, \lambda_f, t) &\stackrel{\Delta}{=} \lambda_f V(\epsilon_f) R_g^c R_b^g(t) R_i^b(t) \\ G_2(\epsilon_f, \lambda_f, t) &\stackrel{\Delta}{=} \boldsymbol{\Omega}(\epsilon_f) R_g^c R_b^g(t) - \lambda_f V(\epsilon_f) R_g^c R_b^g(t) \left[\mathbf{d}_{g/b}^b + R_g^b(t) \mathbf{d}_{c/g}^g \right]_\times \\ G_3(\epsilon_f, \lambda_f) &\stackrel{\Delta}{=} \boldsymbol{\Omega}(\epsilon_f) R_g^c - \lambda_f V(\epsilon_f) R_g^c \left[\mathbf{d}_{c/g}^g \right]_\times. \end{aligned}$$

Proof: Using Figure 9.1 we see that the inertial position of the camera frame, expressed in the inertial frame is given by

$$\begin{aligned} \mathbf{p}_{c/i}^i &= \mathbf{p}_{b/i}^i + \mathbf{d}_{g/b}^i + \mathbf{d}_{c/g}^i \\ &= \mathbf{p}_{b/i}^i + R_b^i \mathbf{d}_{g/b}^b + R_b^i R_g^b \mathbf{d}_{c/g}^g. \end{aligned} \quad (9.5)$$

And that the rotation matrices satisfy

$$R_i^c = R_g^c R_b^g R_i^b.$$

Since the vectors are expressed in the inertial frame, we can differentiate with respect to time to obtain

$$\dot{\mathbf{v}}_{c/i}^i = \mathbf{v}_{b/i}^i + \dot{R}_b^i \mathbf{d}_{g/b}^b + R_b^i \dot{\mathbf{d}}_{g/b}^b + \dot{R}_b^i R_g^b \mathbf{d}_{c/g}^g + R_b^i \dot{R}_g^b \mathbf{d}_{c/g}^g + R_b^i R_g^b \dot{\mathbf{d}}_{c/g}^g.$$

Since the position of the gimbal is fixed in the body frame, we have $\dot{\mathbf{d}}_{g/b}^b = 0$, and since the position of the camera is fixed in the gimbal frame we have that $\dot{\mathbf{d}}_{c/g}^g = 0$. From Equation (??) we have that $\dot{R}_b^i = R_b^i(\omega_{b/i})^\times$ and $\dot{R}_g^b = R_g^b(\omega_{g/b})^\times$. Therefore we have

$$\dot{\mathbf{v}}_{c/i}^i = \mathbf{v}_{b/i}^i + R_b^i(\omega_{b/i})^\times \mathbf{d}_{g/b}^b + R_b^i(\omega_{b/i})^\times R_g^b \mathbf{d}_{c/g}^g + R_b^i R_g^b (\omega_{g/b})^\times \mathbf{d}_{c/g}^g,$$

or expressed in the camera frame

$$\dot{\mathbf{v}}_{c/i}^c = R_g^c R_b^g R_i^b \mathbf{v}_{b/i}^i + R_g^c R_b^g (\omega_{b/i})^\times \mathbf{d}_{g/b}^b + R_g^c R_b^g (\omega_{b/i})^\times R_g^b \mathbf{d}_{c/g}^g + R_g^c (\omega_{g/b})^\times \mathbf{d}_{c/g}^g.$$

Using the fact that $\mathbf{a}^\times \mathbf{b} = -\mathbf{b}^\times \mathbf{a}$ we gives

$$\begin{aligned} \dot{\mathbf{v}}_{c/i}^c &= R_g^c R_b^g R_i^b \mathbf{v}_{b/i}^i - R_g^c R_b^g (\mathbf{d}_{g/b}^b)^\times \omega_{b/i}^b - R_g^c R_b^g (R_g^b \mathbf{d}_{c/g}^g)^\times \omega_{b/i}^b - R_g^c (\mathbf{d}_{c/g}^g)^\times \omega_{g/b}^g \\ &= [R_g^c R_b^g R_i^b] \mathbf{v}_{b/i}^i - [R_g^c R_b^g (\mathbf{d}_{g/b}^b + R_g^b \mathbf{d}_{c/g}^g)^\times] \omega_{b/i}^b - [R_g^c (\mathbf{d}_{c/g}^g)^\times] \omega_{g/b}^g. \end{aligned} \quad (9.6)$$

Also, since angular velocities add, we have

$$\omega_{c/i}^c = \omega_{c/g}^c + \omega_{g/b}^c + \omega_{b/i}^c.$$

Assuming that the camera is stationary with respect to the gimbal, i.e., $\omega_{c/g}^c = 0$ gives

$$\omega_{c/i}^c = R_g^c \omega_{g/b}^g + R_g^c R_b^g \omega_{b/i}^b. \quad (9.7)$$

Using Equations (9.6) and (9.7) in Equation (9.1) gives the following expression for the pixel motion

$$\begin{aligned} \dot{\epsilon}_f &= \lambda_f \left[V(\epsilon_f) R_g^c R_b^g R_i^b \right] (\mathbf{v}_{b/i}^i - \mathbf{v}_{f/i}^i) \\ &\quad + \left[\Omega(\epsilon_f) R_g^c R_b^g - \lambda_f V(\epsilon_f) R_g^c R_b^g (\mathbf{d}_{g/b}^b + R_g^b \mathbf{d}_{c/g}^g)^\times \right] \omega_{b/i}^b \\ &\quad + \left[\Omega(\epsilon_f) R_g^c - \lambda_f V(\epsilon_f) R_g^c (\mathbf{d}_{c/g}^g)^\times \right] \omega_{g/b}^g. \end{aligned} \quad (9.8)$$

■

Corollary 9.1.3 *If the distance to the feature is large relative to the lengths $\mathbf{d}_{g/b}^b$ and $\mathbf{d}_{c/g}^g$, then the pixel motion is*

$$\dot{\epsilon}_f = G_1(\epsilon_f, \lambda_f, t) (\mathbf{v}_{b/i}^i - \mathbf{v}_{f/i}^i) + G'_2(\epsilon_f, t) \omega_{b/i}^b + G'_3(\epsilon_f) \omega_{g/b}^g.$$

where

$$\begin{aligned} G'_2(\epsilon_f, t) &\stackrel{\triangle}{=} \Omega(\epsilon_f) R_g^c R_b^g(t) \\ G'_3(\epsilon_f, \lambda_f) &\stackrel{\triangle}{=} \Omega(\epsilon_f) R_g^c. \end{aligned}$$

Proof: If the distance to the feature is large relative to the lengths $\mathbf{d}_{g/b}^b$ and $\mathbf{d}_{c/g}^g$ then

$$\begin{aligned}\lambda_f \left[\mathbf{d}_{g/b}^b + R_g^b \mathbf{d}_{c/g}^g \right]_{\times} &\approx 0 \\ \lambda_f \left[\mathbf{d}_{c/g}^g \right]_{\times} &\approx 0.\end{aligned}$$

■

Corollary 9.1.4 *If the camera is fixed in the body frame (no gimbal), and the distance to the feature is large relative to the lengths $\mathbf{d}_{c/b}^b$, then the pixel motion is*

$$\dot{\epsilon}_f = \lambda_f V(\epsilon_f) R_b^c R_i^b(t) (\mathbf{v}_{b/i}^i - \mathbf{v}_{f/i}^i) + \Omega(\epsilon_f) R_b^c \omega_{b/i}^b.$$

Lemma 9.1.5 *The rank of the matrices $V(\epsilon_f)$ and $\Omega(\epsilon_f)$ is two for all ϵ_f . If the gimbal is stationary, and $\lambda_z \neq \infty$ (feature depth of zero), then*

$$\text{rank}(G_1) = \text{rank}(G_2) = \text{rank}(G_3) = 2.$$

Proof: From Equation (9.2) it is clear that $\text{rank}(V(\epsilon_\ell)) = 2$ for any ϵ_ℓ . To show that $\Omega(\epsilon_\ell)$ also has rank of two, note from Equation (9.3) that if $\epsilon_x = \epsilon_y = 0$, then the rows of $\Omega(\epsilon_\ell)$ are $(0, -1, 0)$ and $(1, 0, 0)$ which are linearly independent. Now suppose that $\epsilon_y \neq 0$, then

$$c_1 \begin{pmatrix} \epsilon_x \epsilon_y \\ 1 + \epsilon_y^2 \end{pmatrix} + c_2 \begin{pmatrix} \epsilon_y \\ -\epsilon_x \end{pmatrix} = 0,$$

implies that $c_2 = -c_1 \epsilon_x$ and therefore $c_1(1 + \epsilon_x^2 + \epsilon_y^2) = 0$, which implies that $c_1 = c_2 = 0$, and therefore the first and last columns of $\Omega(\epsilon_f)$ are linearly independent. A similar argument can be made for the case of $\epsilon_x \neq 0$. Therefore $\text{rank}(\Omega(\epsilon_f)) = 2$ for any ϵ_f . Since R_a^b are rotation matrices, Sylvester's inequality and assumption A3 imply that $\text{rank}(G_1) = 2$.

The matrix G_2 can be written as

$$G_2 = \left(\Omega R_g^c R_b^g, \quad \lambda_f V R_g^c R_b^g \right) \begin{pmatrix} I_3 \\ -(\mathbf{d}_{g/b}^b + R_g^b(t) \mathbf{d}_{c/g}^g)^\times \end{pmatrix},$$

where the first matrix is rank two following an argument similar to the previous paragraph, and the second matrix is clearly rank 3. Therefore $\text{rank}(G_2) = 2$ by Sylvester's inequality. A similar argument can be made for G_3 . ■

9.2 The Body-Level Frame

The landing and target following problem will be cast in the body-level frame. The basic idea is that the body-level frame is the unrolled and un-pitched body frame. The heading direction for the body

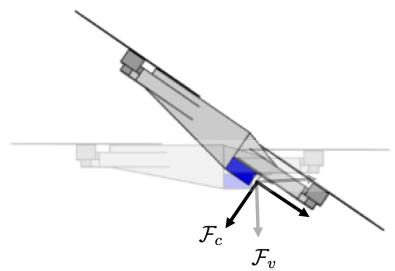


Figure 9.2: The body-level frame results by rotating the body frame by the negative roll and pitch angles. The only rotation in the body-level frame is due to yaw. TOC

frame and the body-level frame will be identical, but the z -axis of the body level frame will always point down along the gravity vector. Letting ℓ denote the body-level frame, we have that

$$R_b^i = R_\ell^i R_b^\ell,$$

or

$$R_\ell^i = R_b^i (R_b^\ell)^\top.$$

To make things concrete, if ϕ , θ , and ψ are the roll, pitch, and yaw Euler angles, then

$$\begin{aligned} R_b^i &= R_z(\psi) R_y(\theta) R_x(\phi) \\ &\triangleq \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{pmatrix}. \end{aligned}$$

In this case $R_\ell^i = R_z(\psi)$ and $R_b^\ell = R_y(\theta) R_x(\phi)$.

Since the origin of the body-level frame is coincident with the origin of the body frame, we have that

$$\mathbf{p}_{\ell/i} = \mathbf{p}_{b/i}$$

$$\mathbf{v}_{\ell/i} = \mathbf{v}_{b/i}.$$

Lemma 9.2.1 *The normalized pixel coordinates in the camera frame $\bar{\epsilon}_f^c$, and the scaling factor λ_f^c are transformed to the body-level frame as*

$$\begin{aligned} \bar{\epsilon}_f^\ell &= \frac{R_b^\ell R_c^b \bar{\epsilon}_f^c}{\mathbf{e}_3^\top R_b^\ell R_c^b \bar{\epsilon}_f^c} \\ \lambda_f^\ell &= \lambda_f^c \mathbf{e}_3^\top R_b^\ell R_c^b \bar{\epsilon}_f^c, \end{aligned}$$

where

$$\lambda_f^\ell \bar{\epsilon}_f^\ell = \mathbf{p}_{f/\ell}^\ell.$$

Proof: The normalized pixel coordinates in the camera frame $\bar{\epsilon}_f^c$ can be transformed to the level frame as

$$\epsilon_f^\ell = R_b^\ell R_c^b \bar{\epsilon}_f^c,$$

but in this case the last element of ϵ_f^ℓ will not necessarily be equal to unity. Therefore, to obtain normalized coordinates we use

$$\bar{\epsilon}_f^\ell = \frac{R_b^\ell R_c^b \bar{\epsilon}_f^c}{\mathbf{e}_3^\top R_b^\ell R_c^b \bar{\epsilon}_f^c}.$$

Noting that

$$\begin{aligned}\lambda_f^\ell \bar{\epsilon}_f^\ell &= \lambda_f^\ell \left(\frac{R_b^\ell R_c^b \bar{\epsilon}_f^c}{\mathbf{e}_3^\top R_b^\ell R_c^b \bar{\epsilon}_f^c} \right) \\ &= \mathbf{p}_{f/\ell}^\ell \\ &= R_b^\ell R_c^b \mathbf{p}_{f/c}^c \\ &= \lambda_f^c R_b^\ell R_c^b \bar{\epsilon}_f^c\end{aligned}$$

which implies that

$$\frac{\lambda_f^\ell}{\mathbf{e}_3^\top R_b^\ell R_c^b \bar{\epsilon}_f^c} (R_b^\ell R_c^b \bar{\epsilon}_f^c) = \lambda_f^c R_b^\ell R_c^b \bar{\epsilon}_f^c$$

which gives the results. \blacksquare

The pixel motion in the body-level frame becomes particularly simply.

Lemma 9.2.2 *Let $\mathbf{v}_{b/f}^i$ be the velocity of the body relative to the feature, and let $\boldsymbol{\omega}_{\ell/i}^\ell = \boldsymbol{\omega}_{\ell/i}^i = (0, 0, \omega_z)^\top$ be the angular velocity of the level frame, then the pixel motion in the body-level frame virtual camera is given by*

$$\dot{\epsilon}_f^\ell = G_f \boldsymbol{\nu}^\ell,$$

where

$$\begin{aligned}\boldsymbol{\nu}^\ell &= (\mathbf{v}_{b/f}^{i\top}, \omega_z)^\top \\ G_f &= \left(\lambda_f^\ell V(\boldsymbol{\epsilon}_f^\ell) R_i^\ell, \boldsymbol{\Omega}(\boldsymbol{\epsilon}_f^\ell) \mathbf{e}_3 \right).\end{aligned}$$

Proof: From Corollary 9.1.4, replacing the body frame with the body-level frame, we have

$$\dot{\epsilon}_f^\ell = \lambda_f V(\boldsymbol{\epsilon}_f^\ell) R_i^c R_i^\ell(t) (\mathbf{v}_{\ell/i}^i - \mathbf{v}_{f/i}^i) + \boldsymbol{\Omega}(\boldsymbol{\epsilon}_f^\ell) R_i^c \boldsymbol{\omega}_{\ell/i}^\ell.$$

Noting that the virtual camera frame and the body-level frame are identical, i.e., $R_i^c = I$ gives

$$\dot{\epsilon}_f^\ell = \lambda_f V(\boldsymbol{\epsilon}_f^\ell) R_i^\ell(t) (\mathbf{v}_{\ell/i}^i - \mathbf{v}_{f/i}^i) + \boldsymbol{\Omega}(\boldsymbol{\epsilon}_f^\ell) \boldsymbol{\omega}_{\ell/i}^\ell.$$

The result follows by noting that

$$\begin{aligned}\boldsymbol{\Omega}(\boldsymbol{\epsilon}_f^\ell) \boldsymbol{\omega}_{\ell/i}^\ell &= \boldsymbol{\Omega}(\boldsymbol{\epsilon}_f^\ell) \begin{pmatrix} 0 \\ 0 \\ \omega_z \end{pmatrix} \\ &= \boldsymbol{\Omega}(\boldsymbol{\epsilon}_f^\ell) \mathbf{e}_3 \omega_z.\end{aligned}$$

\blacksquare

9.3 Landing and target following using visual servoing

This section describes how Lemma 9.2.2 can be used to design algorithms for landing a multirotor on a moving target, and for following a moving target.

The basic idea is to maneuver the multirotor so that a set of feature points move to specified locations in the image plane. Let

$$\mathbf{s} \triangleq \begin{pmatrix} \boldsymbol{\epsilon}_{f_1}^\ell \\ \boldsymbol{\epsilon}_{f_2}^\ell \\ \vdots \\ \boldsymbol{\epsilon}_{f_m}^\ell \end{pmatrix}, \quad (9.9)$$

be a set of feature point and let

$$\mathbf{s}_d(t) = \begin{pmatrix} \boldsymbol{\epsilon}_{d_1}^\ell(t) \\ \boldsymbol{\epsilon}_{d_2}^\ell(t) \\ \vdots \\ \boldsymbol{\epsilon}_{d_m}^\ell(t) \end{pmatrix}, \quad (9.10)$$

be the set of time-varying desired feature locations, and let $\tilde{\mathbf{s}} \triangleq \mathbf{s} - \mathbf{s}_d$. Differentiating $\tilde{\mathbf{s}}$ gives

$$\begin{aligned} \dot{\tilde{\mathbf{s}}} &= \dot{\mathbf{s}} - \dot{\mathbf{s}}_d \\ &= \begin{pmatrix} G_{f_1} \\ \dots \\ G_{f_m} \end{pmatrix} \boldsymbol{\nu}^\ell - \dot{\mathbf{s}}_d \\ &= G_m \boldsymbol{\nu}^\ell - \dot{\mathbf{s}}_d, \end{aligned}$$

where

$$G_m \triangleq \begin{pmatrix} G_{f_1} \\ \dots \\ G_{f_m} \end{pmatrix}.$$

Lemma 9.3.1 *Let $\{\boldsymbol{\epsilon}_{f_1}, \dots, \boldsymbol{\epsilon}_{f_m}\}$ be a set of 2D normalized pixel locations, where at least two points are not scaled version of each other, i.e., there exists $i \neq j$, such that $\boldsymbol{\epsilon}_{f_j} \neq \mu \boldsymbol{\epsilon}_{f_i}$ for any μ , then $\text{rank}(G_m) = 4$.*

Proof: Without loss of generality, let $\boldsymbol{\epsilon}_{f_1}$ and $\boldsymbol{\epsilon}_{f_2}$ be two features points that are not scaled version of each other. Consider the matrix

$$\hat{G} = \begin{pmatrix} G_{f_1} \\ G_{f_2} \end{pmatrix}.$$

We will show that $\text{rank}(\hat{G}) = 4$, which implies that $\text{rank}(G_m) = 4$, since adding additional rows cannot decrease the rank. Toward that

end, note that

$$\begin{aligned} G_{f_i} &= \left(\lambda_{f_i} V(\epsilon_{f_i}) R_\ell^i, \quad \Omega(\epsilon_{f_i}) \mathbf{e}_3 \right) \\ &= \left(\lambda_{f_i} \begin{pmatrix} -I_{2 \times 2}, & \epsilon_{f_i} \end{pmatrix} \begin{pmatrix} R_\psi & \mathbf{0}_{2 \times 1} \\ \mathbf{0}_{1 \times 2} & 1 \end{pmatrix}, \quad J\epsilon_{f_i} \right) \\ &= \left(-\lambda_{f_i} R_\psi, \quad \lambda_{f_i} \epsilon_{f_i}, \quad J\epsilon_{f_i} \right), \end{aligned}$$

where $R_\psi \in SO(2)$ is the 2×2 rotation matrix in the N-E plane, and

$J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ corresponds to a rotation in the image plane by $-\pi/2$.

Therefore

$$\begin{aligned} \hat{G} &= \begin{pmatrix} -\lambda_{f_1} R_\psi, & \lambda_{f_1} \epsilon_{f_1}, & J\epsilon_{f_1} \\ -\lambda_{f_2} R_\psi, & \lambda_{f_2} \epsilon_{f_2}, & J\epsilon_{f_2} \end{pmatrix} \\ &= \begin{pmatrix} -\lambda_{f_1} I_{2 \times 2}, & \lambda_{f_1} \epsilon_{f_1}, & J\epsilon_{f_1} \\ -\lambda_{f_2} I_{2 \times 2}, & \lambda_{f_2} \epsilon_{f_2}, & J\epsilon_{f_2} \end{pmatrix} \begin{pmatrix} R_\psi & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & I_{2 \times 2} \end{pmatrix} \\ &= \begin{pmatrix} -\lambda_{f_1} I_{2 \times 2}, & H(\lambda_{f_1}, \epsilon_{f_1}) \\ -\lambda_{f_2} I_{2 \times 2}, & H(\lambda_{f_2}, \epsilon_{f_2}) \end{pmatrix} \begin{pmatrix} R_\psi & \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} & I_{2 \times 2} \end{pmatrix}, \end{aligned}$$

where

$$H(\lambda, \epsilon) \triangleq \begin{pmatrix} \lambda \epsilon, & J\epsilon \end{pmatrix}.$$

Therefore $\text{rank}(\hat{G}) = 4$ if

$$\det \begin{pmatrix} -\lambda_{f_1} I_{2 \times 2}, & H(\lambda_{f_1}, \epsilon_{f_1}) \\ -\lambda_{f_2} I_{2 \times 2}, & H(\lambda_{f_2}, \epsilon_{f_2}) \end{pmatrix} \neq 0.$$

Using the Schur identity

$$\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det A \det (D - CA^{-1}B)$$

we get that

$$\begin{aligned} \det \begin{pmatrix} -\lambda_{f_1} I_{2 \times 2}, & H(\lambda_{f_1}, \epsilon_{f_1}) \\ -\lambda_{f_2} I_{2 \times 2}, & H(\lambda_{f_2}, \epsilon_{f_2}) \end{pmatrix} &= -\lambda_{f_1} \det \left(H(\lambda_{f_2}, \epsilon_{f_2}) - \frac{\lambda_{f_2}}{\lambda_{f_1}} H(\lambda_{f_1}, \epsilon_{f_1}) \right) \\ &= -\lambda_{f_1} \det \begin{pmatrix} \lambda_{f_2} \epsilon_{2x}, & \epsilon_{2y} \\ \lambda_{f_2} \epsilon_{2y}, & -\epsilon_{2x} \end{pmatrix} - \frac{\lambda_{f_2}}{\lambda_{f_1}} \begin{pmatrix} \lambda_{f_1} \epsilon_{1x}, & \epsilon_{1y} \\ \lambda_{f_1} \epsilon_{1y}, & -\epsilon_{1x} \end{pmatrix} \\ &= -\lambda_{f_1} \det \begin{pmatrix} \lambda_{f_2} (\epsilon_{2x} - \epsilon_{1x}), & \epsilon_{2y} - \frac{\lambda_{f_2}}{\lambda_{f_1}} \epsilon_{1y} \\ \lambda_{f_2} (\epsilon_{2y} - \epsilon_{1y}), & \epsilon_{2x} - \frac{\lambda_{f_2}}{\lambda_{f_1}} \epsilon_{1x} \end{pmatrix} \\ &= -\lambda_{f_2} ((\epsilon_{2x} - \epsilon_{1x})(\lambda_{f_2} \epsilon_{1x} - \lambda_{f_1} \epsilon_{2x}) - (\epsilon_{2y} - \epsilon_{1y})(\lambda_{f_1} \epsilon_{2y} - \lambda_{f_2} \epsilon_{1y})) \\ &= -\lambda_{f_2} \begin{pmatrix} \lambda_{f_1} & \lambda_{f_2} \end{pmatrix}^\top \begin{pmatrix} -\epsilon_{2x} & \epsilon_{2y} \\ \epsilon_{1x} & -\epsilon_{1y} \end{pmatrix} (\epsilon_{f_2} - \epsilon_{f_1}), \end{aligned}$$

which requires that both $\epsilon_{f_2} \neq \epsilon_{f_1}$ and that $E = \begin{pmatrix} -\epsilon_{2x} & \epsilon_{2y} \\ \epsilon_{1x} & -\epsilon_{1y} \end{pmatrix}$ is full rank. But E loses rank when

$$\det \begin{pmatrix} -\epsilon_{2x} & \epsilon_{2y} \\ \epsilon_{1x} & -\epsilon_{1y} \end{pmatrix} = \epsilon_{2x}\epsilon_{1y} - \epsilon_{1x}\epsilon_{2y} = \boldsymbol{\epsilon}_{f_1}^\top J \boldsymbol{\epsilon}_{f_2} = 0.$$

Since J represents a rotation in the image of $\pi/2$, E loses rank only when $\boldsymbol{\epsilon}_{f_1} = \mu \boldsymbol{\epsilon}_{f_2}$ for some μ . \blacksquare

We can now state the main result for visual servoing control.

Theorem 9.3.2 *Let \mathbf{s} and $\mathbf{s}_d(t)$ be defined as in Equations (9.9) and (9.10), with $m = 2$ feature points that are not scaled versions of each other, if the desired body velocity and desired heading vector are given by*

$$\begin{aligned} \mathbf{v}_{d/i}^i &= \left(I_{3 \times 3}, 0_{3 \times 1} \right) \boldsymbol{\nu}^\ell \\ \mathbf{s}_{\psi_d} &= R_\ell^i \mathbf{e}_1, \end{aligned}$$

where R_ℓ^i satisfies

$$\dot{R}_\ell^i = R_\ell^i \left[\left(0_{3 \times 3}, \mathbf{e}_3 \right) \boldsymbol{\nu}^\ell \right]_\times,$$

where

$$\boldsymbol{\nu}^\ell = G_2^{-1} (\dot{\mathbf{s}}_d - K \tilde{\mathbf{s}}), \quad (9.11)$$

where $K = K^\top > 0$ is a positive definite gain matrix, then

$$\tilde{\mathbf{s}}(t) \rightarrow 0.$$

Proof: Defining the Lyapunov function $V = \frac{1}{2} \|\tilde{\mathbf{s}}\|$ and differentiating gives

$$\begin{aligned} \dot{V} &= \tilde{\mathbf{s}}^\top \dot{\tilde{\mathbf{s}}} \\ &= \tilde{\mathbf{s}}^\top (G_2 \boldsymbol{\nu}^\ell - \dot{\mathbf{s}}_d). \end{aligned}$$

Since the feature points are not co-linear, G_2 is invertible by Lemma 9.3.1.

Selecting $\boldsymbol{\nu}^\ell$ according to Equation (9.11) gives $\dot{V} = -\tilde{\mathbf{s}}^\top K \tilde{\mathbf{s}}$ from which we conclude that $\tilde{\mathbf{s}} \rightarrow 0$. Since $\boldsymbol{\nu}^\ell = (\mathbf{v}_{b/i}^{i\top}, \omega_z)^\top$ we have that

$$\begin{aligned} \mathbf{v}_{b/i}^i &= \left(I \quad 0 \right) \boldsymbol{\nu}^\ell \\ \boldsymbol{\omega}_{\ell/i}^i &= \left(0_{3 \times 3} \quad \mathbf{e}_3 \right) \boldsymbol{\nu}^\ell. \end{aligned}$$

The theorem is completed by noting that

$$\mathbf{s}_\psi = \begin{pmatrix} \cos \psi \\ \sin \psi \\ 0 \end{pmatrix} = \begin{pmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{e}_1 = R_\ell^i \mathbf{e}_1.$$

\blacksquare

If we have more than two features in the image plane, then we have the following corollary.

Corollary 9.3.3 *Let \mathbf{s} and $\mathbf{s}_d(t)$ be defined as in Equations (9.9) and (9.10), with $m > 2$ feature points, where at least two points are not scaled versions of each other. If the desired body velocity and desired heading vector are given by*

$$\begin{aligned}\mathbf{v}_{d/i}^i &= \left(I_{3 \times 3}, 0_{3 \times 1} \right) \boldsymbol{\nu}^\ell \\ \mathbf{s}_{\psi_d} &= R_\ell^i \mathbf{e}_1,\end{aligned}$$

where R_ℓ^i satisfies

$$\dot{R}_\ell^i = R_\ell^i \left[\left(0_{3 \times 3}, \mathbf{e}_3 \right) \boldsymbol{\nu}^\ell \right]_\times,$$

where

$$\boldsymbol{\nu}^\ell = G_2^\dagger (\dot{\mathbf{s}}_d - K \tilde{\mathbf{s}}),$$

where $K = K^\top > 0$ is a positive definite gain matrix and A^\dagger is the pseudo-inverse of A , then

$$\lim_{t \rightarrow \infty} \|\tilde{\mathbf{s}}(t)\|$$

is minimized.

9.4 Multiple View Geometry-Transformations of a plane

RWB: Add the stuff about affine homography transformations to show how to manipulate the desired pixels \mathbf{s}_d . Use a discussion similar to the following pages from Multiple View Geometry.

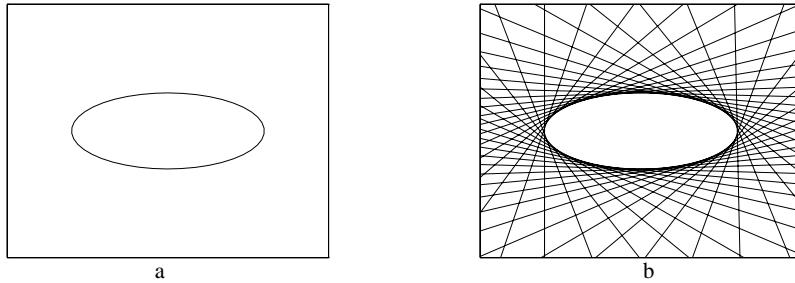


Fig. 2.2. (a) Points \mathbf{x} satisfying $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$ lie on a point conic. (b) Lines \mathbf{l} satisfying $\mathbf{l}^T \mathbf{C}^* \mathbf{l} = 0$ are tangent to the point conic \mathbf{C} . The conic \mathbf{C} is the envelope of the lines \mathbf{l} .

in figure 2.2. A dual conic has five degrees of freedom. In a similar manner to points defining a point conic, it follows that five lines in general position define a dual conic.

Degenerate conics. If the matrix \mathbf{C} is not of full rank, then the conic is termed degenerate. Degenerate point conics include two lines (rank 2), and a repeated line (rank 1).

Example 2.8. The conic

$$\mathbf{C} = \mathbf{l} \mathbf{m}^T + \mathbf{m} \mathbf{l}^T$$

is composed of two lines \mathbf{l} and \mathbf{m} . Points on \mathbf{l} satisfy $\mathbf{l}^T \mathbf{x} = 0$, and are on the conic since $\mathbf{x}^T \mathbf{C} \mathbf{x} = (\mathbf{x}^T \mathbf{l})(\mathbf{m}^T \mathbf{x}) + (\mathbf{x}^T \mathbf{m})(\mathbf{l}^T \mathbf{x}) = 0$. Similarly, points satisfying $\mathbf{m}^T \mathbf{x} = 0$ also satisfy $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$. The matrix \mathbf{C} is symmetric and has rank 2. The null vector is $\mathbf{x} = \mathbf{l} \times \mathbf{m}$ which is the intersection point of \mathbf{l} and \mathbf{m} . \triangle

Degenerate *line* conics include two points (rank 2), and a repeated point (rank 1). For example, the line conic $\mathbf{C}^* = \mathbf{x} \mathbf{y}^T + \mathbf{y} \mathbf{x}^T$ has rank 2 and consists of lines passing through either of the two points \mathbf{x} and \mathbf{y} . Note that for matrices that are not invertible $(\mathbf{C}^*)^* \neq \mathbf{C}$.

2.3 Projective transformations

In the view of geometry set forth by Felix Klein in his famous “Erlangen Program”, [Klein-39], geometry is the study of properties invariant under groups of transformations. From this point of view, 2D projective geometry is the study of properties of the projective plane \mathbb{P}^2 that are invariant under a group of transformations known as *projectivities*.

A projectivity is an invertible mapping from points in \mathbb{P}^2 (that is homogeneous 3-vectors) to points in \mathbb{P}^2 that maps lines to lines. More precisely,

Definition 2.9. A *projectivity* is an invertible mapping h from \mathbb{P}^2 to itself such that three points \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 lie on the same line if and only if $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$ and $h(\mathbf{x}_3)$ do.

Projectivities form a group since the inverse of a projectivity is also a projectivity, and so is the composition of two projectivities. A projectivity is also called a *collineation*

2.3 Projective transformations

33

(a helpful name), a *projective transformation* or a *homography*: the terms are synonymous.

In definition 2.9, a projectivity is defined in terms of a coordinate-free geometric concept of point line incidence. An equivalent algebraic definition of a projectivity is possible, based on the following result.

Theorem 2.10. *A mapping $h : \mathbb{P}^2 \rightarrow \mathbb{P}^2$ is a projectivity if and only if there exists a non-singular 3×3 matrix H such that for any point in \mathbb{P}^2 represented by a vector x it is true that $h(x) = Hx$.*

To interpret this theorem, any point in \mathbb{P}^2 is represented as a homogeneous 3-vector, x , and Hx is a linear mapping of homogeneous coordinates. The theorem asserts that any projectivity arises as such a linear transformation in homogeneous coordinates, and that conversely any such mapping is a projectivity. The theorem will not be proved in full here. It will only be shown that any invertible linear transformation of homogeneous coordinates is a projectivity.

Proof. Let x_1, x_2 and x_3 lie on a line l . Thus $l^T x_i = 0$ for $i = 1, \dots, 3$. Let H be a non-singular 3×3 matrix. One verifies that $l^T H^{-1} H x_i = 0$. Thus, the points Hx_i all lie on the line $H^{-T} l$, and collinearity is preserved by the transformation.

The converse is considerably harder to prove, namely that each projectivity arises in this way. \square

As a result of this theorem, one may give an alternative definition of a projective transformation (or collineation) as follows.

Definition 2.11. Projective transformation. A planar projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular 3×3 matrix:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad (2.5)$$

or more briefly, $x' = Hx$.

Note that the matrix H occurring in this equation may be changed by multiplication by an arbitrary non-zero scale factor without altering the projective transformation. Consequently we say that H is a *homogeneous* matrix, since as in the homogeneous representation of a point, only the ratio of the matrix elements is significant. There are eight independent ratios amongst the nine elements of H , and it follows that a projective transformation has eight degrees of freedom.

A projective transformation projects every figure into a projectively equivalent figure, leaving all its projective properties invariant. In the ray model of figure 2.1 a projective transformation is simply a linear transformation of \mathbb{R}^3 .

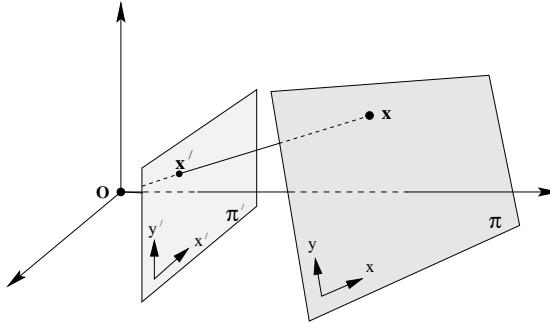


Fig. 2.3. **Central projection maps points on one plane to points on another plane.** The projection also maps lines to lines as may be seen by considering a plane through the projection centre which intersects with the two planes π and π' . Since lines are mapped to lines, central projection is a projectivity and may be represented by a linear mapping of homogeneous coordinates $x' = Hx$.

Mappings between planes. As an example of how theorem 2.10 may be applied, consider figure 2.3. Projection along rays through a common point (the centre of projection) defines a mapping from one plane to another. It is evident that this point-to-point mapping preserves lines in that a line in one plane is mapped to a line in the other. If a coordinate system is defined in each plane and points are represented in homogeneous coordinates, then the *central projection* mapping may be expressed by $x' = Hx$ where H is a non-singular 3×3 matrix. Actually, if the two coordinate systems defined in the two planes are both Euclidean (rectilinear) coordinate systems then the mapping defined by central projection is more restricted than an arbitrary projective transformation. It is called a *perspectivity* rather than a full projectivity, and may be represented by a transformation with six degrees of freedom. We return to perspectivities in section A7.4(p632).

Example 2.12. Removing the projective distortion from a perspective image of a plane.

Shape is distorted under perspective imaging. For instance, in figure 2.4a the windows are not rectangular in the image, although the originals are. In general parallel lines on a scene plane are not parallel in the image but instead converge to a finite point. We have seen that a central projection image of a plane (or section of a plane) is related to the original plane via a projective transformation, and so the image is a projective distortion of the original. It is possible to “undo” this projective transformation by computing the inverse transformation and applying it to the image. The result will be a new synthesized image in which the objects in the plane are shown with their correct geometric shape. This will be illustrated here for the front of the building of figure 2.4a. Note that since the ground and the front are not in the same plane, the projective transformation that must be applied to rectify the front is not the same as the one used for the ground.

Computation of a projective transformation from point-to-point correspondences will be considered in great detail in chapter 4. For now, a method for computing the trans-

2.3 Projective transformations

35

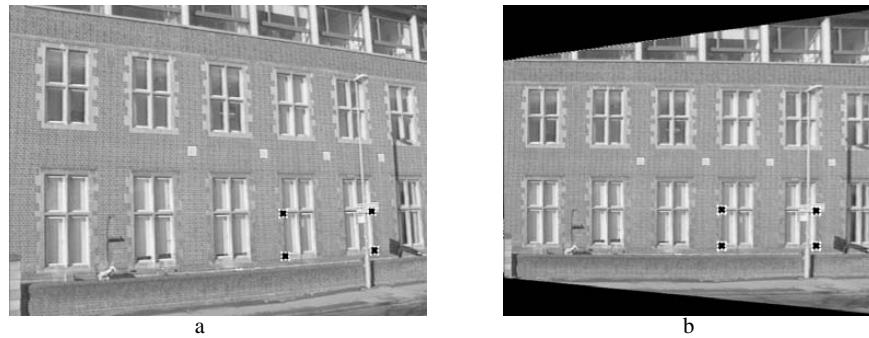


Fig. 2.4. Removing perspective distortion. (a) The original image with perspective distortion – the lines of the windows clearly converge at a finite point. (b) Synthesized frontal orthogonal view of the front wall. The image (a) of the wall is related via a projective transformation to the true geometry of the wall. The inverse transformation is computed by mapping the four imaged window corners to corners of an appropriately sized rectangle. The four point correspondences determine the transformation. The transformation is then applied to the whole image. Note that sections of the image of the ground are subject to a further projective distortion. This can also be removed by a projective transformation.

formation is briefly indicated. One begins by selecting a section of the image corresponding to a planar section of the world. Local 2D image and world coordinates are selected as shown in figure 2.3. Let the inhomogeneous coordinates of a pair of matching points \mathbf{x} and \mathbf{x}' in the world and image plane be (x, y) and (x', y') respectively. We use inhomogeneous coordinates here instead of the homogeneous coordinates of the points, because it is these inhomogeneous coordinates that are measured directly from the image and from the world plane. The projective transformation of (2.5) can be written in inhomogeneous form as

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}.$$

Each point correspondence generates two equations for the elements of H , which after multiplying out are

$$\begin{aligned} x'(h_{31}x + h_{32}y + h_{33}) &= h_{11}x + h_{12}y + h_{13} \\ y'(h_{31}x + h_{32}y + h_{33}) &= h_{21}x + h_{22}y + h_{23}. \end{aligned}$$

These equations are *linear* in the elements of H . Four point correspondences lead to eight such linear equations in the entries of H , which are sufficient to solve for H up to an insignificant multiplicative factor. The only restriction is that the four points must be in “general position”, which means that no three points are collinear. The inverse of the transformation H computed in this way is then applied to the whole image to undo the effect of perspective distortion on the selected plane. The results are shown in figure 2.4b. \triangle

Three remarks concerning this example are appropriate: first, the computation of the rectifying transformation H in this way does not require knowledge of *any* of the camera’s parameters or the pose of the plane; second, it is not always necessary to

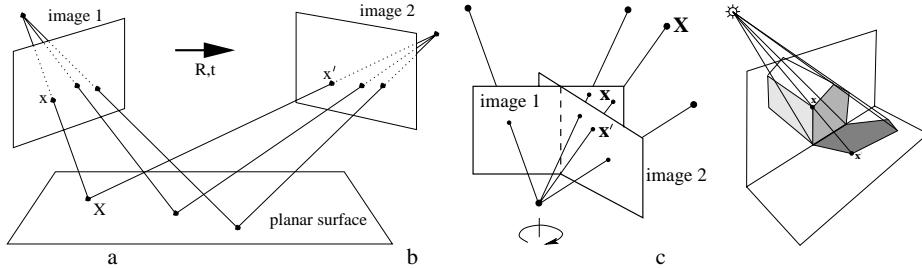


Fig. 2.5. Examples of a projective transformation, $x' = Hx$, arising in perspective images. (a) The projective transformation between two images induced by a world plane (the concatenation of two projective transformations is a projective transformation); (b) The projective transformation between two images with the same camera centre (e.g. a camera rotating about its centre or a camera varying its focal length); (c) The projective transformation between the image of a plane (the end of the building) and the image of its shadow onto another plane (the ground plane). Figure (c) courtesy of Luc Van Gool.

know coordinates for four points in order to remove projective distortion: alternative approaches, which are described in section 2.7, require less, and different types of, information; third, superior (and preferred) methods for computing projective transformations are described in chapter 4.

Projective transformations are important mappings representing many more situations than the perspective imaging of a world plane. A number of other examples are illustrated in figure 2.5. Each of these situations is covered in more detail later in the book.

2.3.1 Transformations of lines and conics

Transformation of lines. It was shown in the proof of theorem 2.10 that if points x_i lie on a line l , then the transformed points $x'_i = Hx_i$ under a projective transformation lie on the line $l' = H^{-T}l$. In this way, incidence of points on lines is preserved, since $l'^T x'_i = l^T H^{-1} H x_i = 0$. This gives the transformation rule for lines:

Under the point transformation $x' = Hx$, a line transforms as

$$l' = H^{-T}l. \quad (2.6)$$

One may alternatively write $l'^T = l^T H^{-1}$. Note the fundamentally different way in which lines and points transform. Points transform according to H , whereas lines (as rows) transform according to H^{-1} . This may be explained in terms of ‘‘covariant’’ or ‘‘contravariant’’ behaviour. One says that points transform *contravariantly* and lines transform *covariantly*. This distinction will be taken up again, when we discuss tensors in chapter 15 and is fully explained in appendix 1(p562).

Transformation of conics. Under a point transformation $x' = Hx$, (2.2) becomes

$$\begin{aligned} x^T C x &= x'^T [H^{-1}]^T C H^{-1} x' \\ &= x'^T H^{-T} C H^{-1} x' \end{aligned}$$

2.4 A hierarchy of transformations

37

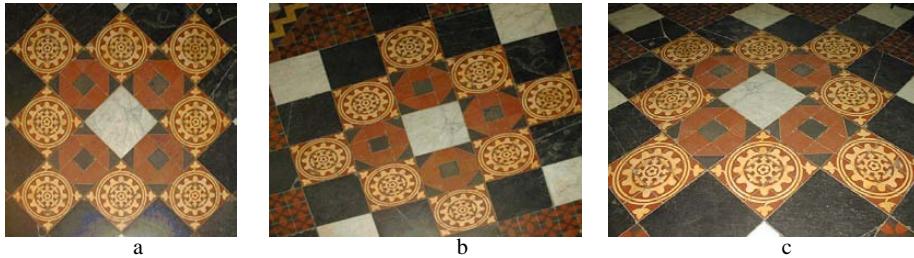


Fig. 2.6. **Distortions arising under central projection.** Images of a tiled floor. (a) **Similarity:** the circular pattern is imaged as a circle. A square tile is imaged as a square. Lines which are parallel or perpendicular have the same relative orientation in the image. (b) **Affine:** The circle is imaged as an ellipse. Orthogonal world lines are not imaged as orthogonal lines. However, the sides of the square tiles, which are parallel in the world are parallel in the image. (c) **Projective:** Parallel world lines are imaged as converging lines. Tiles closer to the camera have a larger image than those further away.

which is a quadratic form $\mathbf{x}'^T \mathbf{C}' \mathbf{x}'$ with $\mathbf{C}' = \mathbf{H}^{-T} \mathbf{C} \mathbf{H}^{-1}$. This gives the transformation rule for a conic:

Result 2.13. Under a point transformation $\mathbf{x}' = \mathbf{H}\mathbf{x}$, a conic \mathbf{C} transforms to $\mathbf{C}' = \mathbf{H}^{-T} \mathbf{C} \mathbf{H}^{-1}$.

The presence of \mathbf{H}^{-1} in this equation may be expressed by saying that a conic transforms *covariantly*. The transformation rule for a dual conic is derived in a similar manner. This gives:

Result 2.14. Under a point transformation $\mathbf{x}' = \mathbf{H}\mathbf{x}$, a dual conic \mathbf{C}^* transforms to $\mathbf{C}'^* = \mathbf{H}\mathbf{C}^* \mathbf{H}^T$.

2.4 A hierarchy of transformations

In this section we describe the important specializations of a projective transformation and their geometric properties. It was shown in section 2.3 that projective transformations form a group. This group is called the *projective linear group*, and it will be seen that these specializations are *subgroups* of this group.

The group of invertible $n \times n$ matrices with real elements is the (real) general linear group on n dimensions, or $GL(n)$. To obtain the projective linear group the matrices related by a scalar multiplier are identified, giving $PL(n)$ (this is a quotient group of $GL(n)$). In the case of projective transformations of the plane $n = 3$.

The important subgroups of $PL(3)$ include the *affine group*, which is the subgroup of $PL(3)$ consisting of matrices for which the last row is $(0, 0, 1)$, and the *Euclidean group*, which is a subgroup of the affine group for which in addition the upper left hand 2×2 matrix is orthogonal. One may also identify the *oriented Euclidean group* in which the upper left hand 2×2 matrix has determinant 1.

We will introduce these transformations starting from the most specialized, the isometries, and progressively generalizing until projective transformations are reached.

This defines a *hierarchy* of transformations. The distortion effects of various transformations in this hierarchy are shown in figure 2.6.

Some transformations of interest are not groups, for example, perspectivities (because the composition of two perspectivities is a projectivity, not a perspectivity). This point is covered in section A7.4(p632).

Invariants. An alternative to describing the transformation *algebraically*, i.e. as a matrix acting on coordinates of a point or curve, is to describe the transformation in terms of those elements or quantities that are preserved or *invariant*. A (scalar) invariant of a geometric configuration is a function of the configuration whose value is unchanged by a particular transformation. For example, the separation of two points is unchanged by a Euclidean transformation (translation and rotation), but not by a similarity (e.g. translation, rotation and isotropic scaling). Distance is thus a Euclidean, but not similarity invariant. The angle between two lines is both a Euclidean and a similarity invariant.

2.4.1 Class I: Isometries

Isometries are transformations of the plane \mathbb{R}^2 that preserve Euclidean distance (from *iso* = same, *metric* = measure). An isometry is represented as

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon \cos \theta & -\sin \theta & t_x \\ \epsilon \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where $\epsilon = \pm 1$. If $\epsilon = 1$ then the isometry is *orientation-preserving* and is a *Euclidean* transformation (a composition of a translation and rotation). If $\epsilon = -1$ then the isometry reverses orientation. An example is the composition of a reflection, represented by the matrix $\text{diag}(-1, 1, 1)$, with a Euclidean transformation.

Euclidean transformations model the motion of a rigid object. They are by far the most important isometries in practice, and we will concentrate on these. However, the orientation reversing isometries often arise as ambiguities in structure recovery.

A planar Euclidean transformation can be written more concisely in block form as

$$\mathbf{x}' = H_E \mathbf{x} = \begin{bmatrix} R & t \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{x} \quad (2.7)$$

where R is a 2×2 rotation matrix (an orthogonal matrix such that $R^\top R = RR^\top = I$), t a translation 2-vector, and $\mathbf{0}$ a null 2-vector. Special cases are a pure rotation (when $t = \mathbf{0}$) and a pure translation (when $R = I$). A Euclidean transformation is also known as a *displacement*.

A planar Euclidean transformation has three degrees of freedom, one for the rotation and two for the translation. Thus three parameters must be specified in order to define the transformation. The transformation can be computed from two point correspondences.

Invariants. The invariants are very familiar, for instance: length (the distance between two points), angle (the angle between two lines), and area.

Groups and orientation. An isometry is orientation-preserving if the upper left hand 2×2 matrix has determinant 1. Orientation-*preserving* isometries form a group, orientation-*reversing* ones do not. This distinction applies also in the case of similarity and affine transformations which now follow.

2.4.2 Class II: Similarity transformations

A similarity transformation (or more simply a *similarity*) is an isometry composed with an isotropic scaling. In the case of a Euclidean transformation composed with a scaling (i.e. no reflection) the similarity has matrix representation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \quad (2.8)$$

This can be written more concisely in block form as

$$\mathbf{x}' = \mathbf{H}_s \mathbf{x} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \quad (2.9)$$

where the scalar s represents the isotropic scaling. A similarity transformation is also known as an *equi-form* transformation, because it preserves “shape” (form). A planar similarity transformation has four degrees of freedom, the scaling accounting for one more degree of freedom than a Euclidean transformation. A similarity can be computed from two point correspondences.

Invariants. The invariants can be constructed from Euclidean invariants with suitable provision being made for the additional scaling degree of freedom. Angles between lines are not affected by rotation, translation or isotropic scaling, and so are similarity invariants. In particular parallel lines are mapped to parallel lines. The length between two points is not a similarity invariant, but the *ratio* of two lengths is an invariant, because the scaling of the lengths cancels out. Similarly a ratio of areas is an invariant because the scaling (squared) cancels out.

Metric structure. A term that will be used frequently in the discussion on reconstruction (chapter 10) is *metric*. The description *metric structure* implies that the structure is defined up to a similarity.

2.4.3 Class III: Affine transformations

An affine transformation (or more simply an *affinity*) is a non-singular linear transformation followed by a translation. It has the matrix representation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.10)$$

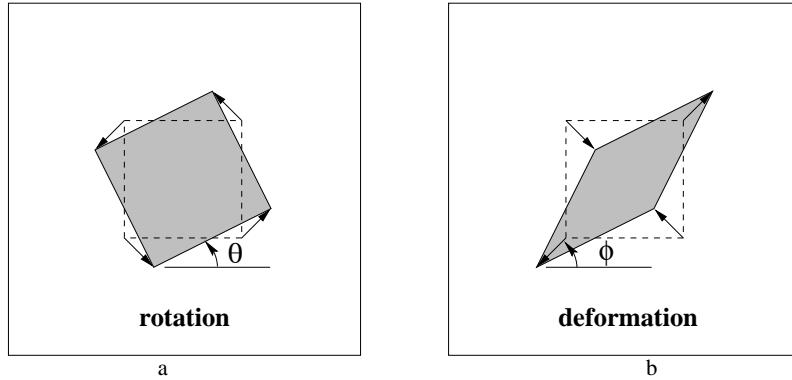


Fig. 2.7. **Distortions arising from a planar affine transformation.** (a) Rotation by $R(\theta)$. (b) A deformation $R(-\phi) D R(\phi)$. Note, the scaling directions in the deformation are orthogonal.

or in block form

$$\mathbf{x}' = \mathbf{H}_A \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \quad (2.11)$$

with \mathbf{A} a 2×2 non-singular matrix. A planar affine transformation has six degrees of freedom corresponding to the six matrix elements. The transformation can be computed from three point correspondences.

A helpful way to understand the geometric effects of the linear component \mathbf{A} of an affine transformation is as the composition of two fundamental transformations, namely rotations and non-isotropic scalings. The affine matrix \mathbf{A} can always be decomposed as

$$\mathbf{A} = R(\theta) R(-\phi) D R(\phi) \quad (2.12)$$

where $R(\theta)$ and $R(\phi)$ are rotations by θ and ϕ respectively, and D is a diagonal matrix:

$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

This decomposition follows directly from the SVD (section A4.4(p585)): writing $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T = (\mathbf{U}\mathbf{V}^T)(\mathbf{V}\mathbf{D}\mathbf{V}^T) = R(\theta)(R(-\phi)D R(\phi))$, since \mathbf{U} and \mathbf{V} are orthogonal matrices.

The affine matrix \mathbf{A} is hence seen to be the concatenation of a rotation (by ϕ); a scaling by λ_1 and λ_2 respectively in the (rotated) x and y directions; a rotation back (by $-\phi$); and finally another rotation (by θ). The only “new” geometry, compared to a similarity, is the non-isotropic scaling. This accounts for the two extra degrees of freedom possessed by an affinity over a similarity. They are the angle ϕ specifying the scaling direction, and the ratio of the scaling parameters $\lambda_1 : \lambda_2$. The essence of an affinity is this scaling in orthogonal directions, oriented at a particular angle. Schematic examples are given in figure 2.7.

Invariants. Because an affine transformation includes non-isotropic scaling, the similarity invariants of length ratios and angles between lines are not preserved under an affinity. Three important invariants are:

- (i) **Parallel lines.** Consider two parallel lines. These intersect at a point $(x_1, x_2, 0)^T$ at infinity. Under an affine transformation this point is mapped to another point at infinity. Consequently, the parallel lines are mapped to lines which still intersect at infinity, and so are parallel after the transformation.
- (ii) **Ratio of lengths of parallel line segments.** The length scaling of a line segment depends only on the angle between the line direction and scaling directions. Suppose the line is at angle α to the x -axis of the orthogonal scaling direction, then the scaling magnitude is $\sqrt{\lambda_1^2 \cos^2 \alpha + \lambda_2^2 \sin^2 \alpha}$. This scaling is common to all lines with the same direction, and so cancels out in a ratio of parallel segment lengths.
- (iii) **Ratio of areas.** This invariance can be deduced directly from the decomposition (2.12). Rotations and translations do not affect area, so only the scalings by λ_1 and λ_2 matter here. The effect is that area is scaled by $\lambda_1 \lambda_2$ which is equal to $\det A$. Thus the area of any shape is scaled by $\det A$, and so the scaling cancels out for a ratio of areas. It will be seen that this does not hold for a projective transformation.

An affinity is orientation-preserving or -reversing according to whether $\det A$ is positive or negative respectively. Since $\det A = \lambda_1 \lambda_2$ the property depends only on the sign of the scalings.

2.4.4 Class IV: Projective transformations

A projective transformation was defined in (2.5). It is a general non-singular linear transformation of *homogeneous* coordinates. This generalizes an affine transformation, which is the composition of a general non-singular linear transformation of *inhomogeneous* coordinates and a translation. We have earlier seen the action of a projective transformation (in section 2.3). Here we examine its block form

$$\mathbf{x}' = H_P \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \mathbf{x} \quad (2.13)$$

where the vector $\mathbf{v} = (v_1, v_2)^T$. The matrix has nine elements with only their ratio significant, so the transformation is specified by eight parameters. Note, it is not always possible to scale the matrix such that v is unity since v might be zero. A projective transformation between two planes can be computed from four point correspondences, with no three collinear on either plane. See figure 2.4.

Unlike the case of affinities, it is not possible to distinguish between orientation preserving and orientation reversing projectivities in \mathbb{P}^2 . We will return to this point in section 2.6.

Invariants. The most fundamental projective invariant is the cross ratio of four collinear points: a ratio of lengths on a line is invariant under affinities, but not under projectivities. However, a ratio of ratios or *cross ratio* of lengths on a line is a projective invariant. We return to properties of this invariant in section 2.5.

2.4.5 Summary and comparison

Affinities (6 dof) occupy the middle ground between similarities (4 dof) and projectivities (8 dof). They generalize similarities in that angles are not preserved, so that shapes are skewed under the transformation. On the other hand their action is homogeneous over the plane: for a given affinity the $\det A$ scaling in area of an object (e.g. a square) is the same anywhere on the plane; and the orientation of a transformed line depends only on its initial orientation, not on its position on the plane. In contrast, for a given projective transformation, area scaling varies with position (e.g. under perspective a more distant square on the plane has a smaller image than one that is nearer, as in figure 2.6); and the orientation of a transformed line depends on both the orientation and position of the source line (however, it will be seen later in section 8.6(p213) that a line's vanishing point depends only on line orientation, not position).

The key difference between a projective and affine transformation is that the vector v is not null for a projectivity. This is responsible for the non-linear effects of the projectivity. Compare the mapping of an ideal point $(x_1, x_2, 0)^T$ under an affinity and projectivity: First the affine transformation

$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ 0 \end{pmatrix}. \quad (2.14)$$

Second the projective transformation

$$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ v_1 x_1 + v_2 x_2 \end{pmatrix}. \quad (2.15)$$

In the first case the ideal point remains ideal (i.e. at infinity). In the second it is mapped to a finite point. It is this ability which allows a projective transformation to model vanishing points.

2.4.6 Decomposition of a projective transformation

A projective transformation can be decomposed into a chain of transformations, where each matrix in the chain represents a transformation higher in the hierarchy than the previous one.

$$H = H_S H_A H_P = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ v^T & v \end{bmatrix} = \begin{bmatrix} A & t \\ v^T & v \end{bmatrix} \quad (2.16)$$

with A a non-singular matrix given by $A = sRK + tv^T$, and K an upper-triangular matrix normalized as $\det K = 1$. This decomposition is valid provided $v \neq 0$, and is unique if s is chosen positive.

2.4 A hierarchy of transformations

43

Each of the matrices H_S, H_A, H_P is the “essence” of a transformation of that type (as indicated by the subscripts S, A, P). Consider the process of rectifying the perspective image of a plane as in example 2.12: H_P (2 dof) moves the line at infinity; H_A (2 dof) affects the affine properties, but does not move the line at infinity; and finally, H_S is a general similarity transformation (4 dof) which does not affect the affine or projective properties. The transformation H_P is an *elation*, described in section A7.3(p631).

Example 2.15. The projective transformation

$$H = \begin{bmatrix} 1.707 & 0.586 & 1.0 \\ 2.707 & 8.242 & 2.0 \\ 1.0 & 2.0 & 1.0 \end{bmatrix}$$

may be decomposed as

$$H = \begin{bmatrix} 2 \cos 45^\circ & -2 \sin 45^\circ & 1 \\ 2 \sin 45^\circ & 2 \cos 45^\circ & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

△

This decomposition can be employed when the objective is to only partially determine the transformation. For example, if one wants to measure length ratios from the perspective image of a plane, then it is only necessary to determine (rectify) the transformation up to a similarity. We return to this approach in section 2.7.

Taking the inverse of H in (2.16) gives $H^{-1} = H_P^{-1} H_A^{-1} H_S^{-1}$. Since H_P^{-1} , H_A^{-1} and H_S^{-1} are still projective, affine and similarity transformations respectively, a general projective transformation may also be decomposed in the form

$$H = H_P H_A H_S = \begin{bmatrix} I & 0 \\ v^T & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} \quad (2.17)$$

Note that the actual values of K , R , t and v will be different from those of (2.16).

2.4.7 The number of invariants

The question naturally arises as to how many invariants there are for a given geometric configuration under a particular transformation. First the term “number” needs to be made more precise, for if a quantity is invariant, such as length under Euclidean transformations, then any function of that quantity is invariant. Consequently, we seek a counting argument for the number of functionally independent invariants. By considering the number of transformation parameters that must be eliminated in order to form an invariant, it can be seen that:

Result 2.16. *The number of functionally independent invariants is equal to, or greater than, the number of degrees of freedom of the configuration less the number of degrees of freedom of the transformation.*

Group	Matrix	Distortion	Invariant properties
Projective 8 dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, order of contact : intersection (1 pt contact); tangency (2 pt contact); inflections (3 pt contact with line); tangent discontinuities and cusps. cross ratio (ratio of ratio of lengths).
Affine 6 dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallelism, ratio of areas, ratio of lengths on collinear or parallel lines (e.g. midpoints), linear combinations of vectors (e.g. centroids). The line at infinity, l_∞ .
Similarity 4 dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratio of lengths, angle. The circular points, I, J (see section 2.7.3).
Euclidean 3 dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Length, area

Table 2.1. **Geometric properties invariant to commonly occurring planar transformations.** The matrix $A = [a_{ij}]$ is an invertible 2×2 matrix, $R = [r_{ij}]$ is a 2D rotation matrix, and (t_x, t_y) a 2D translation. The distortion column shows typical effects of the transformations on a square. Transformations higher in the table can produce all the actions of the ones below. These range from Euclidean, where only translations and rotations occur, to projective where the square can be transformed to any arbitrary quadrilateral (provided no three points are collinear).

For example, a configuration of four points in general position has 8 degrees of freedom (2 for each point), and so 4 similarity, 2 affinity and zero projective invariants since these transformations have respectively 4, 6 and 8 degrees of freedom.

Table 2.1 summarizes the 2D transformation groups and their invariant properties. Transformations lower in the table are specializations of those above. A transformation lower in the table inherits the invariants of those above.

2.5 The projective geometry of 1D

The development of the projective geometry of a line, \mathbb{P}^1 , proceeds in much the same way as that of the plane. A point x on the line is represented by homogeneous coordinates $(x_1, x_2)^\top$, and a point for which $x_2 = 0$ is an ideal point of the line. We will use the notation \bar{x} to represent the 2-vector $(x_1, x_2)^\top$. A projective transformation of a line is represented by a 2×2 homogeneous matrix,

$$\bar{x}' = H_{2 \times 2} \bar{x}$$

and has 3 degrees of freedom corresponding to the four elements of the matrix less one for overall scaling. A projective transformation of a line may be determined from three corresponding points.

9.5 *Visual Servo Control Part I: Basic Approaches, RAM, 2006*

TUTORIAL

Visual Servo Control

Part I: Basic Approaches

BY FRANÇOIS CHAUMETTE AND SETH HUTCHINSON

This article is the first of a two-part series on the topic of visual servo control—using computer vision data in the servo loop to control the motion of a robot. In the present article, we describe the basic techniques that are by now well established in the field. We first give a general overview of the formulation of the visual servo control problem. We then describe the two archetypal visual servo control schemes: image-based and position-based visual servo control. Finally, we discuss performance and stability issues that pertain to these two schemes, motivating the second article in the series, in which we consider advanced techniques.

Introduction

Visual servo control refers to the use of computer vision data to control the motion of a robot. The vision data may be acquired from a camera that is mounted directly on a robot manipulator or on a mobile robot, in which case motion of the robot induces camera motion, or the camera can be fixed in the workspace so that it can observe the robot motion from a stationary configuration. Other configurations can be considered such as, for instance, several cameras mounted on pan-tilt heads observing the robot motion. The mathematical development of all these cases is similar, and in this tutorial we will focus primarily on the former, so-called eye-in-hand, case.

Visual servo control relies on techniques from image processing, computer vision, and control theory. Since it is not possible to cover all of these in depth in a single article, we will focus here primarily on issues related to control, and to those specific geometric aspects of computer vision that are uniquely relevant to the study of visual servo control. We will not specifically address issues related to feature tracking or three-dimensional (3-D) pose estimation, both of which are topics deserving of their own tutorials.

The Basic Components of Visual Servoing

The aim of all vision-based control schemes is to minimize an error $\mathbf{e}(t)$, which is typically defined by

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*. \quad (1)$$

This formulation is quite general, and it encompasses a wide variety of approaches, as we will see below. The parameters in (1) are defined as follows. The vector $\mathbf{m}(t)$ is a set of image

measurements (e.g., the image coordinates of interest points or the image coordinates of the centroid of an object). These image measurements are used to compute a vector of k visual features, $\mathbf{s}(\mathbf{m}(t), \mathbf{a})$, in which \mathbf{a} is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or 3-D models of objects). The vector \mathbf{s}^* contains the desired values of the features.

In Part I of the tutorial (this article), we consider the case of a fixed goal pose and a motionless target, i.e., \mathbf{s}^* is constant, and changes in \mathbf{s} depend only on camera motion. Further, we consider here the case of controlling the motion of a camera with six degrees of freedom (6 DOF); e.g., a camera attached to the end effector of a six degree-of-freedom arm. We will treat more general cases in Part II of the tutorial.

Visual servoing schemes mainly differ in the way that \mathbf{s} is designed. In this article, we will see two very different approaches. First, we describe image-based visual servo control (IBVS), in which \mathbf{s} consists of a set of features that are immediately available in the image data. Then, we describe position-based visual servo control (PBVS), in which \mathbf{s} consists of a set of 3-D parameters, which must be estimated from image measurements.

Once \mathbf{s} is selected, the design of the control scheme can be quite simple. Perhaps the most straightforward approach is to design a velocity controller. To do this, we require the relationship between the time variation of \mathbf{s} and the camera velocity. Let the spatial velocity of the camera be denoted by $\mathbf{v}_c = (v_c, \omega_c)$, with v_c the instantaneous linear velocity of the origin of the camera frame and ω_c the instantaneous angular velocity of the camera frame. The relationship between $\dot{\mathbf{s}}$ and \mathbf{v}_c is given by

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v}_c, \quad (2)$$

in which $\mathbf{L}_s \in \mathbb{R}^{k \times 6}$ is named the *interaction matrix* related to \mathbf{s} . The term *feature Jacobian* is also used somewhat interchangeably in the visual servo literature.

Using (1) and (2), we immediately obtain the relationship between camera velocity and the time variation of the error:

$$\dot{\mathbf{e}} = \mathbf{L}_e \mathbf{v}_c, \quad (3)$$

where $\mathbf{L}_e = \mathbf{L}_s$. Considering \mathbf{v}_c as the input to the robot controller, and if we would like for instance to try to ensure

an exponential decoupled decrease of the error (i.e., $\dot{\mathbf{e}} = -\lambda \mathbf{e}$), we obtain using (3):

$$\mathbf{v}_c = -\lambda \mathbf{L}_e^+ \mathbf{e}, \quad (4)$$

where $\mathbf{L}_e^+ \in \mathbb{R}^{6 \times k}$ is chosen as the Moore-Penrose pseudo-inverse of \mathbf{L}_e , that is $\mathbf{L}_e^+ = (\mathbf{L}_e^\top \mathbf{L}_e)^{-1} \mathbf{L}_e^\top$ when \mathbf{L}_e is of full rank 6. This choice allows $\|\dot{\mathbf{e}} - \lambda \mathbf{L}_e \mathbf{L}_e^+ \mathbf{e}\|$ and $\|\mathbf{v}_c\|$ to be minimal. When $k = 6$, if $\det \mathbf{L}_e \neq 0$ it is possible to invert \mathbf{L}_e , giving the control $\mathbf{v}_c = -\lambda \mathbf{L}_e^{-1} \mathbf{e}$.

In real visual servo systems, it is impossible to know perfectly in practice either \mathbf{L}_e or \mathbf{L}_e^+ . So an approximation or an estimation of one of these two matrices must be realized. In the sequel, we denote both the pseudoinverse of the approximation of the interaction matrix and the approximation of the pseudoinverse of the interaction matrix by the symbol $\widehat{\mathbf{L}}_e^+$. Using this notation, the control law is in fact:

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}}_e^+ \mathbf{e}. \quad (5)$$

This is the basic design implemented by most visual servo controllers. All that remains is to fill in the details: How should \mathbf{s} be chosen? What then is the form of \mathbf{L}_s ? How should we estimate $\widehat{\mathbf{L}}_e^+$? What are the performance characteristics of the resulting closed-loop system? These questions are addressed in the remainder of the article.

Classical Image-Based Visual Servo

Traditional image-based control schemes [1], [2] use the image-plane coordinates of a set of points (other choices are possible, but we defer discussion of these for Part II of the tutorial) to define the set \mathbf{s} . The image measurements \mathbf{m} are usually the pixel coordinates of the set of image points (but this is not the only possible choice), and the parameters \mathbf{a} in the definition of $\mathbf{s} = \mathbf{s}(\mathbf{m}, \mathbf{a})$ in (1) are nothing but the camera intrinsic parameters to go from image measurements expressed in pixels to the features.

The Interaction Matrix

More precisely, for a 3-D point with coordinates $\mathbf{X} = (X, Y, Z)$ in the camera frame, which projects in the image as a 2-D point with coordinates $\mathbf{x} = (x, y)$, we have:

$$\begin{cases} x = X/Z = (u - c_u)/f\alpha \\ y = Y/Z = (v - c_v)/f, \end{cases} \quad (6)$$

where $\mathbf{m} = (u, v)$ gives the coordinates of the image point expressed in pixel units, and $\mathbf{a} = (c_u, c_v, f, \alpha)$ is the set of camera intrinsic parameters: c_u and c_v are the coordinates of the principal point, f is the focal length, and α is the ratio of the pixel dimensions. In this case, we take $\mathbf{s} = \mathbf{x} = (x, y)$, the image plane coordinates of the point. The details of imaging geometry and perspective projection can be found in many computer vision texts, including [3], [4].

Taking the time derivative of the projection equations (6), we obtain

$$\begin{cases} \dot{x} = \dot{X}/Z - X\dot{Z}/Z^2 = (\dot{X} - x\dot{Z})/Z \\ \dot{y} = \dot{Y}/Z - Y\dot{Z}/Z^2 = (\dot{Y} - y\dot{Z})/Z. \end{cases} \quad (7)$$

We can relate the velocity of the 3-D point to the camera spatial velocity using the well-known equation

$$\dot{\mathbf{X}} = -\mathbf{v}_c - \omega_c \times \mathbf{X} \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X. \end{cases} \quad (8)$$

Injecting (8) in (7), and grouping terms we obtain

$$\begin{cases} \dot{x} = -v_x/Z + xv_z/Z + xy\omega_x - (1+x^2)\omega_y + y\omega_z \\ \dot{y} = -v_y/Z + yv_z/Z + (1+y^2)\omega_x - xy\omega_y - x\omega_z \end{cases} \quad (9)$$

which can be written

$$\dot{\mathbf{x}} = \mathbf{L}_x \mathbf{v}_c, \quad (10)$$

where the interaction matrix \mathbf{L}_x related to \mathbf{x} is

$$\mathbf{L}_x = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix}. \quad (11)$$

In the matrix \mathbf{L}_x , the value Z is the depth of the point relative to the camera frame. Therefore, any control scheme that uses this form of the interaction matrix must estimate or approximate the value of Z . Similarly, the camera intrinsic parameters are involved in the computation of x and y . Thus, \mathbf{L}_x cannot be directly used in (4), and an estimation or an approximation $\widehat{\mathbf{L}}_x$ must be used. We discuss this in more detail below.

To control the 6 DOF, at least three points are necessary (i.e., we require $k \geq 6$). If we use the feature vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, by merely stacking interaction matrices for three points we obtain

$$\mathbf{L}_x = \begin{bmatrix} \mathbf{L}_{x_1} \\ \mathbf{L}_{x_2} \\ \mathbf{L}_{x_3} \end{bmatrix}.$$

In this case, there will exist some configurations for which \mathbf{L}_x is singular [5]. Furthermore, there exist four distinct camera poses for which $\mathbf{e} = 0$, i.e., four global minima exist, and it is impossible to differentiate them [6]. For these reasons, more than three points are usually considered.

Approximating the Interaction Matrix

There are several choices available for constructing the estimate $\widehat{\mathbf{L}}_e^+$ to be used in the control law. One popular scheme is, of course, to choose $\widehat{\mathbf{L}}_e^+ = \mathbf{L}_e^+$ if $\mathbf{L}_e = \mathbf{L}_x$ is known; that is, if the current depth Z of each point is available [7]. In practice, these parameters must be estimated at each iteration of the control scheme. The basic methods presented in this article use classical pose estimation methods that will be briefly presented in the next section. Another popular

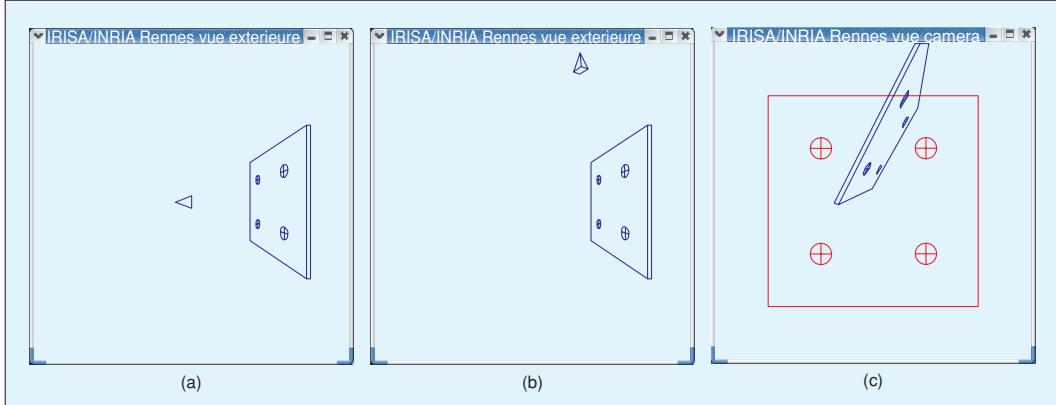


Figure 1. An example of positioning task: (a) the desired camera pose with respect to a simple target, (b) the initial camera pose, and (c) the corresponding initial and desired image of the target.

approach is to choose $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = \mathbf{L}_{\mathbf{e}^*}^+$, where $\mathbf{L}_{\mathbf{e}^*}$ is the value of $\mathbf{L}_{\mathbf{e}}$ for the desired position $\mathbf{e} = \mathbf{e}^* = 0$ [8]. In this case, $\widehat{\mathbf{L}}_{\mathbf{e}}^+$ is constant, and only the desired depth of each point has to be set, which means no varying 3-D parameters have to be estimated during the visual servo. Finally, the choice $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = 1/2(\mathbf{L}_{\mathbf{e}} + \mathbf{L}_{\mathbf{e}^*})^+$ has recently been proposed in [9]. Since $\mathbf{L}_{\mathbf{e}}$ is involved in this method, the current depth of each point must also be available.

We illustrate the behavior of these control schemes with an example. The goal is to position the camera so that it observes a square as a centered square in the image (see Figure 1). We

define \mathbf{s} to include the x and y coordinates of the four points forming the square. Note that the initial camera pose has been selected far away from the desired pose, particularly with regard to the rotational motions, which are known to be the most problematic for IBVS. In the simulations presented in the following, no noise or modeling errors have been introduced in order to allow comparison of different behaviors in perfect conditions.

The results obtained by using $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = \mathbf{L}_{\mathbf{e}^*}^+$ are given in Figure 2. Note that despite the large displacement that is required, the system converges. However, neither the behavior in the image nor the computed camera velocity components nor the 3-D trajectory of the camera present desirable properties far from the convergence (i.e., for the first 30 or so iterations).

The results obtained using $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = \mathbf{L}_{\mathbf{e}}^+$ are given in Figure 3. In this case, the trajectories of the points in the image are almost straight lines, but the behavior induced in 3-D is even less satisfactory than for the case of $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = \mathbf{L}_{\mathbf{e}^*}^+$. The large camera velocities at the beginning of the servo indicate that the condition number of $\widehat{\mathbf{L}}_{\mathbf{e}}^+$ is high at the start of the trajectory, and the camera trajectory is far from a straight line.

The choice $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = 1/2(\mathbf{L}_{\mathbf{e}} + \mathbf{L}_{\mathbf{e}^*})^+$ provides good performance in practice. Indeed, as can be seen in Figure 4, the camera velocity components do not include large oscillations and provide a smooth trajectory both in the image and in 3-D.

A Geometrical Interpretation of IBVS

It is quite easy to provide a geometric interpretation of the behavior of the control schemes defined above. The example illustrated in Figure 5 corresponds to a pure rotation around the optical axis from the initial configuration (shown in blue) to the desired configuration of four coplanar points parallel to the image plane (shown in red).

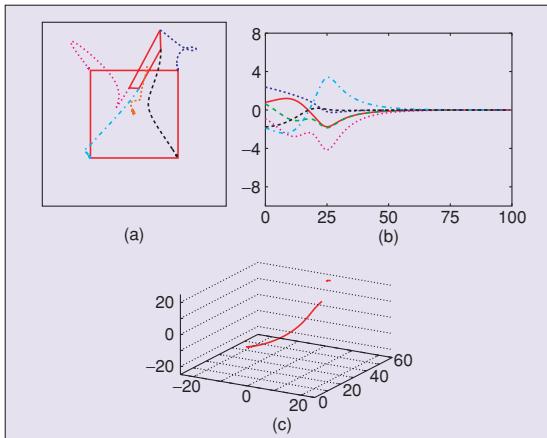


Figure 2. The system behavior using $\mathbf{s} = (x_1, y_1, \dots, x_4, y_4)$ and $\widehat{\mathbf{L}}_{\mathbf{e}}^+ = \mathbf{L}_{\mathbf{e}^*}^+$: (a) image points trajectories including the trajectory of the center of the square, which is not used in the control scheme, (b) \mathbf{v}_c components (cm/s and deg/s) computed at each iteration of the control scheme, and (c) 3-D trajectory of the camera optical center expressed in $\mathcal{R}_{\mathbf{C}^*}$ (cm).

As explained above, using \mathbf{L}_e^+ in the control scheme attempts to ensure an exponential decrease of the error \mathbf{e} . It means that when x and y image point coordinates compose this error, the points' trajectories in the image follow straight lines from their initial to their desired positions, when it is possible. This leads to the image motion plotted in green in the figure. The camera motion to realize this image motion can be easily deduced and is indeed composed of a rotational motion around the optical axis, but is combined with a retreating translational motion along the optical axis [10]. This unexpected motion is due to the choice of the features and to the coupling between the third and sixth columns in the interaction matrix. If the rotation between the initial and desired configurations is very large, this phenomenon is amplified and leads to a particular case for a rotation of π rad where no rotational motion at all will be induced by the control scheme [11]. On the other hand, when the rotation is small, this phenomenon almost disappears. To conclude, the behavior is locally satisfactory (i.e., when the error is small), but it can be unsatisfactory when the error is large. As we will see in the last part of this article, these results are consistent with the local asymptotic stability results that can be obtained for IBVS.

If instead we use $\mathbf{L}_{e^*}^+$ in the control scheme, the image motion generated can easily be shown to be the blue one plotted in Figure 5. Indeed, if we consider the same control scheme as before but starting from \mathbf{s}^* to reach \mathbf{s} , we obtain:

$$\mathbf{v}_c = -\lambda \mathbf{L}_{e^*}^+ (\mathbf{s}^* - \mathbf{s}),$$

which again induces straight-line trajectories from the red points to the blue ones, causing image motion plotted in brown. Going back to our problem, the control scheme computes a camera velocity that is exactly the opposite one:

$$\mathbf{v}_c = -\lambda \mathbf{L}_e^+ (\mathbf{s} - \mathbf{s}^*)$$

and thus generates the image motion plotted in red at the red points. Transformed at the blue points, the camera velocity generates the blue image motion and corresponds once again to a rotational motion around the optical axis, combined now with an unexpected forward motion along the optical axis. The same analysis as before can be done, as for large or small errors. We can add that, as soon as the error decreases significantly, both control schemes get closer and tend to the same one (since $\mathbf{L}_e = \mathbf{L}_{e^*}$ when $\mathbf{e} = \mathbf{e}^*$) with a nice behavior characterized with the image motion plotted in black and a camera motion composed of only a rotation around the optical axis when the error tends towards zero.

If we instead use $\widehat{\mathbf{L}}_e^+ = 1/2(\mathbf{L}_e + \mathbf{L}_{e^*})^+$, it is intuitively clear that considering the mean of \mathbf{L}_e and \mathbf{L}_{e^*} generates the

image motion plotted in black, even when the error is large. In all cases but the rotation around π rad, the camera motion is now a pure rotation around the optical axis, without any unexpected translational motion.

IBVS with a Stereovision System

It is straightforward to extend the IBVS approach to a multi-camera system. If a stereovision system is used, and a 3-D point is visible in both left and right images (see Figure 6), it is possible to use as visual features

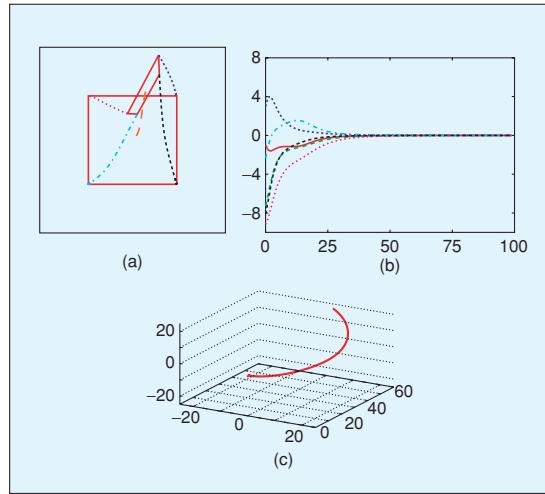


Figure 3. The system behavior using $\mathbf{s} = (x_1, y_1, \dots, x_4, y_4)$ and $\mathbf{L}_e^+ = \mathbf{L}_{e^*}^+$.

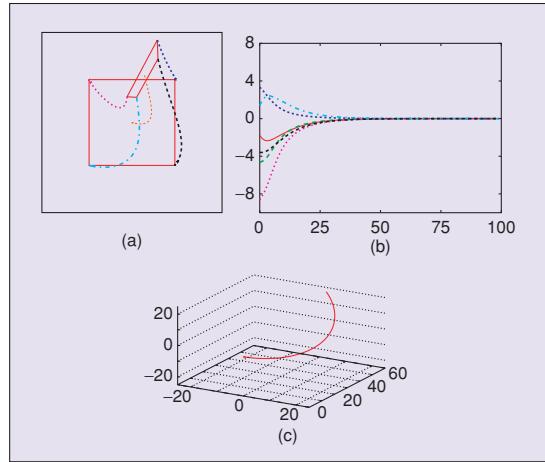


Figure 4. The system behavior using $\mathbf{s} = (x_1, y_1, \dots, x_4, y_4)$ and $\mathbf{L}_e^+ = \frac{1}{2}(\mathbf{L}_e + \mathbf{L}_{e^*})^+$.

$$\mathbf{s} = \mathbf{x}_s = (\mathbf{x}_l, \mathbf{x}_r) = (x_l, y_l, x_r, y_r),$$

i.e., to represent the point by just stacking in \mathbf{s} the x and y coordinates of the observed point in the left and right images [12]. However, care must be taken when constructing the corresponding interaction matrix since the form given in (10) is expressed in either the left or right camera frame. More precisely, we have:

$$\begin{cases} \dot{\mathbf{x}}_l &= \mathbf{L}_{\mathbf{x}_l} \mathbf{v}_l \\ \dot{\mathbf{x}}_r &= \mathbf{L}_{\mathbf{x}_r} \mathbf{v}_r, \end{cases}$$

where \mathbf{v}_l and \mathbf{v}_r are the spatial velocity of the left and right camera respectively, and where the analytical form of $\mathbf{L}_{\mathbf{x}_l}$ and $\mathbf{L}_{\mathbf{x}_r}$ are given by (11).

By choosing a sensor frame rigidly linked to the stereovision system, we obtain:

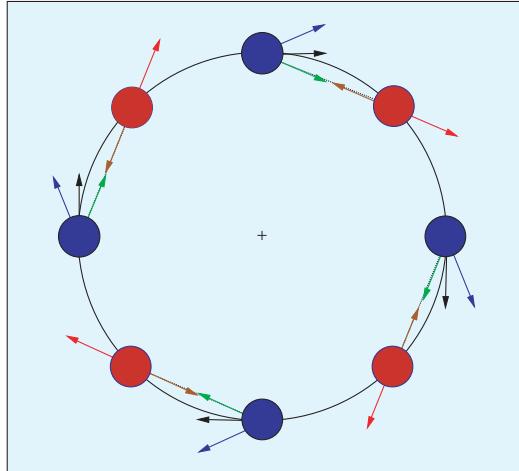


Figure 5. A geometrical interpretation of IBVS.

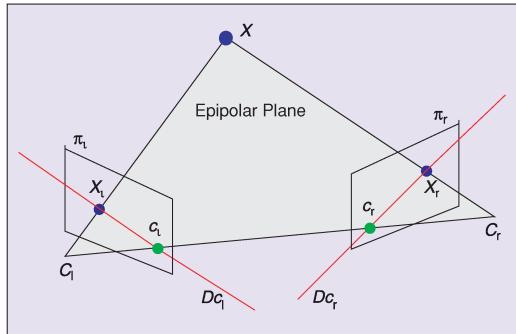


Figure 6. A stereovision system.

$$\dot{\mathbf{x}}_s = \begin{bmatrix} \dot{\mathbf{x}}_l \\ \dot{\mathbf{x}}_r \end{bmatrix} = \mathbf{L}_{\mathbf{x}_s} \mathbf{v}_s,$$

where the interaction matrix related to \mathbf{x}_s can be determined using the spatial motion transform matrix \mathbf{V} to transform velocities expressed in the left or right cameras frames to the sensor frame. \mathbf{V} is given by [13]

$$\mathbf{V} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_{\times} \mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}, \quad (12)$$

where $[\mathbf{t}]_{\times}$ is the skew symmetric matrix associated to the vector \mathbf{t} and where $(\mathbf{R}, \mathbf{t}) \in SE(3)$ is the rigid body transformation from camera to sensor frame. The numerical values for these matrices are directly obtained from the calibration step of the stereovision system. Using this equation, we obtain

$$\mathbf{L}_{\mathbf{x}_s} = \begin{bmatrix} \mathbf{L}_{\mathbf{x}_l}' \mathbf{V}_s \\ \mathbf{L}_{\mathbf{x}_r}' \mathbf{V}_s \end{bmatrix}.$$

Note that $\mathbf{L}_{\mathbf{x}_s} \in \mathbb{R}^{4 \times 6}$ is always of rank 3 because of the epipolar constraint that links the perspective projection of a 3-D point in a stereovision system (see Figure 6). Another simple interpretation is that a 3-D point is represented by three independent parameters, making it impossible to find more than three independent parameters using any sensor observing that point.

To control the 6 DOF of the system, it is necessary to consider at least three points, the rank of the interaction matrix by considering only two points being equal to 5.

Using a stereovision system, since the 3-D coordinates of any point observed in both images can be easily estimated by a simple triangulation process it is possible and quite natural to use these 3-D coordinates in the features set \mathbf{s} . Such an approach would be, strictly speaking, a position-based approach, since it would require 3-D parameters in \mathbf{s} .

Position-Based Visual Servo

Position-based control schemes (PBVS) [2], [14], [15] use the pose of the camera with respect to some reference coordinate frame to define \mathbf{s} . Computing that pose from a set of measurements in one image necessitates the camera intrinsic parameters and the 3-D model of the object observed to be known. This classical computer vision problem is called the *3-D localization problem*. While this problem is beyond the scope of the present tutorial, many solutions have been presented in the literature (see, e.g., [16] and [17]).

It is then typical to define \mathbf{s} in terms of the parameterization used to represent the camera pose. Note that the parameters \mathbf{a} involved in the definition (1) of \mathbf{s} are now the camera intrinsic parameters and the 3-D model of the object.

It is convenient to consider three coordinate frames: the current camera frame \mathcal{F}_c , the desired camera frame \mathcal{F}_{c^*} , and a reference frame \mathcal{F}_o attached to the object. We adopt here the

standard notation of using a leading superscript to denote the frame with respect to which a set of coordinates is defined. Thus, the coordinate vectors ${}^c\mathbf{t}_o$ and ${}^{*}\mathbf{t}_o$ give the coordinates of the origin of the object frame expressed relative to the current camera frame and relative to the desired camera frame, respectively. Furthermore, let $\mathbf{R} = {}^{*}\mathbf{R}_c$ be the rotation matrix that gives the orientation of the current camera frame relative to the desired frame.

We can define \mathbf{s} to be $(\mathbf{t}, \theta\mathbf{u})$, in which \mathbf{t} is a translation vector, and $\theta\mathbf{u}$ gives the angle/axis parameterization for the rotation. We now discuss two choices for \mathbf{t} and give the corresponding control laws.

If \mathbf{t} is defined relative to the object frame \mathcal{F}_o , we obtain $\mathbf{s} = ({}^c\mathbf{t}_o, \theta\mathbf{u})$, $\mathbf{s}^* = ({}^{*}\mathbf{t}_o, \mathbf{0})$, and $\mathbf{e} = ({}^c\mathbf{t}_o - {}^{*}\mathbf{t}_o, \theta\mathbf{u})$. In this case, the interaction matrix related to \mathbf{e} is given by

$$\mathbf{L}_e = \begin{bmatrix} -\mathbf{I}_3 & [{}^c\mathbf{t}_o]_x \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}, \quad (13)$$

in which \mathbf{I}_3 is the 3×3 identity matrix and $\mathbf{L}_{\theta\mathbf{u}}$ is given by [18]:

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2} [\mathbf{u}]_x + \left(1 - \frac{\text{sinc } \theta}{\text{sinc}^2 \frac{\theta}{2}}\right) [\mathbf{u}]_x^2, \quad (14)$$

where $\text{sinc } x$ is the sinus cardinal defined such that $x \text{sinc } x = \sin x$ and $\text{sinc } 0 = 1$.

Following the developments presented at the beginning of the article, we obtain the control scheme

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}_e^{-1}} \mathbf{e},$$

since the dimension k of \mathbf{s} is 6, which is the number of camera degrees of freedom. By setting:

$$\widehat{\mathbf{L}_e^{-1}} = \begin{bmatrix} -\mathbf{I}_3 & [{}^c\mathbf{t}_o]_x \mathbf{L}_{\theta\mathbf{u}}^{-1} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}}^{-1} \end{bmatrix}, \quad (15)$$

we obtain after simple developments:

$$\begin{cases} v_c &= -\lambda(({}^{*}\mathbf{t}_o - {}^c\mathbf{t}_o) + [{}^c\mathbf{t}_o]_x \theta\mathbf{u}) \\ \omega_c &= -\lambda\theta\mathbf{u}, \end{cases} \quad (16)$$

since $\mathbf{L}_{\theta\mathbf{u}}$ is such that $\mathbf{L}_{\theta\mathbf{u}}^{-1} \theta\mathbf{u} = \theta\mathbf{u}$.

This PBVS scheme causes the rotational motion to follow a geodesic with an exponential decreasing speed and so that the translational parameters involved in \mathbf{s} decrease with the same speed. This explains the nice exponential decrease of the camera velocity components in Figure 7. Furthermore, the trajectory in the image of the origin of the object frame follows a pure straight line (here the center of the four points has been selected as this origin). On the other hand, the camera trajectory does not follow a straight line.

Another PBVS scheme can be designed by using $\mathbf{s} = ({}^c\mathbf{t}_c, \theta\mathbf{u})$. In that case, we have $\mathbf{s}^* = \mathbf{0}$, $\mathbf{e} = \mathbf{s}$, and

$$\mathbf{L}_e = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}. \quad (17)$$

Note the decoupling between translational and rotational motions, which allows us to obtain a simple control scheme:

$$\begin{cases} v_c &= -\lambda \mathbf{R}^\top {}^{*}\mathbf{t}_c \\ \omega_c &= -\lambda \theta\mathbf{u}. \end{cases} \quad (18)$$

In this case, as can be seen in Figure 8, if the pose parameters involved in (18) are perfectly estimated, the camera trajectory is a pure straight line, while the image trajectories are less satisfactory than before. Some particular configurations can even be found so that some points leave the camera field of view.

Stability Analysis

In this section, we consider the fundamental issues related to the stability of the controllers. To assess the stability of the closed-loop visual servo systems, we will use Lyapunov analysis. In particular, consider the candidate Lyapunov function defined by the squared error norm $\mathcal{L} = 1/2 \|\mathbf{e}(t)\|^2$, whose derivative is given by

$$\begin{aligned} \dot{\mathcal{L}} &= \mathbf{e}^\top \dot{\mathbf{e}} \\ &= -\lambda \mathbf{e}^\top \mathbf{L}_e \widehat{\mathbf{L}_e^{-1}} \mathbf{e}. \end{aligned}$$

The global asymptotic stability of the system is thus obtained when the following sufficient condition is ensured:

$$\mathbf{L}_e \widehat{\mathbf{L}_e^{-1}} > 0. \quad (19)$$

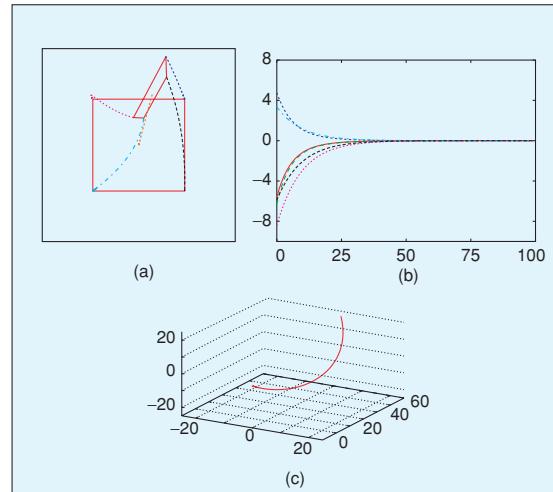


Figure 7. The system behavior using $\mathbf{s} = ({}^c\mathbf{t}_o, \theta\mathbf{u})$.

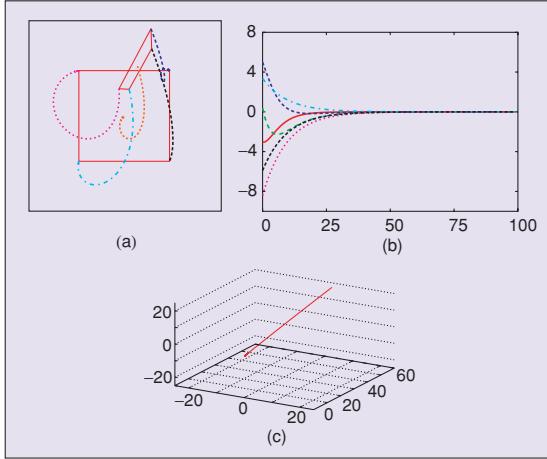


Figure 8. The system behavior using $\mathbf{s} = (\mathbf{t}_c^*, \theta_u)$.

If the number of features is equal to the number of camera degrees of freedom (i.e., $k = 6$), and if the features are chosen and the control scheme designed so that \mathbf{L}_e and $\widehat{\mathbf{L}}_e^+$ are of full rank 6, then condition (19) is ensured if the approximations involved in $\widehat{\mathbf{L}}_e^+$ are not too coarse.

We now consider the particularization of this to the specific cases of IBVS and PBVS.

Stability Analysis of IBVS

As discussed previously, for most IBVS approaches we have $k > 6$. Therefore, condition (19) can never be ensured since $\mathbf{L}_e \widehat{\mathbf{L}}_e^+ \in \mathbb{R}^{k \times k}$ is at most of rank 6; thus, $\mathbf{L}_e \widehat{\mathbf{L}}_e^+$ has a nontrivial null space. In this case, configurations such that $\mathbf{e} \in \text{Ker} \widehat{\mathbf{L}}_e^+$ correspond to local minima. Reaching such a local minimum is illustrated in Figure 9. As can be seen in Figure 9(d), each component of \mathbf{e} has a nice exponential decrease with the same convergence speed, causing straight-line trajectories to be realized in

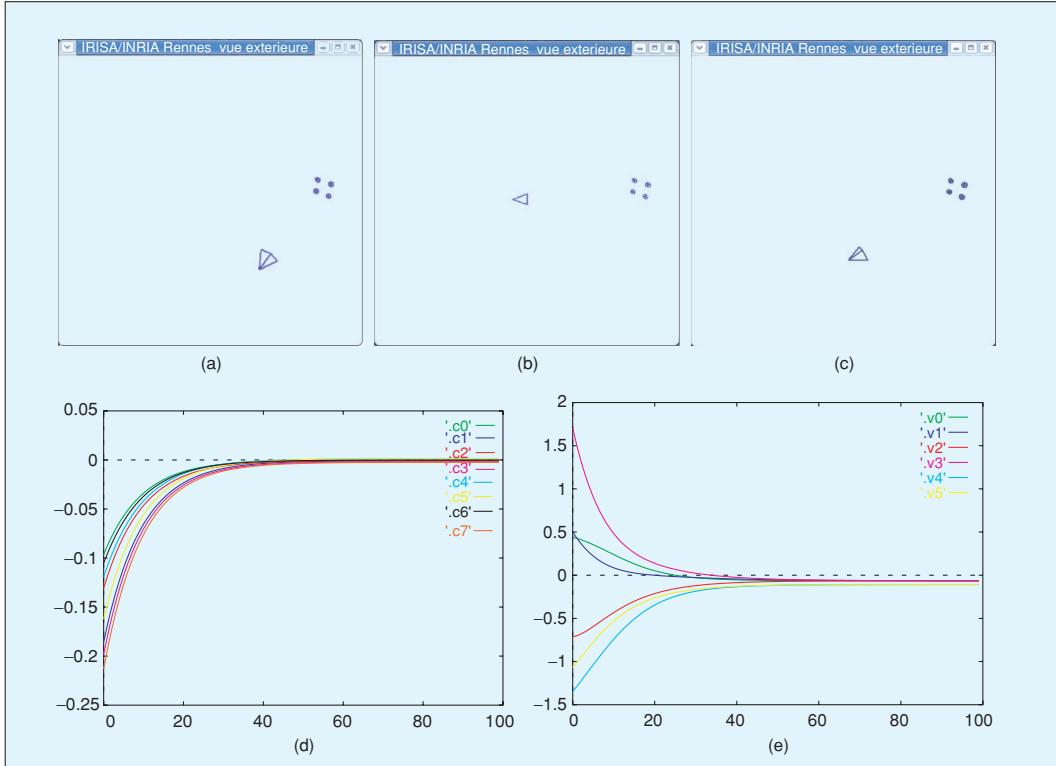


Figure 9. Reaching a local minimum using $\mathbf{s} = (x_1, y_1, \dots, x_4, y_4)$ and $\widehat{\mathbf{L}}_e^+ = \mathbf{L}_e^+$: (a) initial configuration, (b) the desired one, (c) the configuration reached after the convergence of the control scheme, (d) the evolution of the error \mathbf{e} at each iteration of the control scheme, and (f) the evolution of the six components of the camera velocity \mathbf{v}_c .

the image, but the error reached is not exactly zero, and it is clear from Figure 9(c) that the system has been attracted to a local minimum far away from the desired configuration. Thus, only local asymptotic stability can be obtained for IBVS.

To study local asymptotic stability when $k > 6$, let us first define a new error \mathbf{e}' with $\mathbf{e}' = \widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{e}$. The time derivative of this error is given by

$$\begin{aligned}\dot{\mathbf{e}}' &= \widehat{\mathbf{L}}_{\mathbf{e}}^+ \dot{\mathbf{e}} + \widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{e} \\ &= (\widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{L}_{\mathbf{e}} + \mathbf{O}) \mathbf{v}_c,\end{aligned}$$

where $\mathbf{O} \in \mathbb{R}^{6 \times 6}$ is equal to 0 when $\mathbf{e} = 0$, whatever the choice of $\widehat{\mathbf{L}}_{\mathbf{e}}^+$ [19]. Using the control scheme (5), we obtain:

$$\dot{\mathbf{e}}' = -\lambda (\widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{L}_{\mathbf{e}} + \mathbf{O}) \mathbf{e}',$$

which is known to be locally asymptotically stable in a neighborhood of $\mathbf{e} = \mathbf{e}^* = 0$ if

$$\widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{L}_{\mathbf{e}} > 0, \quad (20)$$

where $\widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{L}_{\mathbf{e}} \in \mathbb{R}^{6 \times 6}$. Indeed, only the linearized system $\dot{\mathbf{e}}' = -\lambda \widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{L}_{\mathbf{e}} \mathbf{e}'$ has to be considered if we are interested in the local asymptotic stability [20].

Once again, if the features are chosen and the control scheme designed so that $\mathbf{L}_{\mathbf{e}}$ and $\widehat{\mathbf{L}}_{\mathbf{e}}^+$ are of full rank 6, then condition (20) is ensured if the approximations involved in $\widehat{\mathbf{L}}_{\mathbf{e}}^+$ are not too coarse.

To end the demonstration of local asymptotic stability, we must show that there does not exist any configuration $\mathbf{e} \neq \mathbf{e}^*$ such that $\mathbf{e} \in \text{Ker } \widehat{\mathbf{L}}_{\mathbf{e}}^+$ in a small neighborhood of \mathbf{e}^* and in a small neighborhood of the corresponding pose \mathbf{p}^* . Such configurations correspond to local minima where $\mathbf{v}_c = 0$ and $\mathbf{e} \neq \mathbf{e}^*$. If such a pose \mathbf{p} would exist, it is possible to restrict

the neighborhood around \mathbf{p}^* so that there exists a camera velocity \mathbf{v} to reach \mathbf{p}^* from \mathbf{p} . This camera velocity would imply a variation of the error $\dot{\mathbf{e}} = \mathbf{L}_{\mathbf{e}} \mathbf{v}$. However, such a variation cannot belong to $\text{Ker } \widehat{\mathbf{L}}_{\mathbf{e}}^+$ since $\widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{L}_{\mathbf{e}} > 0$. Therefore, we have $\mathbf{v}_c = 0$ if and only if $\dot{\mathbf{e}} = 0$, i.e., $\mathbf{e} = \mathbf{e}^*$, in a neighborhood of \mathbf{p}^* .

Even though local asymptotic stability can be ensured when $k > 6$, we cannot ensure global asymptotic stability. For instance, as illustrated in Figure 9, there may exist local minima corresponding to configurations where $\mathbf{e} \in \text{Ker } \widehat{\mathbf{L}}_{\mathbf{e}}^+$, which are outside of the neighborhood considered above. Determining the size of the neighborhood where the stability and the convergence are ensured is still an open issue, even if this neighborhood is surprisingly quite large in practice.

Stability Analysis of PBVS

The stability properties of PBVS seem quite attractive. Since $\mathbf{L}_{\theta \mathbf{u}}$ given in (14) is nonsingular when $\theta \neq 2k\pi$, we obtain from (19) the global asymptotic stability of the system since $\mathbf{L}_{\theta \mathbf{u}} \mathbf{L}_{\theta \mathbf{u}}^{-1} = \mathbf{I}_6$, under the strong hypothesis that all the pose parameters are perfect. This is true for both PBVS methods presented previously, since the interactions matrices given in (13) and (17) are full rank when $\mathbf{L}_{\theta \mathbf{u}}$ is nonsingular.

With regard to robustness, feedback is computed using *estimated* quantities that are a function of the image measurements and the system calibration parameters. For the first PBVS method (the analysis for the second method is analogous), the interaction matrix given in (13) corresponds to perfectly estimated pose parameters, while the real one is unknown since the estimated pose parameters may be biased due to calibration errors, or inaccurate and unstable due to noise [11]. The true positivity condition (19) should be in fact written:

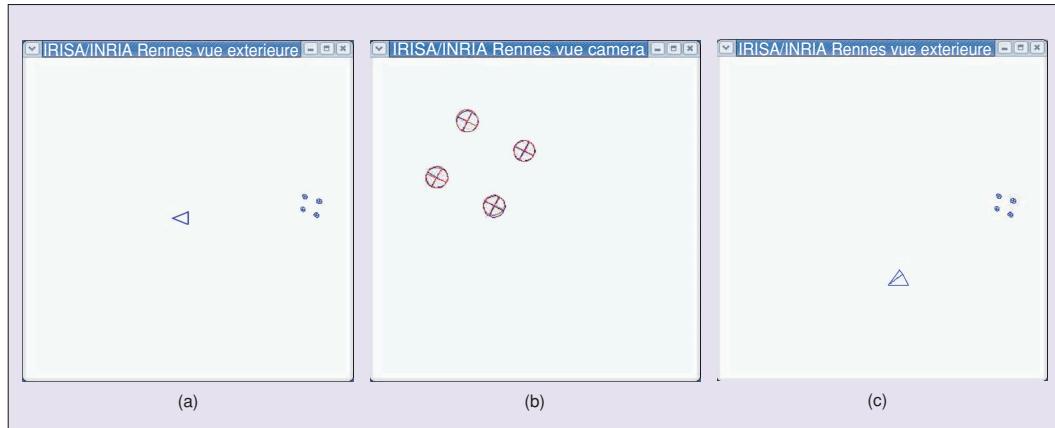


Figure 10. Two different camera poses [(a) and (c)] that provide almost the same image of four coplanar points (b).

$$\widehat{\mathbf{L}_e} \widehat{\mathbf{L}_e^{-1}} > 0, \quad (21)$$

where $\widehat{\mathbf{L}_e^{-1}}$ is given by (15) but where \mathbf{L}_e is unknown and not given by (13). Indeed, even small errors in computing the points position in the image can lead to pose errors that can impact significantly the accuracy and the stability of the system (see Figure 10).

Conclusions

At this point, one might wonder which of IBVS or PBVS is the superior control strategy. As with most engineering questions, there is no definitive answer—only a set of performance tradeoffs to consider.

As for stability issues, we have seen that no strategy provides perfect properties. As for the 3-D parameters involved, a correct estimation is important in IBVS, since they appear in \mathbf{L}_e and thus in the stability condition (20). Poor estimation can thus make the system unstable, but coarse estimations will only imply perturbations in the trajectory performed by the robot to reach its desired pose and will have no effect on the accuracy of the pose reached. On the other hand, a correct estimation of the pose is crucial in PBVS, since it appears both in the error \mathbf{e} to be regulated to 0 and in \mathbf{L}_e . Coarse estimations will thus induce perturbations on the trajectory realized but will have also an effect on the accuracy of the pose reached after convergence.

In fact, in PBVS, the vision sensor is considered as a 3-D sensor. Since the control scheme imposes a behavior of \mathbf{s} , which is here expressed in the Cartesian space, it allows the camera to follow theoretically an optimal trajectory in that space but generally not in the image space. When only one image is used, even small errors in the image measurements can lead to errors in the pose that can impact significantly the accuracy of the system. The main question with this approach is thus: Can a vision sensor be considered as a 3-D sensor?

On the other hand, in IBVS, the vision sensor is considered as a two-dimensional (2-D) sensor since the features are directly expressed in the image space. That is more realistic when a monocular camera is used, and this allows IBVS to be remarkably robust to errors in calibration and image noise. However, IBVS is not without its shortcomings. When the displacement to realize is large, the camera may reach a local minimum or may cross a singularity of the interaction matrix. Furthermore, the camera motion may follow unpredictable, often suboptimal Cartesian trajectories, such as those discussed previously.

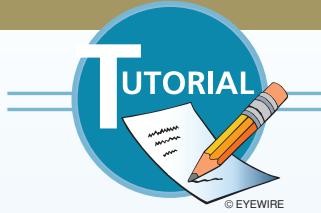
Even if the two basic approaches presented and discussed in this article give in practice satisfactory results in most cases, their respective shortcomings have led to many works and improvements. Results in computer vision about 3-D reconstruction from two views have also been applied successfully, making the knowledge of the 3-D object model unnecessary. These different approaches will form one of the main points presented in Part II of this tutorial.

References

- [1] J. Feddema and O. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 691–700, Oct. 1989.
- [2] L. Weiss, A. Sanderson, and C. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE J. Robot. Automat.*, vol. 3, pp. 404–417, Oct. 1987.
- [3] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2003.
- [4] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. New York: Springer-Verlag, 2003.
- [5] H. Michel and P. Rives, "Singularities in the determination of the situation of a robot effector from the perspective view of three points," INRIA Research Report, Tech. Rep. 1850, Feb. 1993.
- [6] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981.
- [7] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 651–670, Oct. 1996.
- [8] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robotics and Automation*, vol. 8, pp. 313–326, June 1992.
- [9] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 1843–1848, Apr. 2004.
- [10] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. Robot. Automat.*, vol. 17, no. 4, pp. 507–515, 2001.
- [11] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, vol. 237, *Lecture Notes in Control and Information Sciences*, D. Kriegman, G. Hager, and S. Morse, Eds. New York: Springer-Verlag, 1998, pp. 66–78.
- [12] G. Hager, W. Chang, and A. Morse, "Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination," *IEEE Control Syst. Mag.*, vol. 15, pp. 30–39, Feb. 1995.
- [13] R. Paul, *Robot Manipulators: Mathematics, Programming and Control*. Cambridge, MA: MIT Press, 1982.
- [14] W. Wilson, C. Hulls, and G. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 684–696, Oct. 1996.
- [15] B. Thuijls, P. Martinet, L. Cordesses, and J. Gallice, "Position based visual servoing: Keeping the object in the field of vision," in *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 1624–1629, May 2002.
- [16] D. Dementhon and L. Davis, "Model-based object pose in 25 lines of code," *Int. J. Comput. Vis.*, vol. 15, pp. 123–141, June 1995.
- [17] D. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artif. Intell.*, vol. 31, no. 3, pp. 355–395, 1987.
- [18] E. Malis, F. Chaumette, and S. Bouquet, "2-1/2D visual servoing," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 238–250, Apr. 1999.
- [19] E. Malis, "Visual servoing invariant to changes in camera intrinsic parameters," *IEEE Trans. Robot. Automat.*, vol. 20, pp. 72–81, Feb. 2004.
- [20] A. Isidori, *Nonlinear Control Systems*, 3rd ed. New York: Springer-Verlag, 1995.

Address for Correspondence: François Chaumette, IRISA/INRIA Rennes, Campus de Beaulieu, 35 042 Rennes cedex, France. E-mail: Francois.Chaumette@irisa.fr.

9.6 *Visual Servo Control Part II: Advanced Approaches, RAM,
2007*



Visual Servo Control

Part II: Advanced Approaches

This article is the second of a two-part tutorial on visual servo control. In Part I (*IEEE Robotics and Automation Magazine*, vol. 13, no. 4), we introduced fundamental concepts and described basic approaches. Here we discuss more advanced concepts, and present a number of recent approaches.

As described in Part I of the tutorial, visual servo schemes rely on the relationship

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v}_c \quad (1)$$

in which \mathbf{s} is a set of geometrical features whose time derivative $\dot{\mathbf{s}}$ is linearly related to the spatial velocity \mathbf{v}_c of the camera through the interaction matrix \mathbf{L}_s . Using this relationship, control schemes are designed to minimize the error \mathbf{e} between the current value of the visual feature \mathbf{s} and its desired value \mathbf{s}^* : $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$.

A classical proportional control scheme is given by:

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}}_e^+ \mathbf{e}, \quad (2)$$

where \mathbf{L}_e is defined by

$$\dot{\mathbf{e}} = \mathbf{L}_e \mathbf{v}_c, \quad (3)$$

and $\widehat{\mathbf{L}}_e^+$ is an approximation of the pseudo-inverse of \mathbf{L}_e . An approximation must be used because visual servo schemes require the values of three-dimensional (3-D) parameters that are not available directly from the image measurements. Recall that for position-based visual servo (PBVS) schemes 3-D parameters appear both in the error \mathbf{e} and in the interaction matrix, while for basic image-based visual servo

(IBVS), the depth of each point considered appears in the coefficients of the interaction matrix related to the translational motions. This is the case even when $\widehat{\mathbf{L}}_e^+ = \widehat{\mathbf{L}}_{e^*}^+$ is used in the control scheme, although in this case only the depth Z^* of each point for the desired pose is needed, which is generally not difficult to obtain in practice. In all other cases, an estimation of the current depth must be made at each iteration of the control scheme.

We begin Part II of the tutorial by describing two approaches to estimating the interaction matrix. First we describe how epipolar geometry can be used to estimate the 3-D parameters, which can then be used to construct the interaction matrix. We then describe how it is possible to estimate directly the numerical value of $\widehat{\mathbf{L}}_e^+$. With these methods in hand, we then present more advanced techniques in visual servo control. These techniques aim to compensate for the relative shortcomings of the PBVS and IBVS methods. We then consider target tracking tasks, that is, tasks for which the target object is not stationary. Finally, we present a generalization of the modeling issues that allows one to consider both eye-in-hand systems and eye-to-hand systems.

Estimation of 3-D Parameters

If a calibrated stereo vision system is used, all 3-D parameters can be easily determined by triangulation, as evoked in Part I of the tutorial. Similarly, if a 3-D model of the object is known, all 3-D parameters can be computed from a pose estimation algorithm. However, such an estimation can be quite unstable due to image noise. It is also possible to estimate 3-D parameters by using the epipolar geometry that relates the images of the same scene observed from different viewpoints. Indeed, in visual servoing, two images are generally available:

BY FRANÇOIS CHAUMETTE AND SETH HUTCHINSON

Epipolar geometry can be used to estimate the 3-D parameters, which can then be used to construct the interaction matrix.

the current one and the desired one. In contrast, we can estimate the interaction matrix directly as image measurements are made during the visual servoing task. Here, we discuss both approaches.

Epipolar Geometry

Given a set of matches between the image measurements in the current image and in the desired one, the fundamental matrix, or the essential matrix if the camera is calibrated, can be recovered [1], and then used in visual servoing [2]. Indeed, from the essential matrix, the rotation and the translation up to a scalar factor between the two views can be estimated. However, near the convergence of the visual servo, that is when the current and desired images are similar, the epipolar geometry becomes degenerate and it is not possible to estimate accurately the partial pose between the two views. For this reason, using homography is generally preferred.

Let \mathbf{x}_i and \mathbf{x}_i^* denote the homogeneous image coordinates for a point in the current and desired images. Then \mathbf{x}_i is related to \mathbf{x}_i^* by

$$\mathbf{x}_i = \mathbf{H}_i \mathbf{x}_i^*,$$

in which \mathbf{H}_i is a homography matrix.

If all feature points lie on a 3-D plane, then there is a single homography matrix \mathbf{H} such that $\mathbf{x}_i = \mathbf{H}\mathbf{x}_i^*$ for all i . This homography can be estimated using the position of four matched points in the desired and the current images. If all the features points do not belong to the same 3-D plane, then three points can be used to define such a plane and five supplementary points are needed to estimate \mathbf{H} [3].

Once \mathbf{H} is available, it can be decomposed as

$$\mathbf{H} = \mathbf{R} + \frac{\mathbf{t}}{d^*} \mathbf{n}^{*\top}, \quad (4)$$

in which \mathbf{R} is the rotation matrix relating the orientation of the current and desired camera frames, \mathbf{n}^* is the normal to the chosen 3-D plane expressed in the desired frame, d^* is the distance to the 3-D plane from the desired frame, and \mathbf{t} is the translation between current and desired frames. From \mathbf{H} , it is thus possible to recover \mathbf{R} , \mathbf{t}/d^* , and \mathbf{n} . In fact, two solutions for these quantities exist [4], but it is quite easy to select the correct one using some knowledge about the desired pose. It is also possible to estimate the depth of any target point up to a common scale factor [5]. The unknown depth of each point that appears in classical IBVS can thus be expressed as a function of a single constant parameter. Similarly, the pose para-

meters required by PBVS can be recovered up to a scalar factor as for the translation term. The PBVS schemes described in Part I of the tutorial can thus be revisited using this approach, with the new error defined as the translation up to a scalar factor and the angle/axis parameterization of the rotation. This approach has also been used for the hybrid visual servoing schemes described in the next section.

Direct Estimation

The approach described previously can be used to estimate the unknown 3-D parameters that appear in the analytical form of the interaction matrix. It is also possible to estimate directly its numerical value using either an off-line learning step, or an on-line estimation scheme. This idea is only useful for IBVS since, for PBVS, all the coefficients of the interaction matrix are directly obtained from the features \mathbf{s} used in the control scheme.

All the methods proposed to estimate numerically the interaction matrix rely on observations of the variation of features due to a known or measured camera motion. More precisely, if we measure a feature's variation $\Delta\mathbf{s}$ due to a camera motion $\Delta\mathbf{v}_c$, we have from (1)

$$\mathbf{L}_s \Delta\mathbf{v}_c = \Delta\mathbf{s},$$

which provides k equations, while we have $k \times 6$ unknown values in \mathbf{L}_s . Using a set of N independent camera motions with $N > 6$, it is thus possible to estimate \mathbf{L}_s by solving

$$\mathbf{L}_s \mathbf{A} = \mathbf{B},$$

where the columns of $\mathbf{A} \in \mathbb{R}^{6 \times N}$ and $\mathbf{B} \in \mathbb{R}^{k \times N}$ are respectively formed with the set of camera motions and the set of corresponding features variations. The least squares solution is of course given by

$$\widehat{\mathbf{L}}_s = \mathbf{B}\mathbf{A}^+. \quad (5)$$

Methods based on neural networks have also been developed to estimate \mathbf{L}_s [6], [7]. It is also possible to estimate directly the numerical value of \mathbf{L}_s^+ , which provides in practice a better behavior [8]. In that case, the basic relation is

$$\mathbf{L}_s^+ \Delta\mathbf{s} = \Delta\mathbf{v}_c,$$

which provides six equations. Using a set of N measurements, with $N > k$, we now obtain

$$\widehat{\mathbf{L}}_s^+ = \mathbf{A}\mathbf{B}^+. \quad (6)$$

In the first case (5), the six columns of \mathbf{L}_s are estimated by solving six linear systems, while in the second case (6), the k columns of \mathbf{L}_s^+ are estimated by solving k linear systems, which explains the difference in the results.

Estimating the interaction matrix online can be viewed as an optimization problem, and consequently a number of

researchers have investigated approaches that derive from optimization methods. These methods typically discretize the system equation (1), and use an iterative updating scheme to refine the estimate of $\widehat{\mathbf{L}}_s$ at each stage. One such online and iterative formulation uses the Broyden update rule given by [9], [10]

$$\widehat{\mathbf{L}}_s(t+1) = \widehat{\mathbf{L}}_s(t) + \frac{\alpha}{\Delta \mathbf{v}_c^\top \Delta \mathbf{v}_c} (\Delta \mathbf{x} - \widehat{\mathbf{L}}_s(t) \Delta \mathbf{v}_c) \Delta \mathbf{v}_c^\top,$$

where α defines the update speed. This method has been generalized to the case of moving objects in [11].

The main interest of using such numerical estimations in the control scheme is that it avoids all the modeling and calibration steps. It is particularly useful when using features whose interaction matrix is not available in analytical form. For instance, in [12], the main eigenvalues of the Principal Component Analysis of an image have been considered in a visual servoing scheme. The drawbacks of these methods is that no theoretical stability and robustness analysis can be made.

Advanced Visual Servo Control Schemes

We now describe visual servo control schemes which have been proposed to improve the behavior of basic IBVS and PBVS. The first ones combine the respective advantages of these schemes while trying to avoid their shortcomings.

Hybrid Visual Servo

Suppose we have access to a clever control law for ω_c , such as the one used in PBVS

$$\omega_c = -\lambda \theta \mathbf{u}, \quad (7)$$

where $\theta \mathbf{u}$ is obtained either from a pose estimation algorithm if the 3-D model of the object is available, or from the partial pose estimated using the epipolar geometry or the homography. How could we use this in conjunction with traditional IBVS?

Considering a feature vector \mathbf{s}_t and an error \mathbf{e}_t devoted to control the translational degrees of freedom, we can partition the interaction matrix as follows:

$$\begin{aligned} \dot{\mathbf{s}}_t &= \mathbf{L}_{\mathbf{s}_t} \mathbf{v}_c \\ &= [\mathbf{L}_v \quad \mathbf{L}_\omega] \begin{bmatrix} \mathbf{v}_c \\ \omega_c \end{bmatrix} \\ &= \mathbf{L}_v \mathbf{v}_c + \mathbf{L}_\omega \omega_c. \end{aligned}$$

Now, setting $\dot{\mathbf{e}}_t = -\lambda \mathbf{e}_t$, we can solve for the desired translational control input as

$$\begin{aligned} -\lambda \mathbf{e}_t &= \dot{\mathbf{e}}_t = \dot{\mathbf{s}}_t = \mathbf{L}_v \mathbf{v}_c + \mathbf{L}_\omega \omega_c \\ \Rightarrow \mathbf{v}_c &= -\mathbf{L}_v^+ (\lambda \mathbf{e}_t + \mathbf{L}_\omega \omega_c). \end{aligned} \quad (8)$$

We can think of the quantity $(\lambda \mathbf{e}_t + \mathbf{L}_\omega \omega_c)$ as a modified error term, one that combines the original error with the error that would be induced by the rotational motion due to

ω_c . The translational control input $\mathbf{v}_c = -\mathbf{L}_v^+ (\lambda \mathbf{e}_t + \mathbf{L}_\omega \omega_c)$ will drive this error to zero.

The method known as two and one-half-dimensional (2 1/2-D) Visual Servo [13] was the first to exploit such a partitioning in combining IBVS and PBVS. More precisely, in [13], \mathbf{s}_t has been selected as the coordinates of an image point, and the logarithm of its depth, so that \mathbf{L}_v is a triangular always invertible matrix. More precisely, we have $\mathbf{s}_t = (\mathbf{x}, \log Z)$, $\mathbf{s}_t^* = (\mathbf{x}^*, \log Z^*)$, $\mathbf{e}_t = (\mathbf{x} - \mathbf{x}^*, \log \rho_Z)$ where $\rho_Z = Z/Z^*$, and

$$\begin{aligned} \mathbf{L}_v &= \frac{1}{Z^* \rho_Z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix} \\ \mathbf{L}_\omega &= \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \\ -y & x & 0 \end{bmatrix}. \end{aligned}$$

Note that the ratio ρ_Z can be obtained directly from the partial pose estimation algorithm described previously.

If we come back to the usual global representation of visual servo control schemes, we have $\mathbf{e} = (\mathbf{e}_t, \theta \mathbf{u})$ and \mathbf{L}_e given by

$$\mathbf{L}_e = \begin{bmatrix} \mathbf{L}_v & \mathbf{L}_\omega \\ \mathbf{0} & \mathbf{L}_{\theta \mathbf{u}} \end{bmatrix},$$

from which we obtain immediately the control law (7) and (8) by applying (2).

If we consider the example chosen in Part I of the tutorial to compare the behavior of the different control schemes, the results obtained using (8) are given in Figure 1. Here, the point that has been considered in \mathbf{s}_t is the center of gravity \mathbf{x}_g of the target. Note that the image trajectory of this point is a straight line as expected, and that there is a nice decrease of the camera velocity components, which makes this scheme very near the first PBVS approach.

As for stability, it is clear that this scheme is globally asymptotically stable in perfect conditions (refer to the stability

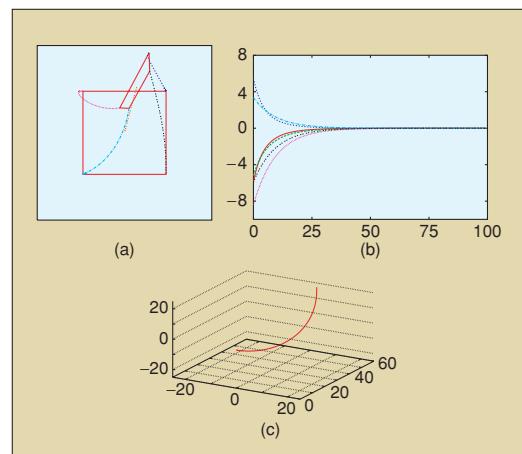


Figure 1. System behavior using $\mathbf{s} = (\mathbf{x}_g, \log(Z_g), \theta \mathbf{u})$.

analysis provided in the Part I of the tutorial: we are here in the simple case $k = 6$). Furthermore, thanks to the triangular form of the interaction matrix \mathbf{L}_e , it has been possible to analyze the stability of this scheme in the presence of calibration errors using the homography estimation approach [14]. Finally, the only unknown constant parameter involved in this scheme, that is Z^* , can be estimated on line using adaptive techniques [15].

Other hybrid schemes can be designed. For instance, in [16], the third component of \mathbf{s}_t is different and has been selected so that all the target points remain in the camera field of view as far as possible. Another example has been proposed in [17]. In that case, \mathbf{s} is selected as $\mathbf{s} = ({}^c\mathbf{t}_c, \mathbf{x}_g, \theta u_z)$ which provides with a block-triangular interaction matrix of the form:

$$\mathbf{L}_e = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{L}'_v & \mathbf{L}'_\omega \end{bmatrix},$$

where \mathbf{L}'_v and \mathbf{L}'_ω can easily be computed. This scheme is so that, in perfect conditions, the camera trajectory is a straight line (since ${}^c\mathbf{t}_c$ is a part of \mathbf{s}), and the image trajectory of the center of gravity of the object is also a straight line (since \mathbf{x}_g is also a part of \mathbf{s}). The translational camera degrees of freedom are devoted to realize the 3-D straight line, while the rotational camera degrees of freedom are devoted to realize the two-dimensional (2-D) straight line and compensate also the 2-D motion of \mathbf{x}_g due to the translational motion. As can be seen in Figure 2, this scheme is particularly satisfactory in practice.

Finally, it is possible to combine differently 2-D and 3-D features. For instance, in [18], it has been proposed to use in \mathbf{s} the 2-D homogeneous coordinates of a set of image points expressed in pixels multiplied by their corresponding depth: $\mathbf{s} = (u_1 Z_1, v_1 Z_1, Z_1, \dots, u_n Z_n, v_n Z_n, Z_n)$. As for classical IBVS, we obtain in that case a set of redundant features, since

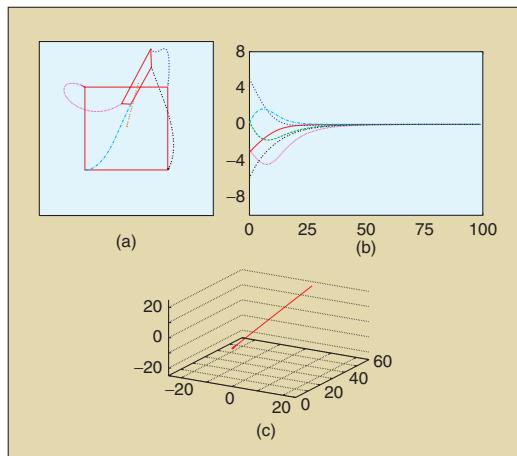


Figure 2. System behavior using $\mathbf{s} = ({}^c\mathbf{t}_c, \mathbf{x}_g, \theta u_z)$.

at least three points have to be used to control the six camera degrees of freedom (we here have $k \geq 9$). However, it has been demonstrated in [19] that this selection of redundant features is free of attractive local minima.

Partitioned Visual Servo

The hybrid visual servo schemes described previously have been designed to decouple the rotational motions from the translational ones by selecting adequate visual features defined in part in 2-D, and in part in 3-D (that is why they have been called 2 1/2-D visual servoing). This work has inspired some researchers to find features that exhibit similar decoupling properties but using only features expressed directly in the image. More precisely, the goal is to find six features so that each of them is related to only one degree of freedom (in which case the interaction matrix is a diagonal matrix), the Grail being to find a diagonal interaction matrix whose elements are constant, as near as possible of the identity matrix, leading to a pure, direct, and simple linear control problem.

The first work in this area partitioned the interaction matrix so as to isolate motion related to the optic axis [20]. Indeed, whatever the choice of \mathbf{s} , we have

$$\begin{aligned} \dot{\mathbf{s}} &= \mathbf{L}_s \mathbf{v}_c \\ &= \mathbf{L}_{xy} \mathbf{v}_{xy} + \mathbf{L}_z \mathbf{v}_z \\ &= \dot{\mathbf{s}}_{xy} + \dot{\mathbf{s}}_z, \end{aligned}$$

in which \mathbf{L}_{xy} includes the first, second, fourth and fifth columns of \mathbf{L}_s , and \mathbf{L}_z includes the third and sixth columns of \mathbf{L}_s . Similarly, $\mathbf{v}_{xy} = (v_x, v_y, \omega_x, \omega_y)$ and $\mathbf{v}_z = (v_z, \omega_z)$. Here, $\dot{\mathbf{s}}_z = \mathbf{L}_z \mathbf{v}_z$ gives the component of $\dot{\mathbf{s}}$ due to the camera motion along and rotation about the optic axis, while $\dot{\mathbf{s}}_{xy} = \mathbf{L}_{xy} \mathbf{v}_{xy}$ gives the component of $\dot{\mathbf{s}}$ due to velocity along and rotation about the camera x and y axes.

Proceeding as before, by setting $\dot{\mathbf{e}} = -\lambda \mathbf{e}$ we obtain

$$-\lambda \mathbf{e} = \dot{\mathbf{e}} = \dot{\mathbf{s}} = \mathbf{L}_{xy} \mathbf{v}_{xy} + \mathbf{L}_z \mathbf{v}_z,$$

which leads to

$$\mathbf{v}_{xy} = -\mathbf{L}_{xy}^+ (\lambda \mathbf{e}(t) + \mathbf{L}_z \mathbf{v}_z).$$

As before, we can consider $(\lambda \mathbf{e}(t) + \mathbf{L}_z \mathbf{v}_z)$ as a modified error that incorporates the original error while taking into account the error that will be induced by \mathbf{v}_z .

Given this result, all that remains is to choose \mathbf{s} and \mathbf{v}_z . As for basic IBVS, the coordinates of a collection of image points can be used in \mathbf{s} , while two new image features can be defined to determine \mathbf{v}_z .

- ◆ Define α , with $0 \leq \alpha < 2\pi$ as the angle between the horizontal axis of the image plane and the directed line segment joining two feature points. It is clear that α is closely related to the rotation around the optic axis.
- ◆ Define σ^2 to be the area of the polygon defined by these points. Similarly, σ^2 is closely related to the translation along the optic axis.

Using these features, \mathbf{v}_z has been defined in [20] as

$$\begin{cases} v_z &= \lambda_{v_z} \ln \frac{\sigma^*}{\sigma} \\ \omega_z &= \lambda_{\omega_z} (\alpha^* - \alpha). \end{cases}$$

IBVS with Cylindrical Coordinates of Image Points

As proposed in [21], another option is to use the cylindrical coordinates (ρ, θ) of the image points instead of their Cartesian coordinates (x, y) . They are defined by

$$\rho = \sqrt{x^2 + y^2} \quad \theta = \arctan \frac{y}{x} \quad (9)$$

as long as $\rho \neq 0$, in which case θ is not defined. We deduce from (9):

$$\dot{\rho} = (\dot{x}\dot{x} + \dot{y}\dot{y})/\rho \quad \dot{\theta} = (x\dot{y} - y\dot{x})/\rho^2.$$

Using the interaction matrix \mathbf{L}_x given in Part I of the tutorial to express \dot{x} and \dot{y} , and then substituting x by $\rho \cos \theta$ and y by $\rho \sin \theta$, we obtain immediately

$$\begin{aligned} \mathbf{L}_\rho &= \left[\begin{array}{cccc} -\frac{c}{Z} & -\frac{s}{Z} & \frac{\rho}{Z} & (1+\rho^2)s & -(1+\rho^2)c & 0 \end{array} \right] \\ \mathbf{L}_\theta &= \left[\begin{array}{cccc} \frac{s}{\rho Z} & \frac{-c}{\rho Z} & 0 & c/\rho & s/\rho & -1 \end{array} \right], \end{aligned}$$

where $c = \cos \theta$ and $s = \sin \theta$. By looking at the third and sixth column of \mathbf{L}_ρ and \mathbf{L}_θ , we can note that ρ is invariant with respect to ω_z while θ is invariant with respect to v_z and linearly related to ω_z , as were the two image features σ and α chosen previously. This explains why the behavior obtained in that case is quite satisfactory. Indeed, by considering the same example as before, and using now $\mathbf{s} = (\rho_1, \theta_1, \dots, \rho_4, \theta_4)$ and $\widehat{\mathbf{L}}_e^+ = \mathbf{L}_{e^*}^+$ (that is a constant matrix), the system behavior shown in Figure 3 has the same nice properties as using $\widehat{\mathbf{L}}_e^+ = (\mathbf{L}_e/2 + \mathbf{L}_{e^*}/2)^+$ with $\mathbf{s} = (x_1, y_1, \dots, x_4, y_4)$. If we go back to the example used in the geometrical interpretation of IBVS described in Part I of the tutorial, the behavior obtained will also be as expected, thanks to the decoupling between the third and sixth columns of the interaction matrix.

Advanced IBVS Schemes

Up until now, for all IBVS schemes presented, we have primarily considered image point coordinates in \mathbf{s} . Other geometrical primitives can of course be used. There are several reasons to do so. First, the scene observed by the camera cannot always be described merely by a collection of points, in which case the image processing provides other types of measurements, such as a set of straight lines or the contours of an object. Second, richer geometric primitives may ameliorate the decoupling and linearizing issues that motivate the design of partitioned systems. Finally, the robotic task to be achieved may be expressed in terms of virtual linkages (or fixtures) between the camera and the observed objects [22], [23], sometimes expressed directly by constraints between primitives, such as for instance point-to-line [24] (which means that an observed point must lie on a specified line).

The main interest of using such numerical estimations in the control scheme is that it avoids all the modeling and calibration steps.

For the first point, it is possible to determine the interaction matrix related to the perspective projection of a large class of geometrical primitives, such as segments, straight lines, spheres, circles, and cylinders. The results are given in [25] and [22]. Recently, the analytical form of the interaction matrix related to any image moments corresponding to planar objects has been computed. This makes possible to consider planar objects of any shape [26]. If a collection of points is measured in the image, moments can also be used [27]. In both cases, moments allow the use of intuitive geometrical features, such as the center of gravity or the orientation of an object. By selecting an adequate combination of moments, it is then possible to determine partitioned systems with good decoupling and linearizing properties [26], [27].

Note that for all these features (geometrical primitives, moments), the depth of the primitive or of the object considered appears in the coefficients of the interaction matrix related to the translational degrees of freedom, as was the case for the image points. An estimation of this depth is thus generally still necessary. Few exceptions occur, using for instance an adequate normalization of moments, which allows in some particular cases to make only the constant desired depth appear in the interaction matrix [27].

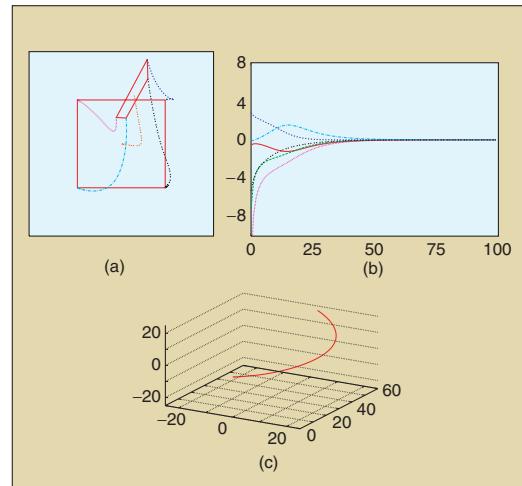


Figure 3. System behavior using $\mathbf{s} = (\rho_1, \theta_1, \dots, \rho_4, \theta_4)$ and $\widehat{\mathbf{L}}_e^+ = \mathbf{L}_{e^*}^+$.

Performance Optimization and Planning

In some sense, partitioned methods represent an effort to optimize system performance by assigning distinct features and controllers to individual degrees of freedom. In this way, the designer performs a sort of off-line optimization when allocating controllers to degrees of freedom. It is also possible to explicitly design controllers that optimize various system performance measures. We describe a few of these in this section.

Optimal Control and Redundancy Framework

An example of such an approach is given in [28] and [29], in which linear quadratic Gaussian (LQG) control design is used to choose gains that minimize a linear combination of state and control input. This approach explicitly balances the trade-off between tracking errors (since the controller attempts to drive $\mathbf{s} - \mathbf{s}^*$ to zero) and robot motion. A similar control approach is proposed in [30] where joint limit avoidance is considered simultaneously with the positioning task.

It is also possible to formulate optimality criteria that explicitly express the observability of robot motion in the image. For example, the singular value decomposition of the interaction matrix reveals which degrees of freedom are most apparent and can thus be easily controlled, while the condition number of the interaction matrix gives a kind of global measure of the visibility of motion. This concept has been called resolvability in [31] and motion perceptibility in [32]. By selecting features and designing controllers that maximize these measures, either along specific degrees of freedom or globally, the performance of the visual servo system can be improved.

The constraints considered to design the control scheme using the optimal control approach may be contradictory in

some cases, leading the system to fail due to local minima in the objective function to be minimized. For example, it may happen that the motion produced to move away from a robot joint limit is exactly the opposite of the motion produced to near the desired pose, which results in a zero global motion. To avoid this potential problem, it is possible to use the gradient projection method, which is classical in robotics. Applying this method to visual servoing has been proposed in [25] and [23]. The approach consists in projecting the secondary constraints \mathbf{e}_s on the null space of the vision-based task \mathbf{e} so that they have no effect on the regulation of \mathbf{e} to 0

$$\mathbf{e}_g = \widehat{\mathbf{L}}_{\mathbf{e}}^+ \mathbf{e} + \mathbf{P}_{\mathbf{e}} \mathbf{e}_s,$$

where \mathbf{e}_g is the new global task considered and $\mathbf{P}_{\mathbf{e}} = (\mathbf{I}_6 - \widehat{\mathbf{L}}_{\mathbf{e}}^+ \widehat{\mathbf{L}}_{\mathbf{e}})$ is such that $\widehat{\mathbf{L}}_{\mathbf{e}} \mathbf{P}_{\mathbf{e}} \mathbf{e}_s = 0, \forall \mathbf{e}_s$. Avoiding the robot joint limits using this approach has been presented in [33]. However, when the vision-based task constrains all the camera degrees of freedom, the secondary constraints cannot be considered since, when $\widehat{\mathbf{L}}_{\mathbf{e}}$ is of full rank 6, we have $\mathbf{P}_{\mathbf{e}} \mathbf{e}_s = 0, \forall \mathbf{e}_s$. In that case, it is necessary to inject the constraints in a global objective function, such as navigation functions which are free of local minima [34], [35].

Switching Schemes

The partitioned methods described previously attempt to optimize performance by assigning individual controllers to specific degrees-of-freedom. Another way to use multiple controllers to optimize performance is to design switching schemes that select at each moment in time which controller to use based on criteria to be optimized.

A simple switching controller can be designed using an IBVS and a PBVS controller as follows [36]. Let the system begin by using the IBVS controller. Consider the Lyapunov function for the PBVS controller given by $\mathcal{L} = \frac{1}{2} \|\mathbf{e}(t)\|^2$, with $\mathbf{e}(t) = (\mathbf{t}_o - \mathbf{t}_s, \theta \mathbf{u})$. If at any time the value of this Lyapunov function exceeds a threshold γ_P , the system switches to the PBVS controller. While using the PBVS controller, if at any time the value of the Lyapunov function for the IBVS controller exceeds a threshold, $\mathcal{L} = \frac{1}{2} \|\mathbf{e}(t)\|^2 > \gamma_I$, the system switches to the IBVS controller. With this scheme, when the Lyapunov function for a particular controller exceeds a threshold, that controller is invoked, which in turn reduces the value of the corresponding Lyapunov function. If the switching thresholds are selected appropriately, the system is able to exploit the relative advantages of IBVS and PBVS, while avoiding their shortcomings.

An example for this system is shown in Figure 4, for the case of a rotation by 160° about the optical axis. Note that the system begins in IBVS mode and the features initially move on straight lines toward their goal positions in the image. However, as the camera retreats, the system switches to PBVS, which allows the camera to reach its desired position by combining a rotational motion around its optic axis and a forward translational motion, producing the circular trajectories observed in the image.

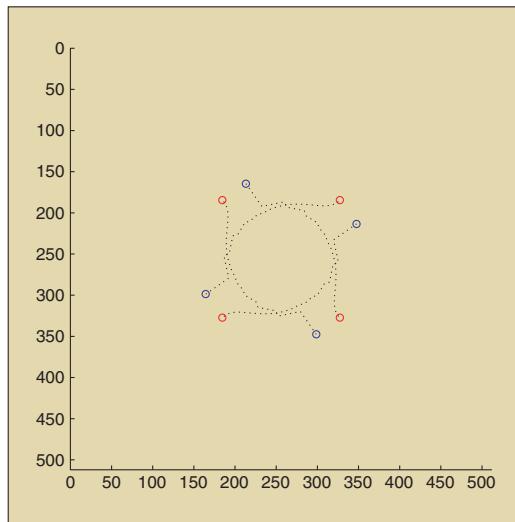


Figure 4. Image feature trajectories for a rotation of 160° about the optical axis using a switched control scheme (initial points position in blue and desired points position in red).

Other examples of temporal switching schemes can be found, such as for instance the one developed in [37] to ensure the visibility of the target observed.

Feature Trajectory Planning

It is also possible to treat the optimization problem offline, during a planning stage. In that case, several constraints can be simultaneously taken into account, such as obstacle avoidance [38], joint limit and occlusions avoidance, and ensuring the visibility of the target [5]. The feature trajectories $\mathbf{s}^*(t)$ that allow the camera to reach its desired pose while ensuring the constraints are satisfied are determined using path planning techniques, such as the well known potential field approach.

Coupling path planning with trajectory following also allows to improve significantly the robustness of the visual servo with respect to modeling errors. Indeed, modeling errors may have large effects when the error $\mathbf{s} - \mathbf{s}^*$ is large, but have few effects when $\mathbf{s} - \mathbf{s}^*$ is small. Once desired features trajectories $\mathbf{s}^*(t)$ such that $\mathbf{s}^*(0) = \mathbf{s}(0)$ have been designed during the planning stage, it is easy to adapt the control scheme to take into account the fact that \mathbf{s}^* is varying, and to make the error $\mathbf{s} - \mathbf{s}^*$ remain small. More precisely, we now have

$$\dot{\mathbf{e}} = \dot{\mathbf{s}} - \dot{\mathbf{s}}^* = \mathbf{L}_e \mathbf{v}_c - \dot{\mathbf{s}}^*,$$

from which we deduce, by selecting as usual $\dot{\mathbf{e}} = -\lambda \mathbf{e}$ as desired behavior

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}}_e^+ \mathbf{e} + \widehat{\mathbf{L}}_e^+ \dot{\mathbf{s}}^*.$$

The new second term of this control law anticipates the variation of \mathbf{s}^* , removing the tracking error it would produce. We will see in the next section that the form of the control law is similar when tracking a moving target is considered.

Target Tracking

We now consider the case of a moving target and a constant desired value \mathbf{s}^* for the features, the generalization to varying desired features $\mathbf{s}^*(t)$ being immediate. The time variation of the error is now given by

$$\dot{\mathbf{e}} = \dot{\mathbf{s}} = \mathbf{L}_e \mathbf{v}_c + \frac{\partial \mathbf{e}}{\partial t}, \quad (10)$$

where the term $\frac{\partial \mathbf{e}}{\partial t}$ expresses the time variation of \mathbf{e} due to the generally unknown target motion. If the control law is still designed to try to ensure an exponential decoupled decrease of \mathbf{e} (that is, once again $\dot{\mathbf{e}} = -\lambda \mathbf{e}$), we now obtain using (10):

$$\mathbf{v}_c = -\lambda \widehat{\mathbf{L}}_e^+ \mathbf{e} - \widehat{\mathbf{L}}_e^+ \frac{\partial \mathbf{e}}{\partial t}, \quad (11)$$

where $\frac{\partial \mathbf{e}}{\partial t}$ is an estimation or an approximation of $\frac{d\mathbf{e}}{dt}$. This term must be introduced in the control law to compensate for the target motion.

Closing the loop, that is injecting (11) in (10), we obtain

A simple switching controller can be designed using an IBVS and a PBVS controller.

$$\dot{\mathbf{e}} = -\lambda \mathbf{L}_e \widehat{\mathbf{L}}_e^+ \mathbf{e} - \mathbf{L}_e \widehat{\mathbf{L}}_e^+ \frac{\partial \widehat{\mathbf{e}}}{\partial t} + \frac{\partial \mathbf{e}}{\partial t}. \quad (12)$$

Even if $\mathbf{L}_e \widehat{\mathbf{L}}_e^+ > 0$, the error will converge to zero only if the estimation $\frac{\partial \mathbf{e}}{\partial t}$ is sufficiently accurate so that

$$\mathbf{L}_e \widehat{\mathbf{L}}_e^+ \frac{\partial \widehat{\mathbf{e}}}{\partial t} = \frac{\partial \mathbf{e}}{\partial t}.$$

Otherwise, tracking errors will be observed. Indeed, by just solving the scalar differential equation $\dot{e} = -\lambda e + b$, which is a simplification of (12), we obtain $e(t) = e(0) \exp(-\lambda t) + b/\lambda$, which converges towards b/λ . On one hand, setting a high gain λ will reduce the tracking error, but on the other hand, setting the gain too high can make the system unstable. It is thus necessary to make b as small as possible.

Of course, if the system is known to be such that $\frac{\partial \mathbf{e}}{\partial t} = 0$ (that is the camera observes a motionless object), no tracking error will appear with the most simple estimation given by $\frac{\partial \mathbf{e}}{\partial t} = 0$. Otherwise, a classical method in automatic control to cancel tracking errors consists in compensating the target motion through an integral term in the control law. In that case, we have

$$\frac{\partial \mathbf{e}}{\partial t} = \mu \sum_j \mathbf{e}(j),$$

where μ is the integral gain, which must be tuned. This scheme allows to cancel the tracking errors only if the target has a constant velocity. Other methods, based on feedforward control, estimate directly the term $\frac{\partial \mathbf{e}}{\partial t}$ through the image measurements and the camera velocity, when it is available. Indeed, from (10), we obtain

$$\frac{\partial \widehat{\mathbf{e}}}{\partial t} = \widehat{\mathbf{e}} - \widehat{\mathbf{L}}_e \widehat{\mathbf{v}}_c,$$

where $\widehat{\mathbf{e}}$ can, for instance, be obtained as $\widehat{\mathbf{e}}(t) = (\mathbf{e}(t) - \mathbf{e}(t - \Delta t))/\Delta t$, with Δt being the duration of the control loop. A Kalman filter [39] or more elaborate filtering methods [40] can then be used to improve the estimated values. If some knowledge about the target velocity or the target trajectory is available, it can of course be used to smooth or predict the motion [41]–[43]. For instance, in [44], the periodic motion of the heart and of the breath are compensated for in an application of visual servoing in medical robotics. Finally, other methods have been developed to remove as fast as possible the perturbations induced by the target motion [28], using for instance predictive controllers [45].

In the joint space, the system equations for both the eye-to-hand configuration and the eye-in-hand configuration have the same form.

Eye-in-Hand and Eye-to-Hand Systems Controlled in the Joint Space

In the previous sections, we considered the six components of the camera velocity as input of the robot controller. As soon as the robot is not able to realize this motion, for instance because it has less than six degrees of freedom, the control scheme must be expressed in the joint space. In this section, we describe how this can be done, and in the process develop a formulation for eye-to-hand systems.

In the joint space, the system equations for both the eye-to-hand configuration and the eye-in-hand configuration have the same form

$$\dot{\mathbf{s}} = \mathbf{J}_s \dot{\mathbf{q}} + \frac{\partial \mathbf{s}}{\partial t}. \quad (13)$$

Here, $\mathbf{J}_s \in \mathbb{R}^{k \times n}$ is the feature Jacobian matrix, which can be linked to the interaction matrix, and n is the number of robot joints.

For an eye-in-hand system [see Figure 5(b)], $(\partial \mathbf{s} / \partial t)$ is the time variation of \mathbf{s} due to a potential object motion, and \mathbf{J}_s is given by

$$\mathbf{J}_s = \mathbf{L}_s {}^c \mathbf{V}_N \mathbf{J}(\mathbf{q}), \quad (14)$$

where

- ${}^c \mathbf{V}_N$ is the spatial motion transform matrix (as defined in Part I of the tutorial) from the vision sensor frame to

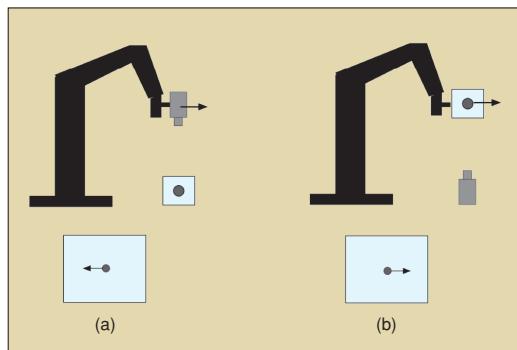


Figure 5. Top: (a) Eye-in-hand system. (b) Eye-to-hand system. Bottom: Opposite image motion produced by the same robot motion.

the end effector frame. It is usually a constant matrix (as soon as the vision sensor is rigidly attached to the end effector). Thanks to the robustness of closed loop control schemes, a coarse approximation of this transform matrix is sufficient in visual servoing. If needed, an accurate estimation is possible through classical hand-eye calibration methods [46].

- ◆ $\mathbf{J}(\mathbf{q})$ is the robot Jacobian expressed in the end effector frame.

For an eye-to-hand system [see Figure 5(b)], $(\partial \mathbf{s} / \partial t)$ is now the time variation of \mathbf{s} due to a potential vision sensor motion and \mathbf{J}_s can be expressed as

$$\dot{\mathbf{s}} = -\mathbf{L}_s {}^c \mathbf{V}_N^N \mathbf{J}(\mathbf{q}) \quad (15)$$

$$= -\mathbf{L}_s {}^c \mathbf{V}_0^0 \mathbf{J}(\mathbf{q}). \quad (16)$$

In (15), the classical robot Jacobian ${}^N \mathbf{J}(\mathbf{q})$ expressed in the end effector frame is used but the spatial motion transform matrix ${}^c \mathbf{V}_N$ from the vision sensor frame to the end effector frame changes all along the servo, and it has to be estimated at each iteration of the control scheme, usually using pose estimation methods.

In (16), the robot Jacobian ${}^0 \mathbf{J}(\mathbf{q})$ is expressed in the robot reference frame and the spatial motion transform matrix ${}^c \mathbf{x}_0$ from the vision sensor frame to that reference frame is constant as long as the camera does not move. In that case, which is convenient in practice, a coarse approximation of ${}^c \mathbf{V}_0$ is usually sufficient.

Once the modeling step is finished, it is quite easy to follow the procedure that has been used above to design a control scheme expressed in the joint space, and to determine sufficient condition to ensure the stability of the control scheme. We obtain, considering again $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$, and an exponential decoupled decrease of \mathbf{e}

$$\dot{\mathbf{q}} = -\lambda \widehat{\mathbf{J}_e} \mathbf{e} - \widehat{\mathbf{J}_e^+} \frac{\partial \widehat{\mathbf{e}}}{\partial t}. \quad (17)$$

If $k = n$, considering the Lyapunov function $\mathcal{L} = \frac{1}{2} \|\mathbf{e}(t)\|^2$, a sufficient condition to ensure the global asymptotic stability is given by

$$\widehat{\mathbf{J}_e} \widehat{\mathbf{J}_e^+} > 0. \quad (18)$$

If $k > n$, we obtain by following the developments of the stability analysis of basic IBVS (see Part I of the tutorial):

$$\widehat{\mathbf{J}_e^+} \widehat{\mathbf{J}_e} > 0 \quad (19)$$

to ensure the local asymptotic stability of the system. Note that the actual extrinsic camera parameters appears in \mathbf{J}_e , while the estimated ones are used in $\widehat{\mathbf{J}_e^+}$. It is thus possible to analyse the robustness of the control scheme with respect to the camera extrinsic parameters. It is also possible to estimate directly the numerical value of \mathbf{J}_e or $\widehat{\mathbf{J}_e^+}$ using, as described previously, the Broyden update rule.

Finally, to remove tracking errors, we have to ensure that

$$\mathbf{J}_e \widehat{\mathbf{J}}_e^+ \frac{\partial \mathbf{e}}{\partial t} = \frac{\partial \mathbf{e}}{\partial t},$$

Finally, let us note that, even if the robot has six degrees of freedom, it is generally not equivalent to first compute \mathbf{v}_c using (2) and then deduce $\dot{\mathbf{q}}$ using the robot inverse Jacobian, and to compute directly $\dot{\mathbf{q}}$ using (17). Indeed, it may occur that the robot Jacobian $\mathbf{J}(\mathbf{q})$ is singular while the feature Jacobian \mathbf{J}_s is not (that may occur when $k < n$). Furthermore, the properties of the pseudo-inverse ensure that using (2), $\|\mathbf{v}_c\|$ is minimal while using (17), $\|\dot{\mathbf{q}}\|$ is minimal. As soon as $\mathbf{J}_e^+ \neq \mathbf{J}^+(\mathbf{q})^N \mathbf{V}_c \mathbf{L}_e^+$, the control schemes will be different and will induce different robot trajectories. The choice of the state-space is thus important.

Conclusion

In this tutorial, we have only considered velocity controllers. It is convenient for most of classical robot arms. However, the dynamics of the robot must of course be taken into account for high speed tasks, or when we deal with mobile nonholonomic or underactuated robots. As for the sensor, we have only considered geometrical features coming from a classical perspective camera. Features related to the image motion or coming from other vision sensors (fish-eye camera, catadioptric camera, echographic probes, etc.) necessitate to revisit the modeling issues to select adequate visual features. Finally, fusing visual features with data coming from other sensors (force sensor, proximity sensors, etc.) at the level of the control scheme will allow to address new research topics. Numerous ambitious applications of visual servoing can also be considered, as well as for mobile robots in indoor or outdoor environments, for aerial, space, and submarine robots, and in medical robotics. The end of fruitful research in the field of visual servo is thus nowhere yet in sight.

Keywords

Visual servo, robot control.

References

- [1] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. New York: Springer-Verlag, 2003.
- [2] R. Basri, E. Rivlin, and I. Shimshoni, "Visual homing: Surfing on the epipoles," *Int. J. Comput. Vision*, vol. 33, pp. 117–137, Sept. 1999.
- [3] E. Malis, F. Chaumette, and S. Boudev, "2 1/2 D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement," *Int. J. Comput. Vision*, vol. 37, pp. 79–97, June 2000.
- [4] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*. Cambridge, MA: MIT Press, 1993.
- [5] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4 pp. 534–549, Aug. 2002.
- [6] I. Suh, "Visual servoing of robot manipulators by fuzzy membership function based neural networks," in *Visual Servoing*, K. Hashimoto, Ed. (World Scientific Series in Robotics and Automated Systems). Singapore: World Scientific Press, 1993, vol. 7, pp. 285–315.
- [7] G. Wells, C. Venaille, and C. Torras, "Vision-based robot positioning using neural networks," *Image Vision Comput.*, vol. 14, pp. 75–732, Dec. 1996.
- [8] J.-T. Lapresté, F. Jurie, and F. Chaumette, "An efficient method to compute the inverse jacobian matrix in visual servoing," in *Proc. IEEE Int. Conf. Robotics Automation*, New Orleans, LA, Apr. 2004, pp. 727–732.
- [9] K. Hosada and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, München, Germany, Sept. 1994, pp. 186–193.
- [10] M. Jägersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Proc. IEEE Int. Conf. Robotics and Automation*, Albuquerque, NM, Apr. 1997, pp. 2874–2880.
- [11] J. Piepmeyer, G.M. Murray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *IEEE Trans. Robot. Automat.*, vol. 20, no. 1, pp. 143–147, Feb. 2004.
- [12] K. Deguchi, "Direct interpretation of dynamic images and camera motion for visual servoing without image feature correspondence," *J. Robot. Mechatron.*, vol. 9, no. 2, pp. 104–110, 1997.
- [13] E. Malis, F. Chaumette, and S. Boudev, "2-1/2D visual servoing," *IEEE Trans. Robot. Automat.*, vol. 15, no. 2, pp. 238–250, Apr. 1999.
- [14] E. Malis and F. Chaumette, "Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods," *IEEE Trans. Robot. Automat.*, vol. 18, no. 2, pp. 176–186, Apr. 2002.
- [15] J. Chen, D. Dawson, W. Dixon, and A. Behal, "Adaptive homography-based visual servo tracking for fixed and camera-in-hand configurations," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 9, pp. 814–825, Sept. 2005.
- [16] G. Morel, T. Leibezeit, J. Szewczyk, S. Boudev, and J. Pot, "Explicit incorporation of 2D constraints in vision-based control of robot manipulators," in *Proc. Int. Symp. Experimental Robotics*, 2000, vol. 250 LNCIS Series, pp. 99–108.
- [17] F. Chaumette and E. Malis, "2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings," in *Proc. IEEE Int. Conf. Robotics Automation*, San Francisco, CA, Apr. 2000, pp. 630–635.
- [18] E. Cervera, A.D. Pobil, F. Berry, and P. Martinet, "Improving image-based visual servoing with three-dimensional features," *Int. J. Robot. Res.*, vol. 22, pp. 821–840, Oct. 2004.
- [19] F. Schramm, G. Morel, A. Micaelli, and A. Lottin, "Extended-2D visual servoing," in *Proc. IEEE Int. Conf. Robotics Automation*, New Orleans, LA, Apr. 2004, pp. 267–273.
- [20] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. Robotics Automat.*, vol. 17, no. 4, pp. 507–515, Aug. 2001.
- [21] M. Iwatsuki and N. Okiyama, "A new formulation of visual servoing based on cylindrical coordinate system," *IEEE Trans. Robot. Automat.*, vol. 21, no. 2, pp. 266–273, Apr. 2005.
- [22] F. Chaumette, P. Rives, and B. Espiau, "Classification and realization of the different vision-based tasks," in *Visual Servoing*, K. Hashimoto, Ed. (Robotics and Automated Systems). Singapore: World Scientific, 1993, vol. 7, pp. 199–228.
- [23] A. Castano and S. Hutchinson, "Visual compliance: task directed visual servo control," *IEEE Trans. Robot. Automat.*, vol. 10, no. 3, pp. 334–342, June 1994.
- [24] G. Hager, "A modular system for robust positioning using feedback from stereo vision," *IEEE Trans. Robot. Automat.*, vol. 13, no. 4, pp. 582–595, Aug. 1997.
- [25] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robot. Automat.*, vol. 8, no. 3, pp. 313–326, June 1992.

Numerous ambitious applications of visual servoing can be considered.

- [26] F. Chaumette, "Image moments: A general and useful set of features for visual servoing," *IEEE Trans. Robot. Automat.*, vol. 20, no. 4, pp. 713–723, Aug. 2004.
- [27] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1116–1127, Dec. 2005.
- [28] N. Papanikopoulos, P. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control," *IEEE Trans. Robot. Automat.*, vol. 9, no. 1, pp. 14–35, Feb. 1993.
- [29] K. Hashimoto and H. Kimura, "LQ optimal and nonlinear approaches to visual servoing," in *Visual Servoing*, K. Hashimoto, Ed. (Robotics and Automated Systems). Singapore: World Scientific, 1993, vol. 7, pp. 165–198.
- [30] B. Nelson and P. Khosla, "Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance," *Int. J. Robot. Res.*, vol. 14, pp. 225–269, June 1995.
- [31] B. Nelson and P. Khosla, "Force and vision resolvability for assimilating disparate sensory feedback," *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, pp. 714–731, Oct. 1996.
- [32] R. Sharma and S. Hutchinson, "Motion perceptibility and its application to active vision-based servo control," *IEEE Trans. Robot. Automat.*, vol. 13, no. 4, pp. 607–617, Aug. 1997.
- [33] E. Marchand, F. Chaumette, and A. Rizzo, "Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, Osaka, Japan, Nov. 1996, pp. 1083–1090.
- [34] E. Marchand and G. Hager, "Dynamic sensor planning in visual servoing," in *Proc. IEEE Int. Conf. Robotics Automation*, Leuven, Belgium, May 1998, pp. 1988–1993.
- [35] N. Cowan, J. Weingarten, and D. Koditschek, "Visual servoing via navigation functions," *IEEE Trans. Robot. Automat.*, vol. 18, no. 4, pp. 521–533, Aug. 2002.
- [36] N. Gans and S. Hutchinson, "An asymptotically stable switched system visual controller for eye in hand robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, Las Vegas, NV, Oct. 2003, pp. 735–742.
- [37] G. Chesi, K. Hashimoto, D. Prattichizio, and A. Vicino, "Keeping features in the field of view in eye-in-hand visual servoing: A switching approach," *IEEE Trans. Robot. Automat.*, vol. 20, no. 5, pp. 908–913, Oct. 2004.
- [38] K. Hosoda, K. Sakamoto, and M. Asada, "Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3D reconstruction," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, vol. 3, Pittsburgh, PA, Aug. 1995, pp. 29–34.
- [39] P. Corke and M. Goods, "Controller design for high performance visual servoing," in *Proc. 12th World Congr. IFAC'93*, Sydney, Australia, July 1993, pp. 395–398.
- [40] F. Bensalah and F. Chaumette, "Compensation of abrupt motion changes in target tracking by visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, Pittsburgh, PA, Aug. 1995, pp. 181–187.
- [41] P. Allen, B. Yoshimi, A. Timcenko, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *IEEE Trans. Robot. Automat.*, vol. 9, no. 2, pp. 152–165, Apr. 1993.
- [42] K. Hashimoto and H. Kimura, "Visual servoing with non linear observer," in *Proc. IEEE Int. Conf. Robotics Automation*, Nagoya, Japan, Apr. 1995, pp. 484–489.
- [43] A. Rizzi and D. Koditschek, "An active visual estimator for dexterous manipulation," *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, pp. 697–713, Oct. 1996.
- [44] R. Ginhoux, J. Gangloff, M. de Mathelin, L. Soler, M.A. Sanchez, and J. Marescaux, "Active filtering of physiological motion in robotized surgery using predictive control," *IEEE Trans. Robot.*, vol. 21, no. 1, pp. 67–79, Feb. 2005.
- [45] J. Gangloff and M. de Mathelin, "Visual servoing of a 6 dof manipulator for unknown 3-D profile following," *IEEE Trans. Robot. Automat.*, vol. 18, no. 4, pp. 511–520, Aug. 2002.
- [46] R. Tsai and R. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand-eye calibration," *IEEE Trans. Robot. Automat.*, vol. 5, no. 3, pp. 345–358, June 1989.

François Chaumette graduated from École Nationale Supérieure de Mécanique, Nantes, France, in 1987. He received the Ph.D. degree in computer science from the University of Rennes, Rennes, France, in 1990. Since 1990, he has been with IRISA in Rennes where he is now "Directeur de Recherches" INRIA and head of the Lagadic group. His research interests include robotics and computer vision, especially visual servoing and active perception. He has coauthored more than 150 papers on the topics of robotics and computer vision, and has served in the last five years on the program committees for the main conferences related to robotics and computer vision. Dr. Cahumette received the AFCET/CNRS Prize for the best French thesis in automatic control in 1991. He also received with Ezio Malis the 2002 King-Sun Fu Memorial Best IEEE Transactions on Robotics and Automation Paper Award. He was Associate Editor of *IEEE Transactions on Robotics* from 2001 to 2005.

Seth Hutchinson received the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988. In 1990, he joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for Advanced Science and Technology. Dr. Hutchinson serves as the Editor-in-Chief for the RAS Conference Editorial Board, and on the editorial boards of the *International Journal of Robotics Research* and the *Journal of Intelligent Service Robotics*. He served as Associate and then Senior Editor for the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, now the IEEE TRANSACTIONS ON ROBOTICS, from 1997 to 2005. In 1996, he was a Guest Editor for a special section of the TRANSACTIONS devoted to the topic of visual servo control, and in 1994 he was Co-Chair of an IEEE Workshop on Visual Servoing. In 1996 and 1998, he coauthored papers that were finalists for the King-Sun Fu Memorial Best Transactions Paper Award. He was Co-Chair of IEEE Robotics and Automation Society Technical Committee on Computer and Robot Vision from 1992 to 1996, and has served on the program committees for more than fifty conferences related to robotics and computer vision. He has published more than 100 papers on the topics of robotics and computer vision, and is coauthor of the books *Principles of Robot Motion: Theory, Algorithms, and Implementations* (Cambridge, MA: MIT Press) and *Robot Modeling and Control* (New York: Wiley).

Address for Correspondence: François Chaumette, IRISA/INRIA Rennes, Campus de Beaulieu, 35 042 Rennes cedex, France. E-mail: Francois.Chaumette@irisa.fr.

9.7 Autonomous Landing of a VTOL UAV on a Moving Platform Using Image-based Visual Servoing, ICRA, 2012

2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Autonomous Landing of a VTOL UAV on a Moving Platform Using Image-based Visual Servoing

Daewon Lee¹, Tyler Ryan² and H. Jin. Kim³

Abstract—In this paper we describe a vision-based algorithm to control a vertical-takeoff-and-landing unmanned aerial vehicle while tracking and landing on a moving platform. Specifically, we use image-based visual servoing (IBVS) to track the platform in two-dimensional image space and generate a velocity reference command used as the input to an adaptive sliding mode controller. Compared with other vision-based control algorithms that reconstruct a full three-dimensional representation of the target, which requires precise depth estimation, IBVS is computationally cheaper since it is less sensitive to the depth estimation allowing for a faster method to obtain this estimate. To enhance velocity tracking of the sliding mode controller, an adaptive rule is described to account for the ground effect experienced during the maneuver. Finally, the IBVS algorithm integrated with the adaptive sliding mode controller for tracking and landing is validated in an experimental setup using a quadrotor.

I. INTRODUCTION

Vertical take-off and landing (VTOL) unmanned aerial vehicles (UAVs) have been used in many applications such as search and rescue, reconnaissance, surveillance and exploration. Generally, these aircraft have highly coupled nonlinear dynamics and fly in unsteady conditions (e.g. due to turbulence or unmodelled dynamics), which has led to the development of various robust and/or nonlinear control techniques. For example, [1] presents an internal-model based error-feedback dynamic regulator that is robust to parametric uncertainty. Sliding mode disturbance observers ([2]) and adaptive-fuzzy control techniques ([3]) have also been used to make the system more robust to external disturbances. A combined controller and observer using neural networks is used in [4] to track reference trajectories.

One feature becoming more common on miniature VTOL UAVs is an onboard camera, which has resulted in a quickly growing library of research making use of this camera. Common isolated vision tasks include target tracking ([5] and [6]) and simultaneous localization and mapping (SLAM, [7]). Additionally, the onboard camera has been used within the control loop such as [8] and [9] which use the camera for position control.

For this paper, we use the adaptive sliding mode controller described in [10] for precision control of a quadrotor while

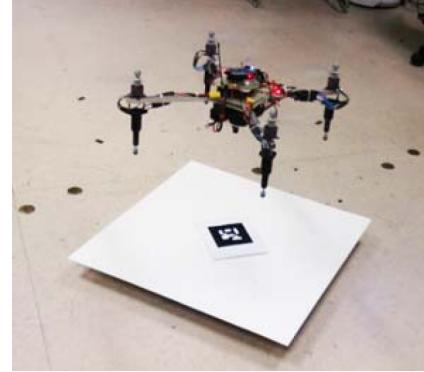


Fig. 1. Quadrotor helicopter and moving ground target under consideration

landing on a moving target (fig. 1). This situation would occur, for example, when landing on a moving ship or truck. While GPS-based control is sufficient for general positioning (assuming the signal is not blocked), the accuracy of GPS receivers fit for use on miniature UAVs is measured in meters, making them unsuitable for precision tasks such as landing. Instead, we demonstrate that image-based visual servoing (IBVS) is capable of providing the necessary precision. IVBS is a technique that performs the majority of the control calculations in 2D image space, rather than the typical 3D body or inertial reference frames. This eliminates the need for expensive 3D position reconstruction calculations allowing for real-time control. Finally, given the need of flying inside ground effect during the landing maneuver, the controller uses a model adaptation technique to dynamically compensate for this system disturbance.

Most similar to our work is [11] which also calculates control in the image frame thereby avoiding full three dimensional reconstruction. The biggest difference between their work and ours is we adaptively compensate for the ground effect which is quite significant during a landing maneuver. Additionally they use a different image-based method for their formulation.

II. SYSTEM

This section describes the dynamics of the quadrotor and the camera systems.

A. Quadrotor

Define the UAV state vector as $\mathbf{x} = [x, y, z, \phi, \theta, \psi]^T$,

¹ Department of Mechanical and Aerospace Engineering, Ph.D Candidate, Seoul National University, Seoul, Republic of Korea lee.daewon@gmail.com

² Department of Mechanical and Aerospace Engineering, Ph.D Student, Seoul National University, Seoul, Republic of Korea ryantr@gmail.com

³ Department of Mechanical and Aerospace Engineering, Associate Professor Seoul National University, Seoul, Republic of Korea hjinkim@snu.ac.kr

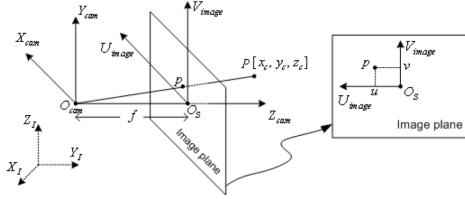


Fig. 2. Geometry and coordinate frames for a pinhole camera model

where x , y and z denote translational position variables and ϕ , θ and ψ denote roll, pitch and yaw, respectively. Define the control input vector as $\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ (inputs for thrust, roll, pitch and yaw respectively). Then system dynamics, including influence of external forces $f_{ex}(\mathbf{x})$, can be represented as eq. (1). (for derivation details, see [10])

$$\ddot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} + f_{ex}(\mathbf{x}) \quad (1)$$

where

$$f(\mathbf{x}) = \begin{bmatrix} 0 & 0 & -F_g & 0 & 0 & 0 \end{bmatrix}^T$$

$$g(\mathbf{x}) = \begin{bmatrix} \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi & 0 & 0 & 0 \\ \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi & 0 & 0 & 0 \\ \cos \phi \cos \theta & 0 & 0 & 0 \\ 0 & l & 0 & 0 \\ 0 & 0 & l & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$f_{ex}(\mathbf{x}) = [0 \ 0 \ F_{gr}(z) \ 0 \ 0 \ 0]^T,$$

l denotes the distance from the motor rotational axis to the system center of gravity (assumed equal for all motors), F_g is gravity force and $F_{gr}(z)$ denotes the ground effect force.

B. Vision System

The camera model we consider in this paper is a pinhole camera model as shown in fig. 2 where $[X_I, Y_I, Z_I]^T$ are the axes of the three-dimensional inertial coordinate frame \mathbb{I} , $[X_{cam}, Y_{cam}, Z_{cam}]^T$ are the axes of the camera frame \mathbb{C} , i.e., the coordinate frame attached to the camera center O_{cam} , $[U_{image}, V_{image}]^T$ are the axes of the image frame \mathbb{S} , O_S denotes the point where the z axis of the camera coordinate frame intersects the image plane, and f is the focal length of the camera. The mapping from a point $P = [x_c, y_c, z_c]^T \in \mathbb{C}$ to a point $p = [u, v]^T \in \mathbb{S}$ can be written as eq. (2).

$$p = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix} \quad (2)$$

Suppose that the camera is moving such that P moves with translational velocity $T = [T_x, T_y, T_z]^T$ and rotational velocity $\Omega = [\omega_x, \omega_y, \omega_z]^T$ both with respect to the camera frame \mathbb{C} . The spatial velocity, $\dot{r} \in \mathbb{R}^6$ can then be defined as (3)

$$\dot{r} = \begin{bmatrix} T \\ \Omega \end{bmatrix} = [T_x, T_y, T_z, \omega_x, \omega_y, \omega_z]^T. \quad (3)$$

Finally, the dynamics of the image space point p can be expressed as eq. (4) [12].

$$\dot{p} = J_p \dot{r}. \quad (4)$$

where

$$J_p = \begin{bmatrix} -\frac{f}{z_c} & 0 & \frac{u}{z_c} & \frac{uv}{f} & -\frac{f^2+u^2}{f} & v \\ 0 & -\frac{f}{z_c} & \frac{v}{z_c} & \frac{f^2+v^2}{f} & -\frac{uv}{f} & -u \end{bmatrix} \quad (5)$$

is called the image Jacobian matrix associated with the image feature p .

III. CONTROL

This section describes image-based visual servoing, including image roll and pitch compensation, and the adaptive sliding mode velocity controller.

A. Image-based Visual Servoing

In image-based visual servoing, the task space is defined in two-dimensional image space and the control law is designed as a function of the image error, e , defined as

$$e = p - p_d \quad (6)$$

where p_d is the desired location of the image feature. For the landing maneuver considered in this paper, the desired image feature is static, i.e. $\dot{p}_d = 0$. Using this and eq. (4) we calculate the error rate as,

$$\dot{e} = J_p \dot{r} \quad (7)$$

Define a Lyapunov candidate function as

$$\mathcal{L} = \frac{1}{2} e^T e \quad (8)$$

$$\dot{\mathcal{L}} = e^T J_p \dot{r} \quad (9)$$

Define the desired camera velocity as

$$\dot{r}_d = -W J_p^T e \quad (10)$$

where W is a positive definite weighting matrix. As $\dot{r} \rightarrow \dot{r}_d$ (to be achieved by the sliding mode controller described in the next section) then

$$\dot{r} \rightarrow -W J_p^T e \quad (11)$$

and eq. (9) becomes

$$\dot{\mathcal{L}} = -e^T (J_p) W (J_p)^T e \leq 0, \quad (12)$$

where $J_p W (J_p)^T$ is a full-rank matrix when the image features are not colinear. This proves the asymptotic convergence of e to zero assuming \dot{r}_d is “well” tracked.

Eq. (10) generates six references (for x , y , z , ϕ , θ , ψ). However, the quadrotor cannot track all the references simultaneously because of its under-actuated property (four motor inputs versus six states). Since the x , y , ϕ and θ channels are highly coupled, we decouple the x and y channels from the ϕ and θ channels.

To do this, we define a virtual coordinate frame \mathbb{V} with the position and yaw angles aligned to match the camera coordinate frame \mathbb{C} , and the z axis is aligned with the inertial

frame z axis, \mathbb{I} . If references are generated from an image in \mathbb{V} , the desired decoupling is achieved.

To map the captured image into \mathbb{V} , first consider an imaginary camera frame which is the same as the true camera frame but rotated to zero roll and pitch. The image features in this imaginary frame become

$$P_i^r = R(1, \phi)R(2, \theta)P_i \quad (13)$$

$$\begin{aligned} p_i^r &= \frac{f}{z_{ci}^r} \begin{bmatrix} x_{ci}^r \\ y_{ci}^r \end{bmatrix} \\ &= f \begin{bmatrix} \frac{u_i \cos \theta + f \sin \theta}{-u_i \cos \phi \sin \theta + v_i \sin \phi + f \cos \phi \cos \theta} \\ \frac{u_i \sin \phi \sin \theta + v_i \cos \phi - f \sin \phi \cos \theta}{-u_i \cos \phi \sin \theta + v_i \sin \phi + f \cos \phi \cos \theta} \end{bmatrix} \quad (14) \\ &= \begin{bmatrix} u_i^r \\ v_i^r \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned} R(1, \phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \\ R(2, \theta) &= \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}. \end{aligned}$$

Here P_i^r is the i -th feature coordinate defined in the roll and pitch compensated camera frame, and p_i^r is the corresponding image feature coordinate of P_i^r . For most normal flight regimes, roll and pitch angles don't approach 90 deg so the denominators in eq. (14) will be non-zero. Interestingly, so far no depth information is needed to perform the roll and pitch compensation.

Assuming that we know the nominal geometry of the target in the camera frame, we can now estimate depth. First, find the position of another feature in image space, p_j^r . Combined with the already measured point p_i^r we can calculate the distance between these points in the image frame, which we call L_s . Let us call L_c the, already known, corresponding distance in the camera frame. Using these, the depth from the camera to the image features can be simply calculated as

$$z_c = \frac{L_c \cdot f}{L_s} \quad (15)$$

At this point we could, in theory, calculate a full three-dimensional reconstruction of the target and perform control without IBVS. However, this simple depth calculation is only a rough estimate, with a focus on computational efficiency over accuracy, and would not produce an accurate enough reconstruction for standard control algorithms. IBVS, however, is less sensitive to this measurement so having only this rough estimate is sufficient.

The last thing we need to address is distortion due to an offset from the origin of the quadrotor body frame to the origin of the camera frame, which is denoted as $[\delta_x, \delta_y, \delta_z]^T$. For this consider another imaginary camera frame which is translated to compensate for the distance caused by rotation of the offset. Define the i -th feature location in this frame

P_i^s , for which following relationship holds:

$$P_i^s = P_i^r + R(1, \phi)R(2, \theta)[\delta_x, \delta_y, \delta_z]^T \quad (16)$$

Now, the only difference between the virtual coordinate frame and the imaginary camera frame is the offset position of the camera. Therefore, the i -th feature coordinate defined in the virtual frame, P_i^v , is

$$P_i^v = P_i^s - [\delta_x, \delta_y, \delta_z]^T \quad (17)$$

$$p_i^v = \frac{f}{Z_{ci}^v} \begin{bmatrix} x_{ci}^v \\ y_{ci}^v \end{bmatrix} \quad (18)$$

where p_i^v is the corresponding image feature coordinate of P_i^v .

B. Adaptive Sliding Mode Controller

To track the translational and rotational velocity references of the UAV given in eq. (10), an adaptive sliding mode controller is adopted. Since $g(\mathbf{x})$ in eq. (1) is a 6-by-4 matrix we need to augment it with a 6-by-2 matrix, \mathbf{g}_s , to make it invertible. This also requires, then, that the input vector \mathbf{u} be augmented with a slack variable vector, \mathbf{u}_s . Eq. (1) can then be rewritten as

$$\ddot{\mathbf{x}} = f(\mathbf{x}) + G(\mathbf{x})U - \nu + f_{ex}(\mathbf{x}), \quad (19)$$

where $G = [g(\mathbf{x}), \mathbf{g}_s]$, $U = [\mathbf{u}^T, \mathbf{u}_s^T]^T$, and $\nu = \mathbf{g}_s \mathbf{u}_s$. If we set

$$\mathbf{g}_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \quad (20)$$

then $\nu = \mathbf{g}_s \mathbf{u}_s = [u_5 \ u_6 \ 0 \ 0 \ 0 \ 0]^T$, so we need to estimate the slack variables u_5 and u_6 .

Define the sliding surface as

$$S = [s_1, s_2, s_3, s_4, s_5, s_6]^T = \dot{\mathbf{x}} - \dot{\mathbf{x}}_r \quad (21)$$

where $\dot{\mathbf{x}}_r$ is a velocity reference vector, $[\dot{x}_r, \dot{y}_r, \dot{z}_r, \dot{\phi}_r, \dot{\theta}_r, \dot{\psi}_r]^T$. As previously mentioned, the quadrotor system is under-actuated such that the x and y state variables cannot be controlled directly from the inputs. Therefore, let us define the velocity references for $\dot{\phi}$ and $\dot{\theta}$ as eqs. (22) and (23) to control the x and y positions.

$$\dot{\phi}_r = (\ddot{y} - \dot{y}_r) + k_\phi (\dot{y} - \dot{y}_r) \quad (22)$$

$$\dot{\theta}_r = (\ddot{x} - \dot{x}_r) + k_\theta (\dot{x} - \dot{x}_r) \quad (23)$$

where k_ϕ and k_θ are proportional gains.

Using the IVBS output from eq. (10) the reference velocity state variable, $\dot{\mathbf{x}}_r$, is given by

$$\begin{aligned} \dot{\mathbf{x}}_r &= E_r \dot{r}_d + \dot{\phi}_r E_4 + \dot{\theta}_r E_5 \\ &= -E_r W J_{p^v}^T e + \dot{\phi}_r E_4 + \dot{\theta}_r E_5 \end{aligned} \quad (24)$$

where $E_r = \text{diag}[1, 1, 1, 0, 0, 1]$, $E_4 = [0, 0, 0, 1, 0, 0]^T$ and $E_5 = [0, 0, 0, 0, 1, 0]^T$.

In order to cancel nonlinear terms in eq. (19), we need to define the estimated values of ν and $f_{ex}(\mathbf{x})$, which we denote as $\hat{\nu}$ and $\hat{f}_{ex}(\mathbf{x})$, respectively. The uncertainty term is defined as

$$\Delta = -\nu + f_{ex} = [u_5, u_6, F_{gr}(z), 0, 0, 0]^T \quad (25)$$

Then equation (19) becomes,

$$\ddot{\mathbf{x}} = f(\mathbf{x}) + G(\mathbf{x})U + \Delta. \quad (26)$$

We set the Lyapunov candidate function as

$$\mathcal{L} = \frac{1}{2}S^T S + \frac{1}{2\Gamma}\tilde{\Delta}^T \tilde{\Delta} \quad (27)$$

and define the control input as

$$U = G^{-1}(\mathbf{x})[-f(\mathbf{x}) - \tilde{\Delta} + \ddot{\mathbf{x}}_r - C_1 S - C_2 \text{sign}(S)] \quad (28)$$

where C_1 is a diagonal weighting matrix and C_2 is a positive constant gain. By setting the uncertainty estimate update law as

$$\dot{\tilde{\Delta}} = \Gamma S \quad (29)$$

where Γ is positive semi-definite weighting matrix, the time derivative of the Lyapunov function becomes negative semi-definite.

$$\dot{\mathcal{L}} = -S^T(C_1 S + C_2 \text{sign}(S)) \leq 0, \quad (30)$$

Therefore the entire system converges to its desired state.

IV. EXPERIMENTAL RESULTS

In this section, the experimental setup is described and the proposed algorithm for autonomous landing on a moving target is validated.

A. Hardware Description

The quadrotor and landing target are shown in fig. 1. The length from each motor rotational axis to the center of the quadrotor is 30 cm, the weight of the entire UAV system is 1700 grams including two batteries and the height is 19 cm. The onboard camera has a 50 degree of field of view (FOV) and captures $320\text{px} \times 240\text{px}$ images at 30fps which are sent to the PC by a 1.2GHz onboard video transmitter. After control calculation, reference commands are sent to the quadrotor by a 2.4GHz radio control (RC) transmitter which is connected to the PC via an Endurance R/C PCTx USB dongle.

The quadrotor carries an inertial measurement unit (IMU) to measure angular rates which are used to estimate the Euler angles. If the target is not detected, a Vicon [13] camera system measures the position from which the translational velocity is calculated. When the target is detected, the Vicon data is used only for translational velocity. The entire control system is depicted in fig 3.

The landing pad is 90 cm long on each side and a fiducial marker is positioned approximately at the center. The four outside corners of the fiducial marker are used as the image features to compute J_p .

B. Mission Scenario

The entire flight sequence is depicted in fig. 4. There are three modes for a mission:

- 1) Patrol Mode: After taking off, the quadrotor patrols over a certain area until the target is detected continuously for a pre-determined time duration (0.2 seconds). In this mode, the position data is obtained from Vicon.

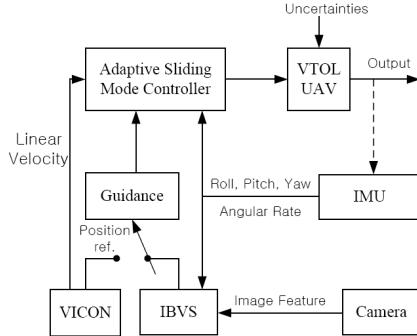


Fig. 3. Structure of the control system for the quadrotor helicopter

- 2) IBVS-Guided Tracking Mode: After the target is detected for 0.2 seconds, the control logic is switched to tracking mode. The purpose of this mode is to stabilize the quadrotor 0.4m over the target before attempting the landing maneuver (since a one-dimensional vertical maneuver is easier than attempting to land immediately via a three-dimensional maneuver). To do this the desired image feature locations are set to correspond to how they would look if the quadrotor were positioned at the desired location. Since the target usually first appears at the edge of the FOV, this maneuver also moves the target to the center where there is less risk of loosing the target while landing.
- 3) IBVS-Guided Landing Mode: After the landing mode is turned on, the desired image features are set to correspond to the position 0.20m directly over the target. If the error norm is below a defined threshold (implying that the quadrotor is at the target height and position) the thrust is turned off so the quadrotor can finish the landing maneuver. This small open-loop drop is necessary since the target is now so close to the camera that even small state errors can cause the camera to lose tracking for a number of reasons: the velocity of the features in image space becomes much higher, nonlinear distortions outside the camera calibration range become much more significant, and the FOV is now very small.

The ground target moves from its initial position (x , y and z) $[0.1 \text{ m}, 1 \text{ m}, 0.07 \text{ m}]$ with a velocity of approximately $[0 \text{ m/s}, -0.07 \text{ m/s}, 0 \text{ m/s}]$. During the tracking and landing modes, if the target is not detected for 2 seconds continuously, then the control logic is reverted back to patrol mode. The entire sequence from take-off to landing is operated automatically.

C. Experimental Results

The proposed landing algorithm is validated by experiment. The vertical dash-dot line (14.2 second) in each of fig. 5 ~ 11 shows the moment when the control logic switches to IBVS-based tracking mode, the vertical dotted line (18.2 second) shows the moment when logic switches to IBVS-based landing mode, and the vertical dashed line

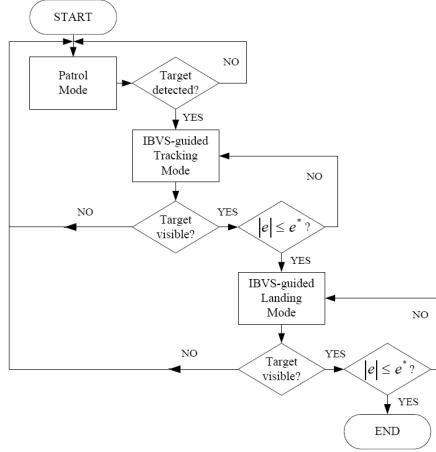


Fig. 4. Autonomous flight sequence

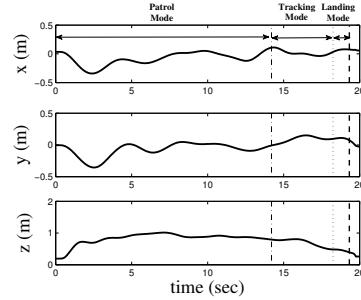
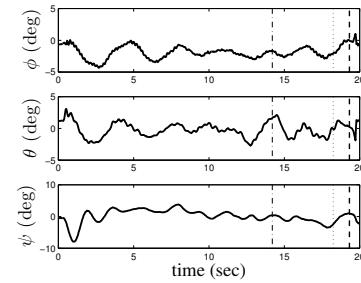
(19.3 second) represents the moment when the mission is completed (i.e. when the thrust is turned off).

Figs. 5 and 6 show position and attitude variables for the quadrotor during the mission. Note that the height of the quadrotor starts from 0.2 meter, which is the height of the markers used by the Vicon system to measure position. Also, the height of the quadrotor at the end of the mission is 0.27 meter, because the height of the target is 0.07 meter.

Fig. 7 shows the four control input signals for thrust, roll, pitch and yaw and fig. 8 shows the reference velocity (eq. (10)) tracking along the x , y and z axes. These reference velocities are generated only once the IBVS-guided control mode is turned on (dash-dot line), so there are no reference velocities (which are recorded as zero) and the controller tracks a guidance command calculated from Vicon position data before this. During the IBVS-guided tracking mode (from dash-dot to dotted line) and landing mode (from dotted to dashed line), the proposed controller shows satisfactory results for tracking performances.

Fig. ?? shows the depth estimation value derived from eq. (15) and the depth value (considered to be the true depth) measured by the Vicon (which is the same as the z position data in fig. 5). Note that eq. (15) calculates the depth between the camera and the target (0.05 m above the camera at rest) while the Vicon measurement is actually the height of the quadrotor markers which are 0.2 meter above the ground at rest. Therefore there is an expected 0.25 m offset between measurements.

Fig. 10 shows the error distance of the four image features in the image frame. Before the dash-dot line the image is not visible (i.e. Vicon control) so the error is set to zero. Between the dash-dot line and the dotted line the control logic is in tracking mode and the IBVS algorithm successfully tracks the image until it achieves the pre-landing setpoint of 0.4m above the target. The sudden change in tracking error at the dotted line is due to the image feature desired location changing once the control logic enters the landing mode,

Fig. 5. Time history of the position of the UAV (x , y and z)Fig. 6. Time history of the attitude of the UAV (ϕ , θ and ψ)

which has a larger desired feature size.

Fig. 11 represents the estimated ground effect as shown in eq. (29). As expected the estimated effect increases rapidly as the quadrotor approaches the ground.

V. CONCLUSION

In this paper, an autonomous landing control algorithm using image-based visual servoing (IBVS) for vertical takeoff and landing (VTOL) unmanned aerial vehicles (UAVs) is described. The ground effect experienced during this maneuver is also compensated for by an adaptive control law. The algorithm is also validated by experiment where three control modes are defined: a patrol mode, a tracking mode and a landing mode. During the patrol mode, since the target is not detected yet, an external position sensor (i.e. Vicon) provides position guidance to the UAV. After

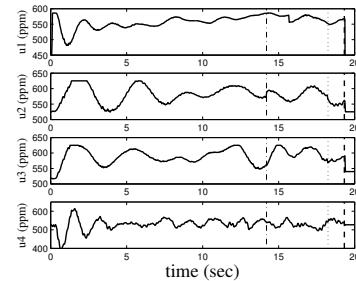


Fig. 7. Time history of the control inputs

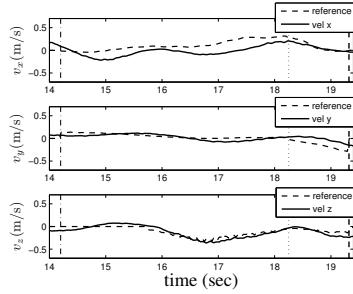


Fig. 8. Velocities of the quadrotor helicopter (solid line) and its references (dashed line) during the IBVS-guided control

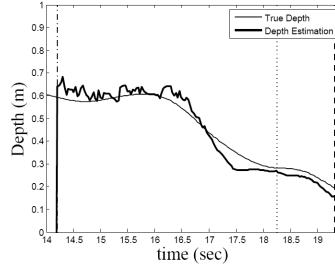


Fig. 9. Depth estimation and its true values

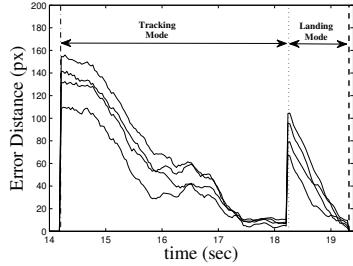


Fig. 10. Error distance of the four corners in the image frame. The sudden change at 18.2 seconds is due to the difference in desired feature locations between tracking mode and landing mode.

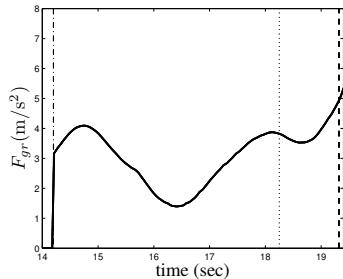


Fig. 11. Estimated ground effect value with the adaptive sliding mode controller

the target is detected, control logic switches to the tracking mode where position control is accomplished by tracking image features in the image frame. This mode positions the UAV above the target where the UAV is now ready to perform the landing maneuver. Now in landing mode, another desired image feature is defined to bring the UAV to the landing target. Once the image feature has converged to its desired position, the throttle is turned off to allow the UAV to land on the target. If the target leaves the FOV for longer than a couple seconds during either tracking or landing modes, the control logic is switched back to the patrol mode where the sequence can be started again once if the target can be found. The experimental results show satisfactory performance with the proposed IBVS in the nonlinear control loop. A video of this can be seen at http://www.youtube.com/watch?v=qrbVVJJzy_Q.

VI. ACKNOWLEDGMENTS

This work was supported by KARI-University Partnership Program (grant No. 2009-09-sunggwa-7) from the Korea Aerospace Research Institute (KARI) and Basic Science Research Program Through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2011-0003656).

REFERENCES

- [1] L. Marconi, A. Isidori, and A. Serrani, Autonomous vertical landing on an oscillating platform: An internal-model based approach, *Automatica*, vol. 38, pp. 2123, 2002.
- [2] L. Besnard, Y. Shitessel and B. Landrum, "Control of a Quadrotor Vehicle Using Sliding Mode Disturbance Observer," Proceedings of the 2007 American Control Conference, pp. 5230-5235, 2007.
- [3] C. Coza and C.J.B. Macnab, "A New Robust Adaptive-Fuzzy Control Method Applied to Quadrotor Helicopter Stabilization," NAFIPS 2006 Annual meeting of the North American Fuzzy Information Society, pp. 454-458, 2006
- [4] T. Dierks and S. Jagannathan, "Output Feedback Control of a Quadrotor UAV Using Neural Networks", *IEEE Transactions on Neural Networks*, VOL. 21, NO. 1, 2010, pp. 50-66
- [5] S. Saripalli, J. Montgomery and G. Sukhatme, Visually-Guided Landing of an Unmanned Aerial Vehicle. *IEEE Transactions on Robotics and Automation*, 19(3) pp. 371-81, 2003.
- [6] S. Lange, N. Sunderhauf, and P. Protzel, A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-Denied Environments, in *Proceedings of the International Conference on Advanced Robotics (ICAR)*, 2009.
- [7] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments", in *International Conference on Robotics and Automation (ICRA)*, 2010.
- [8] L. R. G. Carrillo, E. Randon, A. Sanchez, A. Dzul and R. Lozano, "Stabilization and trajectory tracking of a quad-rotor using vision", *Journal of Intelligent and Robotic Systems*, 61(1-4), 2011, pp 103-118.
- [9] E. Altug, J. P. Ostrowski and R. Mahony, "Control of a Quadrotor Helicopter using Visual Feedback," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pp. 72-77, 2002.
- [10] D. Lee, H. J. Kim, S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter", *Int. J. Control Autom. Syst.* 7(3), 419428 (2009)
- [11] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, Hovering flight and vertical landing control of a vtol unmanned aerial vehicle using optical flow, in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 801-806.
- [12] Hutchinson, S., Hager, G., and Corke, P., "A tutorial on visual servo control", *Robotics and Automation, IEEE Transactions on*, 12(5) 1996, pp. 651-670.
- [13] www.vicon.com

9.8 Using visual servoing to point the gimbal at a target

RWB: Do something similar to the paper below.

Image-Based Pointing and Tracking for Inertially Stabilized Airborne Camera Platform

Zdeněk Hurák, *Member, IEEE*, and Martin Řezáč

Abstract—This paper describes a novel image-based pointing-tracking feedback control scheme for an inertially stabilized double-gimbal airborne camera platform combined with a computer vision system. The key idea is to enhance the intuitive decoupled controller structure with measurements of the camera inertial angular rate around its optical axis. The resulting controller can also compensate for the apparent translation between the camera and the observed object, but then the velocity of this mutual translation must be measured or estimated. Even though the proposed controller is more robust against longer sampling periods of the computer-vision system than the decoupled controller, a sketch of a simple compensation of this delay is also given. Numerical simulations are accompanied by laboratory experiments with a real benchmark system.

Index Terms—Image-based visual servoing, inertial stabilization, line-of-sight stabilization.

I. INTRODUCTION

A. Inertial Line-of-Sight Stabilization on Mobile Carriers

THE very basic control task for steerable cameras or antennas mounted on mobile carriers such as trucks, unmanned aircraft or ships, is to keep the commanded line of sight (optical axis) still even in presence of various disturbing phenomena like mass disbalance, aerodynamic (or wind-induced) torque and possible kinematic coupling between gimbal axes, see Fig. 1. Motivated also by defence technological needs, the topic of inertial stabilization was studied extensively in the past few decades. Several relevant papers from 1970s through 1990s were archived in the selection [1]. Dedication of a full issue of *IEEE Control System Magazine* (February 2008) featuring nice survey papers [2]–[4] confirms that the topic is still relevant for the engineering community. Another recent issue of the same journal brings a rigorous analysis of control problems related to a standard double gimbal system [5], though it is not directly applicable to inertial stabilization.

Manuscript received January 31, 2011; revised May 11, 2011; accepted July 06, 2011. Manuscript received in final form August 05, 2011. Date of publication September 12, 2011; date of current version June 28, 2012. Recommended by Associate Editor F. Caccavale. This work was supported in part by the Ministry of Education of the Czech Republic within the Centre for Applied Cybernetics (IM0567) and by the Ministry of Industry and Trade of the Czech Republic within the Project TIP FR-TI1/265.

The authors are with the Faculty of Electrical Engineering, Czech Technical University, Prague 16627, Czech Republic (e-mail: hurak@fel.cvut.cz; rezac.martin@fel.cvut.cz).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCST.2011.2164541

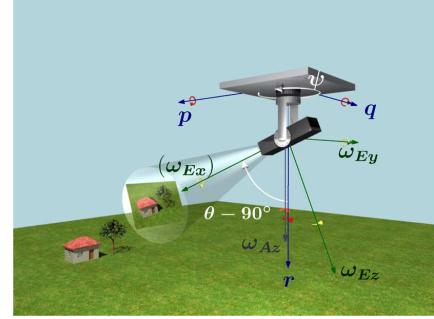


Fig. 1. Basic scenario for inertial line-of-sight stabilization. Depicted in green are the components ω_{Ex} , ω_{Ey} , ω_{Ez} of the vector of inertial angular rate of the elevation frame (as measured by MEMS gyros attached to the camera), blue vectors p , q , r denote the rate components of the base (UAV here). The ω_{Az} component is attached to the outer gimbal (the other two components are not shown). Two white arcs denote the two relative angles. The origins of the coordinate frames are assumed to coincide in the computations.

B. Automatic Visual Tracking on Mobile Carriers

All of the above cited works (including the references made therein) mostly focus on the task of inertial stabilization only. The issue of extending the inertial rate stabilizing feedback loop to visual tracking system is only dealt with at a rather simplistic level in [2] by suggesting the common cascaded control structure for every rotational degree of freedom: a single-input-single-output inner (inertial rate stabilization) loop is accepting commands from the output of the corresponding outer (visual tracking) loop. There are some pitfalls hidden in this decoupled approach, though. This paper will describe the troubles that are encountered when using the classical double-gimbal platform and offer a solution. To the best of the authors knowledge, this is the first formal treatment of visual pointing and tracking for inertially stabilized camera systems. Preliminary versions of this paper were presented at [6] and [7]; this paper includes corrections and some minor theoretical extensions but the most important extension is in supporting the theoretical analysis by reporting on laboratory experiments with a realistic benchmark platform.

C. Benchmark System

The configuration considered in this study is the common two-degree-of-freedom configuration: double gimbal system. The inner gimbal allows for elevation of the payload, the outer gimbal allows for a change in heading (or azimuth) angle. A benchmark system was designed and built within a project coordinated by Czech Air Force and Air Defence Technological

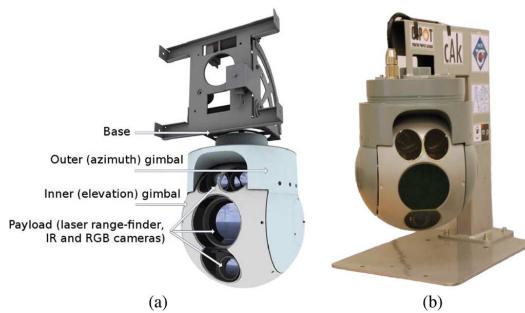


Fig. 2. Platform developed by Czech Air Force and Air Defense Technical Institute in collaboration with Czech Technical University in Prague and ESSA company. (a) 3-D visualization of the platform: both the azimuth and the elevation angles can rotate $n \times 360^\circ$. (b) Photo of the benchmark system. The supporting structure is only used in a lab.

Institute (Vojenský ústav letectva a PVO) in collaboration with Czech Technical University in Prague and ESSA company. The payload consists of a regular RGB camera, infrared camera and laser range-finder, see Fig. 2. Direct drive motors are used for the two axes and MEMS-based gyros (inertial rate sensors) are attached to the payload.

D. Notation for Coordinate Frames and Their Rotations

The paper relies on expressing rotations of coordinate frames with respect to some other coordinate frames. The right-handed orthogonal coordinate frames are represented by triads of vectors $\{x, y, z\}$ and for simplicity they all assume a common origin; this is certainly justifiable when faraway objects are tracked. The coordinate frames and their symbols used in subscripts and superscripts are: the reference coordinate frame $[R]$ aligned with the ground but translated to the center of gravity of the carrier, its z_R axis oriented towards the ground as is common in aerospace applications; the base coordinate frame fixed to the body of the carrier $[B]$ with its x_B axis heading forward and y_B to the starboard; the coordinate frame attached to the outer (azimuth) gimbal $[A]$, which can rotate with respect to the carrier around the $z_B = z_A$ axis; the coordinate frame attached to the inner (elevation) gimbal $[E]$, which can rotate with respect to the azimuth gimbal around the $y_A = y_E$ axis; and finally the coordinate frame attached to the camera $[C]$. Rotation of $[C]$ with respect to $[E]$ is fixed and is used just for the “esthetic” purpose of (re)denoting the camera optical axis as the z_C axis.

The sequence of the two key rotations expressing the pose of the inner gimbal (fixed to camera) with respect to the base (carrier) is visualized in Fig. 3 and for completeness it is given by

$$R_A^B = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

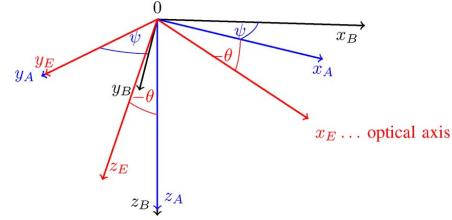


Fig. 3. Composition of rotation of coordinate frames attached to the carrier (base), outer (azimuth) gimbal, and inner (elevation) gimbal. The $[C]$ frame attached to the camera not visualized here, see Fig. 5.

and

$$R_E^A = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2)$$

where the lower and upper indices are used here as “rotation matrix expressing the coordinate triade of the A frame within the B frame”. Applying the right-hand rule, the (outer) azimuth gimbal rotates to right for the positive angle ψ and the (inner) elevation gimbal rotates up for a positive increment in the θ angle. Using the common shorthand notation like $c_\psi = \cos \psi$, the composition of the two rotations is given by the matrix product

$$R_E^B = \begin{bmatrix} c_\psi c_\theta & -s_\psi & -c_\psi s_\theta \\ s_\psi c_\theta & c_\psi & -s_\psi s_\theta \\ -s_\theta & 0 & -c_\theta \end{bmatrix}. \quad (3)$$

Finally, the fixed rotation between the camera frame and the elevation frame is given by

$$R_C^E = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \quad (4)$$

When specifying angular rates, the subscript/superscript scheme used here follows the common style, defined for instance in [8]: one needs to tell which coordinate frame is rotating with respect to which other coordinate frame, and in which coordinate frame is such a vector expressed. For example, $\omega_{A,E}^R$ stands for the angular rate of the Elevation gimbal with respect to the Azimuth gimbal, expressed in the Reference frame. Oftentimes, the notation is relaxed in the paper to avoid cluttering the formulas with indices. For instance, ω_A is a short notation for $\omega_{R,A}^A$, that is, the inertial angular rate of the A gimbal. Its z component is then ω_{Az} .

II. INERTIAL ANGULAR RATE STABILIZATION

In order to make the line of sight insensitive to external disturbances, a simple controller structure can be used. Two decoupled single-input-single-output (SISO) inertial rate controllers suffice, one for each measured (component of the) inertial angular rate. Namely, the following:

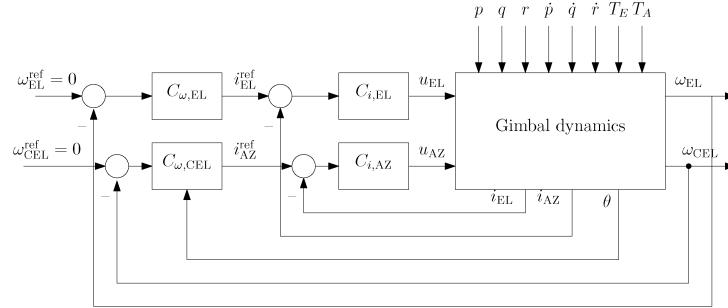


Fig. 4. Inertial stabilization. Two independent (decoupled) SISO feedback loops, one for each rate gyro attached to the body of camera, $\omega_{EL}(= \omega_{E_y})$ and $\omega_{CEL}(= \omega_{E_z})$. The cross-elevation stabilizing controller must contain secant gain correction $1/\cos(\theta)$. The disturbing variables are the carrier roll, pitch, and yaw rates, p, q, r , respectively, their derivatives and external torques around the two motor axes. The innermost current loops are also depicted.

- the inertial angular rate ω_{E_y} (also denoted with the mnemotechnic ω_{EL}) of the payload about the axis of the elevation motor (camera elevation rate);
- and the inertial angular rate ω_{E_z} of the payload around its own vertical axis, also nicknamed camera cross-elevation rate (and denoted ω_{CEL}) since its axis is always orthogonal to the ω_{EL} axis.

This is visualized in Fig. 1. The resulting decoupled controller configuration is in Fig. 4. It is clear from Fig. 1 that the cross-elevation controller must include a secant gain correction $1/\cos(\theta)$, because the motor in the azimuth gimbal cannot directly affect $\omega_{CEL}(= \omega_{E_z})$. It can only do so indirectly through ω_{Az} . It is only when the camera is pointing to the horizon, that is, when $\theta = 0$, that $\omega_{Az} = \omega_{CEL}(= \omega_{E_z})$. See [3] for details.

Even though there is some gyroscopic coupling between the two axes (see [9], [10] for full models or [5] for the simplified version when the base is still), its influence is not worth designing a multiple-input–multiple-output (MIMO) rate controller. This neglected gyroscopic effect can be cast as yet another external disturbing torque and as such left to the rate controller to suppress.

III. IMAGE TRACKER DESCRIPTION

This section gives some details on the automatic image tracker, even though for the purpose of this paper it can be viewed as a black-box device. It serves the purpose of a “relative displacement sensor” and its communication with the rate loop is unidirectional.¹

The image tracker is an algorithm (implemented in a software routine) which recognizes the target object in the input image frame sequence and returns its coordinates in every image frame of the acquired sequence. The target is usually selected by a human operator, who marks the target in the first image frame. The control system then steers the camera in order to get the (image of the) target into the center of the field of view (center of the image frame) and hold it there.

It appears that image tracking of a ground object for the purpose of airborne surveillance is one of the most difficult image tracking tasks [11], [12]. The following are the reasons.

¹So far the image tracker used in the project takes no advantage of availability of the inertial rate measurements.

- Weak visual differences between the tracked object and the background.
- Many localization results assume that the background is static with respect to the camera, which is certainly not the case for cameras carried by aircraft.
- The image of the target object is usually very small. Tracker useful for airborne surveillance must work with objects which projects into images as patches smaller than 10×10 pixels.
- The relative movement of the image of the target can be very fast (multiples of object’s size from frame to frame) mainly due to rotation of camera, either intentional or unwanted.

There are many algorithms for image tracking in different conditions and tasks. The following families of algorithms can be considered.

- *Pattern matching algorithms* compare the captured image of the target with its model. Error function is defined that measures the similarity between the model (represented by an image patch) and the actual image in a specified position. The algorithm searches for a position where the error is at minimum. The representative of this class of algorithms are sum of square differences (SSD) [13] and the basic Kanade-Lucas (K-L) algorithm [14].
- *Feature tracking algorithms* are based on extraction of a small number features (important points) from the image frame and their tracking. The features should satisfactorily describe the object and be easily recognized in next image frame. There are many ways how to define and track the features. Mean-shift algorithm is based on a density analysis of feature space [15], but some more sophisticated algorithm were developed, the examples of which are Kanade-Lucas-Tomasi tracker (KLT) [16], scale invariant feature transform (SIFT) [17], maximally stable extremal regions (MSER) [18] and [19].
- *Object recognition trackers* aim at recognizing an object in the image frame. Machine-learning techniques (for example [20]) are applied on features which are extracted in a preprocessing stage. Target object is recognized and localized in each image. The learning stage of the procedure could be run offline, but then only a “learned” class

of objects could be tracked. Online learning turns out more useful for the present application. An example of this class of algorithms is online boost [21].

These algorithms differ in tracking capabilities when it comes to different types of objects, required conditions, machine time consumption and many other operational characteristics. The experiments conducted with the benchmark system relied on the SSD algorithm. It is easy to implement and is capable of tracking small objects and recognizing similar objects. A major disadvantage of the SSD algorithm is its high machine time consumption.

From the control system viewpoint, the major complication that the image tracker brings into the feedback loop is the transportation delay. The delay consists of computational time for the actual tracker algorithm and the time for image capturing by the hardware. In the benchmark system, a standard PAL video camera with analog output was used, for which 5–20 ms were needed for image capturing. Then it takes another 40 ms (25 Hz frequency) to transmit an analog video signal from the camera. The analog video signal is then captured by a video grabber and after some 10 ms is transferred to the computer. Only then the tracking algorithm could be started. Therefore the minimum transportation delay is around 60 ms plus the computing time of tracker algorithm. It is only the latter that could be minimized by implementing more efficient algorithms.

IV. MODELING THE DYNAMICS FOR POINTING AND TRACKING

Before starting discussions on ways to design and implement a feedback controller for the task of pointing and tracking, a model must be developed. At the initial treatment, the inertial angular rate (feedback) loops can be regarded as perfect within the appropriate frequency range and saturation bounds, that is, the commanded inertial angular rates $\omega_{\text{EL}}^{\text{ref}}$ and $\omega_{\text{CEL}}^{\text{ref}}$ can be regarded as perfectly followed by the inner loops. To develop a mathematical model for this idealized situation, a few basic concepts from the established domain of visual servoing will be given. The next few paragraphs are fully based on the two chapters from [8] dedicated to computer vision and vision-based control. They are given here just for a convenience of a reader nonacquainted with these concepts. Another comprehensive introductory material is [22].

A. Perspective Projection

The objects to be observed are located in the full 3-D world while the camera can only record their 2-D image. The coordinates of the object in the world (on the ground) expressed in the camera frame are given by $P = [x, y, z]^T$. Simplifying a bit the model of the optics, we make the so-called pinhole assumption, which defines the image coordinate frame as follows. At a focal distance λ from the origin of the camera coordinate frame, consider the image plane orthogonal to the optical axis of the camera. The coordinates of the point of intersection of the line connecting the object with the origin are $p = [u, w, \lambda]^T$. The vector $s = [u, w]^T$ thus gives the image coordinates. All this is visualized in Fig. 5. Thanks to the pinhole assumption

$$k[x \ y \ z]^T = [u \ w \ \lambda]^T \quad (5)$$

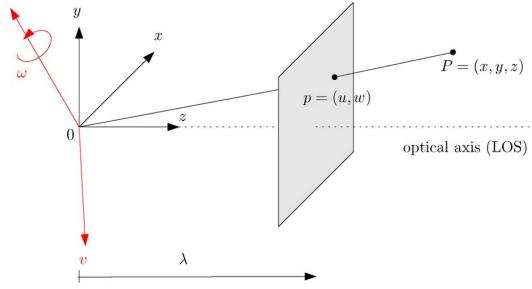


Fig. 5. Coordinates of the object on the ground expressed in the coordinate frame attached to the camera and (after projection) in the image plane. Rotation $\omega_{R,C}^C$ and translation v_C of the camera frame within with respect to the inertial frame is also illustrated (redrawn from [8]).

we have that

$$u = \lambda \frac{x}{z}, \quad w = \lambda \frac{y}{z}. \quad (6)$$

To make this story complete, the coordinates in the image plane should then be quantized and the origin should be moved to the lower left corner to obtain pixel coordinates $[r, c]^T$

$$-\frac{u}{s_x} = (r - o_r), \quad -\frac{w}{s_y} = (c - o_c) \quad (7)$$

where s_x and s_y are the pixel dimensions and o_r and o_c are half the width and height of the image frame in pixels.

Nonetheless, for the analysis in this paper we will stick to u and w variables to make the formulas less involved. In simulations and experiments presented here, the “pixelized” information will be considered centered (again, in the name of simplicity). That is, we will use

$$-\frac{u}{s_x} = r, \quad -\frac{w}{s_y} = c. \quad (8)$$

The computer vision system that processes the images captured by the camera can surely perform this centering before sending the data to the pointing-tracking controller.

B. Camera Motion and the Interaction Matrix

This subsection is again extracted from the nice introduction to image-based visual servoing in the textbook [8]. Consider the movement of the camera in the inertial space characterized by its linear and rotational velocities $v_C = [v_{Cx}, v_{Cy}, v_{Cz}]^T$ and $\omega_C = [\omega_{Cx}, \omega_{Cy}, \omega_{Cz}]^T$, both expressed in the camera frame. Stack them together to form a time-dependent vector $\xi(t) = [v_C(t), \omega_C(t)]^T \in \mathbb{R}^6$. To be rigorous, we should write $\omega_{R,C}^C$ to emphasize that it is an angular rate of the camera frame with respect to the reference (inertial) frame, expressed in the camera frame, and similarly $v_{o_C}^C$ to emphasize that it is a translational velocity of the origin o_C of the camera coordinate frame with respect to the inertial frame, also expressed in the camera frame. But this would yield the equations illegible.

The motion of the object as viewed by the camera is described by the so-called image feature velocity $\dot{s}(t)$, which can be obtained as a derivative of the image feature vector (in the simplest case it is just a position of some significant point). The nice thing

is that it is possible to relate ξ and $\dot{\xi}$ by a transform resembling the concept of Jacobian and denoted often an interaction matrix or image Jacobian

$$\dot{\xi}(t) = L(s, z, \lambda)\xi(t). \quad (9)$$

Next we consider the simplest case of a single-point feature and assume that the ground object does not move. Extension of the results stated here to the case of a moving ground target is feasible, but the resulting interaction matrix will be a function of the velocity of the ground object, which is unknown to the inertial stabilization system (but it may be worth exploring if at least rough estimate of the object velocities can be used). This matrix is derived in [8, p. 415, eq. (12.14)] as

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uw}{\lambda} & -\frac{\lambda^2+u^2}{\lambda} & w \\ 0 & -\frac{\lambda}{z} & \frac{w}{z} & \frac{\lambda^2+w^2}{\lambda} & -\frac{uw}{\lambda} & -u \end{bmatrix} \begin{bmatrix} v_{Cx} \\ v_{Cy} \\ v_{Cz} \\ \omega_{Cx} \\ \omega_{Cy} \\ \omega_{Cz} \end{bmatrix}. \quad (10)$$

The procedure for the derivation is straightforward: first, express the position of a fixed (not moving) point on the ground in the coordinate frame of the moving (rotating and translating) camera, and then project these new coordinates to the image plane. (It is vital to keep in mind within which coordinate frame the velocity vectors are being expressed. This is quite tedious. In this case, both the translational and rotational velocities are indeed considered with respect to the inertial reference frame but are expressed in the camera frame).

It appears useful to highlight the structure in the interaction matrix by writing it as a composition of two parts

$$\dot{\xi} = L_v(u, w, z)v_C + L_\omega(u, w)\omega_C \quad (11)$$

because it turns out that only the part corresponding to the translation of the camera coordinate frame depends on the image depth (distance to the observed ground object) z . The rotational part is independent of z . The focal length λ is regarded as a fixed parameter.

The three components ω_{Cx} , ω_{Cy} and ω_{Cz} define the inertial angular rate vector $\omega_C = [\omega_{Cx}, \omega_{Cy}, \omega_{Cz}]^T$ in the camera coordinate frame $[C]$, which is rotated with respect to the elevation gimbal frame $[E]$ using a fixed (constant) rotation matrix R_E^C

$$\omega_E = R_E^C \omega_C = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \omega_C. \quad (12)$$

While by using the two direct drive motors it is possible, at least partially, to affect the vector ω_E by commanding its two components $\omega_{Ey} (= \omega_{EL})$ and $\omega_{EZ} (= \omega_{CEL})$, it is rather unlikely that the translational velocity v_C will be commanded by the autopilot based on the needs of the pointing-tracking algorithm. (But some projects might allow it.)

Therefore, in order to develop some insight into the model, forget v_C for a moment (assume $v_C = 0$ temporarily, it can be treated as a disturbance later, either estimated or not). Using

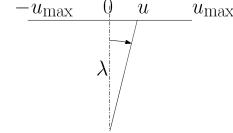


Fig. 6. Relationship between the image coordinate system and the corresponding angle.

the transformation (12) and the rotation part of the interaction matrix (10) we get

$$\begin{bmatrix} \dot{u} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -\frac{uw}{\lambda} & \frac{\lambda^2+u^2}{\lambda} - w \tan \theta \\ \frac{\lambda^2+w^2}{\lambda} & \frac{uw}{\lambda} + u \tan \theta \end{bmatrix} \begin{bmatrix} \omega_{EL}^{\text{ref}} \\ \omega_{CEL}^{\text{ref}} \end{bmatrix} \quad (13)$$

and the camera tilt angle θ evolves according to

$$\dot{\theta}(t) = \omega_{EL}^{\text{ref}}(t). \quad (14)$$

C. Linearization at Distinguished Operating Points

In order to develop an insight into the model (13), consider the situation when $\theta = 0$ (a wing-level flight and the camera pointing towards the horizon) and $w = 0$ (the observed object vertically centered on the screen). The dynamics is then constrained to one dimension and the equation simplifies to

$$\dot{u} = \frac{\lambda^2 + u^2}{\lambda} \omega_{CEL}^{\text{ref}}. \quad (15)$$

The term $(\lambda^2 + u^2)/\lambda$ expresses the nonlinear relationship between the angle and the line segment in the image plane. This is illustrated in Fig. 6.

The focal length λ for the system used by the authors ranges in [4.2, 42] mm. The width of the CCD camera chip is 3.2 mm. Hence, for the maximum zoom, the nonlinear term can be approximated by λ even for u approaching the maximum value, that is, the observed objects is initially located near the borders of the field of view (and the control goal is to bring it to the center). The linear dynamics is then

$$\dot{u} = \lambda \omega_{CEL}^{\text{ref}} \quad (16)$$

that is, the model of dynamics is represented by a pure integrator with a gain λ (given by the optics). For shorter focal lengths (approaching the lower limit of 4.2 mm) this approximation is only valid for correspondingly smaller u , that is, for tracking purposes only, not for (re)pointing over a large part of the image plane.

D. Analysis of Achievable Bandwidth for Pointing and Tracking

The computer vision system works at discrete time instants with the sampling period T_s ranging between something like 0.1 s and 2 s (depending on complexity and performance of the algorithm), which is relatively long compared to 250 Hz of the inner inertial rate loop. This introduces a total delay τ of about $1.5T_s$ into the feedback loop.

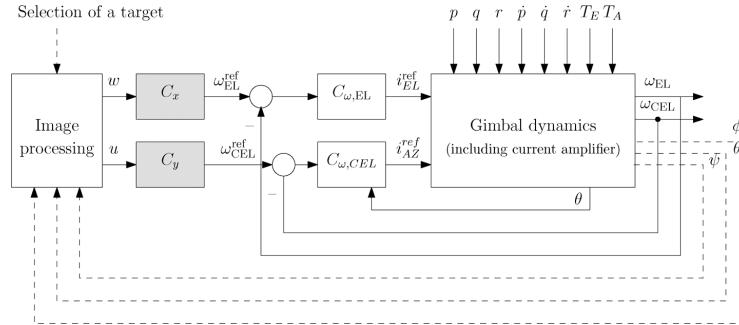


Fig. 7. Naive pointing-tracking system formed by two SISO loops closed around two inertial rate stabilization loops. The dashed lines are not signals truly fed back to the pointing-tracking controller. These are variables representing orientation of the camera which affects the position and orientation of objects in the image plane.

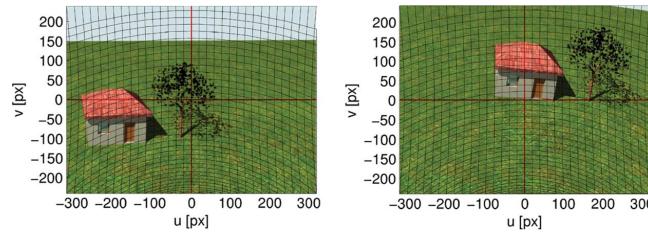


Fig. 8. Illustration of how in an attempt to steer the camera such that the image of the roof of the house gets back to the middle of the field of view using azimuth motor only, the introduced rotation of the camera around its optical axis makes the horizontal movement curved. Consequently, correction in vertical direction using the elevation motor is needed. Curvilinear coordinate system in the image plane corresponds to the initial elevation of camera by $\theta = -54^\circ$ with respect to the body of the aircraft.

It is known that the achievable bandwidth is limited by several properties of the system, delay being one of them. With the sampling period of the image tracker set to $T_s = 0.5$ s, the achievable bandwidth is approximately limited by

$$\omega_{BW} < \frac{1}{\tau} = \frac{1}{1.5T_s} = 1.3 \text{ rad/s} = 0.2 \text{ Hz}. \quad (17)$$

It is derived in [23] from ideal closed-loop transfer functions achievable for systems with a delay τ . Ideally, $T(s) = 1e^{-\tau s}$, therefore $S(s) = 1 - e^{-\tau s}$. By Taylor series expansion $S(s) \approx \tau s$. Therefore $|S(j\omega)|$ crosses 0 dB at about $1/\tau$.

This suggests that the fastest possible pointing-tracking loop will work up to a fraction of 1 Hz if the information from the image tracker is provided twice per second and is delayed one sample period. This roughly corresponds to the classical rule-of-thumb rules [24] for selection of a sampling rate for undelayed systems as 10 to 20 times the closed-loop bandwidth.

V. DECOUPLED POINTING AND TRACKING

Proceeding one step further beyond the mere inertial stabilization, the question of the most suitable feedback control configuration for automatic visual tracking pops up. Shall we use the immediate extension which closes a SISO tracking loop around the corresponding SISO inertial rate loop?

The cascade approach is justified: whereas the inner (inertial rate) loop aims to attenuate the disturbances at middle and high

frequencies, the outer (pointing) loop should be active at low frequencies. This straightforward but naive solution is in Fig. 7.

Insisting on decoupled controllers is plausible from an implementation viewpoint. There is a trick hidden here, though, as seen in Fig. 8. When the automatic computer vision tracker detects a regulation error in the horizontal direction in the image plane while seeing no error in vertical direction, the simple cascaded structure of Fig. 7 would command the azimuth motor only. This motor alone, however, cannot create a purely horizontal motion in the image plane when $\theta \neq 0$. A geometric explanation can be found in Fig. 1: to steer the camera such that the image of an object moves horizontally in an image plane, one would need to command the cross-elevation inertial rate ω_{CEL} (denoted as ω_{Ex} in the figure). However, the motor can only affect the component of the inertial rate in the direction of the azimuth motor axis, that is, ω_{Az} . As soon as there is some misalignment between the two, that is, when the camera is tilted up or down to the ground while the aircraft is in level flight ($\theta \neq 0$), the vector oriented in the azimuth motor axis of length ω_{Az} has some nonzero projection ω_{Ex} to the camera optical axis. Consequently, some unwanted rotation of the image as well as vertical displacement are introduced. Curvilinear coordinate mesh in Fig. 8 is generated by the nonlinear dynamics (13).

However, with sampling rate of the outer (image-based pointing-tracking) loop fast enough, the error introduced by the coupling between the two camera axes would be corrected in the very next step, when the regulation error in vertical direction

in the image plane is detected and a correcting command to the elevation motor would be sent. The currently implemented prototype system achieves sampling rate of 15 Hz, which seems enough to justify this naive approach. Having scanned the available literature, the authors can only suspect that some of the available commercial systems follow this approach too. The motivation for this paper is to improve this scheme, because a bit more advanced and computationally intensive computer vision algorithms can slow down the sampling rate of the outer loop to something like 1 or 2 s.

VI. FEEDBACK LINEARIZATION-BASED VISUAL POINTING AND TRACKING

The key idea for an improvement described in the rest of the paper is that the curvature of the coordinate axes as in Fig. 8 can be compensated for by measuring the third component of the inertial angular rate of the camera body, the one along its optical axis, the so far unused measurement ω_{Ex} . It is available at the sampling rate a few orders of magnitude faster than what the computer vision system provides. Using this information, exact feedback linearization can be implemented in the controller following standard techniques from image-based visual servoing introduced next.

The idea behind image-based visual servoing is that an error “sensed” in the image plane by the image tracker as

$$e(t) = s(t) - s^{\text{ref}} \quad (18)$$

can be eliminated by commanding a proper value of $\xi(t)$, which characterizes the velocity of the camera frame. Note that $s^{\text{ref}} = 0$ when the task is to bring the image of the object into the central position in the image plane by pushing $s(t) = [u(t), w(t)]^T$ to zero. How to find a proper ξ ? Simply by inverting the interaction matrix L . In the case of a single-point feature, the matrix is 2×6 , which suggests that such a solution will not be unique. Which one to pick? It will be shown shortly that there is one important constraint here which makes only one solution acceptable.

In contrast to common robotics tasks, here we cannot influence the translational position of the camera frame (unless there is a bidirectional communication between the UAV autopilot and the inertial stabilization & visual tracking system). Hence the linear velocity $v_C = [v_{Cx}, v_{Cy}, v_{Cz}]^T$ of the camera coordinate origin needs to be taken as given (enforced) from the outside. But then the task of determining ξ at a given time instant consists in solving the linear system (10) with the term corresponding to translation moved to the right-hand side

$$L_\omega(u, w)\omega_C = \dot{s} - L_v(u, w, z)v_C. \quad (19)$$

The 2×3 matrix L_ω has a 1-D nullspace parameterized by

$$\mathcal{N} = \{\omega_c = k[u \ w \ \lambda]^T\} \quad (20)$$

which can be interpreted quite intuitively: rotating the camera about the line connecting the observed point and the origin of the camera frame does not contribute to a change of the coordinates

of the point in the image plane. With the right pseudoinverse of L_ω given by

$$L_\omega^\# = \begin{bmatrix} 0 & \frac{\lambda}{\lambda^2+u^2+w^2} \\ -\frac{\lambda}{\lambda^2+u^2+w^2} & 0 \\ \frac{u}{\lambda^2+u^2+w^2} & -\frac{u}{\lambda^2+u^2+w^2} \end{bmatrix} \quad (21)$$

all solutions are parametrized by a single constant k

$$\omega_C = L_\omega^\# \dot{s} - L_\omega^\# L_v v_C + k[u \ w \ \lambda]^T. \quad (22)$$

Substituting and abusing k since it is arbitrary we get

$$\omega_C = \frac{1}{z(\lambda^2+u^2+w^2)} \begin{bmatrix} \lambda^2 v_y - \lambda v_z w + \lambda \dot{w} z + k u \\ -\lambda^2 v_x + \lambda v_z u - \lambda \dot{z} w + k w \\ -\lambda u v_y + \lambda w v_x - u \dot{v} z + \dot{u} w z + k \lambda \end{bmatrix}. \quad (23)$$

What we have obtained so far is a procedure which for given velocities \dot{u} and \dot{w} of a point feature in the image plane computes the required angular velocity vector ω_C (it does not hurt now to use the full notation $\omega_{R,C}^C$) of the camera. A single arbitrary parameter k can be used to give some choice, which is the key idea to be exploited next.

A. Simple Proportional Image-Based Pointing and Tracking

In order to pull $s(t)$ to the vicinity of $(0,0)$ in the image plane, a cascade control structure can be used: the pointing-tracking controller sets the reference rate vector $\dot{s}^{\text{ref}}(t)$ such that its actual value $\dot{s}(t)$ goes to zero. One simple approach is to require exponential stability, that is, both $u(t)$ and $w(t)$ go to zero values according to

$$\dot{s}(t) = As(t) \quad (24)$$

where A has nonnegative eigenvalues. The simplest solution can be obtained by restricting A to a diagonal matrix $A = -\alpha I$ for some real positive α and then

$$\dot{s}(t) = -\alpha s(t). \quad (25)$$

The larger the α , the faster the error in the image plane goes to zero. Practical considerations of the choice of this parameter are discussed at the end of this section. Now, how can we force this error to evolve as in (25)? Noting that $\dot{s}(t)$ is related to the camera inertial angular velocities according to (23), we can conclude that asymptotically stable image error is guaranteed if the camera inertial velocities follow the reference value

$$\omega_C^{\text{ref}} = \frac{1}{z(\lambda^2+u^2+w^2)} \begin{bmatrix} \lambda^2 v_y - \lambda v_z w - \lambda \alpha w z + k u \\ -\lambda^2 v_x + \lambda v_z u + \lambda \alpha u z + k w \\ -\lambda u v_y + \lambda w v_x + k \lambda \end{bmatrix}. \quad (26)$$

It is not clear at this moment whether and how such a rotation rate of the camera can be established by the two motors. It is the free parameter k that can help to pick such a reference inertial velocity vector ω_C^{ref} of the camera that is realizable by the two motors.

B. Establishing the Camera Inertial Rate Using Two Motors

Once we know the required inertial angular rate of the camera, what remains is to express it via constant transformation R_C^E in the inner gimbal frame

$$\begin{aligned}\omega_E^{\text{ref}} &= R_C^E \omega_C^{\text{ref}} \\ &= \frac{1}{z(\lambda^2 + u^2 + w^2)} \begin{bmatrix} -\lambda uv_y + \lambda wv_x + k\lambda \\ -\lambda^2 v_y + \lambda v_z w + \lambda \alpha w z - ku \\ \lambda^2 v_x - \lambda v_z u - \lambda \alpha u z - kw \end{bmatrix}. \quad (27)\end{aligned}$$

The task for the inertial angular rate control system is to follow this velocity by commanding the two motors. It is important to keep track of the corresponding frames. The resulting ω_E^{ref} is fully labeled as $\omega_{R,E}^{\text{ref}}$ as it gives the required inertial rotation rate of the inner gimbal. Its true value is measured by the three-axis MEMS gyro fixed to the inner gimbal.

Now comes the key part. Having only two motors, it is not possible to set all the three components of the vector of inertial angular velocity independently. But the free scalar parameter k can be used to pick a specific triple that requires no change with respect to the current value of ω_{Ex} (inertial angular rate of the camera around its optical axis). The value of ω_{Ex} must be available to the controller then. Solving (27) for the value of k guaranteeing that the x component of the vector on the right is equal to the measured ω_{Ex} gives

$$k = -uv_x + uw_y + \frac{z(\lambda^2 + u^2 + w^2)}{\lambda} \omega_{Ex}. \quad (28)$$

Substituting this value back to the expressions for the other two components of the reference angular rate vector, the expressions for the controller outputs follow

$$\begin{aligned}\omega_{EL}^{\text{ref}} = \omega_{Ey}^{\text{ref}} &= \frac{\alpha w \lambda}{\lambda^2 + u^2 + w^2} - \frac{\omega_{Ex} u}{\lambda} \\ &- \frac{\lambda^2 v_y - \lambda w v_z - u w v_x + u^2 v_y}{z(\lambda^2 + u^2 + w^2)} \quad (29)\end{aligned}$$

$$\begin{aligned}\omega_{CEL}^{\text{ref}} = \omega_{Ez}^{\text{ref}} &= -\frac{\alpha u \lambda}{\lambda^2 + u^2 + w^2} - \frac{\omega_{Ex} w}{\lambda} \\ &+ \frac{\lambda^2 v_x - \lambda u v_z - u w v_y + w^2 v_x}{z(\lambda^2 + u^2 + w^2)}. \quad (30)\end{aligned}$$

The expressions (29) and (30) for the controllers are structured such that three terms can be immediately recognized in each controller: a term corresponding to a regulation error in the corresponding axis as seen in the image plane, a term compensating for the rotation around the camera optical axis and finally a term attenuating the influence of mutual translational motion of the camera and the ground object.

In order to get an insight into this new controller and compare it with the originally proposed decoupled one, consider again the easy situation when the carrier is in level flight and the camera is pointing towards the horizon ($\theta = 0$). Neglect the translational velocities v_C . The observed object is vertically centered in the image plane, that is, $w = 0$. The expressions in (29) and (30) simplify to

$$\omega_{Ey}^{\text{ref}} = 0 \quad (31)$$

$$\omega_{Ez}^{\text{ref}} = -\alpha \frac{\lambda}{\lambda^2 + u^2} u. \quad (32)$$

Compare this simplified controller and the model of the system (15) valid for the same conditions. Apparently, the nonlinear term $\lambda/(\lambda^2 + u^2)$ serves just to invert the nonlinearity in the model. And this is what the controller does in general. It inverts the nonlinearity. In other words, it performs feedback linearization. Consideration of the inertial angular rate ω_{Ex} of the camera around its optical axis is another measure that the controller takes to invert the nonlinearity. For the maximum zoom ($\lambda = 42$ mm), the nonlinear term $\lambda/(\lambda^2 + u^2)$ is sufficiently close to λ and therefore the controller's action is driven by

$$\omega_{Ez}^{\text{ref}} = -\alpha \frac{1}{\lambda} u. \quad (33)$$

The simplification can take place even in a more general situation $u, w \neq 0$ but small, and λ large (and v_C still neglected). The general expression for the controller output then reduces to

$$\omega_{Ey}^{\text{ref}} = \frac{\alpha w}{\lambda} - \frac{\omega_{Ex} u}{\lambda} \quad (34)$$

$$\omega_{Ez}^{\text{ref}} = -\frac{\alpha u}{\lambda} - \frac{\omega_{Ex} w}{\lambda}. \quad (35)$$

This reduced controller reveals the key enhancement with respect to the fully decoupled design: the controller output contains contribution from the angular rate of the camera around the optical axis!

C. Summary of Controller Structure for Pointing and Tracking

The feedback-linearizing pointing-tracking controller in (29) and (30) does not preserve the decoupled structure (no longer two separate pointing-tracking controllers). Each of the two controllers accepts not only both the “measured” position errors, that is, u and w , but also it follows that:

- 1) the x -component of the vector ω_E describing the inertial angular rate of the camera around the optical axis;
- 2) estimates of the aircraft translational velocity with respect to the ground, expressed in the camera coordinate frame (v_{Cz} describes how fast the camera is approaching the target);
- 3) an estimate of depth z of the image, that is, the distance from the camera to the ground target.

Moreover, the technical parameter that the controller must be aware of is the focal length λ . An upgrade of the naive scheme proposed in Fig. 7 can thus be seen in Fig. 9.

The key challenge in implementing this controller fully is in providing the controller with the extra measurements and/or estimates of the three components of the translational velocity v_C and the distance z to the object. These could be approached using inertial measurement unit (IMU) in combination with a laser range-finder and possibly also in combination with a computer vision system. For instance, the depth z and the “towards the object” velocity v_{Cz} is sometimes estimated from the apparent size of an object in the image (covering the image of the object by some polygon and computing its area, which is suggested in [8]). This technique can turn out of limited use here, though, because the images of observed objects can span just a few pixels and determination of v_{Cz} is then very inaccurate.

On the other hand, these new “complications” caused by the requirements of measuring the translational velocities are not really new and tied to the proposed control scheme. They are

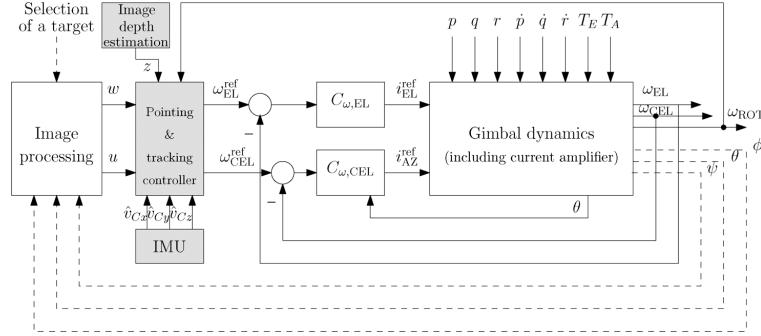


Fig. 9. Full feedback system with an image-based pointing controller aware of the angular rate about the optical axis and the translational motion. The pointing-tracking controller implements (29) and (30).

equally valid even with the (naive) decoupled control. Unless the translation velocity is known, one simply cannot tell whether the image is moving due to undamped aircraft oscillations or because the aircraft is approaching the object. But now, with the systematic analysis documented in this paper, the structure of the ideal controller is known. It is up to an engineer to decide whether or not to ignore the translational motion and regard its effects as unmeasured disturbance. Such disturbance is only significant at low frequencies and can be left for the image-based pointing loop to attenuate.

D. Practical Considerations for Setting the Image Dynamics

It is well known that the basic version of image-based visual servoing can be inefficient in terms of a requested manipulator movement needed to establish the requested image features trajectory. Several approaches have been proposed in the literature, see [25]–[27], that tackle the problem by separating the rotational and translational motion around and along the optical axis from the remaining controlled degrees of freedom. These issues have not been studied in the present situation since the only two controlled degrees of freedom are two rotations. The defective behavior of *camera retreat* is not present.

On the other hand, there is one practical aspect of setting the dynamics in the image plane that must be taken into consideration: placing the eigenvalues of the linearized system (25) too far in the left half plane makes the motional response of the system too fast for the image tracker. The image features then travel so fast in the image plane that the image tracker loses the grasp of the object (most image tracking algorithms such as [28] explore the nearest neighborhood only).

VII. NUMERICAL SIMULATIONS

Closed-loop responses with the two controllers were simulated for the image of the observed object initially located out of the center of the image frame. The control goal is to bring the observed object into the center of the field of view.

Three simulations were run (always with both controllers): Two with the sampling rate 15 Hz for different initial locations of the image of the object. And one for the slower sampling at 2 Hz. Results are visualized in Figs. 10–12.

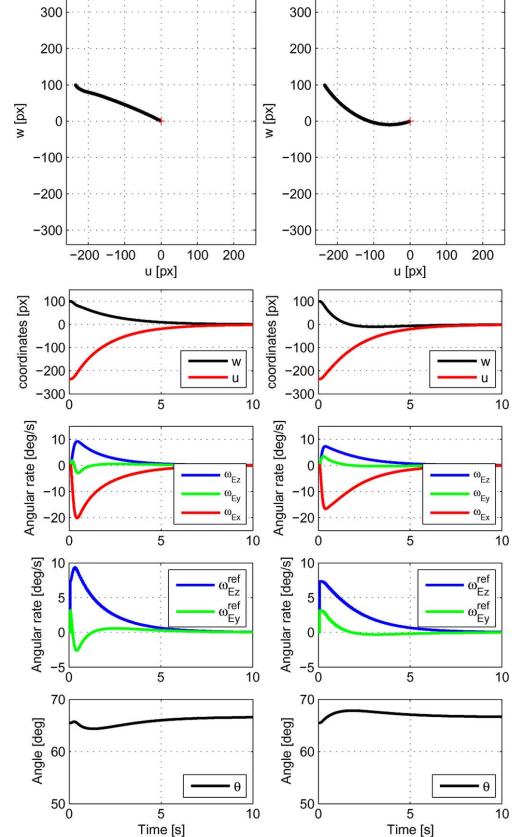


Fig. 10. Simulation 1: Responses of relevant quantities for $\theta(0) = 65.5^\circ$, tracker sampling rate $f_{sp} = 15$ Hz, $\alpha = 0.46$. Left: The proposed algorithm, right: the original decoupled approach.

Unlike in the design stage, in the simulations the inner (inertial rate) loops are not assumed to work ideally. That is, for the purpose of simulations their transfer functions are not iden-

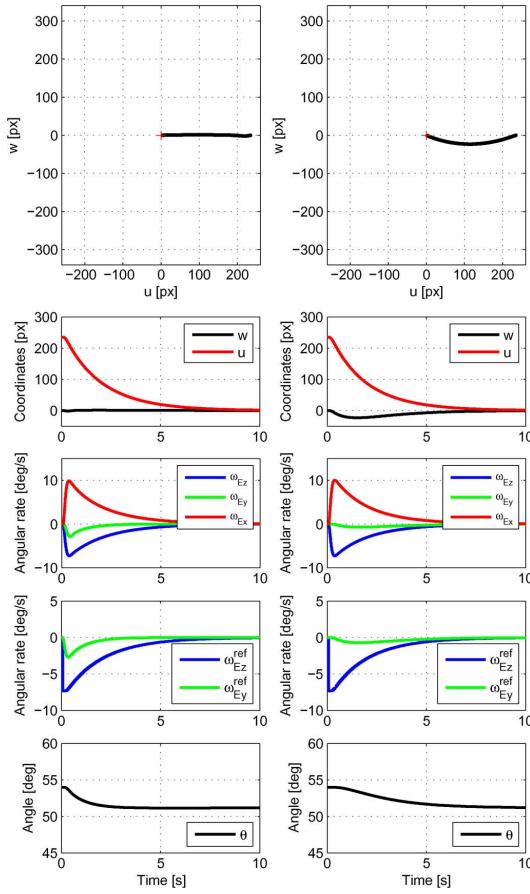


Fig. 11. Simulation 2: Responses of relevant quantities for $\theta(0) = 54^\circ$, tracker sampling rate $f_{sp} = 15$ Hz, $\alpha = 0.46$. Left: The proposed algorithm, right: the original decoupled approach.

tically equal to 1. The two SISO inertial rate loops are just standard feedback interconnections of a first-order system and P or PI regulator. Hence they can be modeled by low pass filters with a given bandwidth. However, the laboratory experiments described in the next section (see 15 and Figs. 16) reveal that this bandwith depends on the magnitude of the step in the reference inertial rate. This behavior is a consequence of a constraint on the amplitude of the control voltage inside the rate loops; a nonlinear model should be used to describe the rate loop more precisely. However, in order to obtain just a rough estimate of the system response, a low-order low pass filter seems satisfactory. The laboratory experiments with a real device then give a true assessment of the system performance.

No translation between the carrier and the observed object was assumed. The only physical parameters are those of the optics: focal length $\lambda = 42$ mm and resolution of the camera CCD chip is 640×480 pixels (u and w were scaled to pixels for visualization purposes).

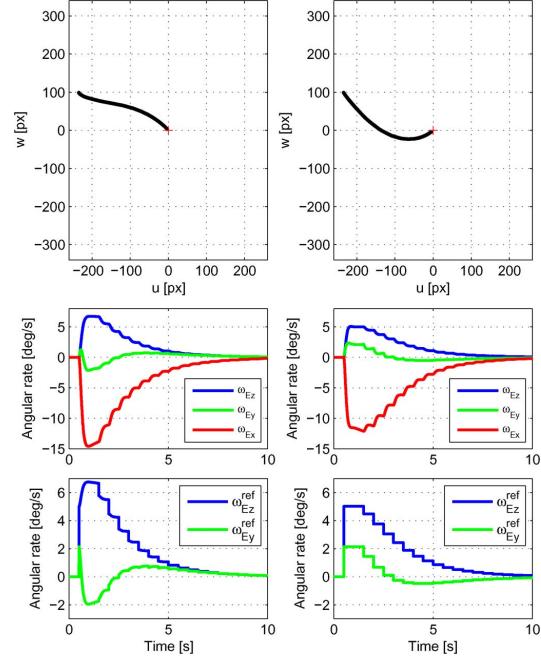


Fig. 12. Simulation 3: Responses of relevant quantities for $\theta(0) = 65.5^\circ$, tracker sampling frequency $f_{sp} = 2$ Hz, $\alpha = 0.31$. Left: The proposed algorithm; right: the original decoupled approach.

As discussed in the section devoted to the image tracker, the process of extracting the pointing tracking error from the visual information devours some computational time, hence u and w are only updated with the sampling rate ranging from 0.5 to 15 Hz, depending on the complexity of the algorithm. For simulations we choose 15 and 2 Hz. In addition, the data is always available to the controller with a delay of one sampling period. The sampling rate of the inner loop is up to two orders of magnitude higher (250 Hz, the maximum provided by the inertial angular rate sensors used in the project).

The simulation results confirm that the new controller struggles to follow a linear path in the image plane. Not only is the linear behavior easier to analyze but also is more plausible for a human operator. Some deterioration of a perfectly linear path in the image plane is visible. This is due to the nonideal inertial rate loops that were considered in the simulations.

The third simulation with the slower image tracker sampling rate of 2 Hz demonstrates that the new proposed algorithm updates the reference values for camera inertial rates at a faster sampling rate than the original decoupled controller.

VIII. COMPENSATING FOR THE ONE-PERIOD DELAY IN THE COMPUTER VISION SYSTEM

The simulation results in the previous section differ from the linear expectations not only because of the nonideal inertial loops but also because of the delays introduced by the computer-vision system. The image tracker not only constrains the

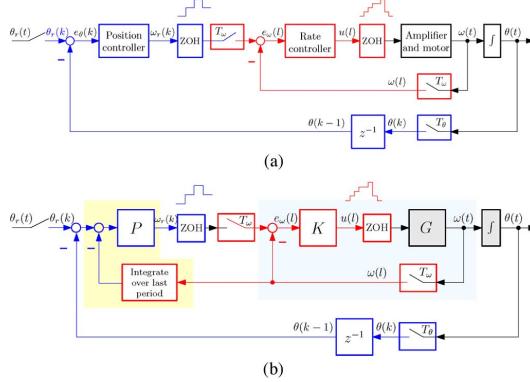


Fig. 13. (a) The inner (inertial rate) loop works at a fast sampling rate. The outer (image-based pointing) loop works at a slow sampling rate and suffers from a one-sampling-period delay. (b) Solution: integrate the rate ω over one slow sampling period, then subtract from the computed position error once the delayed data from the computer vision system arrive, and reset the integrator.

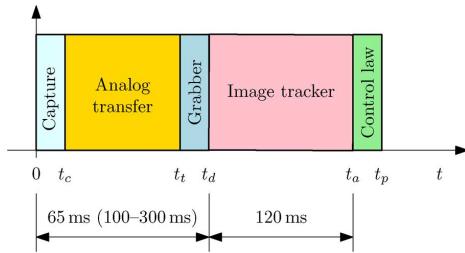


Fig. 14. Composition of the total delay in the visual pointing and tracking feedback loop: t_c is the time needed to capture the image (5–20 ms), which depends on the scene brightness; $t_t - t_c$ is the time for an analog transfer of the image (40 ms); $t_d - t_t$ is the time for digitization by a grabber. These three activities together can take about 65 ms when using the proprietary drivers provided with the camera, and 100–300 ms when using DirectX technology. The next tasks finishing at t_a is the actual visual tracking algorithm (120 ms on a common PC platform). The final task finishing at t_p is responsible for the pointing and tracking control law and consumes a negligible computational time. The sampling period is then given by the total delay. Alternatively, another image capturing can start right after t_d .

outer (pointing) loop to operate at a much slower sampling rate than the inner (inertial rate) loop but it also introduces one sampling period delay into the outer loop, see Fig. 13(a). This is a common problem encountered in visual servoing as discussed, for instance, in [29]. The cascade structure used in this particular system offers a simple solution in Fig. 13(b).

The composition of the total delay is visualized in Fig. 14. The sampling period is then given by this delay.

The simple solution consists in integrating the inertial angular rate ω as measured by the gyros over one slow sampling period and then subtracting from the computed position error once the delayed data from the vision system arrive. The discrete-integrator is then reset. This intuitive solution is investigated in [30] where the authors show that it is similar but not completely equivalent to a modified Smith predictor, which offers even better performance. The idea, being unrelated to the central topic, is not elaborated on further.

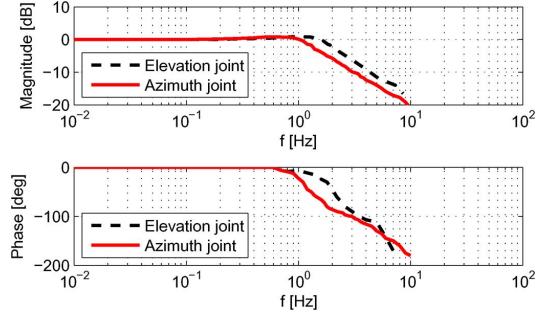


Fig. 15. Measured frequency responses for both inertial rate loops. The required inertial rate as an input and the true (measured) inertial rate as an output. The amplitude of the reference inertial rate was set up to $73^\circ/\text{s}$, which is 10× higher than in Fig. 16.

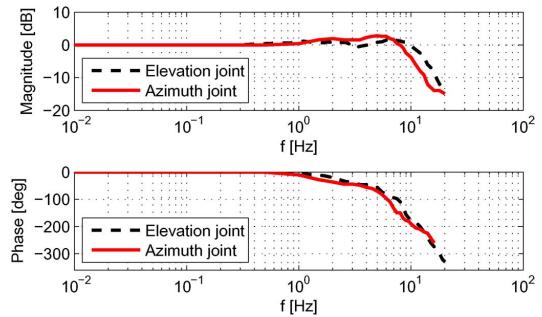


Fig. 16. Measured frequency responses for both inertial rate loops. The required inertial rate as an input and the true (measured) inertial rate as an output. The amplitude of the reference inertial rate was set up to $7.3^\circ/\text{s}$, which is 10× lower than in Fig. 15.

IX. LABORATORY EXPERIMENTS

The benchmark system introduced in the paper was used to validate the functionality of the proposed control scheme and compare its performance with the intuitive decoupled controller. The experimental test was conducted in an indoor lab while the camera platform was carried by a fixed laboratory stand. Therefore it was only the objects on the ceiling rather than on the floor that could be tracked conveniently. Both the new and the original (naive) decoupled controllers were tested only for the faster sampling rate of 15 Hz of the automatic image tracker. The controllers already included the simple heuristic delay compensation sketched above (although it was not much needed with this fast sampling rate).

The bandwidth of the inertial rate loop can be experimentally demonstrated to be at least 1 Hz, see Figs. 15 and 16. Therefore the assumption that the rate loop guarantees tracking of the required inertial rates up to a fraction of Hz is satisfied.

Three experiments were conducted and the measurements are visualized here in Figs. 17–19, the experimental data for the new algorithm always on the left and the data for the original decoupled scheme on the right.

The first two experiments were quite similar: the platform was in both cases sitting peacefully on the desk and the only differ-

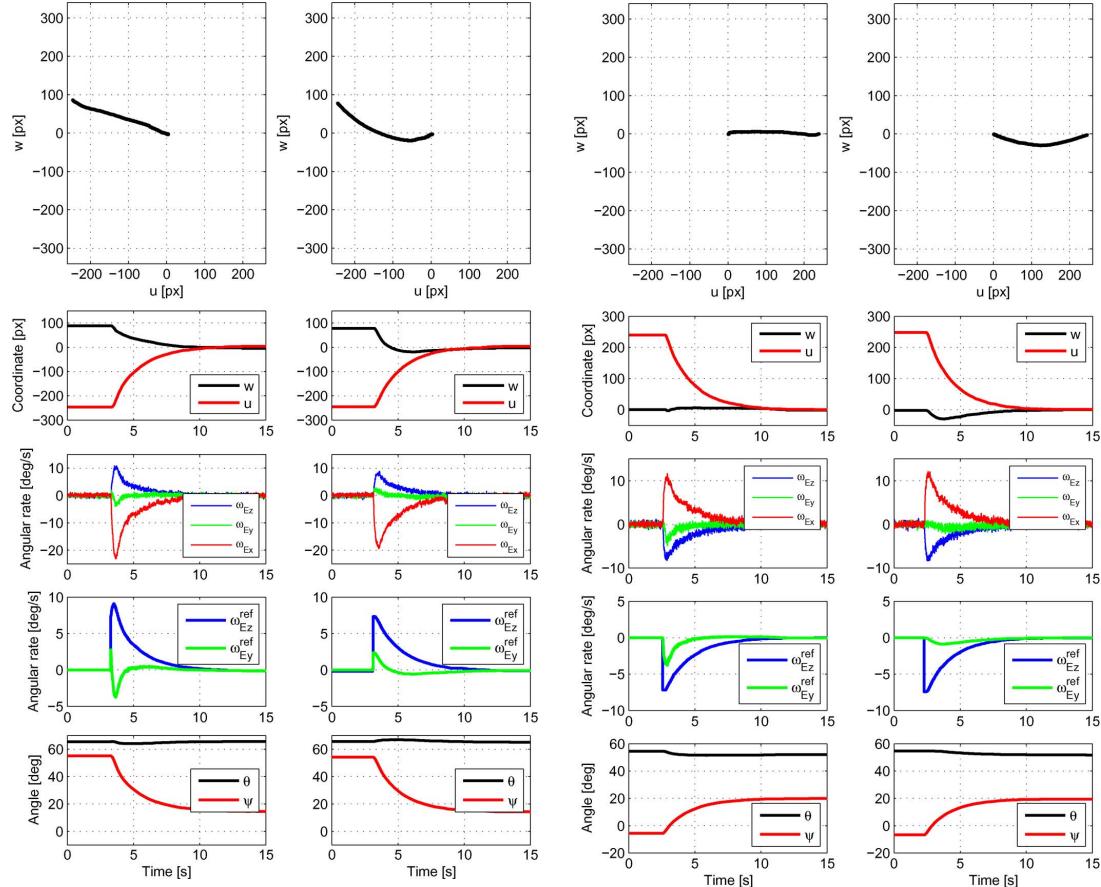


Fig. 17. Experiment 1: Responses of relevant quantities for $\theta(0) = 65.5^\circ$, tracker sampling rate $f_{sp} = 15$ Hz, $\alpha = 0.46$. Left: The proposed algorithm; right: the original decoupled approach.

ence was the initial elevation of the camera ($\theta(0) = 65.5^\circ$ and $\theta(0) = 54^\circ$ as in the simulations) and the position of the point to be tracked in the image plane. The measurements are in Figs. 17 and Fig. 18 and can be easily compared with the simulation results in Figs. 10 and Fig. 11. Apparently, the responses for both control methodologies are quite similar as for the time scale and control magnitude (but see the difference in one of the controller outputs ω_{Ey}^{ref} in Fig. 17). The key difference is that the new algorithm achieves linear trajectory in the image plane as desired.

The third experiment validates the pointing and tracking performance even in presence of a disturbing rotational motion of the carrier. Namely, the optical axis of the camera was initially pointing to the ceiling with the elevation $\theta(0) = 70^\circ$ and the laboratory stand was rotated manually around its vertical axis (orthogonal to the surface of the desk). The measured outcomes are in Fig. 19 both for the new algorithm and for the decoupled scheme, both of which use the same inner (inertial angular rate stabilization) feedback loop. The conclusion is that both algorithms exhibit the same characteristic behavior already known

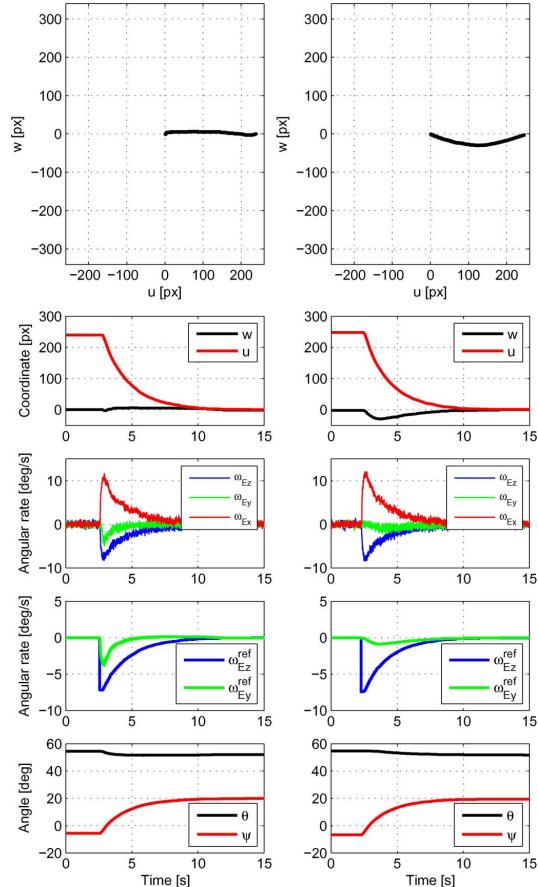


Fig. 18. Experiment 2: (left) The proposed algorithm; (right): the original decoupled approach. The initial elevation angle was set to $\theta(0) = 54^\circ$. The image tracker sampling rate $f_{sp} = 15$ Hz. The controller parameter $\alpha = 0.46$.

from the simulations and static experiments. Actually, the measurements related to the new controller appear a bit more disturbed but this is only due to the fact that the disturbing motion was induced manually, hence a bit differently in both cases.

X. CONCLUSION

This paper presented a systematic procedure for designing and implementing a pointing and tracking image-based controller for an airborne camera platform with an inertial line-of-sight stabilization already designed and implemented. The proposed scheme uses extra information from an inertial angular rate sensor; namely, the angular velocity of the payload (camera, laser) around its optical axis. This extra measurement is provided by a MEMS gyro at a much faster sampling rate than the pointing-tracking error produced by a computer vision system. Moreover, the proposed controller can take into consideration the measured or estimated translation velocity of the aircraft

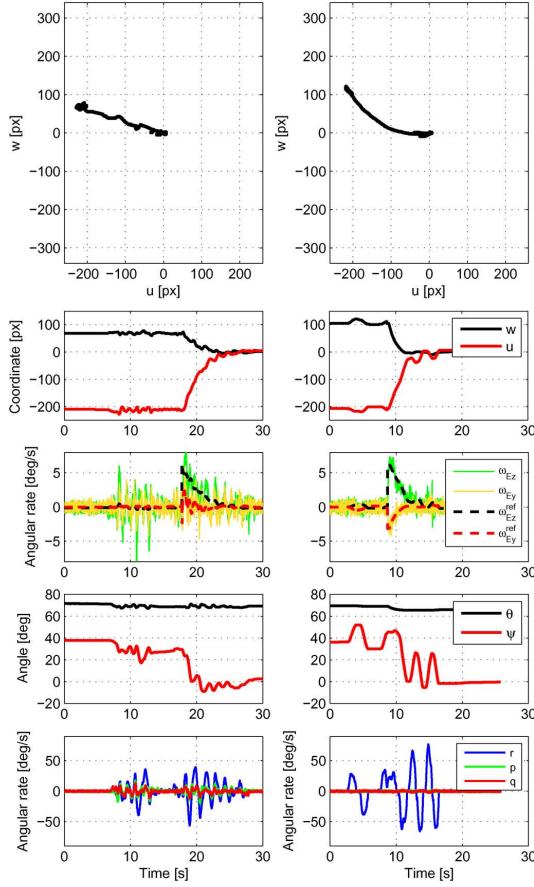


Fig. 19. Experiment 3: Responses of relevant quantities for the pointing exposed to external (recorded) disturbances p, q, r . $\theta(0) = 70^\circ$, tracker sampling rate $f_{\text{sp}} = 15$ Hz, $\alpha = 0.46$. Left: The proposed algorithm; right: the original decoupled approach.

with respect to the observed target (to compensate for the parallactic phenomenon).

The essence of the proposed design technique is that of a feed-back linearization. The resulting controller enforces linear dynamics in the image plane. This not only makes the analysis and design systematic but putting it on the well-explored ground, but also makes the response of the system a bit more friendly for a human operator as the system follows linear paths in the image plane during (re)pointing.

The proposed scheme was thoroughly simulated and verified by practical laboratory experiments with a realistic benchmark system and compared against the more intuitive decoupled control scheme. Possible simplifications were discussed and practical pitfalls were highlighted.

ACKNOWLEDGMENT

The authors acknowledge consulting with J. Hilkert (Alpha-Theta Technologies) and D. Kostić (TU Eindhoven). The bench-

mark platform was co-developed by J. Žohá (CTU FEE), M. Bartoš (ESSA Prague Ltd.), and J. Nohýl (VTÚLaPVO). The credit for the implementation of the image tracker algorithm goes to P. Krsek (CTU FEE).

REFERENCES

- [1] *Selected Papers on Precision Stabilization and Tracking Systems for Acquisition, Pointing and Control Applications*, ser. SPIE Milestone Series, M. Masten and L. Stockum, Eds.. Bellingham, WA: SPIE, 1996, vol. MS 123.
- [2] M. Masten, "Inertially stabilized platforms for optical imaging systems: Tracking dynamic targets with mobile sensors," *IEEE Control Syst. Mag.*, vol. 28, no. 2, pp. 47–64, Feb. 2008.
- [3] J. M. Hilkert, "Inertially stabilized platform technology: Concepts and principles," *IEEE Control Syst. Mag.*, vol. 28, no. 2, pp. 26–46, Feb. 2008.
- [4] J. Debruin, "Control systems for mobile satcom antennas," *IEEE Control Syst. Mag.*, vol. 28, no. 1, pp. 86–101, Feb. 2008.
- [5] J. Osborne, G. Hicks, and R. Fuentes, "Global analysis of the double-gimbal mechanism," *IEEE Control Syst. Mag.*, vol. 28, no. 4, pp. 44–64, Aug. 2008.
- [6] Z. Hurák and M. Řezáč, "Combined line-of-sight inertial stabilization and visual tracking: Application to an airborne camera platform," presented at the 48th IEEE Conf. Decision Control, Shanghai, China, 2009.
- [7] Z. Hurák and M. Řezáč, "Control design for image tracking with an inertially stabilized airborne camera platform," presented at the SPIE Defence, Security, Sensing, Orlando, FL, 2010.
- [8] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York: Wiley, 2006.
- [9] A. Rue, "Stabilization of precision electrooptical pointing and tracking systems," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-5, no. 5, pp. 805–819, Sep. 1969.
- [10] A. Rue, "Precision stabilization systems," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-10, no. 1, pp. 34–42, Jan. 1974.
- [11] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna, A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt, "Aerial video surveillance and exploitation," *Proc. IEEE*, vol. 89, no. 10, pp. 1518–1539, Oct. 2001.
- [12] J. Nygård, P. Skoglund, M. Ulvik, and T. Höglström, "Navigation aided image processing in UAV surveillance: Preliminary results and design of an airborne experimental system," *J. Robot. Syst.*, vol. 21, no. 2, pp. 63–72, Feb. 2004.
- [13] Q. Zhu, K. Cheng, and H. Zhang, "SSD tracking using dynamic template and log-polar transformation," in *Proc. ICME*, 2004, pp. 723–726.
- [14] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.
- [15] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, May 2003.
- [16] T. Kanade and C. Tomasi, "Detection and tracking of point features," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, 1991.
- [17] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vision*, 1999, p. 1150.
- [18] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image Vision Comput.*, vol. 22, no. 10, pp. 761–767, Sep. 2004.
- [19] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (MSER) tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Patt. Recog.*, 2006, vol. 1, pp. 553–560.
- [20] P. Viola and M. Jones, "Robust real-time object detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2002.
- [21] H. Grabner, J. Sochman, H. Bischof, and J. Matas, "Training sequential on-line boosting classifier for visual tracking," in *Proc. 19th Int. Conf. Patt. Recog. (ICPR)*, 2008, pp. 1–4.
- [22] F. Chaumette and S. Hutchinson, "Visual servo control. i. Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006.
- [23] S. Skogstad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 2nd ed. New York: Wiley, 2005.
- [24] G. F. Franklin, J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.

- [25] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1998, pp. 705–711.
- [26] E. Malis, F. Chaumette, and S. Boudev, "2-1/2 d visual servoing," *IEEE Trans. Robot. Autom.*, vol. 15, no. 2, pp. 238–250, Apr. 1999.
- [27] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 507–515, Aug. 2001.
- [28] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *Int. J. Comput. Vision*, vol. 56, no. 3, pp. 221–255, 2004.
- [29] M. Nemani, T. Tsao, and S. Hutchinson, "Multi-rate analysis and design of visual feedback digital servo-control system," *J. Dyn. Syst., Meas., Control*, vol. 116, no. 1, pp. 45–55, Mar. 1994.
- [30] Z. Hurák and M. Řezáč, "Delay compensation in a dual-rate cascade visual servomechanism," presented at the 49th Conf. Decision Control, Atlanta, GA, 2010.



Zdeněk Hurák (M'00) received the Ing. (M.Sc.) degree in aerospace electrical engineering (*summa cum laude*) from Military Academy, Brno, Czech Republic, in 1997. He was awarded Boeing Fellowship in 1999 to stay for three months at Iowa State University, Ames. He received the Ph.D. in cybernetics and robotics supervised by Prof. M. Šebek (thesis on ℓ_1 -optimal control) from the Czech Technical University (CTU), Prague, Czech Republic, in 2004.

He was a visiting researcher with TU Eindhoven, Eindhoven, The Netherlands, in 2008. He is currently an Assistant Professor with the Department of Control Engineering, Faculty of Electrical Engineering, CTU. His research interests include optimal and robust control and applications to electromechanical systems such as inertially stabilized camera platforms, piezoelectric micro-manipulators, and non-contact micro-manipulation using dielectrophoresis.

Prof. Hurák chairs Control Systems Chapter of Czech-Slovak Section of IEEE and is a member of the editorial board of *Kybernetika* journal and Czech industry-oriented *Automa* journal.



Martin Řezáč received the Ing. degree (M.Sc.) in cybernetics and measurement with a major in control engineering (*summa cum laude*) from the Czech Technical University, Prague, Czech Republic, in 2008. He was one of the two key developers of the control system for the benchmark platform and his diploma thesis was awarded Dean's prize. He is currently pursuing the Ph.D. degree in the domain of inertial stabilization and estimation in combination with visual servoing.

He spent six months with Eindhoven University of Technology, Eindhoven, The Netherlands, thanks to the Erasmus Socrates Program.

10

Visual Tracking

In this chapter we will introduce algorithms for target tracking and target following. Target tracking algorithms produce an estimate state of moving targets in the environment, in addition to their uncertainty. Target following algorithms maneuver the multirotor to follow one of the targets. The architecture discussed in this chapter is shown in Figure 10.1.

As shown in Figure 10.1, the KLT tracker produces a set of feature pairs. The feature pairs are used to estimate the homography transformation that maps features from the image at time $k - 1$ to the image at time k . The feature pairs are also used to estimate the essential matrix between frames. The homography and essential matrix are used to segment the feature pairs into pairs that move in the image due to the ego-motion of the multi-rotor, and pairs that are actually moving in the scene. The homography matrix is then used to transform all moving feature pairs, as well all target tracks to the current camera frame. Moving features are processed by the Recursive-RANSAC multiple target tracker to produce states (position, velocity, acceleration) and covariances of all moving tracks, no the image plane. The tracks are then processed by a track selection block to select the specific track to follow. The target follower commands a velocity and heading vector that forces the multirotor to follow the target. The velocity controller then controls the multirotor to follow that velocity.

Section 10.3 discusses the homography matrix, and how it can be estimated. Section 10.2 discusses the essential matrix estimation. The segmentation algorithm is discussed in Section ??, and feature propagation is discussed in Section 10.4. Section 10.5 describes the R-RANSAC tracking algorithm. Track selection is discussed in Section 10.6, and target following algorithms are described in Section 10.7.

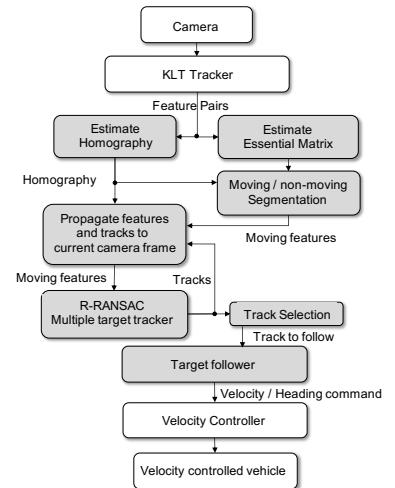


Figure 10.1: The architecture for target tracking and following.

10.1 The Euclidian homography matrix

This section gives a brief derivation of the Euclidian homography matrix between two camera poses. The relevant geometry is shown in Figure 10.2.

Suppose that \mathbf{p}_f is a feature point that lies on a plane defined by the normal (unit) vector \mathbf{n} . Let $\mathbf{p}_{f/a}^a$ and $\mathbf{p}_{f/b}^b$ be the position of \mathbf{p}_f relative to frames a and b , expressed in those frames respectively.

Then as shown in Figure 10.2 we have

$$\mathbf{p}_{f/b}^b = R_a^b \mathbf{p}_{f/a}^a + \mathbf{p}_{a/b}^b,$$

where R_a^b is the rotation matrix from frame a to frame b , and $\mathbf{p}_{a/b}^b$ is the position of frame a relative to frame b expressed in frame b .

Let d_a be the distance from the origin of frame a to the planar scene, and observe that

$$d_a = \mathbf{n}^\top \mathbf{p}_{f/a}^a \quad \Rightarrow \quad \frac{\mathbf{n}^\top \mathbf{p}_{f/a}^a}{d_a} = 1.$$

Therefore, we get that

$$\begin{aligned} \mathbf{p}_{f/b}^b &= R_a^b \mathbf{p}_{f/a}^a + \mathbf{p}_{a/b}^b \\ &= R_a^b \mathbf{p}_{f/a}^a + \mathbf{p}_{a/b}^b \left(\frac{\mathbf{n}^\top \mathbf{p}_{f/a}^a}{d_a} \right) \\ &= \left(R_a^b + \frac{\mathbf{p}_{a/b}^b \mathbf{n}^\top}{d_a} \right) \mathbf{p}_{f/a}^a. \end{aligned} \quad (10.1)$$

Let $\mathbf{p}_{f/a}^a = (p_{xa}, p_{ya}, p_{za})^\top$ and $\mathbf{p}_{f/b}^b = (p_{xb}, p_{yb}, p_{zb})^\top$, and let $\bar{\epsilon}^a = (p_{xa}/p_{za}, p_{ya}/p_{za}, 1)^\top$ represent the normalized homogeneous coordinates of $\mathbf{p}_{f/a}^a$ projected onto image plane a , and similarly for $\bar{\epsilon}^b$. Then Equation (10.1) can be written as

$$\frac{p_{zb}}{p_{za}} \bar{\epsilon}^b = \left(R_a^b + \frac{\mathbf{p}_{a/b}^b \mathbf{n}^\top}{d_a} \right) \bar{\epsilon}^a. \quad (10.2)$$

Defining the scalar $\gamma = p_{zb}/p_{za}$ we get

$$\gamma \bar{\epsilon}^b = \mathring{H}_a^b \bar{\epsilon}^a, \quad (10.3)$$

where

$$\mathring{H}_a^b \triangleq \left(R_a^b + \frac{\mathbf{p}_{a/b}^b \mathbf{n}^\top}{d_a} \right) \quad (10.4)$$

is called the Euclidian homography matrix between frames a and b ¹.

Equation (10.3) demonstrates that the Euclidian homography matrix \mathring{H}_a^b transforms the normalized homogeneous pixel location of \mathbf{p} in frame a into a homogeneous pixel location of \mathbf{p} in frame b . The scaling

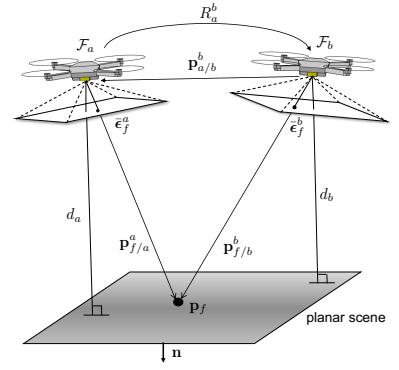


Figure 10.2: The geometry for the derivation of the homography matrix between two camera poses.

¹ M K Kaiser, N R Gans, and W E Dixon. Vision-based estimation for guidance, navigation, and control of an aerial vehicle. *IEEE Transactions on Aerospace and Electronic Systems*, 46(3):1064–1077, jul 2010

factor γ , which is feature point dependent, is required to put ϵ^b in normalized homogeneous coordinates, where the third element is unity.

If the camera is not calibrated, then the actual pixel location is given by

$$\bar{\mathbf{m}} = K_c \bar{\epsilon}$$

where K_c is the camera calibration matrix ². In the uncalibrated case the homography equation becomes

$$\gamma \bar{\mathbf{m}}^b = H_a^b \bar{\mathbf{m}}^a, \quad (10.5)$$

where

$$H_a^b = K_c \mathring{H}_a^b K_c^{-1}$$

is called the homography matrix.

From Equation (10.3), we see that since R_a^b has three degrees of freedom, \mathbf{n} has two degrees of freedom, and $\mathbf{p}_{a/b}^b / d_a$ has three degrees of freedom, that $H_a^b \in \mathbb{R}^{3 \times 3}$ has eight degree of freedom. The extra degree of freedom is absorbed in the scale factor γ . Therefore, the homography matrix can only be determined up to a scale factor. With eight degrees of freedom, the most common algorithms for determining H from image data require at least four point correspondences ³. Since point correspondence is an inherently noisy process, where the noise may not be structured (i.e., the noise contains mismatches that are not zero mean Gaussian), a RANSAC process is typically used to determine the homography from several hundred potential point correspondences.

10.1.1 Computation of the homography matrix

In this section we will describe how the homography matrix can be computed from four matching features. Let $(\bar{\mathbf{m}}^a, \bar{\mathbf{m}}^b)$ be a matching pair of pixels from frames a and b respectively. Then the features are related through the homography matrix as

$$\gamma_m \bar{\mathbf{m}}^b = H_a^b \bar{\mathbf{m}}^a.$$

Note that $H_a^b \in \mathbb{R}^{3 \times 3}$ implies that there are nine elements, or nine degrees of freedom in H_a^b . However, note that if H_a^b is scaled by a non-zero scalar λ that

$$(\lambda \gamma_m) \bar{\mathbf{m}}^b = (\lambda H_a^b) \bar{\mathbf{m}}^a,$$

and therefore that (λH_a^b) represents the same homography relationship between pixels. Therefore H_a^b is only unique up to a scale factor, implying that there are only eight degrees of freedom in H_a^b .

Taking the cross product of both sides with $\bar{\mathbf{m}}^b$ gives

$$\left[\bar{\mathbf{m}}^b \right]_{\times} H_a^b \bar{\mathbf{m}}^a = 0. \quad (10.6)$$

² Yi Ma, Stefano Soatto, Jan Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003

³ Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003

Let $\bar{\mathbf{m}}^b = (m_{xb}, m_{yb}, 1)^\top$, and let

$$H_a^b = \begin{pmatrix} \mathbf{h}_1^\top \\ \mathbf{h}_2^\top \\ \mathbf{h}_3^\top \end{pmatrix},$$

then Equation (10.6) can be written as

$$\begin{pmatrix} 0_{1 \times 3}, & -m_{zb}\bar{\mathbf{m}}^{a\top}, & m_{yb}\bar{\mathbf{m}}^{a\top} \\ m_{zb}\bar{\mathbf{m}}^{a\top}, & 0_{1 \times 3}, & -m_{xb}\bar{\mathbf{m}}^{a\top} \\ -m_{yb}\bar{\mathbf{m}}^{a\top}, & m_{xb}\bar{\mathbf{m}}^{a\top} & 0_{1 \times 3} \end{pmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0,$$

where the matrix is rank two. Taking the first two rows gives

$$\begin{pmatrix} 0_{1 \times 3}, & -m_{zb}\bar{\mathbf{m}}^{a\top}, & m_{yb}\bar{\mathbf{m}}^{a\top} \\ m_{zb}\bar{\mathbf{m}}^{a\top}, & 0_{1 \times 3}, & -m_{xb}\bar{\mathbf{m}}^{a\top} \end{pmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0,$$

which represents two equations in nine unknowns. We can obtain eight equations in nine unknowns using four matching pairs. The resulting equation will be of the form

$$A\mathbf{h} = 0, \quad (10.7)$$

where $A \in \mathbf{R}^{8 \times 9}$ and $\mathbf{h} \in \mathbf{R}^9$ are the stacked rows of H_a^b . Therefore \mathbf{h} is in the null space of A . Since the null space is a vector space $\lambda\mathbf{h}$ is also in the null space of A , and solving Equation (10.7) results in H_a^b that is unique up to a scale factor. Equation (10.7) can be solved by finding the SVD of A as

$$U \begin{bmatrix} \Sigma & \mathbf{0} \end{bmatrix} \begin{bmatrix} V_1^\top \\ \mathbf{v}_2^\top \end{bmatrix} = \text{svd}(A),$$

and setting $\mathbf{h} = \mathbf{v}_2$.

RWB: add details to the following paragraph Given the homography matrix, the relative pose R_a^b , the unit vector \mathbf{n} , and the vector $\mathbf{t}_{a/b}^b = \mathbf{p}_{a/b}^b/d_a$ can be reconstructed. Details are found in (Malis & Varga)⁴, and are implemented in the openCV function `decomposeHomographyMat`.

10.1.2 Group Velocity

RWB: This doesn't fit. The objective of this section is to derive the kinematic evolution equations for the homography matrix given the linear and angular velocity vectors of the camera. The discussion in this section closely follows⁵. We will show that the homography evolves according to the equation

$$\dot{H} = HA,$$

The homography can be calculated using the openCV function `findHomography`. The `findHomography` function scales the elements of H so that the (3,3) element is equal to one. Given the homography matrix H_a^b , the Euclidian homography can be found (up to a scale factor) as $\lambda\hat{H}_a^b = K_c^{-1}H_a^bK_c$. The scale factor λ is usually selected so that $\det(\hat{H}_a^b) = 1$, implying that $\hat{H}_a^b \in SL(3)$, the special linear group.

⁴ Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Technical Report RR-6303, INRIA, <https://hal.inria.fr/inria-00174036v3>, sep 2007

⁵

where $H \in SL(3)$ is the homography matrix $A \in \mathfrak{sl}(3)$ is called the group velocity. The objective of this section is to relate the group velocity A to the linear velocity v and the angular velocity ω of the platform. If $p \in \mathbf{R}^3$ is the position in inertial coordinates, and R is the rotation matrix from the inertial to the body frame, we assume that the position and rotation of the platform evolve according to

$$\begin{aligned}\dot{p}^i &= R_{i/b}v^b \\ \dot{R}_{i/b} &= R_{i/b}(\omega_{b/i}^b)^\times,\end{aligned}$$

where $\omega_{b/i}^b = (p, q, r)^\top$ and

$$\omega^\times = \begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix}.$$

Using Equation (10.4), the time derivative of the Euclidian homography matrix is given by

$$\begin{aligned}\dot{H} &= \gamma \left(\dot{R} + \frac{h(\dot{\mathbf{d}}\mathbf{n}^\top + \mathbf{d}\dot{\mathbf{n}}^\top - \mathbf{d}\mathbf{n}^\top \dot{h})}{h^2} \right) + \dot{\gamma}(R + \frac{\mathbf{d}\mathbf{n}^\top}{h}) \\ &= \gamma \left(R\Omega^\times + \frac{\dot{\mathbf{d}}\mathbf{n}^\top + \mathbf{d}\dot{\mathbf{n}}^\top}{h} - \frac{\mathbf{d}\mathbf{n}^\top}{h} \frac{\dot{h}}{h} \right) + \frac{\dot{\gamma}}{\gamma} H.\end{aligned}$$

Using the fact that $\dot{\mathbf{d}} = R\mathbf{v}$, that $\dot{h} = -\mathbf{n}^\top \mathbf{v}$, and $\dot{\mathbf{n}} = (\Omega^\times)^\top \mathbf{n}$ RWB: Need to explain these relationships, we have

$$\begin{aligned}\dot{H} &= \gamma \left(R\Omega^\times + \frac{R\mathbf{v}\mathbf{n}^\top + \mathbf{d}\mathbf{n}^\top \Omega^\times}{h} + \frac{\mathbf{d}\mathbf{n}^\top}{h} \frac{\mathbf{n}^\top \mathbf{v}}{h} \right) + \frac{\dot{\gamma}}{\gamma} H \\ &= \gamma \left(\left(R + \frac{\mathbf{d}\mathbf{n}^\top}{h} \right) \Omega^\times + \left(R + \frac{\mathbf{d}\mathbf{n}^\top}{h} \right) \frac{\mathbf{v}\mathbf{n}^\top}{h} \right) + \frac{\dot{\gamma}}{\gamma} H \\ &= H \left(\Omega^\times + \frac{\mathbf{v}\mathbf{n}^\top}{h} + \frac{\dot{\gamma}}{\gamma} I \right).\end{aligned}$$

Therefore

$$A = \Omega^\times + \frac{\mathbf{v}\mathbf{n}^\top}{h} + \frac{\dot{\gamma}}{\gamma} I.$$

However, we also know that since $H \in SL(3)$, that $A \in \mathfrak{sl}(3)$, which implies that $\text{tr}(A) = 0$. Since

$$\text{tr} \left(\Omega^\times + \frac{\mathbf{v}\mathbf{n}^\top}{h} + \frac{\dot{\gamma}}{\gamma} I \right) = \frac{\mathbf{n}^\top \mathbf{v}}{h} + 3 \frac{\dot{\gamma}}{\gamma},$$

we have that

$$\frac{\dot{\gamma}}{\gamma} = -\frac{1}{3} \frac{\mathbf{n}^\top \mathbf{v}}{h},$$

which gives

$$A = \Omega^\times + \frac{\mathbf{v}\mathbf{n}^\top}{h} - \frac{1}{3} \frac{\mathbf{n}^\top \mathbf{v}}{h} I.$$

10.1.3 Homography Filter

RWB: This doesn't fit. A complementary filter for homographies is proposed in ⁶. The derivation of the filter is not included in the paper. The purpose of this note is to include a derivation.

The adjoint operator is defined in ⁷ as

$$Ad_H X = H X H^{-1},$$

where $H \in SL(3)$ and $X \in \mathfrak{sl}(3)$.

For $H \in SL(3)$, the projection onto $\mathfrak{sl}(3)$ is given by

$$\mathbb{P}(H) = \left(H - \frac{\text{tr}(H)}{3} I \right)$$

The natural evolution of the homography matrix is given by

$$\dot{H} = HA.$$

Define the estimate of the homography by \hat{H} and let the homography error be $\tilde{H} = \hat{H}^{-1}H$. The the complementary filter suggested in ⁸ is given by

$$\dot{\hat{H}} = \hat{H} Ad_{\tilde{H}} \left(A - k \mathbb{P}(\tilde{H}^\top (I - \tilde{H})) \right). \quad (10.8)$$

The filter can be derived by recalling that if A is invertible then

$$\frac{d}{dt} A^{-1} = -A^{-1} \dot{A} A^{-1}.$$

Then

$$\begin{aligned} \dot{\hat{H}} &= \dot{H} \tilde{H}^{-1} + H \frac{d}{dt} \tilde{H}^{-1} \\ &= HA \tilde{H}^{-1} - H \tilde{H}^{-1} \dot{\tilde{H}} \tilde{H}^{-1} \\ &= \hat{H} \tilde{H} A \tilde{H}^{-1} - \hat{H} \tilde{H} \tilde{H}^{-1} \dot{\tilde{H}} \tilde{H}^{-1} \\ &= \hat{H} \tilde{H} \left(A - \tilde{H}^{-1} \dot{\tilde{H}} \right) \tilde{H}^{-1} \\ &= \hat{H} Ad_{\tilde{H}} \left(A - \tilde{H}^{-1} \dot{\tilde{H}} \right). \end{aligned}$$

The idea is to approximate $\tilde{H}^{-1} \dot{\tilde{H}}$ in a way that results in filter convergence. We note that since $\tilde{H} \in SL(3)$ that

$$\dot{\tilde{H}} = \tilde{H} \tilde{A},$$

where $\tilde{A} \in \mathfrak{sl}(3)$. Therefore, in approximating $\tilde{H}^{-1} \dot{\tilde{H}}$, the projection operator in Equation (10.8) ensures that the approximation is in $\mathfrak{sl}(3)$. Therefore, a first guess at a homography filter might be

$$\dot{\hat{H}} = \hat{H} Ad_{\tilde{H}} (A - k \mathbb{P}(X)).$$

Now, as outlined in ⁹, choose the Lyapunov function

⁶ Robert Mahony, Tarek Hamel, Pascal Morin, and Ezio Malis. Nonlinear complementary filters on special linear group. *International Journal of Control*, 85(10):1557–1573, oct 2012b

⁷ Robert Mahony, Tarek Hamel, Pascal Morin, and Ezio Malis. Nonlinear complementary filters on special linear group. *International Journal of Control*, 85(10):1557–1573, oct 2012b

⁸ Robert Mahony, Tarek Hamel, Pascal Morin, and Ezio Malis. Nonlinear complementary filters on special linear group. *International Journal of Control*, 85(10):1557–1573, oct 2012b

$$L = \frac{1}{2} \|\tilde{H} - I\|_F^2 = \frac{1}{2} \text{tr} \left((\tilde{H} - I)^\top (\tilde{H} - I) \right).$$

Differentiating we have

$$\dot{L} = \text{tr} \left((\tilde{H} - I)^\top \dot{\tilde{H}} \right),$$

where

$$\begin{aligned} \dot{\tilde{H}} &= \frac{d}{dt} (\hat{H}^{-1} H) \\ &= \hat{H}^{-1} \dot{H} + \frac{d}{dt} (\hat{H}^{-1}) H \\ &= \hat{H}^{-1} H A - \hat{H}^{-1} \dot{\hat{H}} \hat{H}^{-1} H \\ &= \tilde{H} A - \hat{H}^{-1} (\hat{H} A d_{\hat{H}} (A - k \mathbb{P}(X))) \tilde{H} \\ &= \tilde{H} A - \tilde{H} (A - k \mathbb{P}(X)) \tilde{H}^{-1} \tilde{H} \\ &= -k \tilde{H} \mathbb{P}(X). \end{aligned}$$

Therefore

$$\dot{L} = -k \text{tr} \left((\tilde{H} - I)^\top \tilde{H} \mathbb{P}(X) \right),$$

which motivates the choice of

$$X = \tilde{H}^\top (\tilde{H} - I)$$

to give

$$\dot{L} = -k \text{tr} \left((\tilde{H} - I)^\top \tilde{H} \mathbb{P}(\tilde{H}^\top (\tilde{H} - I)) \right),$$

which can be shown to be negative unless $\tilde{H} = I$.

Note that when using the filter (??), \tilde{H} is computed using the measurement of H , which we will denote as H_m . Therefore, the homography filter is

$$\dot{H} = H_m \left(A - k \mathbb{P}(H_m^\top \hat{H}^{-\top} (I - \hat{H}^{-1} H_m)) \right) H_m^{-1} \hat{H}. \quad (10.9)$$

10.2 Relative pose estimation

RWB: Just pull out what we need from this paper.

An Iterative Pose Estimation Algorithm based on Epipolar Geometry with Application to Multi-Target Tracking

Jacob H. White, Randal W. Beard

Abstract—This paper introduces a new algorithm for estimating the relative pose of a moving camera using consecutive frames of a video sequence. State-of-the-art algorithms for calculating the relative pose between two images use matching features to estimate the essential matrix. The essential matrix is then decomposed into the relative rotation and normalized translation between frames. To be robust to noise and feature match outliers, these methods generate a large number of essential matrix hypotheses from randomly selected minimal subsets of feature pairs, and then score these hypotheses on all feature pairs.

Alternatively, the algorithm introduced in this paper calculates relative pose hypotheses by directly optimizing the rotation and normalized translation between frames, rather than calculating the essential matrix and then performing the decomposition. The resulting algorithm improves computation time by an order of magnitude. If an IMU is available, it is used to seed the optimizer, and in addition, we reuse the best hypothesis at each iteration to seed the optimizer thereby reducing the number of relative pose hypotheses that must be generated and scored. These advantages greatly speed up performance and enable the algorithm to run in real-time on low cost embedded hardware. We show application of our algorithm to visual multi-target tracking (MTT) in the presence of parallax and demonstrate its real-time performance on a 640×480 video sequence captured on a UAV. Video results are available at <https://youtu.be/HhK-p2hXNnU>.

Index Terms—Multi-target tracking, epipolar geometry, pose estimation, unmanned aircraft systems, aerial robotics, vision-based flight.

I. INTRODUCTION

ESTIMATING camera motion from a video sequence has many applications in robotics including target tracking, visual odometry, and 3D scene reconstruction. These applications often require on-board processing of the video sequence in real-time and thereby impose size, weight, and power (SWAP) constraints on the computing platform.

One method to estimate motion from a video sequence is to calculate the essential matrix between consecutive frames. The essential matrix relates the homogeneous image coordinates between frames using the epipolar constraint. After the essential matrix has been determined, it can be decomposed into a rotation and a normalized translation to determine the relative motion of the camera between frames. In order to be robust to noise and feature mismatches, the essential matrix is typically estimated by generating a large number

of hypotheses from five-point minimum subsets of matching features, and selecting the best hypothesis using either Random Sample Consensus (RANSAC) [1] or Least Median of Squares (LMedS) [2]. When using RANSAC, the hypotheses are scored by counting the number of inlier points from the entire set. When using LMedS, the hypotheses are scored by calculating the median error.

State of the art methods calculate essential matrix hypotheses directly from each five-point minimum subset. One of the best known methods is Nister's algorithm [3]. Nister showed that for five matching points, there are potentially ten essential matrices that satisfy the constraints, each corresponding to a real root of a tenth-order polynomial generated from the data. There are many open-source implementation of Nister's five-point algorithm including OpenCV's `findEssentialMat` function [4]. However, constructing, solving, and extracting the essential matrix from this tenth-order polynomial is complex and can be computationally expensive. Furthermore, since each minimum subset produces up to ten hypotheses, it can be time consuming to score them.

As an alternative to directly calculating essential matrix solutions, some authors [5], [6], [7], [8], [9] propose solving for the essential matrix using non-linear optimization algorithms such as Gauss-Newton (GN) and Levenberg-Marquardt (LM). Since the essential matrix has nine entries but only five degrees of freedom, the optimization is performed on the five dimensional essential matrix manifold. There are a number of ways to define the essential matrix manifold. Some authors define the manifold using a rotation and translation unit vector, which are elements of $SO(3)$ and S^2 respectively [5], [6]. Others define the manifold using two elements of $SO(3)$ [8], [9].

A third method of optimizing on a manifold is described in [7]. This approach called LM Basis calculates the four essential matrix basis vectors in the nullspace of the essential matrix equation using SVD. The four coefficients to these matrices are solved for on the S^3 manifold. In contrast to the previously described methods which operate on five-dimensional manifolds, this method uses a three-dimensional manifold.

During each iteration of the optimization algorithm, the optimizer step is solved for in terms of the three or five degrees of freedom along the manifold. The computational requirements of the resulting scheme are significantly less than Nister's five point algorithm. However, one weakness of optimization-based solvers is that they only find one of

*This work has been funded by the Center for Unmanned Aircraft Systems (C-UAS), a National Science Foundation Industry/University Cooperative Research Center (I/UCRC) under NSF award Numbers IIP-1161036 and CNS-1650547, along with significant contributions from C-UAS industry members.

the ten possible essential matrices at a time. Finding all solutions requires additional optimization runs with different initialization points. The optimization method is also sensitive to initial conditions, which can cause the optimizer to fail to produce a valid solution. For example, GN may diverge if the initial guess is too far from the true solution. LM can be used to prevent increases in the cost function, but still may fail to converge. Because of the need to run the optimizer multiple times from different initial conditions, these existing optimization-based solvers might not necessarily be faster than the direct essential matrix solvers if the same level of accuracy is desired. However, not all of the ten possible solutions are needed in order to achieve comparable accuracy to direct essential matrix solvers if the best solution can be found the first time.

After the essential matrix is found, it must then be decomposed into a rotation and normalized translation. Given an essential matrix, there are four possible rotation-translation pairs [10]. The correct rotation-translation pair is typically determined using the Cheirality check that ensures that matching features are in front of both cameras. However, the Cheirality check is sensitive to noise in the image and frequently returns the wrong decomposition.

The main contribution in this paper is a novel optimization-based algorithm that directly solves for the relative pose using the epipolar constraint in the cost function. If an IMU is available, then it is used to seed the optimization algorithm at the next time step. When an IMU is not available, since the rotation and translation between consecutive video frames is similar to nearby frames, we use the relative pose estimate from the previous time step to initialize the optimization at the current time step. At each iteration, we use the current best hypothesis to seed the LM algorithm.

We show that this approach significantly reduces the number of hypotheses that must be generated and scored to estimate the pose, thus allowing real-time execution of the algorithm on a Jetson TX2 processor.

The remainder of the paper is organized as follows. The problem is formally stated in Section II. The new pose estimation algorithm is developed in Section III. Application of the algorithm to target tracking in the presence of parallax is described in Section IV. Simulation and flight results on a quadrotor UAV are presented in Section V, and conclusions are given in section VI.

II. PROBLEM DESCRIPTION

The problem geometry is shown in Fig. 1 where a UAV captures images at time instants k_1 and k_2 . The camera frame at time k_i is denoted \mathcal{F}^{k_i} . Between time instances, the UAV rotates by $R_{k_1}^{k_2} \in SO(3)$, where $R_{k_1}^{k_2}$ transforms coordinates in \mathcal{F}^{k_1} into coordinates in \mathcal{F}^{k_2} , and is translated by $\xi_{k_1/k_2}^{k_2} = \xi_{k_1/I}^{k_2} - \xi_{k_2/I}^{k_2} \in \mathbb{R}^3$, which denotes the position of \mathcal{F}^{k_1} relative to \mathcal{F}^{k_2} expressed in \mathcal{F}^{k_2} , and where $\xi_{k/I}$ is the position of frame \mathcal{F}^k in the inertial frame. Superscripts on vectors are used to denote the coordinate frame in which the vector is expressed.

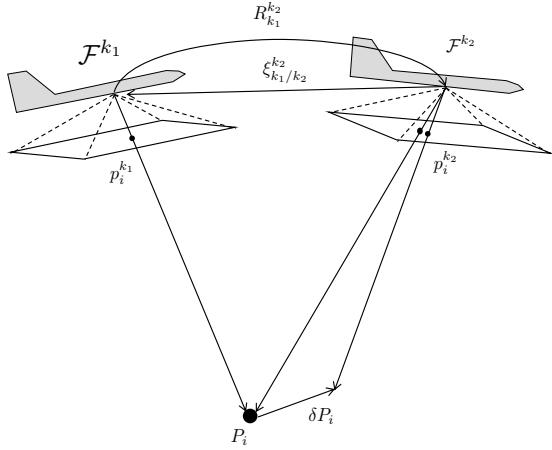


Fig. 1. The geometry for the derivation of the essential matrix

Suppose that the UAV observes the point P_i in both frames, and let $P_i^{k_1} = (X_i^{k_1} \ Y_i^{k_1} \ Z_i^{k_1})^\top$ and $P_i^{k_2} = (X_i^{k_2} \ Y_i^{k_2} \ Z_i^{k_2})^\top$ be the 3D position of point i with respect to camera frames \mathcal{F}^{k_1} and \mathcal{F}^{k_2} , then from the geometry in Fig. 1 we have

$$P_i^{k_2} = R_{k_1}^{k_2} P_i^{k_1} + \xi_{k_1/k_2}^{k_2}. \quad (1)$$

Left multiplying each side of the equation by $(\xi_{k_1/k_2}^{k_2})_\times$, where

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_\times \triangleq \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}, \quad (2)$$

gives

$$(\xi_{k_1/k_2}^{k_2})_\times P_i^{k_2} = (\xi_{k_1/k_2}^{k_2})_\times R_{k_1}^{k_2} P_i^{k_1} + (\xi_{k_1/k_2}^{k_2})_\times \xi_{k_1/k_2}^{k_2}. \quad (3)$$

The last term on the right is zero, thus

$$(\xi_{k_1/k_2}^{k_2})_\times P_i^{k_2} = (\xi_{k_1/k_2}^{k_2})_\times R_{k_1}^{k_2} P_i^{k_1}. \quad (4)$$

Left-multiply each side of Equation (4) by $(P_i^{k_2})^\top$ give

$$(P_i^{k_2})^\top (\xi_{k_1/k_2}^{k_2})_\times P_i^{k_2} = (P_i^{k_2})^\top (\xi_{k_1/k_2}^{k_2})_\times R_{k_1}^{k_2} P_i^{k_1}. \quad (5)$$

However, the left side of Equation (5) is always zero because the cross product $(\xi_{k_1/k_2}^{k_2})_\times P_i^{k_2}$ gives a vector perpendicular to $P_i^{k_2}$, and therefore

$$(P_i^{k_2})^\top (\xi_{k_1/k_2}^{k_2})_\times R_{k_1}^{k_2} P_i^{k_1} = 0. \quad (6)$$

Let

$$p_i \triangleq g(P_i) \triangleq \frac{P_i}{e_3^\top P_i},$$

where $e_3 = (0, 0, 1)^\top$ be the normalized homogeneous image coordinates of P_i , then $p_i^{k_1}$ and $p_i^{k_2}$ are the normalized

homogeneous image coordinates of point i in camera frame \mathcal{F}^{k_1} and \mathcal{F}^{k_2} respectively. Since the right side of Equation (6) is zero, any scalar multiple of $P_i^{k_1}$ and $P_i^{k_2}$ will also satisfy the equation, which gives the so-called epipolar constraint

$$\left(p_i^{k_2} \right)^T \left(\xi_{k_1/k_2}^{k_2} \right)_x R_{k_1}^{k_2} p_i^{k_1} = 0,$$

that relates homogeneous coordinates of matching features in two images. Since the epipolar constraint holds for any scaling of $\xi_{k_1/k_2}^{k_2}$, we define $q = \frac{\xi}{\|\xi\|}$ to get

$$\left(p_i^{k_2} \right)^T \left(q_{k_1/k_2}^{k_2} \right)_x R_{k_1}^{k_2} p_i^{k_1} = 0. \quad (7)$$

The essential matrix is defined as

$$E(R_{k_1}^{k_2}, q_{k_1/k_2}^{k_2}) \triangleq \left(q_{k_1/k_2}^{k_2} \right)_x R_{k_1}^{k_2}. \quad (8)$$

Since $q_{k_1/k_2}^{k_2} \in S^2$ and $R_{k_1}^{k_2} \in SO(3)$, there are five degrees of freedom associated with E .

The key idea behind our pose estimation algorithm is to optimize the rotation and translation on the manifold $SO(3) \times S^2$ subject to the constraint of Equation (7). The essential matrix is only constructed when it is needed to update the rotation and translation. In contrast, many approaches in the literature solve for the essential matrix directly, which requires decomposing the essential matrix later to obtain the desired rotation and translation.

In the remainder of the paper we drop the frame notation and simply write $R \in SO(3)$ and $q \in S^2$ for the rotation and normalized translation, except where needed for clarity.

III. ITERATIVE POSE ESTIMATION ALGORITHM

The new pose estimation algorithm proposed in this paper is shown schematically in Fig. 2. At each time step, features

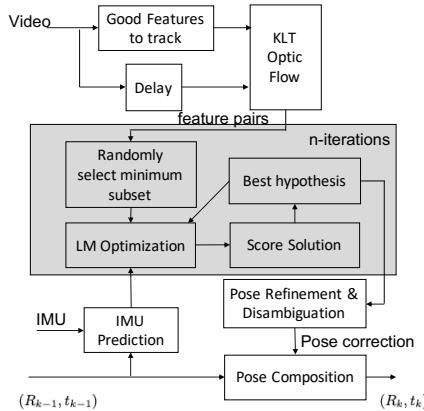


Fig. 2. Block diagram of the proposed recursive pose estimation algorithm.

are detected in one frame using, for example, OpenCV's `goodFeaturesToTrack` [11] and then matched to the next frame using, for example, OpenCV's LK optical flow [12]. As the camera moves, `goodFeaturesToTrack` is only used on regions of the image that do not currently

contain features. The corresponding feature pairs are input to the pose estimation algorithm.

As shown in Fig. 2, given the set of matching features between images, at each iteration of the optimization loop, a set of N-matching features are randomly selected and the Levenberg-Marquardt (LM) algorithm is used to minimize the Sampson error, as explained in Section III-A. The solution of the LM Optimization algorithm is then scored using either Least Median Squares (LMedS) or an inlier count similar to the Random Sample Consensus (RANSAC) algorithm, as described in Section III-B. If IMU measurements are available, they are used to predict the relative pose between frames, as explained in Section III-C, and are used to seed the LM optimization algorithm. In addition, the algorithm retains the current best hypothesis and uses the best hypothesis to seed the next LM optimization, as also explained in Section III-C. After n-iterations the best hypothesis is then refined using all inlier data, and the rotation and translation are disambiguated, as explained in Section III-D. Finally, the relative pose between frames is composed with the estimated pose at the prior time step to produce the new estimated pose, as explained in Section III-E.

A. Levenberg-Marquardt (LM) Optimization

Since the LM optimization block is the core element of the pose estimation algorithm, we explain it first in this section. The Levenberg-Marquardt (LM) algorithm is a standard technique for solving nonlinear least squares problems of the form

$$\beta^* = \arg \min r(z, \beta)^T r(z, \beta),$$

where $z \in Z$ is a known data vector, $\beta \in B$ is a parameter vector, and $r : Z \times B \rightarrow \mathbb{R}^N$ is a residual vector to be minimized. The optimization problem is solved by iteratively incrementing β by a small δ in a direction that decreases the squared residual

$$S(\beta) = r(x, \beta)^T r(z, \beta).$$

Using the approximation

$$r(z, \beta + \delta) \approx r(z, \beta) + J(z, \beta)\delta,$$

where

$$J(z, \beta) \triangleq \frac{\partial r}{\partial \beta}(z, \beta)$$

is the Jacobian of r with respect to the parameters β , we get that

$$S(\beta + \delta) \approx [r(z, \beta) - J(z, \beta)\delta]^T [r(z, \beta) - J(z, \beta)\delta].$$

Taking the partial of $S(\beta + \delta)$ with respect to δ and setting equal to zero gives

$$J^T r = J^T J \delta. \quad (9)$$

Solving for δ from this equation would result in the Gauss-Newton optimization method. In contrast, gradient descent would replace the right hand side of Equation (9) by $\lambda \delta$, where

λ is a scalar. The LM algorithm is a hybrid between Gauss Newton and gradient descent, requiring δ to satisfy

$$J^\top r = (J^\top J + \lambda I)\delta. \quad (10)$$

The general LM optimization algorithm is therefore given by the iteration

$$\beta_{\ell+1} = \beta_\ell + \delta_\ell, \quad (11)$$

where

$$\delta_\ell = [J(z, \beta_\ell)^\top J(z, \beta_\ell) + \lambda_\ell I]^{-1} J(z, \beta_\ell)^\top r(z, \beta_\ell),$$

where λ_ℓ is selected at each iteration to ensure that the residual $r(z, \beta)$ decreases.

The remainder of this section explains how the LM algorithm is adapted to the problem of finding relative pose between two camera frames. Let $\mathcal{M}_i^k = (p_i^{k-1}, p_i^k)$ be defined as the i^{th} matching feature pair at time k , and supposing that there are M_k matching features at time k , let $\mathcal{I}_k = \{1, 2, \dots, M_k\}$ denote the index set. Then $\mathcal{M}_{\mathcal{I}_k}$ will denote the set of all matching feature pairs at time k . If $\mathcal{J} \subset \mathcal{I}_k$, then $\mathcal{M}_{\mathcal{J}} \subset \mathcal{M}_{\mathcal{I}_k}$ denotes a subset of matching feature pairs.

The residual function will be defined in terms of the well-known Sampson error, which scales the epipolar error by the reprojection error in each image [13]. If E^k is the essential matrix at time k , then the associated residual based on the Sampson's error is defined by

$$r_i(\mathcal{M}_i^k, E_k) = \frac{p_i^{k\top} E^k p_i^{k-1}}{\sqrt{\|\Pi_{e_3} E^k p_i^{k-1}\|^2 + \|\Pi_{e_3} E^{k\top} p_i^k\|^2}}, \quad (12)$$

where $\Pi_x = I - xx^\top$ and $e_3 = (0, 0, 1)^\top$. The residual associated with the set $\mathcal{M}_{\mathcal{J}}$ is a vector of length $|\mathcal{J}|$ constructed by stacking the residual for each matching pair in $\mathcal{M}_{\mathcal{J}}$, and will be denoted as $r(\mathcal{M}_{\mathcal{J}}, E_k)$. Recall from Equation (8) that the essential matrix can be written as

$$E(\tilde{R}, \tilde{q}) = \tilde{q}_\times \tilde{R},$$

where $\tilde{R} \in SO(3)$ and $\tilde{q} \in S^2$, and the 'tilde' indicates that we are seeking a correction over one time step. Therefore, there are five degrees of freedom in E . To make the math more transparent, we over-parameterize $\tilde{q} \in S^2$ by using an element of $SO(3)$. Accordingly, given $\tilde{q} \in S^2$ we define the matrix \tilde{Q} such that the last column of \tilde{Q}^\top is \tilde{q} , the first column of \tilde{Q}^\top is selected as any unit vector that is orthogonal to \tilde{q} and the second column of \tilde{Q}^\top is selected so that \tilde{Q} is orthogonal. Then $\tilde{q} = \tilde{Q}^\top e_3$, and

$$E(\tilde{R}, \tilde{Q}) = [\tilde{Q}^\top e_3]_\times \tilde{R} = \tilde{Q}^\top e_3 \times \tilde{Q} \tilde{R}.$$

Therefore, the residual will be denoted as $r(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q})$, where a subset of matching pairs $\mathcal{M}_{\mathcal{J}}$ plays the role of the data vector z in the previous discussion, and \tilde{R} and \tilde{Q} play the role of β .

To form the Jacobian $J(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q})$, we need to take the partial derivative of the residual $r(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q})$ with respect to the three degrees of freedom in \tilde{R} and the two degrees of

freedom in \tilde{Q} . Toward that end, and to simplify notation, we define the boxplus operator [14] as

$$\begin{aligned} \boxplus : SO(3) \times \mathbb{R}^3 &\rightarrow SO(3) \\ S \boxplus \delta &= \exp(\delta_\times)S, \end{aligned}$$

where

$$\exp(\delta_\times) = I + \sin(\|\delta\|) \frac{\delta_\times}{\|\delta\|} + (1 - \cos(\|\delta\|)) \frac{\delta_\times^2}{\|\delta\|^2}.$$

If $\delta = \theta \hat{\omega}$ where $\hat{\omega}$ is unit length, then

$$\frac{\partial \tilde{R} \boxplus \delta R}{\partial \theta} = \frac{\partial}{\partial \theta} \exp(\theta \hat{\omega}_\times) \tilde{R} \Big|_{\theta=0} \quad (13)$$

$$= \hat{\omega}_\times \exp(\theta \hat{\omega}_\times) \tilde{R} \Big|_{\theta=0} \quad (14)$$

$$= \hat{\omega}_\times R. \quad (15)$$

Let $\delta_R = (\delta_{R,1}, \delta_{R,2}, \delta_{R,3})^\top$ represent the (local) three degrees of freedom in \tilde{R} , then the partial derivatives of $\tilde{R} \in SO(3)$ along each of the three degrees of freedom are

$$\frac{\partial \tilde{R} \boxplus \delta_R}{\partial \delta_{R,j}} = (e_j)_\times \tilde{R}, \quad j = 1, 2, 3,$$

where $e_1 = (1, 0, 0)^\top$, $e_2 = (0, 1, 0)^\top$, and $e_3 = (0, 0, 1)^\top$.

The normalized translation vector is represented as $\tilde{q} = \tilde{Q}^\top e_3 \in S^2$ where $\tilde{Q} \in SO(3)$ only has two degrees of freedom. Let $\delta_Q = (\delta_{Q,1}, \delta_{Q,2})^\top$ represent the (local) two degrees of freedom in \tilde{Q} , and let

$$B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix},$$

then the partial derivatives of \tilde{Q} along each of the two degrees of freedom are

$$\frac{\partial \tilde{Q} \boxplus B \delta_Q}{\partial \delta_{Q,j}} = [e_j]_\times \tilde{Q}, \quad j = 1, 2.$$

Defining $\delta = (\delta_1, \delta_2, \delta_3, \delta_4, \delta_5)^\top = (\delta_R^\top, \delta_Q^\top)^\top$, then the Jacobian $J(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q}) \in \mathbb{R}^{|\mathcal{J}| \times 5}$ is given by

$$J(\mathcal{M}_{\mathcal{J}}, \tilde{R}, \tilde{Q}) = \begin{pmatrix} \frac{\partial r_1}{\partial \delta_1} & \frac{\partial r_1}{\partial \delta_2} & \frac{\partial r_1}{\partial \delta_3} & \frac{\partial r_1}{\partial \delta_4} & \frac{\partial r_1}{\partial \delta_5} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_1} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_2} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_3} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_4} & \frac{\partial r_{|\mathcal{J}|}}{\partial \delta_5} \end{pmatrix}, \quad (16)$$

where

$$\frac{\partial r_i}{\partial \delta_j} = \frac{\sqrt{s_i} (p_i^k)^\top \frac{\partial E}{\partial \delta_j} p_i^{k-1} - (p_i^k)^\top E p_i^{k-1} \frac{1}{\sqrt{s_i}} \frac{\partial s_i}{\partial \delta_j}}{s_i}, \quad (17)$$

with

$$s_i = \|\Pi_{e_3} E p_i^{k-1}\|^2 + \|\Pi_{e_3} E^\top p_i^k\|^2, \quad (18)$$

where

$$\begin{aligned} \frac{\partial s_i}{\partial \delta_j} &= p_i^{k-1\top} \frac{\partial E^\top}{\partial \delta_j} \Pi_{e_3} E p_i^{k-1} + p_i^{k-1\top} E^\top \Pi_{e_3} \frac{\partial E}{\partial \delta_j} p_i^{k-1} \\ &\quad + p_i^{k\top} \frac{\partial E}{\partial \delta_j} \Pi_{e_3} E^\top p_i^k + p_i^{k\top} E \Pi_{e_3} \frac{\partial E^\top}{\partial \delta_j} p_i^k, \end{aligned}$$

and where $\frac{\partial E}{\partial \delta_j}$ are given by

$$\begin{aligned}\frac{\partial E}{\partial \delta_1}(\tilde{R}, \tilde{Q}) &= \tilde{Q}^\top e_{3 \times} \tilde{Q} e_{1 \times} \tilde{R} \\ \frac{\partial E}{\partial \delta_2}(\tilde{R}, \tilde{Q}) &= \tilde{Q}^\top e_{3 \times} \tilde{Q} e_{2 \times} \tilde{R} \\ \frac{\partial E}{\partial \delta_3}(\tilde{R}, \tilde{Q}) &= \tilde{Q}^\top e_{3 \times} \tilde{Q} e_{3 \times} \tilde{R} \\ \frac{\partial E}{\partial \delta_4}(\tilde{R}, \tilde{Q}) &= \tilde{Q}^\top e_{2 \times} \tilde{Q} \tilde{R} \\ \frac{\partial E}{\partial \delta_5}(\tilde{R}, \tilde{Q}) &= -\tilde{Q}^\top e_{1 \times} \tilde{Q} \tilde{R}.\end{aligned}$$

In applying the Levenberg-Marquardt algorithm, we begin with $\lambda = 10^{-4}$ and then adaptively modify λ as follows. If the residual error grows after an LM step, then the update is rejected and λ is doubled. On the other hand, if the error is decreased, then we accept the step and divide λ by two. This ensures that the residual monotonically decreases, while allowing the LM algorithm to converge faster if the residual function is well-behaved. The Levenberg-Marquart optimization algorithm for \tilde{R} and \tilde{Q} is summarized in Algorithm 1.

Algorithm 1 LM Optimization for Pose Refinement

```

1: procedure LM OPTIMIZATION( $\mathcal{M}_{\mathcal{J}}, \tilde{R}_0, \tilde{q}_0$ )
2:   Select  $\tilde{Q}_0 \in SO(3)$  such that  $\tilde{q}_0 = \tilde{Q}_0^\top e_3$ .
3:    $\ell \leftarrow 0, \lambda \leftarrow 10^{-4}$ .
4:   repeat
5:     repeat
6:       Compute Jacobian  $J_\ell = J(\mathcal{M}_{\mathcal{J}}, \tilde{R}_\ell, \tilde{Q}_\ell)$ 
7:         from Equation (16).
8:       Compute  $\delta_\ell \in \mathbb{R}^5$  as
9:        $\delta_\ell \leftarrow (J_\ell^\top J_\ell + \lambda I)^{-1} J_\ell^\top r_\ell$ 
10:      Update  $\tilde{R}$  as  $\tilde{R}_{\ell+1} \leftarrow \tilde{R}_\ell \boxplus \delta_{\ell,1:3}$ 
11:      Update  $\tilde{Q}$  as  $\tilde{Q}_{\ell+1} \leftarrow \tilde{Q}_\ell \boxplus \delta_{\ell,4:5}$ 
12:       $\lambda \leftarrow 2\lambda$ 
13:      until  $\|r_\ell\| < \|r_{\ell+1}\|$ 
14:       $\ell \leftarrow \ell + 1, \lambda \leftarrow \lambda/2$ 
15:    until  $\|r_{\ell+1}\| < \epsilon$ 
16: end procedure
```

B. Outlier Rejection

The optic flow algorithm shown in Fig. 2 produces M_k feature pairs, but it is well known that the process is not robust, and that some of the feature pairs will not actually correspond to the same physical location in the environment. Therefore, the LM optimization algorithm given in Algorithm 1 must be made robust to false feature matches. The most common methods reported in the literature are Random Sample Consensus (RANSAC) [1] and Least Median of Squares (LMedS) [2]. In this paper we will compare the efficacy of these two methods.

First note that since the essential matrix manifold $SO(3) \times S^2$ contains five degrees of freedom, and each feature correspondence gives one constraint, that only $N = 5$ feature points are needed to generate a pose using the LM optimization in Algorithm 1, assuming that the feature pairs are true correspondences, and that the 3D points are not co-linear [10].

To make the process robust to outliers, the basic idea is to solve the LM optimization many times on randomly selected subsets of $|\mathcal{J}| = 5$ feature pairs, to score each pose hypothesis using all of the feature pairs, and then to select the pose with the best score.

Recall that the set of matching feature pairs at time k is given by $\mathcal{M}_{\mathcal{I}_k}$. The basic idea of the RANSAC and LMedS algorithms is to randomly select a minimum subset $\mathcal{M}_{\mathcal{J}} \subset \mathcal{M}_{\mathcal{I}_k}$ where $|\mathcal{J}| = 5 \ll |\mathcal{I}_k|$. The output of Algorithm 1 then forms a model hypothesis associated with $\mathcal{M}_{\mathcal{J}}$. The model hypothesis is then scored against all feature matches in $\mathcal{M}_{\mathcal{I}_k}$. This process is repeated n times, always retaining the model hypothesis with the best score.

In this paper we consider the following two scoring functions:

$$S_{\text{RANSAC}}(\mathcal{M}_{\mathcal{I}_k}, \tilde{R}, \tilde{Q}) \triangleq \sum_{\mathcal{M}_i^k \in \mathcal{M}_{\mathcal{I}_k}} \mathbb{I}(\|r(\mathcal{M}_i^k, \tilde{R}, \tilde{Q})\| > \tau) \quad (19)$$

$$S_{\text{LMedS}}(\mathcal{M}_{\mathcal{I}_k}, \tilde{R}, \tilde{Q}) \triangleq \text{median}_{\mathcal{M}_i^k \in \mathcal{M}_{\mathcal{I}_k}} \{r^2(\mathcal{M}_i^k, \tilde{R}, \tilde{Q})\}, \quad (20)$$

where $\mathbb{I}(\text{boolean})$ is the indicator function (1 when boolean is true, and 0 when boolean is false), and where τ is the RANSAC threshold that must be selected. Note that for RANSAC, the score is the total number of outliers with a residual greater than τ , whereas for LMedS, the score is the square of the median residual over all matching features.

To determine the number of iterations required for RANSAC or LMedS, following [1] we assume that the event that a matching feature pair is an outlier is a binomial distribution with

ϵ = Probability that \mathcal{M}_i^k is a false matching pair.

If N is the number of matching points used to generate a model hypothesis, then the probability that all point correspondences used to generate the model are inliers is

$$p_m = (1 - \epsilon)^N. \quad (21)$$

If n model hypotheses are generated, then the probability that at least one of models is generated using only outliers is

$$p = 1 - (1 - (1 - \epsilon)^N)^n. \quad (22)$$

Solving for the number of hypotheses needed to achieve a desired confidence level p , gives

$$n(p) = \frac{\log(1 - p)}{\log(1 - (1 - \epsilon)^N)}. \quad (23)$$

For example, achieving a 99% confidence ratio when the outlier ratio is 50% and when minimal subsets of $N = 5$ matching features are used to create model hypotheses requires $n = 145$ RANSAC or LMedS iterations.

C. IMU Predication and LM Seeding

In this section, we address the issue of initializing the LM optimization given in Algorithm 1. We are particularly interested in the robotic situation where an IMU, synchronized to the camera, is available to resolve the scale ambiguity. We will also discuss the case where IMU measurements are not available. The discussion in this section follows in some respects, the development in [15].

For the sake of clarity, in this section we will again explicitly specify the different coordinate frames. Let $R_k^I \in SO(3)$ be the rotation from the camera frame at time k , \mathcal{F}^k to the inertial frame \mathcal{F}^I , and let $\xi_{k/I}^k$ and $v_{k/I}^k$ be the position and velocity of the camera at time k relative to the inertial frame, expressed in the camera frame \mathcal{F}^k , let $a_{k/I}^k$ be the measured specific acceleration of the camera expressed in \mathcal{F}_k , and $\omega_{k/I}^k$ be the measured angular velocity of the camera relative to the inertial frame, as expressed in the camera frame \mathcal{F}^k , where we have assumed that the IMU biases are known and have been removed from the measurements. Then the kinematics for the camera are given by

$$\begin{aligned}\dot{R}_k^I &= R_k^I \left(\omega_{k/I}^k \right)_\times \\ \dot{v}_{k/I}^k &= R_k^{I\top} g^I + a_{k/I}^k \\ \dot{\xi}_{k/I}^k &= v_{k/I}^k\end{aligned}\quad (24)$$

where g^I is the gravity vector in the inertial frame. Let T_s be the sample period of the IMU, and assume that there are m IMU samples between camera frames. We will use the notation $\kappa_0, \kappa_1, \kappa_2, \dots, \kappa_m$ to denote the intermediate sample from $k-1$ to k , implying that the time instance κ_0 corresponds to the time at image frame $k-1$, and κ_m corresponds to the time at image frame k . Then, integrating over one sample of the IMU and assuming that the measurements are constant over the sample period, we get

$$\begin{aligned}R_{\kappa_{i+1}}^I &= R_{\kappa_i}^I \exp \left((\omega_{\kappa_i/I}^k) \times T_s \right) \\ v_{\kappa_{i+1}/I}^{\kappa_{i+1}} &= R_{\kappa_i}^{\kappa_{i+1}} v_{\kappa_i/I}^{\kappa_i} + R_{\kappa_{i+1}}^{I\top} g^I T_s + R_{\kappa_i}^{\kappa_{i+1}} a_{\kappa_i/I}^{\kappa_i} T_s \\ \xi_{\kappa_{i+1}/I}^{\kappa_{i+1}} &= R_{\kappa_i}^{\kappa_{i+1}} \xi_{\kappa_i/I}^{\kappa_i} + v_{\kappa_{i+1}/I}^{\kappa_{i+1}},\end{aligned}$$

where

$$R_{\kappa_i}^{\kappa_{i+1}} = R_{\kappa_{i+1}}^{I\top} R_{\kappa_i}^I = \exp \left((\omega_{\kappa_i/I}^k) \times T_s \right)^\top.$$

The predicted pose after m IMU samples is therefore

$$\tilde{R} = R_{\kappa_0}^{\kappa_m} = R_{\kappa_m}^{I\top} R_{\kappa_0}^I \quad (25)$$

$$\tilde{q} = \frac{R_{\kappa_0}^{\kappa_m} \xi_{\kappa_0/I}^{\kappa_0} - \xi_{\kappa_m/I}^{\kappa_m}}{\| R_{\kappa_0}^{\kappa_m} \xi_{\kappa_0/I}^{\kappa_0} - \xi_{\kappa_m/I}^{\kappa_m} \|}. \quad (26)$$

The predicted relative pose (\tilde{R}, \tilde{q}) is used to initialize Algorithm 1.

When an IMU is not available, Algorithm 1 can be seeded with the identity rotation $\tilde{R} = I$ and a randomly selected translation unit vector \tilde{q} . This method we will call the “random initialization” method. Another alternative is to initialize Algorithm 1 with the best hypothesis from the previous time step. We will denote this method as the “prior” method. A third

alternative, that can also be used with or without the IMU, is to initialize Algorithm 1 with the best RANSAC/LMedS hypothesis that has been found so far from previous LM iterations. Since each hypothesis depends on the previous best hypothesis, when the first hypothesis is selected randomly, we will denote this method as the “random recursive” method. When the first hypothesis is the best hypothesis from the IMU, or the prior time step in the case of no IMU, we call it the “prior recursive” method. In Section V we will show results using each of these initialization techniques.

D. Pose Refinement and Disambiguation

Even the best hypothesis from RANSAC and LMedS usually has some error due to noise on the feature matches in the minimum subset used to create the hypothesis. The estimate can be improved by using least-squares optimization over all inlier feature matches. An advantage of our method over traditional five point or eight point algorithms is that the size of the feature set $\mathcal{M}_{\mathcal{J}}$ in Algorithm 1 is not limited to a fixed number of points. Therefore, the relative pose can be refined by instantiating Algorithm 1 on the entire set of inlier feature matches.

To determine inlier points, we use the robust inlier detection method described in [16] by setting

$$\hat{\sigma} = 1.4826 \left[1 + \frac{5}{|\mathcal{I}_k| - N} \right] \sqrt{\text{median}\{r(\mathcal{M}_i^k, \tilde{R}, \tilde{q})^2\}},$$

where $N = 5$ is the number of feature matches used to create the model and $|\mathcal{I}_k|$ is the total number of feature matches at time k . A feature match \mathcal{M}_i^k is determined to be an inlier if the Sampson residual $r(\mathcal{M}_i^k, \tilde{R}, \tilde{q})^2 < 2.5\hat{\sigma}$.

It is well known that if the epipolar constraint (7) is satisfied for (\tilde{R}, \tilde{q}) , that it is also satisfied for (\tilde{R}', \tilde{q}) , $(\tilde{R}, -\tilde{q})$, and $(\tilde{R}', -\tilde{q})$, where \tilde{R}' is a 180 degree rotation about \tilde{q} i.e.,

$$\tilde{R}' = (I + 2\tilde{q}_x^2) \tilde{R}.$$

Algorithm 1 will find one of these four solutions, but the result may not correspond to the correct pose, particularly in the case when the IMU is not present and the initial translation is selected randomly. This is a well-known problem with 5-point and 8-point solvers of the essential matrix. The correct pose is typically determined using the Cheirality check, which involves triangulating each feature match to determine its 3D position, and then selecting the relative pose that puts all 3D points in front of the camera. However, the Cheirality check often gives spurious results. As an alternative, since the two possible rotations \tilde{R} and \tilde{R}' are always 180 degrees apart, we can pick the rotation with the smallest angle and use the Cheirality check to find the correct translation. Since

$$\text{tr}(\tilde{R}) = 1 + 2 \cos \theta,$$

where θ is the eigen-angle, and since we are expecting small camera deviations between time samples, we select \tilde{R} or \tilde{R}' based on which one has the largest trace. The translation \tilde{q} or $-\tilde{q}$ is then selected based on the Cheirality check.

E. Pose Composition

The global pose at time $k - 1$ is given by $(R_{k-1}^I, \xi_{k-1/I}^I)$. The current estimated incremental pose is $(\tilde{R}, \tilde{q}) = (R_{k-1}^k, q_{k-1/k}^k)$. We use the IMU prediction to scale q which gives

$$\xi_{k-1/k}^k = \left\| R_{k_0}^{\kappa_m} \xi_{k_0/I}^{\kappa_0} / \xi_{\kappa_m/I}^{\kappa_m} \right\| \tilde{q}. \quad (27)$$

Accordingly, the pose at time k is

$$R_k^I = R_{k-1}^I R_{k-1}^{k\top} \quad (28)$$

$$\xi_{k/I}^I = \xi_{k-1/I}^I - R_{k-1}^I R_{k-1}^{k\top} \xi_{k-1/k}^k. \quad (29)$$

F. Algorithm Summary

A summary of the proposed iterative pose estimation scheme is give in Algorithm 2 below.

Algorithm 2 Iterative Pose Estimation

```

1: procedure ITERATIVE POSE ESTIMATION
2:   Input:  $(I_{k-1}, I_k), (a_{\kappa_i/I}^{\kappa_i}, \omega_{\kappa_i/I}^{\kappa_i})_{i=1}^m, (R_{k-1}^I, \xi_{k-1/I}^I)$ 
3:   From images  $I_{k-1}$  and  $I_k$  determine set
4:   of matching features  $\mathcal{M}_{\mathcal{I}_k}$ .
5:   Initialize relative pose  $(\tilde{R}_0, \tilde{q}_0)$  using either:
6:     (1) Initialization:  $\tilde{R}_0 = I, \tilde{q}_0 = \text{random}$ .
7:     (2) Prior: best hypothesis from previous step.
8:     (3) IMU: initialize using Equations (25)-(26).
9:   Determine  $n(p)$  from Equation (23).
10:  for  $i = 1$  to  $n$  do
11:    Randomly generate  $\mathcal{J}_i \subset \mathcal{I}_k$  where  $|\mathcal{J}_i| = 5$ ,
12:    with corresponding features  $\mathcal{M}_{\mathcal{J}_i}$ .
13:    Let  $(\tilde{R}_i, \tilde{q}_i) \leftarrow \text{LM Optimization}(\mathcal{M}_{\mathcal{J}_i}, \tilde{R}_0, \tilde{q}_0)$ 
14:    using Algorithm 1.
15:    Score  $(\tilde{R}_i, \tilde{q}_i)$  using Equations (19) or (20).
16:    Let  $(\tilde{R}^*, \tilde{q}^*) \leftarrow \arg \max_i S(\mathcal{M}_{\mathcal{I}_k}, \tilde{R}_i, \tilde{q}_i)$  be the
17:    pose correction with the current lowest score.
18:  end for
19:  Using  $(\mathcal{I}_k, \tilde{R}^*, \tilde{q}^*)$ , determine inliers and refine the
20:  pose with those inliers, following Section III-D.
21:  Update the pose according to Equations (27)–(29).
22: end procedure

```

IV. MOTION DETECTION AND TRACKING IN THE PRESENCE OF PARALLAX

One application of relative pose estimation is motion detection and tracking in the presence of parallax. Motion detection is a valuable source of information in target tracking applications. It can be used to track objects without any prior knowledge about their appearance, in contrast to many trackers that are designed to track specific classes of objects.

There are many successful image-based background subtraction techniques in the literature that work on stationary cameras. In order for image differencing techniques to work on a moving camera, the images must first be aligned using a homography. While this works well for planar scenes, if there is parallax, artifacts will appear in the difference image. If the parallax is small enough in comparison to the movement of objects in the scene, the effects of parallax can be

reduced using simple morphological operations and gradient suppression [17].

In the presence of strong parallax, however, a better motion model that accounts for depth variation must be used. There are several methods in the literature that use a geometric model to describe the motion of tracked points in the scene over time. For example, [18] uses orthographic projections to segment moving objects, which works well if the camera is far from the scene or has a narrow field of view. Another approach maintains an affinity matrix and uses principal component analysis to segment moving objects [19]. Another approach uses multiple-frame geometric constraints [20], [21]. However, all of these methods can be computationally prohibitive. In contrast, the technique proposed in this paper exploits the two-frame epipolar constraint and is therefore computationally simple, enabling real-time performance.

A. Motion Detection Algorithm

Given two consecutive frames, with point correspondences detected in each frame, the objective is to determine which points are from stationary objects and which are from moving objects. In this section we assume that the relative pose between cameras (\tilde{R}, \tilde{q}) has been calculated using Algorithm 2. The goal is to design a detector $\phi(\mathcal{M}_i^k, \tilde{R}, \tilde{q})$ which returns 1 if the feature pair \mathcal{M}_i^k is sourced from a moving object and 0 if it sourced from a stationary background object. The output of the motion detector is used as an input to a tracking algorithm that produces target tracks as described in Section IV-B.

The essential matrix relates points in one image to the other image with the epipolar constraint. In other words, the essential matrix maps a point in one image to a line in the other image. The location where the point in the other image appears along this line depends on the feature's depth to the camera. As the camera translates, points that are closer to the camera will appear to move more than the points that are far away. This effect is known as parallax.

There are two degrees of freedom for the apparent motion of each point in the image plane. One of these degrees of freedom can be explained by the epipolar constraint if the real-world point is stationary. However, motion along this degree of freedom can also be explained by object motion in the world frame. Hence the source of any movement along this degree of freedom is ambiguous without additional information. The second degree of freedom for apparent motion of points in the image plane is perpendicular to the epipolar constraint. Thus the only possible source of motion along this degree of freedom is movement in the real-world frame.

Let $g : \mathbb{R}^3 \rightarrow \mathbb{P}^2$ defined by $g(P) = \frac{P}{e_3^T P}$ be the prospective projection operator. With reference to Fig. 1 let $P_i \in \mathbb{R}^3$ be a feature in the world that is imaged by the camera at time k_1 to produce feature $p_i^{k_1} = g(P_i^{k_1})$. At time time k_2 , the feature has moved in the world to position $P_i + \delta P$, and the camera, which has moved by $(\tilde{R}, \tilde{\xi})$ images the feature to produce

$$p_i^{k_2} = g(P_i^{k_2}) = g(\tilde{R}P_i^{k_1} + \tilde{\xi} + \delta P).$$

To streamline the notation, we will drop the i subscript in the following discussion. The epipolar line in frame \mathcal{F}^{k_2} corresponding to the point p^{k_1} is given by

$$\ell = Ep^{k_1} = \tilde{q}_x \tilde{R}p^{k_1} = \frac{\tilde{\xi}_x \tilde{R}P^{k_1}}{\|\tilde{\xi}\| e_3^\top P^{k_1}},$$

where $e_3^\top P^{k_1}$ is the distance that P^{k_1} is from the camera in frame \mathcal{F}^{k_1} and is therefore positive. Note also that the perpendicular to the epipolar line in \mathcal{F}^{k_2} is given by

$$e_3 \times \ell = e_3 \times Ep^{k_1} = \frac{e_{3x} \tilde{\xi}_x \tilde{R}P^{k_1}}{\|\tilde{\xi}\| e_3^\top P^{k_1}},$$

since e_3 in the camera frame is directed along the optical axis. Our proposed motion detection algorithm is based on the following theorem.

Theorem 4.1: Let P be a point in the world frame and δP its displacement between times k_1 and k_2 . Suppose that the camera moves by $(\tilde{R}, \tilde{\xi})$ between k_1 and k_2 and that $\tilde{\xi} \neq 0$, and let $p^{k_1} = g(P^{k_1})$ and $p^{k_2} = g(P^{k_2}) = g(\tilde{R}P^{k_1} + \tilde{\xi} + \delta P)$ be the image projection of the point P in frame \mathcal{F}^{k_1} and $P + \delta P$ in \mathcal{F}^{k_2} , respectively. Let $\ell = \tilde{q}_x \tilde{R}p^{k_1}$ be the epipolar line in \mathcal{F}^{k_2} corresponding to point p^{k_1} in \mathcal{F}^{k_1} , where $\tilde{q} = \tilde{\xi}/\|\tilde{\xi}\|$, and define

$$\begin{aligned} v_{\parallel} &= (p^{k_2} - g(\tilde{R}p^{k_1}))^\top (e_3 \times \ell) \\ v_{\perp} &= (p^{k_2} - g(\tilde{R}p^{k_1}))^\top \ell, \end{aligned}$$

which can be interpreted as the rotation corrected displacement of the projected position of P in \mathcal{F}^{k_2} projected along the perpendicular to the epipolar line, and the epipolar line, respectively. Then $v_{\perp} \neq 0$ implies that $\delta P \neq 0$, and $v_{\parallel} < 0$ implies that $\delta P \neq 0$.

Proof: Note that

$$\begin{aligned} p^{k_2} - g(\tilde{R}p^{k_1}) &= g(\tilde{R}P^{k_1} + \tilde{\xi} + \delta P) - g(\tilde{R}p^{k_1}) \\ &= \frac{\tilde{R}P^{k_1} + \tilde{\xi} + \delta P}{e_3^\top \tilde{R}P^{k_1} + e_3^\top \tilde{\xi} + e_3^\top \delta P} - \frac{\tilde{R}P^{k_1}}{e_3^\top \tilde{R}P^{k_1}} \\ &= \frac{(e_3^\top \tilde{R}P^{k_1})\tilde{\xi} - (e_3^\top \tilde{\xi})\tilde{R}P^{k_1} + (e_3^\top \tilde{R}P^{k_1})\delta P - (e_3^\top \delta P)\tilde{R}P^{k_1}}{(e_3^\top \tilde{R}P^{k_1} + e_3^\top \tilde{\xi} + e_3^\top \delta P)e_3^\top \tilde{R}P^{k_1}}. \end{aligned} \quad (30)$$

Since $\tilde{\xi}^\top \tilde{\xi} \times \tilde{R}P^{k_1} = (\tilde{R}P^{k_1})^\top \tilde{\xi} \times \tilde{R}P^{k_1} = 0$ we have that

$$v_{\perp} = \frac{(e_3^\top \tilde{R}P^{k_1})\delta P^\top \tilde{\xi} \times \tilde{R}P^{k_1}}{(e_3^\top P^{k_2})(e_3^\top \tilde{R}P^{k_1})(e_3^\top P^{k_1}) \|\tilde{\xi}\|}.$$

Clearly $v_{\perp} \neq 0$ implies that $\delta P \neq 0$.

For v_{\parallel} , note that Equation (30) can be written as

$$p^{k_2} - g(\tilde{R}p^{k_1}) = \frac{e_{3x} \tilde{\xi}_x \tilde{R}P^{k_1} + e_3 \times \delta P \tilde{R}P^{k_1}}{(e_3^\top P^{k_2})(e_3^\top \tilde{R}P^{k_1})}.$$

Therefore

$$v_{\parallel} = \frac{\left\| e_{3x} \tilde{\xi}_x \tilde{R}P^{k_1} \right\|^2 + (e_3 \times \delta P \tilde{R}P^{k_1})^\top (e_{3x} \tilde{\xi}_x \tilde{R}P^{k_1})}{(e_3^\top P^{k_2})(e_3^\top \tilde{R}P^{k_1})(e_3^\top P^{k_1}) \|\tilde{\xi}\|^2}.$$

Since the denominator and the first term in the numerator are always positive, the only way for v_{\parallel} to be negative is for the second term in the numerator to be negative and to dominate the first term, which requires that $\delta P \neq 0$. ■

Therefore, a decision function for whether a point is moving is given by

$$\phi(P_i^k, \tilde{R}, \tilde{q}) = \begin{cases} 1 & \text{if } |v_{\perp}| > \tau \text{ or } v_{\parallel} < -\tau \\ 0 & \text{otherwise.} \end{cases} \quad (31)$$

where τ is a positive threshold. Due to small errors in calculating the relative pose and inaccuracies with camera calibration, a threshold of one pixel is the tightest constraint that can be used.

B. Recursive-RANSAC Tracker

Moving points found using Equation (31) are the input to the Recursive-RANSAC (R-RANSAC) multiple target tracking algorithm described in [22], [23]. In essence, the R-RANSAC algorithm fits motion model trajectories to sequences of points. For this paper we use a constant velocity motion model. Tracks are initialized when a sufficient number of new data indicate a moving object, and existing tracks are propagated forward using a Kalman Filter. Probabilistic Data Association [24] is used to account for measurement association uncertainty. Each track is given an inlier score based on the percentage of time steps in which the track is detected. R-RANSAC also has a track management system that merges similar tracks and removes tracks that have an inlier score lower than the minimum threshold. For more details see [22], [23].

V. RESULTS

The performance of the pose estimation algorithm will be demonstrated in simulation and with real flight tests. In the simulation study outlined in Section V-A we know the true pose and so we will be able to assess the accuracy of pose estimation. In the flight test outlined in Section V-B, we apply pose estimation to motion detection and tracking as described in Section IV.

A. Pose Estimation - Synthetic Video

Algorithm 2 was tested on a synthetic video sequence of a UAV inside a city generated using the BYU Holodeck simulator [25]. The two-minute video sequence (3600 frames) includes aggressive rotational and translational motions. A screenshot of the video sequence is shown in Fig. 3. The purpose for using the simulator is to compare estimated pose to true pose.

We will consider three error metrics in evaluating Algorithm 2. The rotational error give by

$$e_R = \left\| \log(R_{true} R_{est}^{-1}) \right\| = \frac{\text{tr}(R_{true} R_{est}^{-1}) - 1}{2}$$

is the smallest rotation angle between the estimated and true rotation. The translation error given by

$$e_q = \cos^{-1}(q_{true}^\top q_{est}) \quad (32)$$



Fig. 3. Screenshot of the holodeck video sequence

is the angle between the estimated and true normalized translation. The rotation and translation error metrics do not penalize pose disambiguation errors since they return the same metric for all four possible rotation-translation pairs. Therefore the third metric is the pose disambiguation metric defined as the percentage of time that Algorithm 2 selects the correct rotation and translation. Both the rotational and translational error are measured in radians. The LMedS Sampson error is also computed. Unless otherwise noted, all error metrics are averaged over the entire video sequence of 3599 frame pairs.

The error over time for the OpenCV Nister/LMedS polynomial solver [4], LM Basis [7], and Algorithm 2 are shown in Fig. 4. All three algorithms give low error for the UAV trajectory. Notice how the rotation error seems to be proportional to the total rotation, while the translation error becomes very large as the true translation approaches zero.

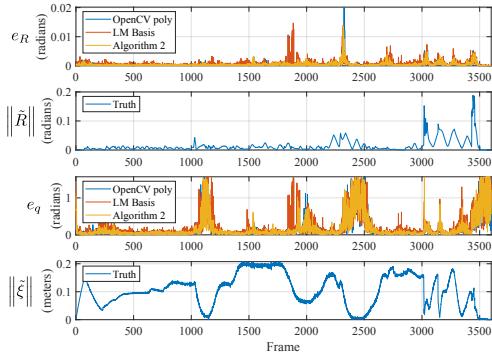


Fig. 4. Incremental rotation and translation error over entire video sequence. True incremental translation and rotation are also shown.

In order to provide a fair comparison of initialization schemes for Algorithm 2 with the OpenCV five-point polynomial solver, the IMU was not used in this section for initialization. Alternatively we compare random initialization,

where both \tilde{R}_0 and \tilde{q}_0 are selected randomly, random recursive initialization, where the first LM optimization at each time step is initialized randomly, but subsequent LM optimizations at that time step use the best pose hypothesis prior, where $(\tilde{R}_0, \tilde{q}_0)$ is the best pose from the previous time step, and prior recursive, where the first LM optimization at each time step is the best pose from the previous time step, but subsequent LM optimizations at that time step use the best pose hypothesis. All results use LMedS for outlier rejection. The LM optimization is repeated 100 times with five matching features at each iteration. The mean error across the entire video sequence is plotted in Fig. 5.

This result shows the importance of initializing the optimizer with a prior. The random initialization method performs the worst out of all four methods, while initializing the optimizer with a prior from the previous time step or the best LMedS hypothesis far from the current time step significantly reduces the error. IMU prediction will further improve these results. After 100 iterations, the LMedS error for the initialization methods that use prior information is comparable to the OpenCV five-point polynomial solver, despite the fact that only one hypothesis is generated per subset instead of an average of about four hypotheses. LM Basis also generates hypotheses from random seeds, resulting in higher error than methods that initialize from a prior. Note that while LM Basis generates up to 10 hypotheses per subset, it removes duplicates and hypothesis that do not appear to converge, resulting in an average of only 1.82 hypotheses per subset. Dropping these hypotheses early may increase the error based on iteration number, but results in a lower error when compared against time.

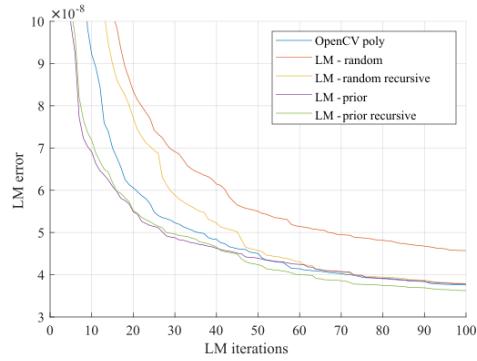


Fig. 5. Comparison of LM seeding methods.

Fig. 6 shows the error of Algorithm 2 compared to the OpenCV 5-point algorithm, but with the x-axis changed to be time instead of number of iterations. When under a time constraint, Algorithm 2 significantly outperforms the OpenCV solver [4] and LM Basis [7].

For outlier rejection, RANSAC and LMedS were also compared. For RANSAC the algorithm was tested with 19 different thresholds. For LMedS, the algorithm was run once, because there is no threshold parameter to tune. For each run

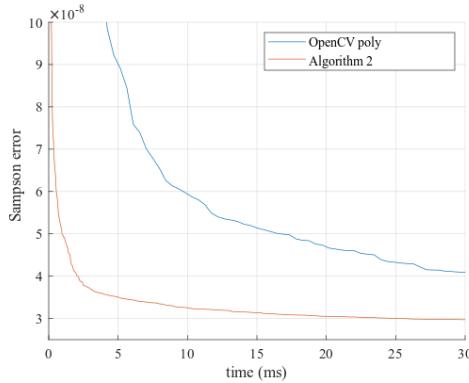


Fig. 6. Sampson error over time.

the average truth rotation and translation error over the entire video sequence were calculated. As shown in Fig. 7, LMedS performs well without requiring a threshold. However, in order for RANSAC to perform as well as LMedS, the threshold must be tuned to within an order of magnitude of the optimal threshold.

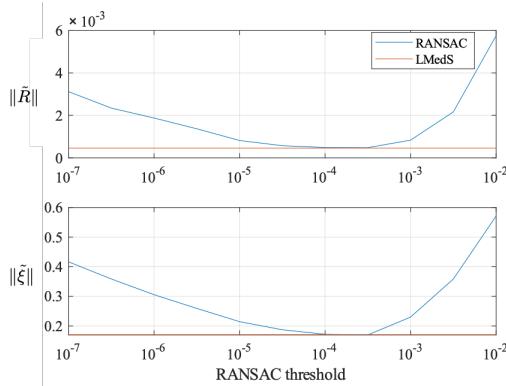


Fig. 7. Average rotation and translation errors for RANSAC and LMedS

Table I shows the results of the rotation and translation disambiguation algorithms. The first row within each group shows a baseline comparison, where no method was used for pose disambiguation. The baseline method gives poor results. However, it is worth noting that Algorithm 2 returns the correct rotation, even without any form of pose disambiguation. This is likely because it is seeded at the first frame with the identity rotation, and in every frame thereafter the best hypothesis from the previous iteration is used as the seed to the optimizer. The second row in each group shows the results when the cheirality check was used to determine the best of the four possible rotation/translation pairs. The translation direction is often correct but the rotation is correct only about half of the time. The third row in each group shows the results

TABLE I
POSE DISAMBIGUATION COMPARISON

Solver	Pose disambiguation method	Rotation correct	Translation correct	Both correct
OpenCV	none	50.2%	14.7%	6.8%
OpenCV	cheirality	54.0%	93.2%	52.3%
OpenCV	trace + cheirality	100.0%	96.5%	96.5%
LM Basis	other + cheirality	100.0%	95.8%	95.8%
Algorithm 2	none	100.0%	57.2%	57.2%
Algorithm 2	cheirality	40.9%	92.1%	40.8%
Algorithm 2	trace + cheirality	100.0%	96.5%	96.5%

TABLE II
RELATIVE POSE REFINEMENT

Relative pose solver	Refine (success rate)	Rot err (radians)	Trans err (radians)	LMedS err (Sampson)
OpenCV poly	-	4.843e-04	1.742e-01	3.769e-08
LM Basis	No	7.076e-04	2.059e-01	4.762e-08
LM Basis	Yes	4.159e-04	1.708e-01	5.059e-08
Algorithm 2	No	4.640e-04	1.699e-01	3.653e-08
Algorithm 2	Yes (54.5%)	3.850e-04	1.596e-01	3.540e-08

of using the matrix trace to determine which rotation is correct and the cheirality check to determine the correct translation direction. This third pose disambiguation method consistently outperforms the other methods.

The best hypothesis was refined using Levenberg-Marquardt to optimize the Sampson cost over all inliers (Section III-D). The refined relative pose is only kept if the new relative pose successfully reduces the LMedS error. Table II compares the average error with and without refinement. Refining the best relative pose hypothesis significantly reduces all three error metrics. LM Basis also refines the best hypothesis and successfully reduces the translational and rotational error. However, even with refinement, LM Basis has a higher error than Algorithm 2.

Table III compares the computation time for relative pose estimation between the OpenCV implementation [4], LM Basis [7], and Algorithm 2. The algorithms were both implemented on a laptop with a 2.1 GHz Intel i7 CPU running Linux. The breakdown of the time required to generate each hypothesis set is shown in Figure 8. The most time-consuming part of the OpenCV solver is finding the zeros of the tenth-order polynomial. The most time-consuming part of algorithm 2 is the Eigen matrix solver. Note that while LM Basis and Algorithm 2 take about the same amount of time to generate hypotheses per subset, LM Basis generates up to 10 hypotheses per subset, while Algorithm 2 only generates one hypothesis. The faster optimization used in the LM Basis algorithm is likely due to solving simpler equations which require inverting a 3×3 matrix instead of a 5×5 matrix.

B. Motion Detection and Tracking - Flight Video

The motion detection algorithm was tested on a moving camera video sequence taken from a multi-rotor UAV. Fig. 9 shows the results of the motion detection algorithm. Note that the stationary points have zero perpendicular velocity and a positive parallax velocity, while the moving points have a non-zero perpendicular velocity component.

TABLE III
COMPUTATION TIME

	OpenCV poly	LM Basis	Algorithm 2
Hypothesis generation	100*0.404 ms= 40.4 ms	100*27.9 us= 2.79 ms	100*23.4 us= 2.34 ms
Hypothesis scoring (avg 400 pts)	400*17.3 ns= 6.91 ms	182*13.9 ns= 2.52 ms	100*8.99 ns= 0.90 ms
Refinement	-	2.17 ms	5.91 ns
Pose disambiguation	0.32 ms	1.86 ms	0.14 ms
Total	47.7 ms	9.57 ms	3.97 ms

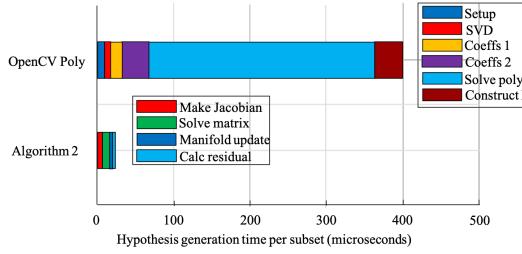


Fig. 8. Time required to generate each hypothesis set.

The computation times of the motion detection and tracking algorithm are shown in Table IV. For faster processing the video was scaled to 640x480. The motion detection and tracking algorithm is running on a Linux desktop computer with a 4GHz Intel i7 CPU. On average about 800 points are detected, tracked, and fed to Algorithm 2 each frame. Notice that the OpenCV feature detection and tracking are the most time-consuming components of the tracking algorithm and consume 70% of the total CPU usage. The complete algorithm takes 29 milliseconds to run per frame, which means it is capable of running in real-time at 34 frames per second (FPS).

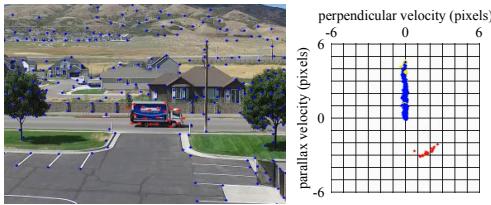


Fig. 9. Video motion detection results. Each point position (left) and its corresponding net velocity (right) are plotted. Points with a net perpendicular velocity greater than one pixel are classified as moving points (red), while points with a velocity below this threshold are classified as stationary points (blue).

VI. CONCLUSION

In this paper we have presented a relative pose estimation algorithm for solving the rotation and translation between consecutive frames, that requires at least five matching feature points per frame, and is capable of running in real-time. We show the importance of seeding the Levenberg-Marquardt optimizer with an initial pose estimate and demonstrate that this initial estimate significantly improves the performance of the



Fig. 10. Recursive RANSAC tracks

TABLE IV
MOTION DETECTION AND TRACKING COMPUTATION TIMES

Tracking Component	Computation Time
Good Features to Track	9.2 ms
LK optical flow	12 ms
Compute (R, \bar{q}) (Algorithm 2)	3.0 ms
Recursive RANSAC	0.4 ms
Other	4.4 ms
Total	29 ms (34 FPS)

algorithm. We have applied the algorithm to detecting motion and tracking multiple targets from a UAV and demonstrated real-time performance of this tracking algorithm on a 640x480 video sequence. Future work includes applications to 3D scene reconstruction and more complex tracking methods.

REFERENCES

- [1] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [2] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.
- [3] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.
- [4] "OpenCV 3.1: Open source computer vision library," <https://github.com/opencv/opencv/releases/tag/3.1.0>, 2015.
- [5] Y. Ma, J. Košecká, and S. Sastry, "Optimization criteria and geometric algorithms for motion and structure estimation," *International Journal of Computer Vision*, vol. 44, no. 3, pp. 219–249, 2001.
- [6] T. Botterill, S. Mills, and R. Green, "Refining essential matrix estimates from RANSAC," in *Proceedings Image and Vision Computing New Zealand*, 2011, pp. 1–6.
- [7] ———, "Fast RANSAC hypothesis generation for essential matrix estimation," in *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*. IEEE, 2011, pp. 561–566.
- [8] U. Helmke, K. Hüper, P. Y. Lee, and J. Moore, "Essential matrix estimation using Gauss-Newton iterations on a manifold," *International Journal of Computer Vision*, vol. 74, no. 2, pp. 117–136, 2007.
- [9] M. Sarkis, K. Diepold, and K. Hüper, "A fast and robust solution to the five-point relative pose problem using Gauss-Newton optimization on a manifold," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 1. IEEE, 2007, pp. I–681.
- [10] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.
- [11] J. Shi et al., "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*. IEEE, 1994, pp. 593–600.

- [12] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.
- [13] S. Belongie, "Cse 252b: Computer vision II, lecture 11," <https://cseweb.ucsd.edu/classes/sp04/cse252b/notes/lec11/lec11.pdf>, 2006.
- [14] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.
- [15] C. Forster, L. Carbone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, February 2017.
- [16] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. John Wiley & Sons, 2005, vol. 589.
- [17] V. Santhaseelan and V. K. Asari, "Moving object detection and tracking in wide area motion imagery," in *Wide Area Surveillance*. Springer, 2014, pp. 49–70.
- [18] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1219–1225.
- [19] A. Elqursh and A. Elgammal, "Online moving camera background subtraction," in *European Conference on Computer Vision*. Springer, 2012, pp. 228–241.
- [20] J. Kang, I. Cohen, G. Medioni, and C. Yuan, "Detection and tracking of moving objects from a moving platform in presence of strong parallax," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 10–17.
- [21] S. Dey, V. Reilly, I. Saleemi, and M. Shah, "Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7576 LNCS, no. PART 5, pp. 860–873, 2012.
- [22] P. C. Niedfeldt and R. W. Beard, "Multiple target tracking using recursive RANSAC," in *2014 American Control Conference*, June 2014, pp. 3393–3398.
- [23] P. C. Niedfeldt, K. Ingersoll, and R. W. Beard, "Comparison and analysis of recursive-RANSAC for multiple target tracking," *IEEE Transaction on Aerospace and Electronic Systems*, vol. 53, no. 1, pp. 461–476, feb 2017.
- [24] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Systems*, vol. 29, no. 6, 2009.
- [25] "BYU holodeck: A high-fidelity simulator for deep reinforcement learning," <https://github.com/byu-pcl/holodeck>, 2018.

Jacob H. White received the B.S. and M.S. degrees in electrical engineering from Brigham Young University in 2015 and 2019, respectively. His interests are autonomous systems and machine learning.



Randal W. Beard received the B.S. degree in electrical engineering from the University of Utah, Salt Lake City in 1991, the M.S. degree in electrical engineering in 1993, the M.S. degree in mathematics in 1994, and the Ph.D. degree in electrical engineering in 1995, all from Rensselaer Polytechnic Institute, Troy, NY. Since 1996, he has been with the Electrical and Computer Engineering Department at Brigham Young University, Provo, UT, where he is currently a professor. His primary research focus is autonomous control of unmanned air vehicles, multiple target tracking, and multivehicle coordination and control. He is a past associate editor for the IEEE Control Systems Magazine, the Journal of Intelligent and Robotic Systems, and the IEEE Transactions on Automatic Control. He is a fellow of the IEEE.

10.3 Moving feature segmentation

RWB: need to write this up.

Step 1. Find homography H_a^b between frames using matching feature pairs, and the RANSAC algorithm to identify outliers as $\bar{\mathcal{P}}$

Step 2. Find the essential matrix E_a^b between frames using matching pairs.

Step 3. Find all pairs in $\bar{\mathcal{P}}$ that do not satisfy the epipolar constraint $\bar{\epsilon}^{b\top} E_a^b \bar{\epsilon}^a = 0$, and label them as \mathcal{P} .

Step 4. Return the set \mathcal{P} as the set of potentially moving tracks.

10.4 Feature and track propagation

We have shown uncalibrated pixel are transformed between frames by the homography matrix as

$$\gamma_m \tilde{\mathbf{m}}^b = H_a^b \tilde{\mathbf{m}}^a.$$

The visual multiple target tracking algorithm produces pixel velocities, pixel accelerations, and 2×2 covariance matrices associated with each of these quantities. In this section, we show how to transform pixel velocities, accelerations, and covariances using the homography matrix.

Throughout this section we will use the following notation. The homography matrix will be decomposed into block elements as

$$\begin{pmatrix} H_1 & \mathbf{h}_2 \\ \mathbf{h}_3^\top & h_4 \end{pmatrix} \triangleq H_a^b,$$

and the homogeneous pixel coordinates are decomposed as

$$\begin{pmatrix} \hat{\mathbf{m}} \\ 1 \end{pmatrix} \triangleq \tilde{\mathbf{m}}.$$

Given the relationship

$$\gamma_m \begin{pmatrix} \hat{\mathbf{m}}^b \\ 1 \end{pmatrix} = \begin{pmatrix} H_1 & \mathbf{h}_2 \\ \mathbf{h}_3^\top & h_4 \end{pmatrix} \begin{pmatrix} \hat{\mathbf{m}}^a \\ 1 \end{pmatrix},$$

it is tempting to compute the pixel velocity $\dot{\mathbf{m}}^b$ as

$$\gamma_m \begin{pmatrix} \dot{\mathbf{m}}^b \\ 0 \end{pmatrix} = \begin{pmatrix} H_1 & \mathbf{h}_2 \\ \mathbf{h}_3^\top & h_4 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{m}}^a \\ 0 \end{pmatrix},$$

and conclude that $\gamma_m \dot{\hat{\mathbf{m}}}^b = H_1 \dot{\hat{\mathbf{m}}}^a$, or even that $\dot{\hat{\mathbf{m}}}^b = H_1 \dot{\hat{\mathbf{m}}}^a$. However, doing so ignores the fact that γ_m is dependent on the pixel $\hat{\mathbf{m}}^a$.

To derive the correct relationship, observe that

$$\begin{aligned} \gamma_m \begin{pmatrix} \hat{\mathbf{m}}^b \\ 1 \end{pmatrix} &= \begin{pmatrix} H_1 & \mathbf{h}_2 \\ \mathbf{h}_3^\top & h_4 \end{pmatrix} \begin{pmatrix} \hat{\mathbf{m}}^a \\ 1 \end{pmatrix} \\ \iff \begin{pmatrix} \gamma_m \hat{\mathbf{m}}^b \\ \gamma_m \end{pmatrix} &= \begin{pmatrix} H_1 \hat{\mathbf{m}}^a + \mathbf{h}_2 \\ \mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4 \end{pmatrix}, \end{aligned}$$

which implies that

$$\begin{aligned} \gamma_m &= \mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4 \\ \hat{\mathbf{m}}^b &= \frac{H_1 \hat{\mathbf{m}}^a + \mathbf{h}_2}{\mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4}. \end{aligned}$$

Defining the function

$$g(\hat{\mathbf{m}}, H) \triangleq \frac{H_1 \hat{\mathbf{m}} + \mathbf{h}_2}{\mathbf{h}_3^\top \hat{\mathbf{m}} + h_4}, \quad (10.10)$$

we have that 2D-pixels are transformed between frames as

$$\hat{\mathbf{m}}^b = g(\hat{\mathbf{m}}^a, H_a^b).$$

Therefore, the 2D pixel velocity is transformed as

$$\begin{aligned} \dot{\hat{\mathbf{m}}}^b &= \frac{\partial g}{\partial \hat{\mathbf{m}}} \Big|_{\hat{\mathbf{m}}=\hat{\mathbf{m}}^a} \dot{\hat{\mathbf{m}}}^a \\ &= G(\hat{\mathbf{m}}^a, H_a^b) \dot{\hat{\mathbf{m}}}^a, \end{aligned} \quad (10.11)$$

where

$$\begin{aligned} G(\hat{\mathbf{m}}^a, H_a^b) &= \frac{\partial g}{\partial \hat{\mathbf{m}}} \Big|_{\hat{\mathbf{m}}^a} \\ &= \frac{(\mathbf{h}_3^\top \hat{\mathbf{m}} + h_4) H_1 - (H_1 \hat{\mathbf{m}} + \mathbf{h}_2) \mathbf{h}_3^\top}{(\mathbf{h}_3^\top \hat{\mathbf{m}} + h_4)^2} \Big|_{\hat{\mathbf{m}}=\hat{\mathbf{m}}^a} \\ &= \frac{(\mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4) H_1 - (H_1 \hat{\mathbf{m}}^a + \mathbf{h}_2) \mathbf{h}_3^\top}{(\mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4)^2}. \end{aligned} \quad (10.12)$$

By similar logic, we get that accelerations transform as

$$\ddot{\hat{\mathbf{m}}}^b = G(\hat{\mathbf{m}}^a, H_a^b) \ddot{\hat{\mathbf{m}}}^a + \dot{G}(\hat{\mathbf{m}}^a, \dot{\hat{\mathbf{m}}}^a, H_a^b) \dot{\hat{\mathbf{m}}}^a,$$

where

$$\dot{G}(\hat{\mathbf{m}}^a, \dot{\hat{\mathbf{m}}}^a, H_a^b) = \frac{-(2\mathbf{h}_3^\top \dot{\hat{\mathbf{m}}}^a + h_4) H_1}{(\mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4)^2} + \frac{2(\mathbf{h}_3^\top \dot{\hat{\mathbf{m}}}^a + h_4)(H_1 \hat{\mathbf{m}}^a + \mathbf{h}_2) \mathbf{h}_3^\top}{(\mathbf{h}_3^\top \hat{\mathbf{m}}^a + h_4)^3}.$$

The next lemma shows how position covariances are transformed between images.

Lemma 10.4.1 Suppose that H_a^b is the homography matrix between frames a and b and that $\hat{\mathbf{m}}^a$ is a random variable representing pixel locations in frame a with mean $\hat{\mu}^a$ and covariance Σ_p^a . Suppose that $\hat{\mathbf{m}}^a$ is transformed according to $\hat{\mathbf{m}}^b = g(\hat{\mathbf{m}}^a, H_a^b)$ where g is defined in Equation (10.10), then the mean and covariance of $\hat{\mathbf{m}}^b$ are given by

$$\begin{aligned}\hat{\mu}^b &= g(\hat{\mu}^a, H_a^b) \\ \Sigma_p^b &= G(\hat{\mu}^a, H_a^b) \Sigma_p^a G^\top (\hat{\mu}^a, H_a^b)\end{aligned}$$

where G is defined in Equation (10.12).

Proof: Since pixel locations transform through g , it is clear that the mean transforms as

$$\hat{\mu}^b = g(\hat{\mu}^a).$$

For small deviations from the mean, we have that

$$g(\hat{\mathbf{m}}^a) \approx g(\hat{\mu}^a) + G(\hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a).$$

Since the covariance of $\hat{\mathbf{m}}^a$ is given by

$$\Sigma_p^a = E\{(\hat{\mathbf{m}}^a - \hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a)^\top\},$$

we get that

$$\begin{aligned}\Sigma_p^b &= E\{(\hat{\mathbf{m}}^b - \hat{\mu}^b)(\hat{\mathbf{m}}^b - \hat{\mu}^b)^\top\} \\ &= E\{(g(\hat{\mathbf{m}}^a) - g(\hat{\mu}^a))(g(\hat{\mathbf{m}}^a) - g(\hat{\mu}^a))^\top\} \\ &\approx E\{(g(\hat{\mu}^a) + G(\hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a) - g(\hat{\mu}^a))(g(\hat{\mu}^a) + G(\hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a) - g(\hat{\mu}^a))^\top\} \\ &= E\{(G(\hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a))(G(\hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a))^\top\} \\ &= G(\hat{\mu}^a)E\{(\hat{\mathbf{m}}^a - \hat{\mu}^a)(\hat{\mathbf{m}}^a - \hat{\mu}^a)^\top\}G^\top(\hat{\mu}^a) \\ &= G(\hat{\mu}^a)\Sigma_p^a G^\top(\hat{\mu}^a).\end{aligned}$$

■

Similarly, the next lemma shows how the covariance of the velocity is transformed between images.

Lemma 10.4.2 Suppose that H_a^b is the homography matrix between frames a and b and that $\dot{\mathbf{m}}^a$ is a random variable representing pixel velocity in frame a at pixel location $\hat{\mathbf{m}}^a$ with mean $\dot{\mu}^a$ and covariance Σ_v^a . Suppose that the pixel locations $\hat{\mathbf{m}}^a$ is transformed according to $\dot{\mathbf{m}}^b = g(\hat{\mathbf{m}}^a, H_a^b)$ where g is defined in Equation (10.10), then the mean and covariance of $\dot{\mathbf{m}}^b$ are given by

$$\dot{\mu}^b = G(\hat{\mu}^a, H_a^b)\dot{\mu}^a \tag{10.13}$$

$$\Sigma_v^b = G(\hat{\mu}^a, H_a^b)\Sigma_v^a G^\top(\hat{\mu}^a, H_a^b) \tag{10.14}$$

where G is defined in Equation (10.12).

Proof: From Equation (10.11) it is clear that the mean velocity transforms according to Equation (10.13). The covariance calculation is standard for linear transformations and is given by

$$\begin{aligned}\Sigma_v^b &= E\{(\dot{\mathbf{m}}^b - \dot{\boldsymbol{\mu}}^b)(\dot{\mathbf{m}}^b - \dot{\boldsymbol{\mu}}^b)^\top\} \\ &= E\{(G\dot{\mathbf{m}}^a - G\dot{\boldsymbol{\mu}}^a)(G\dot{\mathbf{m}}^a - G\dot{\boldsymbol{\mu}}^a)^\top\} \\ &= E\{G(\hat{\mathbf{m}}^a - \hat{\boldsymbol{\mu}}^a)(\hat{\mathbf{m}}^a - \hat{\boldsymbol{\mu}}^a)^\top G^\top\} \\ &= GE\{(\hat{\mathbf{m}}^a - \hat{\boldsymbol{\mu}}^a)(\hat{\mathbf{m}}^a - \hat{\boldsymbol{\mu}}^a)^\top\}G^\top \\ &= G\Sigma_v^a G^\top.\end{aligned}$$

■

10.5 Recursive-RANSAC

RWB: Need a clear explanation of the R-RANSAC algorithm. Use this paper as a starting point.

2014 American Control Conference (ACC)
June 4-6, 2014. Portland, Oregon, USA

Multiple Target Tracking using Recursive RANSAC

Peter C. Niedfeldt and Randal W. Beard
Department of Electrical and Computer Engineering
Brigham Young University, Provo, UT 84602

Abstract—Estimating the states of multiple dynamic targets is difficult due to noisy and spurious measurements, missed detections, and the interaction between multiple maneuvering targets. In this paper a novel algorithm, which we call the recursive random sample consensus (R-RANSAC) algorithm, is presented to robustly estimate the states of an unknown number of dynamic targets. R-RANSAC was previously developed to estimate the parameters of multiple static signals when measurements are received sequentially in time. The R-RANSAC algorithm proposed in this paper reformulates our previous work to track dynamic targets using a Kalman filter. Simulation results using synthetic data are included to compare R-RANSAC to the GM-PHD filter.

I. INTRODUCTION

Multiple target tracking (MTT) continues to be an active field of research due to inherent difficulties caused by noisy measurements, false detections, missed detections, and the interactions between multiple maneuvering targets. There exists a wide range of target tracking applications in both civilian and military applications. Some of these applications include tracking pedestrians [1], air and ground vehicles [2], bacteria [3], air traffic control [4], and many others. In this paper we present a new algorithm that efficiently and robustly tracks multiple targets in clutter.

Classical MTT techniques include nearest neighbor [5], multiple hypothesis tracking (MHT) [6], and the joint probabilistic data association (JPDA) [7] algorithms. Numerous variations and applications of these algorithms can be found in the literature. Unfortunately, there are limitations with each of these algorithms: nearest neighbor is not robust, MHT is computationally expensive and difficult to implement, and JPDA requires a priori information on the number of targets to be tracked and makes a hard data association each time step. Recently, the probabilistic hypothesis density (PHD) filter has successfully applied random finite set theory to efficiently track an unknown number of targets in highly-cluttered environments [8]. A significant advantage of the PHD filter is that the computations are linear in the number of measurements. However, a disadvantage is that additional steps are required to maintain track continuity. Also, the estimated number of targets has a large variance in clutter [9], and Erdnic et al. note that the PHD filter is sensitive to track loss after very few missed detections [10].

This research was made with Government support through the DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

KALMANSAC is another algorithm that robustly tracks a dynamic target using causal measurements [11]. Both KALMANSAC and the recursive-RANSAC algorithm developed in this paper rely on the principles of the random sample consensus (RANSAC) algorithm. RANSAC was developed as a new paradigm to regression analysis given a batch of data [12]. Traditional regression techniques form an estimate using all or most of the available data set, and then remove observations inconsistent with the hypothesis before producing a final estimate. Alternatively, RANSAC forms numerous hypothesis estimates using the minimum number of required points and computes the support of each hypothesis, where the hypothesis support refers to the number of measurements, or *inliers*, whose residual is less than a specified threshold. The hypothesis with the greatest support is selected and smoothed using the inlier set, allowing RANSAC to quickly and accurately estimate the parameters of an underlying signal in clutter. For a recent survey of the many variations of RANSAC, see [13].

The KALMANSAC algorithm utilizes the RANSAC paradigm when tracking a dynamic target. KALMANSAC first selects a random minimum subset of measurements to estimate the most likely state estimate, which is subsequently used to estimate the most likely inlier/outlier measurement labeling over the measurement set. These steps are iterated until a stopping criterion is satisfied. The resulting measurement labeling is used to seed the iteration of subsequent time-steps. Since KALMANSAC computes the maximum a posterior estimate, the underlying measurement distribution, excluding the outliers, must be unimodal. However, this assumption does not hold when there exist multiple targets since there is no mechanism for data association between targets.

In this paper, we extend RANSAC to recursively estimate multiple dynamic targets using the recursive-RANSAC (R-RANSAC) framework, previously developed for estimating the parameters of multiple static signals [14]. While KALMANSAC uses iteration to compute the MAP estimate, R-RANSAC stores multiple hypothesis tracks in memory to allow subsequent inlier measurement to refine the current estimate. By tracking multiple hypotheses across several time steps, R-RANSAC can naturally track multiple targets without prior knowledge of the number of existing targets [15].

The R-RANSAC algorithm is summarized as follows. At each time step, existing tracks are propagated using the Kalman filter. Current measurements that are inliers to existing tracks are used to update those tracks. The remaining

current measurements are outliers and are used in conjunction with a random minimum set of previous measurements to hypothesize the initial states of a new track that characterizes the outlier. This new track is propagated forward and updated using measurements that are within the inlier threshold to obtain a current state estimate of the target. Tracks with high support are identified as valid target tracks.

The rest of the paper is organized as follows. Section II describes the problem and notation. In Section III the initial states of a single target using RANSAC using maximum likelihood. Section IV presents the R-RANSAC algorithm to recursively estimate the states of multiple targets. Simulations are presented in Section V using synthetic data.

II. PROBLEM DESCRIPTION

We desire to estimate the states of multiple dynamic targets given a set of measurements and a dynamic model. Let the surveillance region be defined as a subset of the measurement space, $\mathcal{R}_y \subset \mathbb{R}^m$. Let $\mathbf{x}_i \in \mathbb{R}^n$ be the n states of the i^{th} target whose dynamics are given by

$$\mathbf{x}_i[k] = A\mathbf{x}_i[k-1] + \mathbf{w}_i[k], \quad (1)$$

where $\mathbf{w} \in \mathbb{R}^n$ is wide-sense stationary (WSS), zero-mean Gaussian process noise with covariance Q . Let $\mathbf{y}_i \in \mathcal{R}_y$ be the measurement of the i^{th} target, given by

$$\mathbf{y}_i[k] = C\mathbf{x}_i[k] + \mathbf{v}_i[k], \quad (2)$$

where $\mathbf{v} \in \mathbb{R}^m$ is a WSS, zero-mean Gaussian process with covariance R .

Let $M[k]$ be the unknown number of true targets at time k , where at each time step the probability that the i^{th} target is observed is modeled by the probability of detection $p_i \in (0, 1]$. Define a sensor *scan* as the set of ψ_k observations received after processing the sensor data at time k . The observation $\mathbf{y}_j, j = 1, \dots, \psi_k$, can be a measurement of an underlying true target or instead a realization of some random distribution defined over the surveillance region.

A false measurement, or clutter, is received according to a Poisson distribution with parameter λ_{GE} , where λ_{GE} is the expected amount of clutter per scan. Unless prior knowledge of the clutter distribution is known, we assume it is independent and uniformly distributed over the surveillance region as in [16]. We assume that target evolution, target measurements, and clutter measurements are all independent from each other, and that the system (A, C) is observable. Finally, the probability of detection is state independent. These assumptions are common in the literature [8], [5].

III. COMPUTING THE INITIAL STATES FROM BATCH ESTIMATION

Least-squares estimation finds the optimal parameters to fit a model to a batch of data by minimizing the mean-squared error. Alternatively, the random sample consensus (RANSAC) algorithm generates hypotheses using the fewest necessary measurements from a batch of data; the hypothesis with the most support, or inliers, is selected as the best estimate. Inliers are measurements whose residual is less than a threshold τ_R .

In the following, we estimate the current states of a targets using maximum likelihood and the RANSAC framework.

A. Single Target with No Clutter

Consider the scenario when there is a single target that is measured with probability $p_1 = 1$ and that the probability of clutter is $\lambda_{\text{GE}} = 0$. Under these assumptions, there is exactly one measurement per time step that corresponds to the target of interest. For simplicity, we define $k_N \triangleq k - N + 1$ as the initial time index of the measurement window of length N . Consider a sequence of N measurements characterized by

$$\begin{bmatrix} \mathbf{y}[k_N] \\ \mathbf{y}[k_N+1] \\ \vdots \\ \mathbf{y}[k] \end{bmatrix} = \begin{bmatrix} C\mathbf{x}[k_N] \\ C\mathbf{x}[k_N+1] \\ \vdots \\ C\mathbf{x}[k] \end{bmatrix} + \begin{bmatrix} \mathbf{v}[k_N] \\ \mathbf{v}[k_N+1] \\ \vdots \\ \mathbf{v}[k] \end{bmatrix}. \quad (3)$$

Using (1), we write (3) in terms of $\mathbf{x}[k_N]$, as shown in (10). Define $\mathbf{Y}_k = [\mathbf{y}[k_N], \dots, \mathbf{y}[k]]^\top$, $\mathbf{W}_k = [\mathbf{w}[k_N], \dots, \mathbf{w}[k]]^\top$, $\mathbf{V}_k = [\mathbf{v}[k_N], \dots, \mathbf{v}[k]]^\top$, and

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix}, \quad G = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & C & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & CA^{N-2} & CA^{N-3} & \dots & C \end{bmatrix}.$$

Matrix \mathcal{O} is $mN \times n$ and describes the expected measurements evolving from the initial state $\mathbf{x}[k_N]$ to subsequent time steps; it is the observability matrix if $N = n$. Matrix G has dimensions $mN \times nN$ and propagates the process noise.

We can write (10) as

$$\mathbf{Y}_k = \mathcal{O}\mathbf{x}[k_N] + \xi_k, \quad (4)$$

where $\xi_k = GW_k + V_k$ is the combined propagated process noise and the measurement noise. Since \mathbf{w} and \mathbf{v} are i.i.d. and zero-mean Gaussian, then ξ is also zero-mean Gaussian with covariance Ξ given by

$$\Xi = E[\xi_k \xi_k^\top] = GE[W_k W_k^\top] G^\top + E[V_k V_k^\top]. \quad (5)$$

Define $\mathbf{Q} = E[W_k W_k^\top]$ and $\mathbf{R} = E[V_k V_k^\top]$, where, due to the assumed independence of the process and measurement noise, \mathbf{Q} and \mathbf{R} are both block diagonal matrices. Using this notation, (5) can be written as

$$\Xi = G\mathbf{Q}G^\top + \mathbf{R}. \quad (6)$$

Briefly consider the scenario where there is no process or measurement noise, i.e., $\mathbf{w} = \mathbf{v} = 0$. In this case, (4) reduces to $\mathbf{Y}_k = \mathcal{O}\mathbf{x}[k_N]$ and assuming \mathcal{O} is full rank, the initial state $\mathbf{x}[k_N]$ is found using least-squares, as

$$\mathbf{x}[k_N] = (\mathcal{O}^\top \mathcal{O})^{-1} \mathcal{O}^\top \mathbf{Y}_k. \quad (7)$$

By observability, this is true if $N \geq n$. When including the process and measurement noise, the maximum likelihood estimate (MLE) of $\mathbf{x}[k_N]$ and the covariance are given by

$$\hat{\mathbf{x}}[k_N] = (\mathcal{O}^\top \Xi^{-1} \mathcal{O})^{-1} \mathcal{O}^\top \Xi^{-1} \mathbf{Y}_k \quad (8)$$

$$P[k_N] = (\mathcal{O}^\top \Xi^{-1} \mathcal{O})^{-1}. \quad (9)$$

$$\begin{bmatrix} \mathbf{y}[k_N] \\ \mathbf{y}[k_N+1] \\ \vdots \\ \mathbf{y}[k] \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix} \mathbf{x}[k_N] + \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & C & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & CA^{N-2} & CA^{N-3} & \dots & C \end{bmatrix} \begin{bmatrix} \mathbf{w}[k_N] \\ \mathbf{w}[k_N+1] \\ \vdots \\ \mathbf{w}[k] \end{bmatrix} + \begin{bmatrix} \mathbf{v}[k_N] \\ \mathbf{v}[k_N+1] \\ \vdots \\ \mathbf{v}[k] \end{bmatrix} \quad (10)$$

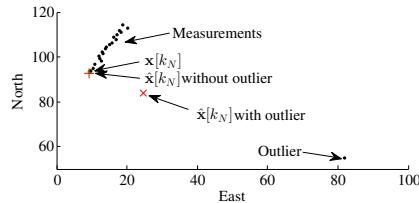


Fig. 1. MLE diverges when even one false measurement is introduced.

Note that (8) is similar to the pseudoinverse in (7), but weighted by the noise covariance. After computing the initial states $\mathbf{x}[k_N]$, the Kalman filter is used to estimate the current states. The initial state is updated using the measurement $y[k_N]$, then the states are propagated forward and updated for each subsequent time step.

Unfortunately, even a single false measurement may cause the MLE to diverge. A simple example is shown in Fig. 1, and is analogous to Fischler and Bolles' demonstration that one false measurement can corrupt the least-squares solution [12]. In the following, we use a minimum subset of data to find the initial states of a dynamic target. In Section III-C we then use RANSAC to robustly estimate the target track.

B. Multiple Targets in Clutter

Let the number of true targets be $M[k] \geq 1$ and the clutter rate $\lambda_{GE} \geq 0$. Under these conditions, multiple measurements are received at each time step. As in the previous section, the objective is to estimate the initial states $\mathbf{x}_i[k_N], i = 1, \dots, M[k]$. We do so by applying the RANSAC paradigm and using only a minimal number of randomly selected measurements b to estimate the initial states, where b is defined as the observability index of the system. The observability index is determined by finding the smallest natural number b such that $\text{rank}(\mathcal{O}_b) = \text{rank}(\mathcal{O}_{b+1})$, where

$$\mathcal{O}_b = [C \quad CA \quad \dots \quad CA^{b-1}]^\top.$$

Recall that ψ_k describes the number of measurements received during the k^{th} measurement scan. Let $\Psi_k = \sum_{\kappa=k_N}^k \psi_\kappa$ be the total number of measurements in the measurement window at time step k . For convenience, define $\mathbb{S} = \binom{\Psi_k}{b}$ as the set of $\binom{\Psi_k}{b} = \frac{\Psi_k!}{b!(\Psi_k-b)!}$ possible combinations of b observations, where each $S \in \mathbb{S}$ contains the minimum subset of points necessary to estimate the initial states $\mathbf{x}[k_N]$ according to the observability index b .

Since we receive multiple measurements per scan, we define an $m\Psi_k \times mN$ matrix Φ_k to associate the correct time indices to all the measurements received during that

time step. Let $\mathbf{1}_{\psi_k}$ be defined as a column vector of ψ_k ones, and let I_m be defined as a $m \times m$ identity matrix. Also, let the \otimes operator be the standard Kronecker product. Define

$$\Phi_k = \begin{bmatrix} \mathbf{1}_{\psi_{k_N}} \otimes I_m & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{1}_{\psi_{k_N+1}} \otimes I_m & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{1}_{\psi_k} \otimes I_m \end{bmatrix}. \quad (11)$$

With Φ_k , the windowed measurements are expressed as

$$\begin{bmatrix} \mathbf{y}_1[k_N] \\ \vdots \\ \mathbf{y}_{\psi_{k_N}}[k_N] \\ \mathbf{y}_1[k-N+2] \\ \vdots \\ \mathbf{y}_{\psi_k}[k] \end{bmatrix} = \Phi_k \mathcal{O} \mathbf{x}_i[k_N] + \Phi_k G_k W_k + \Phi_k V_k$$

$$Y_k = \Phi_k \mathcal{O} \mathbf{x}_i[k_N] + \Phi_k \xi_k. \quad (12)$$

The estimated initial state $\hat{\mathbf{x}}_i[k_N]$ can be found by modifying Equation (8). Let \mathbf{q} be a random vector with b elements uniformly distributed over the set $\{1, 2, \dots, \Psi_k\}$. Define $S_{\mathbf{q}} \in \mathbb{S}$ as a minimum subset of b measurements randomly selected from the set of $\binom{\Psi_k}{b}$ measurement combinations, where \mathbf{q} defines the indices of the selected measurements. To improve performance, we restrict $S_{\mathbf{q}}$ to contain no more than one measurement per scan. We define a binary indicator matrix that allows us to select specific measurements of interest from the measurement vector Y_k . In general, define $\mathbf{d} = [d_1, \dots, d_d]^\top$ to be a vector where each element is an index to a specific measurement in Y_k . Let $B(\mathbf{d})$ be defined as a binary indicator matrix of dimensions $md \times m\Psi_k$, where

$$B(\mathbf{d}) = \begin{bmatrix} \mathbf{0}_{m \times m(d_1-1)} & I_m & \mathbf{0}_{m \times m(\Psi_k-d_1)} \\ \vdots & \vdots & \vdots \\ \mathbf{0}_{m \times m(d_d-1)} & I_m & \mathbf{0}_{m \times m(\Psi_k-d_d)} \end{bmatrix}. \quad (13)$$

Starting with Equation (12), the binary indicator matrix $B(\mathbf{q})$ is pre-multiplied to both sides. Letting $\bar{Y} = B(\mathbf{q}) Y_k$, $\bar{\mathcal{O}} = B(\mathbf{q}) \Phi_k \mathcal{O}$, and $\bar{\xi}_k = B(\mathbf{q}) \Phi_k \xi_k$, we have

$$\bar{Y} = \bar{\mathcal{O}} \mathbf{x}_i[k_N] + \bar{\xi}_k, \quad (14)$$

where the noise term $\bar{\xi}_k$ has covariance $\bar{\Xi}$ given by

$$\begin{aligned} \bar{\Xi} &= E[\bar{\xi}_k \bar{\xi}_k^\top] \\ &= B(\mathbf{q}) \Phi_k E[\xi_k \xi_k^\top] \Phi_k^\top B(\mathbf{q})^\top \\ &= B(\mathbf{q}) \Phi_k \Xi \Phi_k^\top B(\mathbf{q})^\top. \end{aligned} \quad (15)$$

The MLE of the initial states and covariance is given by

$$\hat{x}[k_N] = (\bar{\mathcal{O}}^T \Xi^{-1} \bar{\mathcal{O}})^{-1} \bar{\mathcal{O}}^T \Xi^{-1} \bar{Y} \quad (16)$$

$$P[k_N] = (\bar{\mathcal{O}}^T \Xi^{-1} \bar{\mathcal{O}})^{-1}. \quad (17)$$

Inliers to the track described by (16) can be found by finding the measurements whose residual is less than a threshold τ_R ,

$$\chi[k] = \left\{ \kappa \in \{1, \dots, \Psi_k\} : \|B(\kappa)Y_k - B(\kappa)\Phi_k \mathcal{O}\hat{x}[k_N]\|_\infty < \tau_R \right\}. \quad (18)$$

C. RANSAC for Dynamic Targets

Since the ML estimate in (16) is computed using a random minimum subset of measurements, the estimate will only accurately estimate the i^{th} target when the b measurements all measure the i^{th} target. For this reason, we use RANSAC to generate and test multiple hypotheses. The RANSAC algorithm consists of two repeated steps: A hypothesis generation step and a hypothesis validation step. During hypothesis generation, random minimum subsets of measurements are used to form a set of hypothesis tracks \hat{x}' . During hypothesis validation, the support for each hypothesis is determined by counting the number of measurements whose residual to the hypothesis estimate is within a specified threshold τ_R . After ℓ hypotheses are tested, the hypothesis with the most support is selected and smoothed over the associated set of inliers [12]. An optional early termination threshold $\gamma \in (0, 1]$ can be used to stop generating hypotheses when $|\chi[k]| > \gamma N$.

The RANSAC implementation is summarized in Algorithm 1. Line 3 uses (16) to estimate a hypothesis of the initial conditions. Line 4 uses the hypothesized initial condition and (18) to find the subset of measurements are within a threshold τ_R of the predicted states for that time step, where $|\cdot|$ denotes the size of a set. These steps are then repeated up to ℓ times to find the hypothesis with the most inliers. Lines 11-20 are the Kalman filter prediction and update step.

Figure 2 gives a simple example. Suppose we want to fit the current measurement $y[k]$ to previous data assuming a target whose dynamics are described by a constant acceleration model. In this case, three measurements are needed to estimate the initial states at time k_N . For each hypothesis, two additional measurements are randomly selected from previous measurements to form a minimum subset, which is used to estimate the initial states $\hat{x}'[k_N]$. Two hypotheses are considered, where the hypothesis with the most inliers is selected to be the most accurate. The final estimate $\hat{x}[k]$ is found by propagating the initial states, updating as necessary using the associated inlier measurements with a Kalman filter.

IV. RECURSIVE RANSAC FOR DYNAMIC TARGETS

In the previous section, we designed a RANSAC-based technique to estimate the target states from a window of N measurements using a measurement at time k . The recursive RANSAC (R-RANSAC) algorithm can be used when subsequent measurements are received to update the set of stored tracks. The R-RANSAC algorithm consists of

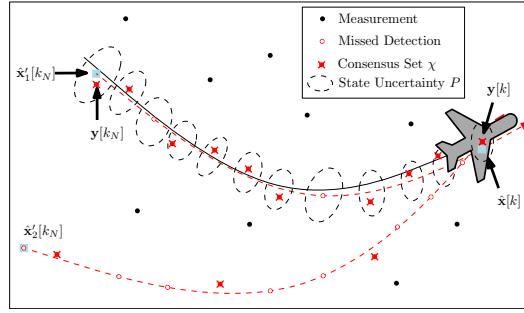


Fig. 2. RANSAC hypothesis tracks \hat{x}'_1 and \hat{x}'_2 are generated to fit the current measurement $y[k]$. The hypothesis described by \hat{x}'_1 is selected as the best track since it has ten inliers, as opposed to four for the other hypothesis. The final estimate is smoothed by propagating the initial states $\hat{x}'_1[k_N]$ using a Kalman filter and updating as necessary using the associated inliers.

Algorithm 1 RANSAC for Dynamic Targets

```

1: for  $\ell$  iterations, or until  $|\chi'[k]| > \gamma N$  do
2:   Select  $S_q \in \mathbb{S}$ 
3:    $\hat{x}'[k_N] = (\bar{\mathcal{O}}^T \Xi^{-1} \bar{\mathcal{O}})^{-1} \bar{\mathcal{O}}^T \Xi^{-1} \bar{Y}$ 
4:    $P'[k_N] = (\bar{\mathcal{O}}^T \Xi^{-1} \bar{\mathcal{O}})^{-1}$ 
5:    $\chi'[k] =$ 
    $\left\{ \kappa \in \{1, \dots, \Psi_k\} : \|B(\kappa)Y_k - B(\kappa)\Phi_k \mathcal{O}\hat{x}'[k_N]\|_\infty < \tau_R \right\}$ 
6:   Store track if  $|\chi'[k]|$  is larger than all previous tracks.
7: end for
8: for  $\kappa = k_N : k$  do
9:   if  $\kappa \neq k_N$  then
10:     $\hat{x}[\kappa] = \hat{x}[\kappa-1], P[\kappa] = AP[\kappa-1]A^T + Q$ 
11:   end if
12:   for  $j \in \{1, \dots, \psi_\kappa\} : y_j[\kappa] \in \chi'[k]$  do
13:      $L = P[\kappa] C^T (R + C P[\kappa] C^T)^{-1}$ 
14:      $P[\kappa] = (I_n - LC) P[\kappa]$ 
15:      $\hat{x}[\kappa] = \hat{x}[\kappa] + L(y_j[\kappa] - C\hat{x}[\kappa])$ 
16:   end for
17: end for

```

three main steps and is summarized in Algorithm 2. The first step in Line 2 is to predict existing R-RANSAC tracks forward using the Kalman prediction step. The second step in Line 4 classifies each measurement as either an inlier or an outlier. Line 6 uses each outlier to seed the RANSAC algorithm described in Algorithm 1 to generate a new track, while Line 8 uses inliers to the i^{th} track to update that track using the Kalman update step.

The remaining steps are for R-RANSAC track management. In Line 11, the consensus set for the i^{th} track is updated and the *inlier ratio* $\rho[i] = \frac{|\chi_i[k]|}{N}$ is recomputed. Lines 12 and 13 merges similar tracks and prunes the tracks to keep the best M tracks. Finally, Line 14 identifies as good tracks those whose inlier ratio and lifetime are above a threshold.

The R-RANSAC algorithm is related to the multiple hypothesis algorithm (MHT) [6] in that a set of hypotheses

Algorithm 2 Recursive RANSAC

```

1: for each  $k$  do
2:    $\hat{\mathbf{x}}_i[k] = A\hat{\mathbf{x}}_i[k-1], \forall i = \{1, \dots, \mathcal{M}\}$ .
3:   for each  $\mathbf{y}_j[k], \forall j = \{1, \dots, \psi_k\}$  do
4:      $\mathcal{I} = \{i : |\mathbf{y}_j[k] - A\hat{\mathbf{x}}_i[k]| < \tau_R\}, \forall i = \{1, \dots, \mathcal{M}\}$ .
5:     if  $|\mathcal{I}| = 0$  then
6:       Create new track found using Algorithm 1,
      where each  $S_q$  is chosen such that  $\mathbf{y}_j[k] \in S$ 
7:     else
8:       Update  $i^{\text{th}}$  track using Kalman update  $\forall i \in \mathcal{I}$ .
9:     end if
10:    end for
11:    Update  $\chi_i[k]$  and  $\rho_i[k] \forall i = \{1, \dots, \mathcal{M}\}$ .
12:    Eliminate similar tracks.
13:    Prune to keep  $\mathcal{M}$  best tracks
14:    Determine good track,  $\rho_i[k] \geq \tau_\rho$  and  $\kappa_i \geq \tau_\kappa$ .
15:  end for

```

are stored in memory between time steps. Similar to MHT, the R-RANSAC framework naturally identifies target births and deaths. However, the complexity of the MHT algorithm is proportional to $O(c^{E[\psi_k]})$ to account for all data associations, whereas the worst-case R-RANSAC performance is proportional $O(E[\psi_k]^2)$ resulting from performing RANSAC on each measurement. Where MHT builds a hypothesis/track tree to describe the data associations, R-RANSAC maintains a fixed number of tracks that are most probable given the current measurements. In short, R-RANSAC is much easier to code and computationally more efficient than MHT, but sacrifices the potential of optimal estimates.

Three additional parameters must be specified for the R-RANSAC algorithm: the number of stored tracks \mathcal{M} , the track similarity threshold τ_x , and the good track threshold τ_ρ . Each parameter can be tuned to account for various trade-offs depending on the simulation. The number of stored tracks \mathcal{M} should be greater than the expected number of true targets. Note that storing up to \mathcal{M} tracks in memory inherently allows up to \mathcal{M} targets at a time within the surveillance region, regardless of their birth or death time during the simulation. Similar tracks are pruned if the Mahalanobis distance is less than a threshold τ_x , leaving the track with the most inliers. The good track threshold τ_ρ can be increased to reduce the presence of false tracks, and decreased to reduce missed tracks. Increasing the measurement window length N leads to greater stability in the estimated number of good tracks. However, large N requires an increased delay before recognizing good tracks with sufficient inlier ratio and hinders detection of short-lived targets. We define an optional, but frequently used parameter to accept as good tracks those that persist for more than τ_κ time steps [17].

V. RESULTS

During the last decade, the probability hypothesis density (PHD) filter has shown promise when applied to multiple target tracking (MTT). The theoretical and mathematical framework behind the PHD filter was introduced in [18].

Vo and Ma present a closed-form solution to the PHD recursion in [8], where they assume Gaussian dynamics and model the underlying random finite sets as Gaussian mixtures, referred to as the GM-PHD filter. A survey of PHD filter implementations is found in [19]. It has been shown that in some scenarios the PHD filter outperforms classical algorithms such as the multiple hypothesis tracker and the joint probabilistic data association filter [9]. In this section, we compare the GM-PHD filter as presented in [8] to the R-RANSAC algorithm.

Twelve targets are initialized as summarized Table I. Ten of the targets persist throughout the 200 second simulation, while two targets are born after the simulation starts and die before it ends. The simulation step size is $\delta k = \frac{1}{3}$ second. The state dynamics are defined by a 2D, constant-velocity model, and the targets are measured with probability of detection $p_D = 0.95$. The target states are defined by $\mathbf{x} = [n \ e \ \dot{n} \ \dot{e}]$, and modeled as described by Example 1 in [8]. The PHD filter parameters are also the same in [8]. The parameters for R-RANSAC are: number of RANSAC iterations $\ell = 10$, inlier threshold $\tau_R = 3\sigma_\epsilon$, where σ_ϵ is the measurement noise. The good track threshold is $\tau_\rho = 0.75$. The number of stored measurement scans is $N = 25$, the number of stored tracks is $\mathcal{M} = 25$, and the early termination threshold is $\gamma = \tau_\rho$. As with the PHD filter, the similar track threshold is when the Mahalanobis distance between two tracks is less than $\tau_x = 4$. Finally, tracks must persist longer than $\tau_\kappa = 10$ time steps before being labeled a good track.

Both algorithms are capable of real-time performance. The algorithms require almost the same amount of computation, with R-RANSAC requiring 0.0739 sec per iteration and the PHD filter requiring 0.0754 sec per iteration in MATLAB on a 3 GHz Core™2 Duo processor. However, the true computational complexity is dependent on the ratio of targets to clutter. As the number of average clutter returns increases, the PHD filter becomes notably faster than R-RANSAC since RANSAC must be used more frequently in the inner loop of the filter. Conversely, R-RANSAC becomes notably faster as the ratio of targets to clutter increases since R-RANSAC only performs a Kalman update. This trade-off will be explored further in future work.

In this example, the ability of the PHD filter and R-RANSAC to detect the targets is approximately equal, at 95.7% and 96.0% respectively. The mean RMS error is comparable, with the R-RANSAC algorithm slightly more accurate with an RMS error of 5.6 m, compared to 7.6 m for the PHD filter. Most importantly, the false track rate is significantly lower for R-RANSAC at 0.023 false tracks per time step compared to 0.062 for the PHD filter. The number of targets over time and the inlier ratio are shown in Figure 4.

We note that track management can be incorporated by assigning a label to each good track which persists between time steps. Figure 4 shows the results when targets 1 and 3 and targets 9 and 10 cross at times 45 and 100, respectively. In the first case, the targets are moving so slow that their tracks were merged; in the second case, track uniqueness persisted during the crossing targets.

Target Number	1	2	3	4	5	6	7	8	9	10	11	12
Birth (sec)	0	0	0	0	0	0	0	0	0	0	25	25
Death (sec)	200	200	200	200	200	200	200	200	200	200	175	150
North Pos (m)	200	-200	200	600	600	-600	800	800	-900	-900	400	-800
East Pos (m)	0	-200	200	600	-600	-600	-800	800	-900	900	-400	-800
North Vel (m/s)	0	0	0	0	0	3	0	-5	0	0	0	9
East Vel (m/s)	2	0	-3	0	5	0	4	0	9	-9	1	0

TABLE I

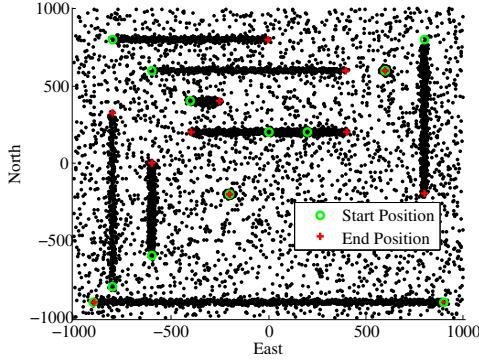


Fig. 3. Simulation measurement history.

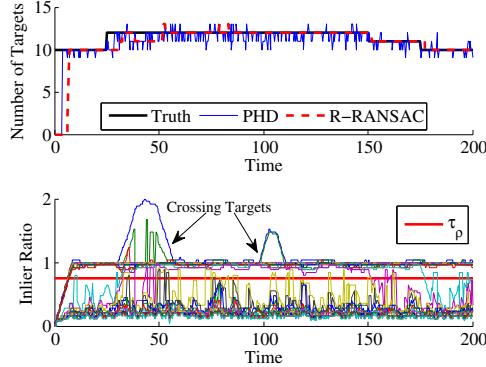


Fig. 4. Simulation results showing the true and estimated number of targets and the inlier ratio of the R-RANSAC filter during the simulation.

VI. CONCLUSION

In this work, we have applied the recursive RANSAC (R-RANSAC) framework developed in [14] to track dynamic targets. The new algorithm is capable of both estimating the true, dynamic number of targets and their states in clutter. R-RANSAC is compared to the state-of-the-art GM-PHD and is shown to be comparable in terms of accuracy and computational complexity.

Several avenues of possible research exist to extend and apply R-RANSAC. Similar to the PHD filter, we are considering methods of incorporating prior knowledge of target birth and spawning. This may improve the ability of R-RANSAC

to more rapidly detect targets, although good tracks would still need to satisfy the good track threshold. Finally, direct comparisons with classical tracking algorithms, such as JPDA and MHT, should be explored.

REFERENCES

- [1] N. A. Tsokas and K. J. Kyriakopoulos, "Multi-Robot Multiple Hypothesis Tracking for Pedestrian Tracking", *Autonomous Robots*, vol. 32, no. 1, pp. 63–79, Nov. 2011.
- [2] G. Thomasidis, L. Spinoulas, P. Lytrivis, M. Ahrholdt, G. Grubb, and A. Amditis, "Multiple Hypothesis Tracking for Automated Vehicle Perception", in *IEEE Intelligent Vehicles Symposium*, 2010, pp. 1122–1127.
- [3] T. M. Wood, C. A. Yates, D. A. Wilkinson, and G. Rosser, "Simplified Multitarget Tracking Using the PHD Filter for Microscopic Video Data", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 702–713, 2012.
- [4] J. K. Kuchar and L. C. Yang, "A Review of Conflict Detection and Resolution Modeling Methods", *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 179–189, 2000.
- [5] Y. Bar-Shalom and T.E. Fortmann, *Tracking and Data Association*, Academic Press, 1988.
- [6] D. Reid, "An Algorithm for Tracking Multiple Targets", *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.
- [7] T.E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-Target Tracking Using Joint Probabilistic Data Association", in *9th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, 1980, vol. 19, pp. 807–812.
- [8] B.-N. Vo and W.-K. Ma, "The Gaussian Mixture Probability Hypothesis Density Filter", *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.
- [9] R.P.S. Mahler, *Statistical Multisource-Multitarget Information Fusion*, Artech House, Norwood, 2007.
- [10] O. Erdinc, P. Willett, and Y. Bar-Shalom, "Probability Hypothesis Density Filter for Multitarget Multisensor Tracking", in *7th International Conference on Information Fusion*, 2005, pp. 146–153.
- [11] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto, "KALMANSAC: Robust Filtering by Consensus", in *Tenth IEEE International Conference on Computer Vision*, 2005, pp. 633–640.
- [12] M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [13] S. Choi, T. Kim, and W. Yu, "Performance Evaluation of RANSAC Family", *Proceedings of the British Machine Vision Conference*, 2009.
- [14] P.C. Niedfeldt and R.W. Beard, "Recursive RANSAC: Multiple Signal Estimation with Outliers", in *9th IFAC Symposium on Nonlinear Control Systems*, 2013.
- [15] P.C. Niedfeldt and R.W. Beard, "Convergence Analysis of the Recursive RANSAC Multiple Target Tracking Algorithm", *Under Review in IEEE Transactions on Automatic Control*.
- [16] P.H.S. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry", *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.
- [17] S.S. Blackman, "Multiple Hypothesis Tracking for Multiple Target Tracking", *Aerospace and Electronic Systems Magazine, IEEE*, vol. 19, no. 1, pp. 5–18, 2004.
- [18] R.P.S. Mahler, "A Theoretical Foundation for the Stein-Winter" Probability Hypothesis Density (PHD) Multitarget Tracking Approach", Tech. Rep., Lockheed Martin, 2000.
- [19] R.P.S. Mahler, "A Survey of PHD filter and CPHD filter Implementations", in *Defense and Security Symposium*, 2007.

10.6 Track Selection

The R-RANSAC multiple target tracking algorithm returns a set of track, together with covariance information. One of these tracks needs to be selected to follow. In the following, we list several possible options for target following.

10.6.1 Target closest to the image center

One option is to follow the track that is closest to the image center. If visual-MTT returns a set of normalized pixel coordinates ϵ_i for the tracks, then select the track that minimizes $\|\epsilon_i\|$.

10.6.2 Template matching

Suppose that we have a template image of the target that we desire to follow. The idea is to do a template match located at each potential track.¹⁰ The template image will be of a certain size and rotation. Unfortunately, the `openCV` function `matchTemplate` is not scale and rotation invariant. Therefore, given a track location ϵ_i , extract a set of scaled and rotated image blocks centered at ϵ_i . The template is then compared to each of these scaled and rotated images, and the best score is assigned to that track. The track with the best overall score is selected as the track to follow.

A variation of this scheme is when the template, is actually a set of templates, and the scaled and rotated windows from the image are compared to each member of the set. As the target is followed, additional images of the target can be added to the set of templates to facilitate future tracking.

Template matching does not need to be performed at each time step, but only at initialization, or when tracks are lost.

¹⁰ The `openCV` function `matchTemplate` can be used for template matching.

10.6.3 Machine learning

A variation of template matching is to use a machine learning algorithm to identify the track that best matches a learned model. As with template matching, scale and translation invariance needs to be insured.

10.6.4 User input

A final method for track selection is to query a user about which track should be followed. After the user has been queried, templates of the track can be stored in memory, and template matching can be used for continued tracking.

10.7 Target following

The tracking problem is best formulated in the body-level frame, as discussed in Section 9.2.

Recall that in the body-level frame, the kinematics are given simply as

$$\dot{\mathbf{p}}_{\ell/i}^i = \mathbf{v}_{b/i}^i,$$

where $\mathbf{v}_{b/i}^i$ is considered as the input signal to the velocity controller, in addition to the heading vector $\mathbf{s}_\psi = R_\ell^i \mathbf{e}_1$.

For the target following problem, we will assume that the position of the target in the inertial frame is given by $\mathbf{p}_{t/i}^i$, with velocity $\mathbf{v}_{t/i}^i$. The objective is to follow the target at some desired stand-off vector $\mathbf{p}_{d/\ell}^\ell$. We will assume that the desired stand-off vector $\mathbf{p}_{d/\ell}^\ell$ is constant in the level frame. For example, if the camera is fixed in the body frame, and is oriented so that it points down from the nose with an elevation angle of $-\alpha_0$ degrees, then a stand-off vector of

$$\mathbf{p}_{d/\ell}^\ell = D \begin{pmatrix} \cos \alpha_0 \\ 0 \\ \sin \alpha_0 \end{pmatrix}$$

is equivalent to commanding the multirotor to follow the target from a slant-range of length D . When perfect tracking is achieved, the set of all possible positions for the multirotor would form a cone about the target with side length D .

Let the relative position of the target to the multirotor is given by

$$\mathbf{p}_{t/\ell}^\ell = R_i^\ell \mathbf{p}_{t/i}^i - R_i^\ell \mathbf{p}_{\ell/i}^i.$$

The control objective is to maneuver the vehicle so that

$$\mathbf{p}_{d/t}^\ell \triangleq \mathbf{p}_{d/\ell}^\ell - \mathbf{p}_{t/\ell}^\ell \quad (10.15)$$

approaches zero asymptotically, or is ultimately uniformly bounded.

Differentiating Equation (10.15) gives

$$\begin{aligned} \dot{\mathbf{p}}_{d/t}^\ell &= \dot{\mathbf{p}}_{d/\ell}^\ell - \dot{\mathbf{p}}_{t/\ell}^\ell \\ &= -\dot{\mathbf{p}}_{t/\ell}^\ell \\ &= \mathbf{v}_{\ell/i}^i - \mathbf{v}_{t/i}^i \\ &= \mathbf{v}_{b/i}^i - \mathbf{v}_{t/i}^i, \end{aligned}$$

where have used the fact that $\mathbf{p}_{d/\ell}^\ell$ is constant, and the fact that the position and velocity of the body and body-level frame are identical.

If the target velocity in the body-level frame is known, and if $\mathbf{p}_{t/\ell}^\ell$ can be estimated, then the appropriate body velocity would be

$$\mathbf{v}_{b/i}^i = \mathbf{v}_{t/i}^i - K(\mathbf{p}_{d/\ell}^\ell - \mathbf{p}_{t/\ell}^\ell), \quad (10.16)$$

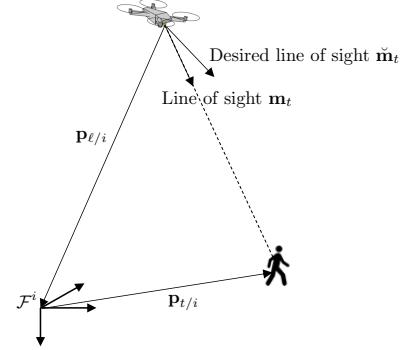


Figure 10.3: Definitions and geometry for target following.

where $K = K^\top > 0$ so that $\dot{\mathbf{p}}_{d/t}^\ell = -k\mathbf{p}_{d/t}^\ell$ implies that $\mathbf{p}_{d/t}^\ell(t) \rightarrow 0$.

Suppose that the pixel center of the object being tracked is transformed to the body-level frame according to Lemma 9.2.1, and is given by $\bar{\epsilon}_{t/\ell}^\ell$, then

$$\mathbf{p}_{t/\ell}^\ell = \lambda \bar{\epsilon}_{t/\ell}^\ell.$$

Suppose that the size S of the object being tracked is known, and that the (calibrated) size in the image ϵ_S can be estimated using image processing techniques, then from similar triangles we have $S/\lambda = \epsilon_S/1$, or

$$\lambda = \frac{S}{\epsilon_S}.$$

Therefore $\mathbf{p}_{t/\ell}^\ell$ can be estimated from image data. If the target velocity is not known, or cannot be estimated, then implement

$$\mathbf{v}_{b/i}^i = -R_\ell^i K(\mathbf{p}_{d/\ell}^\ell - \frac{S}{\epsilon_S} \bar{\epsilon}_{t/\ell}^\ell). \quad (10.17)$$

The orientation \mathbf{s}_ψ is currently a free variable, and we could think of different schemes to select the orientation. If the objective is to do a tail-chase, then \mathbf{s}_ψ should be selected to align with the target velocity $\mathbf{v}_{t/i}^i$.

If $\mathbf{v}_{t/i}^i$ is known, then \mathbf{s}_ψ can be selected by projecting $\mathbf{v}_{t/i}^i$ on to the North-East plane as $\Pi_{\mathbf{e}_3} \mathbf{v}_{t/i}^i$ and then normalizing to obtain

$$\mathbf{s}_\psi = \frac{\Pi_{\mathbf{e}_3} \mathbf{v}_{t/i}^i}{\|\Pi_{\mathbf{e}_3} \mathbf{v}_{t/i}^i\|}. \quad (10.18)$$

If $\mathbf{v}_{t/i}^i$ is not known, then it can be approximated from image data as follows. When $\mathbf{v}_{b/i}^i$ is selected as in Equation (10.17) then

$$\dot{\mathbf{p}}_{d/t}^i = -R_\ell^i K(\mathbf{p}_{d/\ell}^\ell - \mathbf{p}_{t/\ell}^\ell) - \mathbf{v}_{t/i}^i.$$

Therefore,

$$\begin{aligned} \mathbf{v}_{t/i}^i &= -R_\ell^i K(\mathbf{p}_{d/\ell}^\ell - \mathbf{p}_{t/\ell}^\ell) - \dot{\mathbf{p}}_{d/t}^i \\ &= -R_\ell^i \left[K(\mathbf{p}_{d/\ell}^\ell - \frac{S}{\epsilon_S} \bar{\epsilon}_{t/\ell}^\ell) - (\dot{\mathbf{p}}_{d/\ell}^\ell - \lambda \bar{\epsilon}_{t/\ell}^\ell - \lambda \dot{\bar{\epsilon}}_{t/\ell}^\ell) \right] \\ &= -R_\ell^i \left[K(\mathbf{p}_{d/\ell}^\ell - \frac{S}{\epsilon_S} \bar{\epsilon}_{t/\ell}^\ell) - \frac{S \dot{\epsilon}_S}{\epsilon_S^2} \bar{\epsilon}_{t/\ell}^\ell + \frac{S}{\epsilon_S} \dot{\bar{\epsilon}}_{t/\ell}^\ell \right]. \end{aligned}$$

The heading vector is then selected using Equation (10.22).

10.8 Target following 2.0

The tracking problem is best formulated in the body-level frame, as discussed in Section 9.2.

Recall that $\Pi_{\mathbf{x}} = I - \mathbf{x}\mathbf{x}^\top$. If \mathbf{x} is a unit vector, the $\Pi_{\mathbf{x}}$ is a projection matrix on to the plane orthogonal to \mathbf{x} .

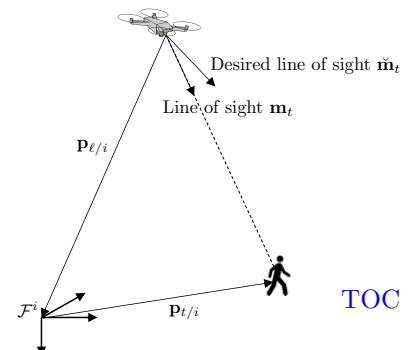


Figure 10.4: Definitions and geometry for target following.

Recall that in the body-level frame, the kinematics are given simply as

$$\dot{\mathbf{p}}_{\ell/i}^i = \mathbf{v}_{b/i}^i, \quad (10.19)$$

where $\mathbf{v}_{b/i}^i$ is considered as the input signal to the velocity controller, in addition to the heading vector $\mathbf{s}_\psi = R_\ell^i \mathbf{e}_1$.

The normalized line-of-sight vector between the multirotor and the target is given by

$$\mathbf{m}_t \triangleq \frac{\mathbf{p}_{t/i} - \mathbf{p}_{b/i}}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|}.$$

Note that the normalized line of sight vector can be computed from the calibrated pixel coordinates as

$$\mathbf{m}_t = \frac{\boldsymbol{\epsilon}_t}{\|\boldsymbol{\epsilon}_t\|}.$$

Recall from Property 5.1.3 that

$$\begin{aligned} \frac{d\mathbf{m}_t}{dt} &= \Pi_{\mathbf{m}_t} \frac{\dot{\mathbf{p}}_{t/i} - \dot{\mathbf{p}}_{b/i}}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|} \\ &= \Pi_{\mathbf{m}_t} \frac{\dot{\mathbf{p}}_{t/i} - \mathbf{v}_{b/i}}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|}, \end{aligned} \quad (10.20)$$

where $\Pi_{\mathbf{n}} = I - \mathbf{n}\mathbf{n}^\top$ is the projection operator onto the plane orthogonal to the normal vector \mathbf{n} .

The tracking objective is to maneuver the multirotor so that the normalized line-of-sight vector \mathbf{m}_t is equal to a desired normalized line-of-sight vector \mathbf{m}_d , as shown in Figure 10.4. For example, if the camera is fixed in the body frame, and is oriented so that it points down from the nose with an elevation angle of $-\alpha_0$ degrees, and the objective is to keep the target in the center of the field of view, then the desired normalized line-of-sight vector would be

$$\mathbf{m}_d = \begin{pmatrix} \cos \alpha_0 \\ 0 \\ \sin \alpha_0 \end{pmatrix}.$$

The intuition behind the control law introduced below is illustrated in Figure 10.5. The error between the line-of-sight vector \mathbf{m}_t and the desired line-of-sight vector \mathbf{m}_d can be expressed as

$$\boldsymbol{\nu}_1 = \Pi_{\mathbf{m}_d^v} \mathbf{m}_t^v$$

where $\boldsymbol{\nu}_1$ is the projection of \mathbf{m}_t onto the plane orthogonal to \mathbf{m}_d .

Note that when $\mathbf{m}_t = \mathbf{m}_d$, then $\boldsymbol{\nu}_1 = 0$.

We will assume for the moment that the multirotor will only be commanded to perform lateral motion. Therefore, projecting $\boldsymbol{\nu}_1$ onto the lateral plane gives the error vector

$$\boldsymbol{\nu} = \Pi_{\mathbf{e}_3} \Pi_{\mathbf{m}_d^v} \mathbf{m}_t^v.$$

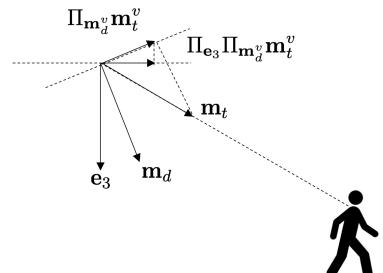


Figure 10.5: target following geometry

The multirotor velocity is selected to be proportional to this error as

$$\mathbf{v}_{b/i}^i = k \Pi_{\mathbf{e}_3} \Pi_{\mathbf{m}_d^v} \mathbf{m}_t^v. \quad (10.21)$$

Theorem 10.8.1 Suppose that the equations of motion for the multirotor are given in Equation (10.19), and that the target velocity satisfies $\mathbf{v}_{t/i}^i = 0$, and the vehicle velocity is given by Equation (10.21). If the altitude of the multi-rotor is greater than the altitude of the target, and if the desired line-of-sight vector points down, i.e., $\mathbf{e}_3^\top \mathbf{m}_d^v$, then $\mathbf{m}_t(t) \rightarrow \mathbf{m}_d$.

Proof: Consider the Lyapunov function candidate

$$\begin{aligned} V &= \frac{1}{2} \|\mathbf{m}_d \times \mathbf{m}_t\|^2 \\ &= \frac{1}{2} \|[\mathbf{m}_d]_\times \mathbf{m}_t\|^2 \\ &= \frac{1}{2} \mathbf{m}_t^\top [\mathbf{m}_d]_\times^\top [\mathbf{m}_d]_\times \mathbf{m}_t \\ &= -\frac{1}{2} \mathbf{m}_t^\top [\mathbf{m}_d]_\times^2 \mathbf{m}_t \\ &= \frac{1}{2} \mathbf{m}_t^\top \Pi_{\mathbf{m}_d} \mathbf{m}_t, \end{aligned}$$

where we have used the fact that $[\mathbf{x}]_\times^2 = \mathbf{x}\mathbf{x}^\top - \mathbf{x}^\top \mathbf{x}I$, implying that $[\mathbf{m}_d]_\times^2 = -\Pi_{\mathbf{m}_d}$. Taking the derivative of V gives

$$\begin{aligned} \dot{V} &= \mathbf{m}_t^\top \Pi_{\mathbf{m}_d} \dot{\mathbf{m}}_t \\ &= \mathbf{m}_t^\top \Pi_{\mathbf{m}_d} \Pi_{\mathbf{m}_t} \frac{\dot{\mathbf{p}}_{t/i} - \mathbf{v}_{b/i}}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|} \\ &= -\mathbf{m}_t^\top \Pi_{\mathbf{m}_d} \Pi_{\mathbf{m}_t} \frac{\mathbf{v}_{b/i}}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|} \\ &= -\frac{k}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|} \mathbf{m}_t^\top \Pi_{\mathbf{m}_d} \Pi_{\mathbf{m}_t} \Pi_{\mathbf{e}_3} \Pi_{\mathbf{m}_d^v} \mathbf{m}_t^v, \end{aligned}$$

where we have used Equations (10.20) and (10.21). Using the definition of $\Pi_{\mathbf{n}}$, after some algebra, we get that

$$\begin{aligned} \dot{V} &= -\frac{k}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|} \mathbf{m}_t^\top (I - \mathbf{m}_d \mathbf{m}_d^\top) (I - \mathbf{m}_t \mathbf{m}_t^\top) (I - \mathbf{e}_3 \mathbf{e}_3^\top) (I - \mathbf{m}_d \mathbf{m}_d^\top) \mathbf{m}_t^v, \\ &= -\frac{k}{\|\mathbf{p}_{t/i} - \mathbf{p}_{b/i}\|} (1 - (\mathbf{m}_t^\top \mathbf{m}_d)^2) (\mathbf{m}_t^\top \mathbf{m}_d) (1 + (\mathbf{e}_3^\top \mathbf{m}_t)(\mathbf{e}_3^\top \mathbf{m}_d)). \end{aligned}$$

If the altitude of the multirotor is greater than the altitude of the target, then $\mathbf{e}_3^\top \mathbf{m}_t > 0$. In addition, since both \mathbf{m}_t and \mathbf{m}_d point down, we have that $\mathbf{m}_t^\top \mathbf{m}_d > 0$. And since \mathbf{m}_t and \mathbf{m}_d are both unit vectors, Cauchy's inequality gives that $(\mathbf{m}_t^\top \mathbf{m}_d)^2 \leq 1$ where equality hold only when $\mathbf{m}_t = \mathbf{m}_d$. Therefore \dot{V} is negative definite, implying that $\mathbf{m}_t \rightarrow \mathbf{m}_d$. ■

The orientation \mathbf{s}_ψ is currently a free variable, and we could think of different schemes to select the orientation. If the objective is to do a tail-chase, then \mathbf{s}_ψ should be selected to align with the target velocity $\mathbf{v}_{t/i}^i$.

If $\mathbf{v}_{t/i}^i$ is known, then \mathbf{s}_ψ can be selected by projecting $\mathbf{v}_{t/i}^i$ on to the North-East plane as $\Pi_{\mathbf{e}_3}\mathbf{v}_{t/i}^i$ and then normalizing to obtain

$$\mathbf{s}_\psi = \frac{\Pi_{\mathbf{e}_3}\mathbf{v}_{t/i}^i}{\|\Pi_{\mathbf{e}_3}\mathbf{v}_{t/i}^i\|}. \quad (10.22)$$

Recall that $\Pi_{\mathbf{x}} = I - \mathbf{x}\mathbf{x}^\top$. If \mathbf{x} is a unit vector, the $\Pi_{\mathbf{x}}$ is a projection matrix on to the plane orthogonal to

*10.9 Autonomous Flight for Detection, Localization, and Tracking of Moving Targets with Small Quadrotor, RA-L, 2017
(in review)*

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

Autonomous Flight for Detection, Localization, and Tracking of Moving Targets with a Small Quadrotor

Justin Thomas, Jake Welde, Giuseppe Loianno, Kostas Daniilidis, and Vijay Kumar

Abstract—In this work, we address the autonomous flight of a small quadrotor, enabling tracking of a moving object. The 15 cm diameter, 250 g robot relies only on onboard sensors (a single camera and an inertial measurement unit) and computers, and can detect, localize, and track moving objects. Our key contributions include the relative pose estimate of a spherical target as well as the planning algorithm, which considers the dynamics of the underactuated robot, the actuator limitations, and the field of view constraints. We show simulation and experimental results to demonstrate feasibility and performance, as well as robustness to abrupt variations in target motion.

I. INTRODUCTION

Micro Aerial Vehicles (MAVs) equipped with on-board sensors are becoming ideal platforms for autonomous navigation in complex, confined environments for applications such as exploration [1], inspection [2], [3], mapping [4], interaction with the environment [5], and search and rescue [6]. For truly autonomous systems, in addition to autonomous navigation, it is necessary to provide MAVs with the ability to maneuver with respect to objects, which opens the door for additional applications. Most previous works leveraging aerial robots for observation of another object assume that the object of interest is static in the world frame. However, a static object is often not a valid assumption, such as when intercepting malicious aerial vehicles, tracking moving ground-based targets, or landing on a moving vehicle. Since localization methods such as GPS do not provide information about positioning relative to an object, researchers typically consider sensors such as cameras but often overlook the limited field-of-view (FOV) constraint. The FOV constraint has been mitigated by using creative methods such as an upward-facing camera [7] or by leveraging an omnidirectional camera [8], however, aerial robots are more likely to be equipped with downward-facing sensors than upward-facing ones, and omnidirectional cameras typically require cumbersome optics, which reduces agility and payload capacity. Further, most of these works do not model the dynamics of the target or even predict its path, limiting them to quasi-static scenarios.

Thus, the goal of this work is to relax the typical assumptions of a fixed target and an unconstrained field of view, enabling a quadrotor to track a moving target while considering the robot's limited field of view. The example

The authors are with the General Robotics Automation Sensing and Perception (GRASP) laboratory, University of Pennsylvania, Philadelphia {jut@seas, jwelde@seas, loiannog@seas, kostas@cis, kumar@seas}.upenn.edu



Fig. 1. A quadrotor accelerates to track a target moving at 1.5 m/s. The motion is planned in real time and considers the estimated trajectory of the target, the dynamics of the robot, the actuator limitations, and the camera's field of view. All sensing and computation occurs onboard the robot and leverages only one downward-facing camera and an onboard inertial measurement unit.

which will be explored is that of a robot tracking a moving sphere.

Visual odometry methods focus on determining the robot's pose relative to a starting location, not relative to a specific object, making them not directly applicable to our scenario. On the other hand, visual servoing solutions focus on motion relative to a specific object, but they typically require one or more of the following assumptions:

- A1) The dynamics of the robot are first-order.
- A2) The robot is fully actuated.
- A3) The image features are static features on the object.
- A4) The object is stationary in the inertial frame.
- A5) The object does not leave the field of view.

In this work, we overcome these assumptions as they relate to maneuvering a quadrotor relative to a moving target. To accomplish this, we extend our previous planning methods for high-speed grasping and perching [9], [10].

The closest works regarding visual servoing relative to spheres develop the control for a first-order robotic arm (A1 and A2) and assume the sphere is stationary (A4) [11], [12]. Circular markers have also been considered for visual servoing [13]. Our previous work considered servoing using a higher order underactuated system and did not require the image features to correspond to fixed locations on the target, an issue that arises when the object of interest has curvature [7]. While innovative design assisted in keeping the object in the field of view, there wasn't an explicit consideration for the field of view constraints, and

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

there was no way to recover if the object left the image.

There have also been more general approaches for landing or maneuvering multicopters relative to a moving target (A4). Landing on a small carrier vehicle was accomplished by leveraging an onboard IR camera (from a Wii Remote) and IR markers on the landing pad [14]. The very limited field of view (45°) was mitigated with a pan-tilt unit. Landing on a moving target using a downward-facing camera was explored in [15], but there was no explicit consideration for the field of view. Angular dynamics were ignored but enabled landing on a vertically moving target similar to a ship deck [16]. Conveniently, this approach only relied on optical flow of the surface, making the field of view constraint less of an issue. However, the normal of the surface was assumed to be known.

The most relevant work considers tracking a moving target with a quadrotor while avoiding obstacles [17]. The trajectory of the target is estimated, and trajectories are planned to minimize the position error. In this work, we consider a similar approach with some novel extensions. Specifically, we will consider a modified objective function which is more appropriate for aggressive scenarios, and we will incorporate the field of view constraints in the robot's trajectory planner.

This work makes multiple contributions. First, we provide a solution to identify a spherical object's position with respect to the aerial platform using monocular vision. Second, a trajectory planning method enables a robot to track a moving object in real time using the single downward-facing, body-fixed camera while explicitly considering the field of view. To the best of our knowledge, this is the first time that onboard navigation techniques based on a single camera and inertial measurement unit (IMU) are used both to navigate in the world and track a moving target without the need of an external motion capture system or additional onboard sensors.

The high-level goal to enable a robot to track or acquire a moving target will be broken down into a number of sub-tasks. First, the robot must be able to determine a relative pose, which will be discussed in Section II. Since the object may be moving, we also need to estimate its motion and propagate its dynamics in order to increase robustness to occlusions and potential failure when an object may temporarily leave the field of view. Thus, we model the object's and robot's dynamics as well as discuss the robot's controller in Section III. The planning strategy is proposed in Section IV. Finally, results are presented in Section V, and we conclude in Section VI.

II. RELATIVE POSE

In this section, we provide an approach to determine a relative pose from the image of a sphere. Specifically, we consider fitting a cone to a set of 3D points, making our approach agnostic to the camera model. Related works include fitting of ellipses and circles to points in a plane [18], [19]. Unless otherwise noted, vectors in this section will be expressed in the camera frame \mathcal{C} . Let a sphere be represented

by

$$\|\mathbf{X} - \mathbf{C}\|^2 - r^2 = 0 \quad (1)$$

where $\mathbf{C} \in \mathbb{R}^3$ is the center, r is the radius, $\mathbf{X} \in \mathbb{R}^3$ is any point on the surface, and $\|\cdot\|$ represents the Euclidean norm.

We define the projection operator π which maps a point \mathbf{X} in the camera frame to a point $\mathbf{x} \in \mathbb{R}^3$ on the image surface

$$\mathbf{x} = \pi(\mathbf{X}) \equiv \frac{1}{\lambda(\mathbf{X})} \mathbf{X} \quad (2)$$

where the choice of $\lambda : \mathbb{R}^3 \mapsto \mathbb{R}$ is dependent on the camera model. For example, we would choose $\lambda \equiv \|\mathbf{X}\|$ for a spherical camera model or $\lambda \equiv \mathbf{e}_3^T \mathbf{X}$, where $\mathbf{e}_3^T = [0 \ 0 \ 1]$, for a pinhole model. In any case, we can express $\mathbf{X} = \lambda \mathbf{x}$, allowing us to write (1) as

$$\|\lambda \mathbf{x} - \mathbf{C}\|^2 - r^2 = 0 \quad (3)$$

which is quadratic in λ . Considering that points on the contour of the projection represent rays which are tangent to the sphere, we require λ to be unique, which means that, for the contour, the discriminant of (3) must vanish so that

$$\mathbf{x}^T [\mathbf{C} \mathbf{C}^T + (r^2 - \mathbf{C}^T \mathbf{C}) \mathbf{I}] \mathbf{x} = 0, \quad (4)$$

where \mathbf{I} is the identity matrix, and we observe that (4) is a conic. The set of solutions for \mathbf{x} is known as the *tangent cone* to the sphere from the camera origin [20]. In this case, however, \mathbf{x} is an observed point, making this form not ideal for the rest of our formulation where we wish to determine \mathbf{C} .

A. A Geometric Solution

If we assume a spherical camera model for \mathbf{x} so that $\mathbf{x}^T \mathbf{x} = 1$, then (4) simplifies to

$$(\mathbf{x}^T \mathbf{C})^2 + r^2 - \mathbf{C}^T \mathbf{C} = 0. \quad (5)$$

Then, if we let $\mathbf{C} = \gamma \mathbf{c}$ with $\|\mathbf{c}\| = 1$ so that \mathbf{c} represents the bearing to the center of the target, we have

$$(\gamma \mathbf{x}^T \mathbf{c})^2 + r^2 - \gamma^2 = 0, \quad (6)$$

which is simply the Pythagorean constraint as discussed in Figure 2. Squared errors from this constraint can be captured in a non-linear minimization problem to estimate \mathbf{C}

$$\arg \min_{\mathbf{C}} \sum_{i=1}^n \left[(\mathbf{x}_i^T \mathbf{C})^2 + r^2 - \mathbf{C}^T \mathbf{C} \right]^2 \quad (7)$$

given n observed bearings in the tangent cone. This minimization can be seeded with the centroid as an initial guess, and a gradient descent can be used to determine the solution. With this formulation, the geometric error is minimized, however, the computational demands may be higher than is realistic for real-time implementation.

B. An Algebraic Solution

The conic fitting problem can also be approached algebraically, analogous to algorithms used for ellipse fitting [19]. For ease of comparison, we will use similar notation.

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

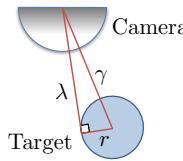


Fig. 2. The geometry for a cross section of a vector $\mathbf{X} = \lambda \mathbf{x}$ with $\|\mathbf{x}\| = 1$ in the tangent cone. The center of the sphere is given by $\mathbf{C} = \gamma \mathbf{c}$ with $\|\mathbf{c}\| = 1$. The top hemisphere represents the spherical camera model, and the circle is the cross section through a hemisphere of the sphere. Note that the Pythagorean theorem holds such that $\lambda^2 + r^2 = \gamma^2$ when a spherical camera model is used for both \mathbf{X} and \mathbf{C} .

1) *Fitting a Cone to a Set of Observations:* Let a conic be defined by

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = 0 \quad (8)$$

with $\mathbf{A} \in \mathbb{R}^{3 \times 3}$, $\mathbf{A} = \mathbf{A}^T$, and $\mathbf{x} = [x \ y \ z]^T$. Note that \mathbf{A} can be arbitrarily scaled and represent the same conic. The constraints for n observations \mathbf{x}_i can be written as

$$\underbrace{\begin{bmatrix} x_1^2 & y_1^2 & z_1^2 & 2x_1y_1 & 2x_1z_1 & 2y_1z_1 \\ x_2^2 & y_2^2 & z_2^2 & 2x_2y_2 & 2x_2z_2 & 2y_2z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & y_n^2 & z_n^2 & 2x_ny_n & 2x_nz_n & 2y_nz_n \end{bmatrix}}_D = \begin{bmatrix} A_{11} \\ A_{22} \\ A_{33} \\ A_{12} \\ A_{13} \\ A_{23} \end{bmatrix} = 0 \quad (9)$$

so that the linear system can be solved using a singular value decomposition (SVD) of D . From this, we can construct \mathbf{A} , which represents the best algebraic-fit conic.

2) *Extracting the Relative Pose from a Conic:* In a frame defined such that the z axis is parallel with the bearing to the target's center, our observations of the sphere boundary form a cone which can be written as

$$\mathbf{x}^T \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \mathbf{x} \equiv \mathbf{x}^T \Lambda \mathbf{x} = 0. \quad (10)$$

A change of basis can be applied to express the cone in camera frame coordinates with $\mathbf{A} \in \mathbb{R}^{3 \times 3}$, $\mathbf{A} = \mathbf{A}^T$

$$\mathbf{x}^T Q \Lambda Q^{-1} \mathbf{x} \equiv \mathbf{x}^T \mathbf{A} \mathbf{x} = 0. \quad (11)$$

Thus, we see that for a general conic defined by \mathbf{A} , we can use an eigenvalue decomposition to determine Q , λ_1 , λ_2 , and λ_3 . The axis of the cone in the camera frame is given by the eigenvector associated with the eigenvalue whose sign is the least common, which, without loss of generality, we can assume is λ_3 . This is also the axis along which the bearing to the target's centroid will lie. For a circular cone, the other two eigenvalues would be equal and have a sign opposing λ_3 . However, in practice, they will not be identical and can be approximated by their average, $\bar{\lambda}_{12}$. Then, using similar triangles, the distance to the centroid of the target can be determined to be $\gamma = r\sqrt{|\bar{\lambda}_{12}/\lambda_3|} + 1$.



Fig. 3. A sample image of a sphere comparing various detection algorithms. The purple frame indicates a simulated image boundary. Red circles denote detected boundary points of the sphere. In this case, more points were actually detected, but points outside of the simulated image boundary were discarded. The centroid of the remaining points is represented by the red pentagram. The computed bearing to the center of the sphere using the geometric solution is denoted by the blue diamond, and the computed bearing to the center of the sphere using the algebraic solution is denoted by the green square. In this case, the centroid does not accurately represent the bearing to the center of the sphere. The geometric solution is best, but it requires more computation time. (JW: The legend here could really be improved by being larger to help differentiate the markers and make the text more readable, if that's not a huge hassle)

C. Discussion

A comparison of these approaches is given in Figure 3. We first note that the geometric error provides the best fit but at the cost of the most computation time (about 23 ms). On the other hand, the algebraic approach is much faster (about 4 ms on the same machine) and, in most cases, provides a sufficiently accurate solution. In general, both solutions are superior to the centroid of the observed boundary points, especially when only part of the target is observed.

If we assume a planar camera model, the second approach remains valid, and we could still recover the information needed from an ellipse in the image plane. In many cases, this may be the preferred approach as there are readily available ellipse trackers such as vpMeEllipse from [21], and most image processing would occur in a flat image.

With these approaches, determining the relative pose from a single image requires knowledge of the sphere's radius. A scale has been estimated online using a Structure from Motion (SfM) approach, but the relative velocity was assumed to be known [22]. In our case, the target is not stationary and the magnitude of its velocity is unknown, so determining the scale online is a more difficult problem which will be left for future work.

III. DYNAMICS AND CONTROL

A. Dynamics of the Object

One of the unique differences between this work and other visual-servoing works is that we no longer require the target object to be fixed in the world. We simply require that the object's path can be approximated and predicted over some horizon using an n^{th} order polynomial in each Cartesian dimension.

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

B. Dynamics of the Robot

The dynamics of the robot are given by

$$m\ddot{\mathbf{x}} = -mg\mathbf{e}_3 + fR\mathbf{e}_3 \quad (12)$$

$$\dot{R} = R\dot{\Omega} \quad (13)$$

$$\mathcal{I}\dot{\Omega} = \mathbf{M} - \Omega \times \mathcal{I}\Omega \quad (14)$$

where m is the mass of the robot, $\mathbf{x} \in \mathbb{R}^3$ is the position, g is gravity, \mathbf{e}_3 is the 3rd standard basis vector, $f \in \mathbb{R}$ and $\mathbf{M} \in \mathbb{R}^3$ are the thrust and moment control inputs to the system, $R \in SO(3)$ is the orientation of the vehicle, and $\hat{\cdot} : \mathbb{R}^3 \mapsto \mathfrak{so}(3)$ is the hat map defined such that, for any two vectors, $\hat{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b}$. Also, \mathcal{I} is the inertial matrix, and Ω is the robot's angular velocity expressed in the robot frame.

C. Control

We leverage a position-based visual servoing (PBVS) control strategy, which allows the use of a common nonlinear controller [23], [24]. The position and velocity errors are

$$\mathbf{e}_x = \mathbf{x} - \mathbf{x}_{\text{des}} \quad \text{and} \quad \dot{\mathbf{e}}_x = \dot{\mathbf{x}} - \dot{\mathbf{x}}_{\text{des}}, \quad (15)$$

respectively, and the thrust is computed as

$$f = (-k_x\mathbf{e}_x - k_v\dot{\mathbf{e}}_x + mg\mathbf{e}_3 + m\ddot{\mathbf{x}}_{\text{des}}) \cdot R\mathbf{e}_3 \quad (16)$$

where k_x and k_v are positive gains and the subscript “des” denotes a desired value. The attitude and angular velocity errors are defined as

$$\mathbf{e}_R = \frac{1}{2} (R_{\text{des}}^T R - R^T R_{\text{des}})^{\vee}, \quad \mathbf{e}_\Omega = \Omega - R^T R_{\text{des}} \Omega_{\text{des}} \quad (17)$$

where $\cdot^{\vee} : \mathfrak{so}(3) \mapsto \mathbb{R}^3$ is the opposite of the hat map. The control moments are computed as

$$\mathbf{M} = -k_R\mathbf{e}_R - k_\Omega\mathbf{e}_\Omega + \Omega \times \mathcal{I}\Omega, \quad (18)$$

where k_R and k_Ω are positive gains. Then, the zero-equilibrium is exponentially stable and, in general, the controller provides “almost global exponential attractiveness” [24].

IV. PLANNING

Since we are interested in aggressive maneuvers so that the quadrotor can commence tracking a quickly moving target, it is important to not only consider dynamic feasibility (considering the relative degree), but also to ensure that actuator and sensor constraints, including the field of view, are not violated. The incorporation of some actuator and sensor constraints was demonstrated, enabling a robot to perform aggressive maneuvers to perch on vertical surfaces, but with no need to consider vision constraints [10], [25]. An extension to plan trajectories for image features was presented, but there were no guarantees that the trajectories would satisfy the sensor and actuator constraints [7]. The approach here will allow for the consideration of the dynamic, sensor, and actuator constraints, including the field of view.

A. Representation of Trajectories

We choose to express trajectories using an n^{th} order polynomial basis with terms $b_k(t)$ so that a trajectory $p(t)$ can be represented by

$$p(t) = \sum_{k=0}^n c_k b_k(t) \quad (19)$$

or with a vector of coefficients $\mathbf{c}_i \in \mathbb{R}^{n+1}$ for dimension i and a basis vector $\mathbf{b}(t) : \mathbb{R} \mapsto \mathbb{R}^{n+1}$

$$p_i(t) = \mathbf{c}_i^T \mathbf{b}(t). \quad (20)$$

We could allow $\mathbf{b}(t)$ to be a standard power basis,

$$\mathbf{b}(t) = [1 \ t \ t^2 \ \dots \ t^n]^T, \quad (21)$$

a Legendre Polynomial basis,

$$\mathbf{b}(t) = [1 \ t \ \frac{1}{2}(3t^2 - 1) \ \frac{1}{2}(5t^3 - 3t) \ \dots]^T, \quad (22)$$

or any basis of the user's choice. The r^{th} derivative can be computed as

$$p_i^{(r)}(t) = \mathbf{c}_i^T \mathbf{b}^{(r)}(t) \quad (23)$$

since \mathbf{c}_i is independent of time. Let $B(t) : \mathbb{R} \mapsto \mathbb{R}^{d(n+1) \times d}$ and $\mathbf{c} \in \mathbb{R}^{d(n+1)}$ be defined as

$$B(t) = \begin{bmatrix} \mathbf{b}(t) \\ \vdots \\ \mathbf{b}(t) \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_d \end{bmatrix} \quad (24)$$

for d dimensions. That is, \mathbf{c} is a stack of the coefficient vectors. Then, we can write the trajectory as $\mathbf{p}^{(r)}(t) : \mathbb{R} \mapsto \mathbb{R}^d$ where

$$\mathbf{p}^{(r)}(t) = (B^{(r)}(t))^T \mathbf{c}. \quad (25)$$

For clarity, we note that this is equivalent to

$$\mathbf{p}^{(r)}(t) = \begin{bmatrix} \mathbf{c}_1^T \\ \mathbf{c}_2^T \\ \vdots \\ \mathbf{c}_d^T \end{bmatrix} \mathbf{b}^{(r)}(t), \quad (26)$$

however, the previous formulation will be useful later.

B. Trajectory of the Target and Robot

The trajectory of the target in dimension i is defined by coefficients $\mathbf{h}_i \in \mathbb{R}^{n+1}$ so that the trajectory for all dimensions is

$$\mathbf{g}^{(r)}(t) = (B^{(r)}(t))^T \mathbf{h} \quad (27)$$

where $\mathbf{h} = [\mathbf{h}_1^T \ \dots \ \mathbf{h}_d^T]^T$. This allows us to fit a polynomial to the dynamic model over some horizon to predict the motion of the target [17]. We define the trajectory of the robot as

$$\mathbf{p}^{(r)}(t) = (B^{(r)}(t))^T \mathbf{c}, \quad (28)$$

which is the same notation outlined previously.

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

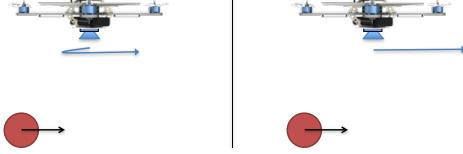


Fig. 4. A one-dimensional example motivating the minimization of velocity error between the robot and the target. In both scenarios pictured above, the robot is assumed to start from rest as the object enters the field of view. On the left hand side, we see a possible trajectory if the position error is minimized, and on the right-hand side, we see a result if the velocity error is minimized. In the position-error case (LHS), it is obvious that the motion is not ideal for larger target velocities. Thus, we are motivated, at least during the initial transient, to minimize the velocity error. Observe that this also could aid in mitigating the field of view constraints.

C. General Planning Strategy

One strategy is to minimize the position error between the target and the robot's trajectories with the inclusion of smoothing on higher derivatives [17]. However, this approach is not ideal in some scenarios (see discussion in Figure 4) because it can produce initial transients that are counterproductive to achieving tracking as quickly as possible.

Instead, we propose minimizing the velocity error during the initial transient. Interestingly, this results in a strategy similar to the ones taken by dragonflies [26], falcons [27], and human outfielders [28]. In these cases, the target is regulated to remain at a constant bearing in the field of view, and the range gap is closed using a strategy that may be captured by τ ("tau") theory [29], which doesn't require knowledge of a distance to the target. For now, we assume that the scale of our target is known a priori, allowing for the direct estimation of the range, and we leave the online range estimation, if necessary at all, for future work.

Minimizing velocity error alone, however, cannot capture desired relative pose. Thus, we consider the planning and tracking problem as having two phases. First, there must be a phase where the robot is accelerating to match the velocity of the target. This transient phase is when the field of view constraints are most likely to be active constraints. The objective of this phase can be expressed as a minimization of the velocity error between the robot and the target. The second phase incorporates the position error and enables planning to intercept the target or track the target from a desired relative pose.

D. A Multi-Objective Cost Function

We are motivated to use a multi-objective cost function to penalize both velocity errors and, when applicable, position errors. Interestingly, a similar approach was used to smoothly change formation shapes of an aerial robot team [30]. We define the error as

$$\mathbf{e}(t) = \mathbf{g}(t) - \mathbf{p}(t). \quad (29)$$

Since we're interested in minimizing specific derivatives, we write a general objective function which computes the

integrated square of the Euclidean error of the r^{th} derivative

$$\mathcal{J}_r = \int_{t_o}^{t_f} \|\mathbf{e}^{(r)}(t)\|^2 dt \quad (30)$$

where $\|\cdot\|$ represents the Euclidean norm. Expanding, we have

$$\mathcal{J}_r = \int_{t_o}^{t_f} (\mathbf{e}^{(r)})^T (\mathbf{e}^{(r)}) dt \quad (31)$$

$$= \mathbf{c}^T Q_r \mathbf{c} - 2\mathbf{h}^T Q_r \mathbf{c} + \mathbf{h}^T Q_r \mathbf{h} \quad (32)$$

where

$$Q_r = \int_{t_o}^{t_f} (B^{(r)}(t))^T (B^{(r)}(t)) dt, \quad (33)$$

so that \mathcal{J}_r can be expressed in quadratic form as

$$\mathcal{J}_r = \mathbf{c}^T Q_r \mathbf{c} + \mathbf{f}^T \mathbf{c} + \alpha, \quad \mathbf{f} = -2Q_r^T \mathbf{h}, \quad \alpha = \mathbf{h}^T Q_r \mathbf{h}. \quad (34)$$

Note that a translation could be included in the object's coefficients, \mathbf{h} , to specify a desired relative pose.

At this point, it is interesting to consider nondimensionalizing the trajectory so that $t_o = 0$ and $t_f = 1$. This would allow us to precompute Q_r for each derivative. An alternative approach is to plan all trajectories to be the same duration. In practice, we have found that numerical stability is best achieved by only non-dimensionalizing trajectories that are over 1 second in duration.

E. Actuator and Sensor Constraints

The field of view of a lens can be modeled as a cone in the camera (or body) frame

$$\mathbf{m}^T A \mathbf{m} \leq 0 \quad (35)$$

where $A = A^T$ and the solutions $\mathbf{m} \in \mathbb{R}^3$ are rays lying within the field of view. Unfortunately, this results in a non positive semidefinite constraint, which means that we can't include it as a quadratic constraint in a Quadratically Constrained Quadratic Program (QCQP). Alternatively, we could model the constraint with an inscribed pyramid similar to the approximation of a coulomb friction cone (see Figure 5 or [31]). Further, because of the rectangular sensor design of most cameras, the cone model may not be best. Thus, we can inscribe a convex pyramid in the field of view, which would provide a set of linear constraints representing the effective field of view.

We incorporate bounds to prevent actuator saturation using our previous approaches [10]. However, we also want to consider the field of view constraints so the object does not leave the image. The simplest way to solve this problem is to prescribe a maximum attitude angle (e.g., $\arccos(\mathbf{e}_3 \cdot R\mathbf{e}_3) \leq \beta_{max}$), and reduce the effective field of view accordingly. A trajectory could then be planned simultaneously using the maximum attitude constraint and the reduced field of view

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

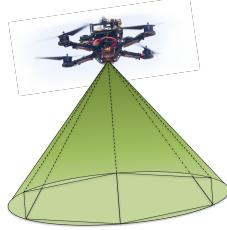


Fig. 5. The field of view of a lens. The cone's representation is not positive semidefinite, which means we cannot use it directly in a QCQP. However, the cone can be approximated with an inscribed pyramid.

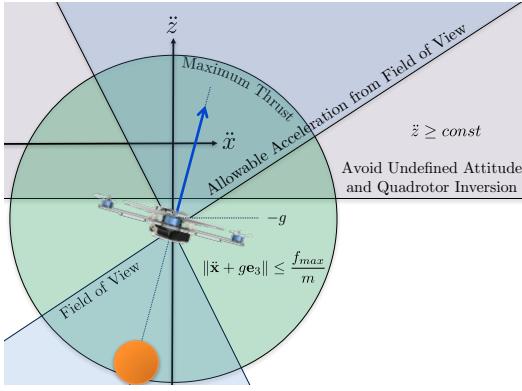


Fig. 6. A visualization of the acceleration constraints in the $x - z$ plane. The large circle represents the maximum thrust bound, and is similar to the illustration in [32]. The half plane bound keeps the quadrotor from inverting and avoids the singularity resulting when there is no thrust. Given the expected bearing between the target and the robot (inclined, dashed line), a conic (or pyramid) section represents the acceleration constraint to keep the object in the field of view.

to constrain the relative positions. However, this approach is more conservative than desired, especially when aggressive maneuvers are necessary. Certainly, we do not want to restrict the maximum attitude.

Instead, we directly incorporate the field of view as constraints in the optimization by using the relative position to rotate the field of view and prescribe constraints on the acceleration (effectively on the attitude). We first determine a set of acceleration constraints when the object would be directly beneath the robot. Then, we rotate the axis to be coincident with the expected bearing from the object to the robot. A schematic of sample acceleration constraints at one instant is given in Figure 6.

F. The Planner

In this subsection, we describe the proposed planning algorithm. We leverage a receding horizon planning strategy to continuously update the planned trajectory based on the target's actions. We first set the default objective function to minimize the velocity error and we include a slight weighting to penalize jerk, which helps to reduce the angular velocity and has been used previously to provide smoothing [17].

Note that we could also penalize the next derivative, snap, which would most directly help to reduce the angular acceleration. Dynamic, actuator, gyro, and vision constraints are incorporated as nonlinear constraints on the coefficients of the trajectory. Then, the problem can be solved using a Sequential Quadratic Program (SQP) solver. In our case, there is the benefit that instead of the QP subproblem being an approximation of a general nonlinear cost function, it is identical to our cost function. Most convergence arguments for SQP problems require that an initial solution or “warm start” is sufficiently close to the actual solution and that the active inequality constraints at the optimal solution are the same ones that are active at the local solution. For more details, we refer the reader to the “Sequential Quadratic Programming Methods” chapter in [33].

When the relative bearing and relative velocity have a positive inner product, then there is no harm in also minimizing position error (see the example in Figure 4). Additionally, there may be situations where the velocity is matched before the previous condition is satisfied. In such cases, we can also incorporate the position term in the objective function once the relative velocity falls below a predefined threshold. While this planning approach does not guarantee completeness since it is dependent on a non-linear optimization, it is viable for real-time applications, and it works well in practice. The algorithm is presented in Algorithm 1, where the weighting for derivative r is given by λ_r , and \mathcal{J}_r is defined by (34).

Algorithm 1 The Planning Algorithm

```

1:  $\mathcal{J} \leftarrow \lambda_1 \mathcal{J}_1 + \lambda_3 \mathcal{J}_3$ 
2: for Each Horizon do
3:   update( $\mathbf{g}(t)$ )
4:   repeat
5:      $\mathbf{p}(t) \leftarrow \text{iterateSQP}(\mathcal{J}, \mathbf{g}(t), \mathbf{p}(t))$ 
6:   until Out of Time
7:    $\mathbf{e}(t) \leftarrow \mathbf{g}(t) - \mathbf{p}(t)$ 
8:   if  $(\mathbf{e} \cdot \dot{\mathbf{e}} \geq 0 \text{ or } \|\dot{\mathbf{e}}\| \leq \text{thresh})$  then
9:      $\mathcal{J} \leftarrow \lambda_0 \mathcal{J}_0 + \lambda_1 \mathcal{J}_1 + \lambda_3 \mathcal{J}_3$ 
10:   end if
11: end for

```

V. RESULTS

In this section, we present our simulation and experimental results. We first present a sample simulation assuming a constant-velocity target starting at

$$\mathbf{x} = \begin{bmatrix} -2.5 \\ 1 \\ 0 \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$

with the robot's initial conditions given by

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad R = \mathbf{I}, \quad \Omega = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

The planning horizon is 1 second with a trajectory update frequency of 10 Hz. The desired relative pose is defined such

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

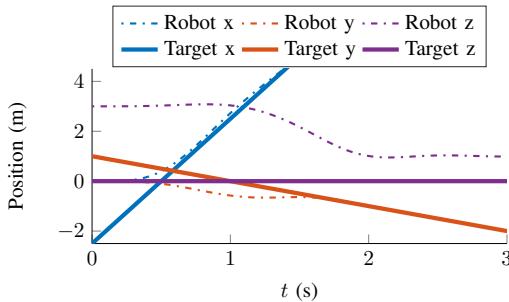


Fig. 7. The nominal positions from the proposed planning strategy.

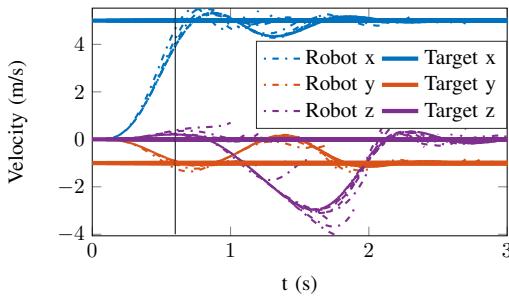


Fig. 8. The planned velocities resulting from the receding horizon planner. All planned trajectories are shown as dash-dotted lines, but only the first 0.1 s of any trajectory is executed before the next one is planned. The planning horizon is 1 s. We observe interesting results like the initial positive velocity in z , which is possibly a result of the field of view constraint. At $t = 0.6$ s (represented by the partial vertical line), the position error is included in the cost function.

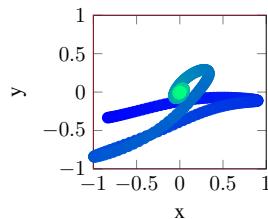


Fig. 9. The path of the bearing to the target in the image, starting on the left and ending up centered. The image boundary is given by the solid boundaries. We observe that the field of view constraints are not violated.

that the robot is 1 m above the target, and the horizontal and vertical fields of view are both assumed to be 90° . The resultant trajectory is plotted in Figure 7. Each new planned trajectory's velocity is plotted in Figure 8. The resultant path of the object in the image is plotted in Figure 9. We see some very exciting results. For example, despite the fact that there is no initial velocity error in the z direction, the robot accelerates upward, helping to keep the object in the field of view (Figure 8). With the same initial conditions and minimizing the position error from the start, the motion is quickly dominated by the visual constraints and results in an infeasible problem.

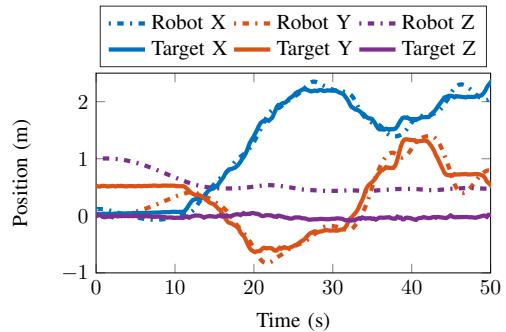


Fig. 10. Experimental results of a robot tracking a rolling sphere. The vehicle quickly locks onto the target and maintains the desired relative pose despite erratic motion in the target's path.

Next we present our experimental results, which were executed in the GRASP (General Robotics Automation Sensing and Perception) lab at the University of Pennsylvania. The total flying area has a volume of $20 \times 6 \times 4$ m 3 . We implemented the entire estimation, planning, and control pipeline, including the robot state estimator, the target's state estimator, a target trajectory predictor, and the trajectory planning algorithm on a Qualcomm Snapdragon™ board, featuring a Qualcomm Hexagon™ DSP and 802.11n WiFi, all packed onto a (58 × 40 mm) board based on the Snapdragon™ 801 processor. A Kalman filter is used for the robot's state estimation at 500 Hz with respect to a fixed reference frame. The estimation is obtained combining IMU data in the prediction phase and distinguishable environment landmarks in the measurement update step. For more details, we refer the reader to our previous work, where reliable flights with speeds up to 5 m/s and angular rates of 800 deg/s are achieved [34]. The quadrotor is only 250 g, and a single downward-facing camera and onboard IMU are used to perform both the state estimation of the robot and the target.

The target is a 7.6 cm diameter Sphero SPRK+ spherical robot, controllable using a smartphone via Bluetooth. A linear regression of the 10 most recent target pose measurements (at 30 Hz) is used to estimate the state of the Sphero assuming a constant-velocity model in the horizontal plane and a constant model in the z direction. Then, the future trajectory of the target is predicted, providing the vector of coefficients, $\mathbf{h}(t)$, for the next horizon.

With this platform, we demonstrate successful results such as the tracking in Figure 10 and the acceleration to track a quickly moving target entering the field of view in Figure 1. In both cases, the object does not leave the field of view, and the quadrotor successfully tracks the target. For more results, including simulations of the approach applied to objects moving in 3D, we refer the reader to the attached video or to the higher-resolution video at <http://www.jtwebs.net/2017-ra1/>.

CONFIDENTIAL. Limited circulation. For review only
IEEE RA-L submission 17-0352.1

VI. CONCLUSION

This work presented a relative pose estimation and trajectory planning strategy to track a moving sphere with an underactuated micro aerial vehicle while considering the dynamic, actuator, and field of view constraints. We validated that all perception and computation can occur onboard a 250 g robot equipped with only one downward-facing camera and an inertial measurement unit. Simulation and experimental results demonstrate successful tracking of a moving target while keeping the object in the field of view.

There are many exciting research opportunities stemming from this work. For example, an online scale or range estimate is needed, which could be inspired from the strategies used by dragonflies. Optical flow (considering the parallax) between the target and background could also help improve the estimate of the relative velocities. Finally, a better-informed strategy is needed to select the cost function weightings, λ_i .

REFERENCES

- [1] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 46–56, Sept 2012.
- [2] T. Ozaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Inspection of Penstocks and Featureless Tunnel-Like Environments using Micro UAVs," in *Field and Service Robotics Conference (FSR)*, Brisbane, Australia, 2013, pp. 123–136.
- [3] J. Cacace, A. Finzi, V. Lippiello, G. Loianno, and D. Sanzone, *Aerial Service Vehicles for Industrial Inspection: Task Decomposition and Plan Execution*. Springer Berlin Heidelberg, 2013, pp. 302–311.
- [4] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of mavs using rgb-d sensors," in *IEEE International Conference on Robotics and Automation*, May 2015, pp. 4021–4028.
- [5] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 57–64, Jan 2016.
- [6] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative Mapping of an Earthquake-Damaged Building via Ground and Aerial Robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.
- [7] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual Servoing of Quadrotors for Perching by Hanging From Cylindrical Objects," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 57–64, jan 2016.
- [8] J. Kim, Y. Jung, D. Lee, and D. H. Shim, "Outdoor Autonomous Landing on a Moving Platform for Quadrotors using an Omnidirectional Camera," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. Orlando, FL: IEEE, may 2014, pp. 1243–1252.
- [9] J. Thomas, J. Polin, K. Sreenath, and V. Kumar, "Avian-Inspired Grasping for Quadrotor Micro UAVs," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Portland: ASME, aug 2013, p. V06AT07A014.
- [10] J. Thomas, G. Loianno, M. Pope, E. W. Hawkes, M. A. Estrada, H. Jiang, M. R. Cutkosky, and V. Kumar, "Planning and Control of Aggressive Maneuvers for Perching on Inclined and Vertical Surfaces," in *International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. Boston: ASME, 2015, pp. 1–10.
- [11] R. T. Fomena and F. Chaumette, "Visual Servoing from Spheres using a Spherical Projection Model," in *IEEE International Conference on Robotics and Automation*, no. April. IEEE, apr 2007, pp. 2080–2085.
- [12] R. Fomena and F. Chaumette, "Improvements on Visual Servoing From Spherical Targets Using a Spherical Projection Model," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 874–886, aug 2009.
- [13] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision Based Position Control for MAVs Using One Single Circular Landmark," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1–4, pp. 495–512, jan 2011.
- [14] K. E. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 61, no. 1–4, pp. 221–238, 2011.
- [15] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2012, pp. 971–976.
- [16] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russo, "Landing a VTOL Unmanned Aerial Vehicle on a Moving Platform Using Optical Flow," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 77–89, feb 2012.
- [17] J. Chen, T. Liu, and S. Shen, "Tracking a Moving Target in Cluttered Environments Using a Quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 446–453.
- [18] W. Gander, G. H. Golub, and R. Strelbel, "Least-squares fitting of circles and ellipses," *BIT*, vol. 34, no. 4, pp. 558–578, dec 1994.
- [19] A. W. Fitzgibbon and R. B. Fisher, "A Buyer's Guide to Conic Fitting," *British Machine Vision Conference*, no. FEBRUARY, pp. 513–522, 1995.
- [20] O. Faugeras, *Three-dimensional Computer Vision: A Geometric Viewpoint*. Cambridge, MA, USA: MIT Press, 1993.
- [21] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics & Automation Magazine*, vol. 12, no. 4, pp. 40–52, dec 2005.
- [22] R. Spica, P. R. Giordano, and F. Chaumette, "Active structure from motion for spherical and cylindrical targets," in *IEEE International Conference on Robotics and Automation*. Hong Kong: IEEE, may 2014, pp. 5434–5440.
- [23] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*. IEEE, may 2011, pp. 2520–2525.
- [24] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *IEEE Conference on Decision and Control*. IEEE, dec 2010, pp. 5420–5425.
- [25] J. Thomas, M. Pope, G. Loianno, E. W. Hawkes, M. A. Estrada, H. Jiang, M. R. Cutkosky, and V. Kumar, "Aggressive Flight for Perching on Inclined Surfaces," *Journal of Mechanisms and Robotics*, vol. 8, no. 5, may 2016.
- [26] R. M. Olberg, A. H. Worthington, and K. R. Venator, "Prey pursuit and interception in dragonflies," *Journal of Comparative Physiology A: Sensory, Neural, and Behavioral Physiology*, vol. 186, no. 2, pp. 155–162, feb 2000.
- [27] S. A. Kane and M. Zamani, "Falcons pursue prey using visual motion cues: new perspectives from animal-borne cameras," *Journal of Experimental Biology*, vol. 217, no. 2, pp. 225–234, jan 2014.
- [28] M. McBeath, D. Shaffer, and M. Kaiser, "How baseball outfielders determine where to run to catch fly balls," *Science*, vol. 268, no. 5210, pp. 569–573, apr 1995.
- [29] L. Peper, R. J. Bootsma, D. R. Mestre, and F. C. Bakker, "Catching balls: how to get the hand to the right place at the right time," *Journal of experimental psychology. Human perception and performance*, vol. 20, no. 3, pp. 591–612, jun 1994.
- [30] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Autonomous Robots*, no. 2001, feb 2012.
- [31] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
- [32] M. Mueller and R. D'Andrea, "A model predictive controller for quadcopter state interception," in *European Control Conference (ECC)*, 2013, pp. 1383–1389.
- [33] "Mixed Integer Nonlinear Programming," in *Mixed Integer Nonlinear Programming*, ser. The IMA Volumes in Mathematics and its Applications, J. Lee and S. Leyffer, Eds. New York, NY: Springer New York, 2012, vol. 154, ch. Sequential, pp. 1–39.
- [34] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, April 2017.

Part III

Vision Based Estimation

11

Attitude Estimation

- overview of IMU sensors
 - complementary filter
 - EKF for Euler angles - include biases
 - MEKF using quaternions - attitude only

11.1 Inertial Measurement Unit (IMU)

11.1.1 Accelerometers

Acceleration transducers (accelerometers) typically employ a proof mass held in place by a compliant suspension as shown in Figure ???. When the casing around the accelerometer experiences an acceleration, the proof mass moves relative to the casing through a distance proportional to the acceleration. The acceleration experienced by the proof mass is converted to a displacement by the springs in the suspension. A simple force balance analysis of the proof mass yields the relationship

$$m\ddot{x} + kx = ky(t),$$

where x is the inertial position of the proof mass and $y(t)$ is the inertial position of the housing – the acceleration of which we want to sense. Given that the deflection of the suspension is $\delta = y - x$, this relation can be expressed as

$$\ddot{x} = \frac{k}{m}\delta.$$

Thus, the acceleration of the proof mass is proportional to the deflection of the suspension. At frequencies below the resonant frequency, the acceleration of the proof mass is the same as the acceleration of the housing. This can be seen by examining the transfer function from the housing position input to the proof mass position output

$$\frac{X(s)}{Y(s)} = \frac{1}{\frac{m}{k}s^2 + 1},$$

or equivalently, the transfer function from the housing acceleration input to the proof mass acceleration output

$$\frac{A_X(s)}{A_Y(s)} = \frac{1}{\frac{m}{k}s^2 + 1}.$$

At frequencies corresponding to $\omega \ll \sqrt{k/m}$, the transfer function $A_X(s)/A_Y(s) \approx 1$ and the displacement of the proof mass is an accurate indicator of the acceleration of the body to which the accelerometer is attached.

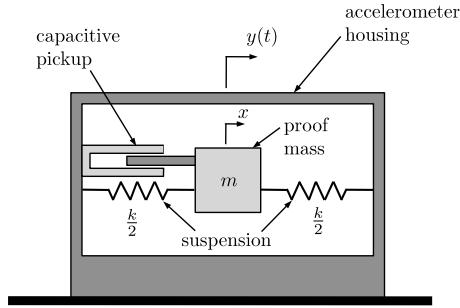


Figure 11.1: Conceptual depiction of MEMS accelerometer.

The accelerometer in Figure ?? is shown with a capacitive transducer to convert the proof mass displacement into a voltage output as is common in many MEMS devices. Other approaches to convert the displacement to a usable signal include piezoelectric, reluctive, and strain-based designs. As with other analog devices, accelerometer measurements are subject to signal bias and random uncertainty. The output of an accelerometer can be modeled as

$$Y_{\text{accel}} = k_{\text{accel}}A + \beta_{\text{accel}} + \eta'_{\text{accel}},$$

where Y_{accel} is in volts, k_{accel} is a gain, A is the acceleration in meters per second squared, β_{accel} is a bias term, and η'_{accel} is zero-mean Gaussian noise. The gain k_{accel} may be found on the data sheet of the sensor. Due to variations in manufacturing, however, it is imprecisely known. A one-time lab calibration is usually done to accurately determine the calibration constant, or gain, of the sensor. The bias term β_{accel} is dependent on temperature and should be calibrated prior to each flight.

For quadrotors, three accelerometers are required to measure the specific acceleration of the body. The accelerometers are mounted near the center of mass of the vehicle, with the sensitive axes the accelerometer aligned with the body axes. Accelerometers measure the specific force in the body frame of the vehicle. Another interpretation is that they measure the difference between the acceleration of the UAV and the gravitational acceleration. To understand this phenomena, imagine that the device shown in Figure ?? were to be turned

ninety degrees and set on a table. The forces acting on the casing will be gravity pulling down, and an equal and opposite normal force pushing up to keep the casing on the table. Therefore, the total acceleration on the casing will be zero. However, since the normal force of the table does not act on the proof mass, it will deflect under the force of gravity and the sensor will measure an acceleration equal to one g . Therefore the measured acceleration is the total acceleration of the casing minus gravity.

If \mathbf{a}^b is the specific acceleration measured in the body frame, and \mathbf{f}^b is the total on the quadrotor as measured in the body frame, then

$$\mathbf{a}^b = \frac{1}{m}\mathbf{f}^b - gR_i^b\mathbf{e}_3,$$

where g is the force of gravity of a unit mass at sea level, and $\mathbf{e}_3 = (0, 0, 1)^\top$.

The voltage output of the accelerometer is sampled by an analog-to-digital converter at a sample rate T_s . Through calibration, this voltage can be converted to a numerical representation of the acceleration in meters per second squared. The measured output of the accelerometer is therefore given by

$$\begin{aligned} \mathbf{y}_{\text{accel}} &= \mathbf{a}^b + \mathbf{b}_{\text{accel}} + \boldsymbol{\eta}_{\text{accel}} \\ &= \frac{1}{m}\mathbf{f}^b - gR_i^b\mathbf{e}_3 + \mathbf{b}_{\text{accel}} + \boldsymbol{\eta}_{\text{accel}}, \end{aligned} \quad (11.1)$$

where $\mathbf{b}_{\text{accel}}$ represents the bias, and $\boldsymbol{\eta}_{\text{accel}}$ represents the sensor noise, which is typically modeled as a zero mean Gaussian random process.

As shown in Chapter ?? the equations of motion for the velocity of a quadrotor are given by

$$\begin{aligned} \dot{\mathbf{v}}^b &= \mathbf{v}^b \times \boldsymbol{\omega}_{b/i}^b + \frac{1}{m}\mathbf{f}^b \\ &= \mathbf{v}^b \times \boldsymbol{\omega}_{b/i}^b + gR_i^b\mathbf{e}^3 - \frac{T}{m}\mathbf{e}_3 - \frac{\mu}{m}\Pi_{\mathbf{e}_3}\mathbf{v}^b, \end{aligned} \quad (11.2)$$

where T is the throttle and $\Pi_{\mathbf{e}_3} = I - \mathbf{e}_3\mathbf{e}_3^\top$. Therefore, the accelerometer measurement can be expressed as

$$\mathbf{y}_{\text{accel}} = -\frac{T}{m}\mathbf{e}_3 - \frac{\mu}{m}\Pi_{\mathbf{e}_3}\mathbf{v}^b + \mathbf{b}_{\text{accel}} + \boldsymbol{\eta}_{\text{accel}}.$$

Alternatively, Equation (11.2) implies that $\frac{1}{m}\mathbf{f}^b = \dot{\mathbf{v}}^b - \mathbf{v}^b \times \boldsymbol{\omega}_{b/i}^b$ and therefore, the output of the accelerometer can be expressed as

$$\mathbf{y}_{\text{accel}} = \dot{\mathbf{v}}^b - \mathbf{v}^b \times \boldsymbol{\omega}_{b/i}^b - gR_i^b\mathbf{e}_3 + \mathbf{b}_{\text{accel}} + \boldsymbol{\eta}_{\text{accel}}. \quad (11.3)$$

In subsequent sections and chapters, we will find both expressions to be useful.

11.1.2 Rate Gyros

MEMS rate gyros typically operate based on the principle of the Coriolis acceleration. In the early 19th century, French scientist G.G. de Coriolis discovered that a point translating on a rotating rigid body experiences an acceleration, now called Coriolis acceleration, that is proportional to the velocity of the point and the rate of rotation of the body

$$\mathbf{a}_C = 2\boldsymbol{\Omega} \times \mathbf{v}, \quad (11.4)$$

where $\boldsymbol{\Omega}$ is the angular velocity of the body in an inertial reference frame, and \mathbf{v} is the velocity of the point in the reference frame of the body.

MEMS rate gyros commonly consist of a vibrating proof mass as depicted in Figure ???. In this figure, the cantilever and proof mass are actuated at their resonant frequency to cause oscillation in the vertical plane. The cantilever is actuated so that the velocity of the proof mass due to these oscillations is a constant amplitude sinusoid

$$v = A\omega_n \sin(\omega_n t),$$

where A is the amplitude of the oscillation and ω_n is the natural frequency of the oscillation. If the sensitive axis of the rate gyro is configured to be the longitudinal axis of the undeflected cantilever, then rotation about this axis will result in a Coriolis acceleration in the horizontal plane described by Equation (11.4) and shown in Figure ???. Similar to the accelerometer, the Coriolis acceleration of the proof mass results in a lateral deflection of the cantilever. This lateral deflection of the cantilever can be detected in several ways: by capacitive coupling, through a piezoelectrically generated charge, or through a change in piezoresistance of the cantilever. Whatever the transduction method, a voltage proportional to the lateral Coriolis acceleration is produced.

With the sensing axis orthogonal to direction of vibration, the ideal output voltage of the rate gyro is proportional to the amplitude of Coriolis acceleration, and is given by

$$\begin{aligned} V_{\text{gyro}} &= k_C |\mathbf{a}_C| \\ &= 2k_C |\boldsymbol{\Omega} \times \mathbf{v}|. \end{aligned}$$

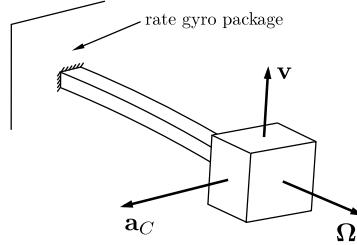
Since $\boldsymbol{\Omega}$, the angular rate of rotation about the sensitive axis of the gyro, and \mathbf{v} are orthogonal

$$|\boldsymbol{\Omega} \times \mathbf{v}| = \Omega |\mathbf{v}|,$$

and

$$\begin{aligned} V_{\text{gyro}} &= 2k_C \Omega |A\omega_n \sin(\omega_n t)| \\ &= 2k_C A\omega_n \Omega \\ &= K_C \Omega, \end{aligned}$$

where K_C is a calibration constant and Ω represents the magnitude and direction (sign) of the angular velocity about the sensitive axis.



The output of a rate gyro can be modeled as

$$Y_{\text{gyro}} = k_{\text{gyro}} \Omega + \beta_{\text{gyro}} + \eta'_{\text{gyro}},$$

where Y_{gyro} corresponds to the measured rate of rotation in volts, k_{gyro} is a gain converting the rate in radians per second to Volts, Ω is the angular rate in radians per second, β_{gyro} is a bias term, and η'_{gyro} is zero mean Gaussian noise. An approximate value for the gain k_{gyro} should be given on the spec sheet of the sensor. To ensure accurate measurements, the value of this gain should be determined through experimental calibration. The bias term β_{gyro} is strongly dependent on temperature and should be calibrated prior to each flight. For low-cost MEMS gyros, drift in this bias term can be significant and so it should be estimated during flight and subtracted from the measurement.

If the sensitive axes of the gyroscope are aligned with the body axes of the UAS, then the measured output of the gyro is given by

$$\mathbf{y}_{\text{gyro}} = \boldsymbol{\omega}_{b/i}^b + \mathbf{b}_{\text{gyro}} + \boldsymbol{\eta}_{\text{gyro}}, \quad (11.5)$$

where \mathbf{b}_{gyro} represents the bias, and $\boldsymbol{\eta}_{\text{gyro}}$ represents the sensor noise, which is typically modeled as a zero mean Gaussian random process.

11.1.3 Magnetometer

The earth's magnetic field has been used as a navigational aid for centuries. The first magnetic compasses are believed to have originated with the Chinese around the first century A.D.. Compasses appeared in Europe around the 11th century A.D. and were used by Christopher Columbus and other world explorers in the late 15th century. The

Figure 11.2: Conceptual depiction of proof mass rate gyro. Ω is the angular velocity of the sensor package to be measured. v is the actuated vibration velocity of the cantilever. a_C is the Coriolis acceleration that results as the sensor package undergoes an angular velocity.

earth's magnetic field continues to provide a means for navigation for a variety of vehicles, including unmanned aircraft.

The magnetic field around the earth behaves similarly to that of a common magnetic dipole with the magnetic field lines running normal to the earth's surface at the poles and parallel to the earth's surface near the equator. Except near the poles, the earth's magnetic field points to magnetic North. A compass measures the direction of the magnetic field locally and provides an indication of heading relative to magnetic North, ψ_m . This is depicted schematically in Figure ???. The declination angle δ is the angle between true North and magnetic North.

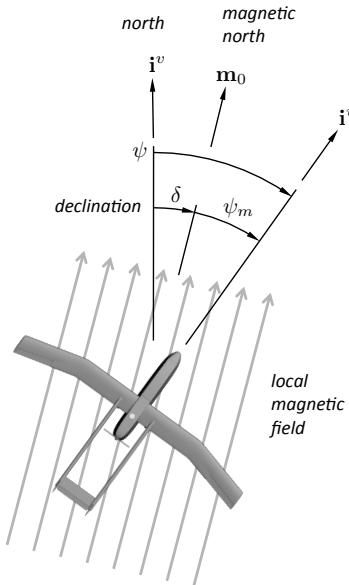


Figure 11.3: Magnetic field and compass measurement.

The earth's magnetic field is three dimensional, with North, East, and Down components that vary with location along the earth's surface. For example, in Provo, Utah, the North component (X) of the magnetic field is 21,053 nT, the East component (Y) is 4520 nT, and the Down component (Z) is 47,689 nT. The declination angle is 12.12 degrees. Figure ?? shows the declination angle over the surface of the earth and illustrates the significant dependence of the magnetic North direction on location. The inclination angle is the angle that the magnetic field makes with the horizontal plane. In Provo, the inclination is 65.7 degrees.

The vector corresponding to the magnetic field at any location can be determined using, for example, the World Magnetic Model (WMM), available from the National Geophysical Data Center (NGDC) ¹. Let \mathbf{m}^i be the value of this vector resolved in inertial coordinates. A magnetometer will measure \mathbf{m}^i resolved in body coordinates, in addi-

tion to a bias term, that may be introduced due to magnetic material in the environment, as well as noise. Therefore, the output of the magnetometer is given by

$$\mathbf{y}_{\text{mag}} = R_i^b \mathbf{m}^i + \mathbf{b}_{\text{mag}} + \boldsymbol{\eta}_{\text{mag}}, \quad (11.6)$$

where the bias term \mathbf{b}_{mag} is assumed to be constant, and $\boldsymbol{\eta}_{\text{mag}}$ is a zero mean Gaussian random process.

11.2 Wahba's Problem

Estimating the attitude of the quadrotor in frame \mathcal{F}^b relative to frame \mathcal{F}^a will require that we can measure a set of unit vector $\{\mathbf{u}_i\}_{i=1}^N$ in both frames. Given measurements of $\{\mathbf{u}_i^a\}_{i=1}^N$ and $\{\mathbf{u}_i^b\}_{i=1}^N$, Wahba's problem is to determine the rotation matrix $R_a^b \in SO(3)$ to minimize

$$J(R_a^b) = \sum_{i=1}^N \alpha_i \left\| \mathbf{u}_i^b - R_a^b \mathbf{u}_i^a \right\|_2,$$

where α_i are weights. When $N = 1$ the problem is underdetermined and a one-dimensional family of rotations can be found. When $N = 2$ and the unit vectors are not co-linear, a unique rotation can be found. When $N > 2$, R_a^b is interpreted as a least squares match that best fits the available data. In the following sections we will present solutions for Wahba's problem in the case when $N = 1$ and when $N \geq 2$.

11.2.1 Single vector measurement

Recall from Equation (??) that a rotation about a unit vector \mathbf{n} by an angle of μ is given by

$$R_{\mathbf{n}, \mu} = I + (\sin \mu) \mathbf{n}^\wedge + (1 - \cos \mu) (\mathbf{n}^\wedge)^2.$$

Given the unit vectors \mathbf{u}_1^a and \mathbf{u}_1^b , the axis of rotation that rotates \mathbf{u}_1^a by a right-handed rotation into \mathbf{u}_1^b is given by

$$\mathbf{n} = \frac{\mathbf{u}_1^a \times \mathbf{u}_1^b}{\|\mathbf{u}_1^a \times \mathbf{u}_1^b\|}.$$

The angle of rotation is given $\mu = \cos^{-1}(\mathbf{u}_1^{a\top} \mathbf{u}_1^b)$. Therefore, a rotation matrix that rotates \mathbf{u}_1^a into \mathbf{u}_1^b is given by

$$\begin{aligned} R_a^b &= I + \sin \left(\cos^{-1}(\mathbf{u}_1^{a\top} \mathbf{u}_1^b) \right) \left(\frac{\mathbf{u}_1^a \times \mathbf{u}_1^b}{\|\mathbf{u}_1^a \times \mathbf{u}_1^b\|} \right)^\wedge + \left(1 - \mathbf{u}_1^{a\top} \mathbf{u}_1^b \right) \left(\frac{\mathbf{u}_1^a \times \mathbf{u}_1^b}{\|\mathbf{u}_1^a \times \mathbf{u}_1^b\|} \right)^\wedge \\ &= I + \sin \left(\cos^{-1}(\mathbf{u}_1^{a\top} \mathbf{u}_1^b) \right) \left(\frac{\mathbf{u}_1^a \times \mathbf{u}_1^b}{\sin(\cos^{-1}(\mathbf{u}_1^{a\top} \mathbf{u}_1^b))} \right)^\wedge + \left(1 - \mathbf{u}_1^{a\top} \mathbf{u}_1^b \right) \left(\frac{\mathbf{u}_1^a \times \mathbf{u}_1^b}{\sin(\cos^{-1}(\mathbf{u}_1^{a\top} \mathbf{u}_1^b))} \right)^\wedge \\ &= I + (\mathbf{u}_1^a \times \mathbf{u}_1^b)^\wedge + \left(\frac{1 - \mathbf{u}_1^{a\top} \mathbf{u}_1^b}{1 + (\mathbf{u}_1^{a\top} \mathbf{u}_1^b)^2} \right) (\mathbf{u}_1^a \times \mathbf{u}_1^b)^\wedge, \end{aligned}$$

where we have used the fact that $\|\mathbf{u} \times \mathbf{v}\| = \|\mathbf{u}\| \|\mathbf{v}\| \sin \theta$, where θ is the angle between \mathbf{u} and \mathbf{v} , and the fact that $\sin(\cos^{-1} x) = \sqrt{1 - x^2}$ when $|x| \leq 1$.

Since any subsequent rotation about \mathbf{u}_1^b does not change \mathbf{u}_1^b , the one-dimensional family of rotation matrices parameterized by the angle μ is given by

$$\begin{aligned} R_a^b(\mu) &= \left(I + (\sin \mu) \mathbf{u}_1^{b\wedge} + (1 - \cos \mu) (\mathbf{u}_1^{b\wedge})^2 \right) \cdot \\ &\quad \left(I + (\mathbf{u}_1^a \times \mathbf{u}_1^b)^\wedge + \left(\frac{1 - \mathbf{u}_1^{a\top} \mathbf{u}_1^b}{1 + (\mathbf{u}_1^{a\top} \mathbf{u}_1^b)^2} \right) (\mathbf{u}_1^a \times \mathbf{u}_1^b)^\wedge \right). \end{aligned}$$

11.2.2 Multiple vector measurements

11.3 Complementary Filter

11.3.1 Model Free Complementary Filter

The objective of the complementary filter described in this section is to produce estimates of the Euler angles ϕ , θ , and ψ , when the roll angle ϕ and the pitch angle θ are small. The complementary filter fuses two types of measurements. The first measurement for each angle comes from the rate gyros which measure the angular rates plus a bias. From Equation (??), we see that when ϕ and θ are small that

$$\begin{aligned} \dot{\phi} &\approx p \\ \dot{\theta} &\approx q \\ \dot{\psi} &\approx r, \end{aligned}$$

where $\omega_{b/i}^b = (p, q, r)^\top$ is the angular rate of the vehicle resolved in the body frame. Resolving Equation (11.5) along each body axes we get

$$\begin{aligned} y_{gyro,x} &= p + b_p + \eta_p \\ y_{gyro,y} &= q + b_q + \eta_q \\ y_{gyro,z} &= r + b_r + \eta_r, \end{aligned}$$

where b_* is a slowly varying bias term, and η_* is a zero mean Gaussian random variable with known variance. Integrating the output of the

rate gyros gives

$$\begin{aligned}\hat{\phi}_{gyro}(t) &\triangleq \int_{-\infty}^t y_{gyro,x}(\tau) d\tau = \phi(t) + \int_{-\infty}^t (b_p(\tau) + \eta_p(\tau)) d\tau \\&= \phi(t) + \beta_\phi(t) \\ \hat{\theta}_{gyro}(t) &\triangleq \int_{-\infty}^t y_{gyro,y}(\tau) d\tau = \theta(t) + \int_{-\infty}^t (b_q(\tau) + \eta_q(\tau)) d\tau \\&= \theta(t) + \beta_\theta(t) \\ \hat{\psi}_{gyro}(t) &\triangleq \int_{-\infty}^t y_{gyro,z}(\tau) d\tau = \psi(t) + \int_{-\infty}^t (b_r(\tau) + \eta_r(\tau)) d\tau \\&= \psi(t) + \beta_\psi(t),\end{aligned}$$

where $\beta_*(t)$ are slowly varying signals, or signals with low frequency content.

The second measurement that will be used in the model-free complementary filter either comes from the accelerometers in the case of the roll and pitch angles, or from a magnetometer, in the case of the yaw angle. Letting $\mathbf{v}^b = (u, v, w)^\top$, using Equation (??) for the rotation matrix in terms of the Euler angles, and resolving Equation (11.3) along each body axis, we get

$$\begin{aligned}y_{accel,x} &= \dot{u} + qw - rv + g \sin \theta + b_x + \eta_x \\y_{accel,y} &= \dot{v} + ru - pw + g \cos \theta \sin \phi + b_y + \eta_y \\y_{accel,z} &= \dot{w} + pv - qu + g \cos \theta \cos \phi + b_z + \eta_z,\end{aligned}$$

where b_* are slowly varying biases, and η_* represent zero mean Gaussian noise. If the bias terms are known, for example through pre-flight calibration, then the roll and pitch angles can be approximated as

$$\begin{aligned}\hat{\phi}_{accel}(t) &\triangleq \tan^{-1} \left(\frac{y_{accel,y} - b_y}{y_{accel,z} - b_z} \right) = \tan^{-1} \left(\frac{\dot{v} + ru - pw + g \cos \theta \sin \phi + \eta_y}{\dot{w} + pv - qu + g \cos \theta \cos \phi + \eta_z} \right) \\&= \phi(t) + \nu_\phi(t) + \eta_\phi \\ \hat{\theta}_{accel}(t) &\triangleq \sin^{-1} \left(\frac{y_{accel,x} - b_x}{g} \right) = \sin^{-1} \left(\frac{\dot{u} + qw - rv + g \sin \theta + \eta_x}{g} \right) \\&= \theta(t) + \nu_\theta(t) + \eta_\theta.\end{aligned}$$

The approximation of ϕ and θ given by the accelerometers will be most accurate when the vehicle is not accelerating, i.e., when $\dot{u} + qw - rv = \dot{v} + ru - pw = \dot{w} + pv - qu = 0$. Since these terms are high frequency signals that are linked through the dynamics to the true roll and pitch angles, ϕ_{accel} and θ_{accel} are only good approximations at low frequencies. Therefore, we assume that ν_ϕ and ν_θ are signals with high frequency content. We note that this assumption is violated in many flight scenarios for fixed wing aircraft like when the vehicle is in a fixed loiter configuration where ν_ϕ and ν_θ are constant.

In the case of the magnetometer, the measurement is first processed by subtracting any known biases, rotating to remove the inclination and declination angles, and then rotating to the body level frame as

$$\mathbf{m}^v = R_b^v(\phi, \theta) R_z^\top(\delta) R_y^\top(\iota)(\mathbf{y}_{\text{mag}} - \mathbf{b}_{\text{mag}}) \quad (11.7)$$

$$= R_b^v(\phi, \theta) R_y(\iota) R_z(\delta) (\mathbf{m}^b + \boldsymbol{\nu}_{\text{mag}} + \boldsymbol{\eta}_{\text{mag}}) \quad (11.8)$$

$$= \begin{pmatrix} c_\theta & s_\theta s_\phi & s_\theta c_\phi \\ 0 & c_\phi & -s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \begin{pmatrix} c_\delta & s_\delta & 0 \\ -s_\delta & c_\delta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_\iota & 0 & s_\iota \\ 0 & 1 & 0 \\ -s_\iota & 0 & c_\iota \end{pmatrix} (\mathbf{m}^b + \boldsymbol{\nu}_{\text{mag}} + \boldsymbol{\eta}_{\text{mag}}), \quad (11.9)$$

where δ is the declination angle, ι is the inclination angle, $\boldsymbol{\eta}$ is zero mean Gaussian noise, and $\boldsymbol{\nu}$ represents all other magnetic interference that comes from, for example, the motor of the vehicle or flying over power lines, etc. The estimate of the heading ψ is therefore given by

$$\begin{aligned} \hat{\psi}_{\text{mag}}(t) &= -\text{atan}2(m_y^v, m_x^v) \\ &= \psi(t) + \nu_\psi(t) + \eta_\psi. \end{aligned}$$

We will assume that ν_ψ is a high frequency signal, and therefore, that a low pass filtered version of $\hat{\psi}_{\text{mag}}$ is essentially ψ over low frequencies.

We will present the derivation of the simple complementary filter for the roll angle ϕ . The derivation for the pitch and yaw angles is similar. The intuitive idea of the complementary filter is to estimate the roll angle by blending a high pass version of $\hat{\phi}_{\text{gyro}}$ and a low pass version of $\hat{\phi}_{\text{accel}}$ as

$$\hat{\phi} = H_{\text{HPF}}(s)\hat{\phi}_{\text{gyro}} + H_{\text{LPF}}(s)\hat{\phi}_{\text{accel}}$$

where $H_{\text{HPF}}(s)$ is a high pass filter and $H_{\text{LPF}}(s)$ is a low pass filter. Since

$$\begin{aligned} \hat{\phi} &= H_{\text{HPF}}(s)\hat{\phi}_{\text{gyro}} + H_{\text{LPF}}(s)\hat{\phi}_{\text{accel}} \\ &= H_{\text{HPF}}(s)[\phi + \beta_\phi] + H_{\text{LPF}}(s)[\phi + \nu_\phi + \eta_\phi] \\ &= [H_{\text{HPF}}(s) + H_{\text{LPF}}(s)]\phi + H_{\text{HPF}}(s)\beta_\phi + H_{\text{LPF}}(s)[\nu_\phi + \eta_\phi], \end{aligned}$$

if the frequency content of β_ϕ is below the cut-off frequency of H_{HPF} and the frequency content of $\nu_\phi + \eta_\phi$ is above the cut-off frequency of H_{LPF} then

$$\hat{\phi} = [H_{\text{HPF}}(s) + H_{\text{LPF}}(s)]\phi,$$

which implies that the filters H_{HPF} and H_{LPF} need to be selected so that

$$H_{\text{HPF}}(s) + H_{\text{LPF}}(s) = 1.$$

For example, if $H_{\text{LPF}}(s) = \frac{k_p}{s+k_p}$ then we need to select $H_{\text{HPF}}(s) = \frac{s}{s+k_p}$. The block diagram for a naive implementation of the complementary filter is shown in Figure ??.

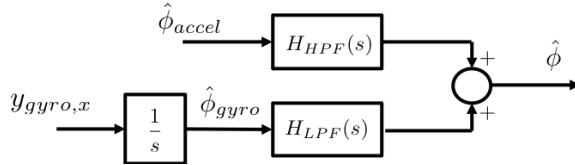


Figure 11.4: Naive complementary filter for roll angle estimation

The implementation of the complementary filter shown in Figure ?? has several drawbacks that include the need to implement two filters and also the fact that bias rejection properties are not obvious. A better implementation strategy is to use a feedback configuration, as explained below. Consider the feedback loop show in Figure ??.

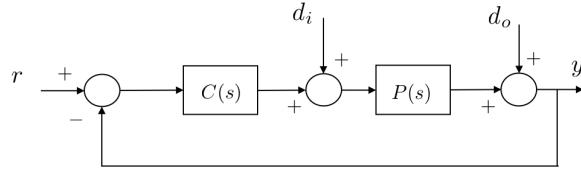


Figure 11.5: Standard feedback loop

Following standard block diagram manipulation we get that

$$y(s) = \left(\frac{1}{1+PC} \right) d_o(s) + \left(\frac{P}{1+PC} \right) d_i(s) + \left(\frac{PC}{1+PC} \right) r(s)$$

where y is the output, r is the reference input, d_o is an output disturbance, and d_i is an input disturbance. The transfer function

$$S = \frac{1}{1+PC}$$

is called the sensitivity function, and the tranfer function

$$T = \frac{PC}{1+PC}$$

is called the complementary sensitivity function. Note that

$$S(s) + T(s) = 1.$$

If $P(s)C(s)$ is a standard loopshape that is large ($>> 1$) at low frequency and small ($<< 1$) at high frequency, then the sensitivity function $S(s)$ is a high pass filter and the complementary sensitivity function $T(s)$ is a low pass filter. Therefore, the feedback structure can be used to implement a complementary filter for the roll angle as shown in Figure ??.

In order to get a first-order filter where

$$S(s) = \frac{1}{1+PC} = \frac{s}{s+k_p} = \frac{1}{1+\frac{k_p}{s}}$$

we set $P(s) = 1/s$ and $C(s) = k_p$ as shown in Figure ??, Figure ??

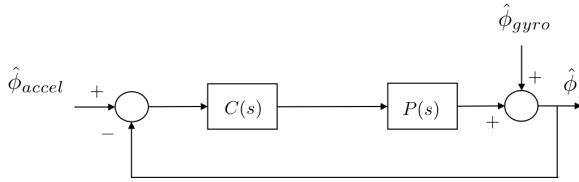


Figure 11.6: Feedback loop implementation of the complementary filter.

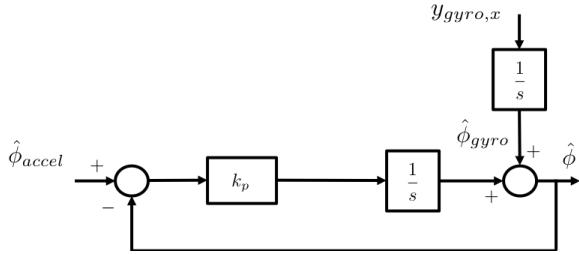


Figure 11.7: Feedback loop implementation of the complementary filter.

also indicates that $\hat{\phi}_{gyro}$ is the integral of the measurement $y_{gyro,x}$. Clearly, the output disturbance of $\hat{\phi}_{gyro}$ is equivalent to an input disturbance of $y_{gyro,x}$ as shown in Figure ??.

As mentioned above, the gyro measurement contains the true roll rate p plus a nearly constant bias b_ϕ . We can use the final value theorem to determine the response of the feedback system shown above to a constant b_ϕ as the input disturbance. The relevant transfer function is

$$\hat{\phi}(s) = \frac{P}{1 + PC} b_\phi(s).$$

The final value theorem gives

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{t \rightarrow \infty} \hat{\phi}(t) \\ &= \lim_{s \rightarrow 0} s \left(\frac{P}{1 + PC} \right) \frac{b}{s} \\ &= \lim_{s \rightarrow 0} \frac{bP}{1 + PC}.\end{aligned}$$

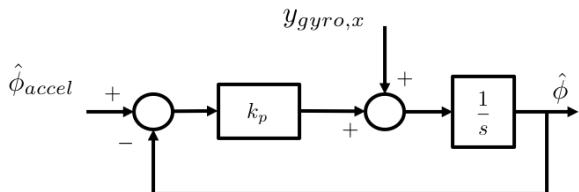
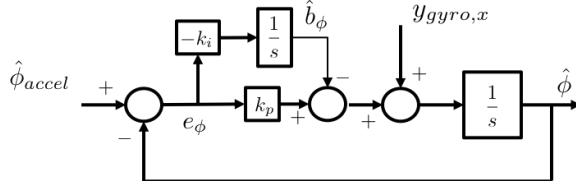


Figure 11.8: Feedback loop implementation of the complementary filter.

When $P = \frac{1}{s}$ and $C = k_p$ we get

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{s \rightarrow 0} \frac{\frac{b}{s}}{1 + \frac{k_p}{s}} \\ &= \lim_{s \rightarrow 0} \frac{b}{s + k_p} \\ &= \frac{b}{k_p}.\end{aligned}$$

Therefore, the effect of the bias can be reduced by increasing k_p but cannot be eliminated. However, using a PI structure for $C(s)$ we can completely remove the effect of the bias. The resulting architecture is shown in Figure ???. When $C(s) = k_p + k_i/s$ the steady state response



to a constant bias is

$$\begin{aligned}\hat{\phi}_{ss} &= \lim_{s \rightarrow 0} \frac{\frac{b}{s}}{1 + \frac{k_p + k_i/s}{s}} \\ &= \lim_{s \rightarrow 0} \frac{bs}{s^2 + k_ps + k_i} \\ &= 0.\end{aligned}$$

From the block diagram, we see that the differential equations that describe the complementary filter are given by

$$\begin{aligned}\dot{\hat{\phi}} &= (y_{gyro,x} - \hat{b}_\phi) + k_p e_\phi \\ \dot{\hat{b}}_\phi &= -k_i e_\phi \\ e_\phi &= \hat{\phi}_{accel} - \hat{\phi}.\end{aligned}\tag{11.10}$$

Note that we have introduced negative signs in the implementation of the integrator to emphasize the fact that the role of the integrator is to estimate the bias and to subtract the bias from the gyro measurement.

We also note that a Lyapunov argument can be used to prove the stability of the complementary filter given in Equation (11.10) in the absence of noise. Indeed, consider the Lyapunov function candidate

$$V = \frac{1}{2}(\phi - \hat{\phi})^2 + \frac{1}{2k_i}(b_\phi - \hat{b}_\phi)^2.$$

Figure 11.9: Feedback loop implementation of the complementary filter.

Differentiating and using the system and filter dynamics gives

$$\begin{aligned}\dot{V} &= (\phi - \hat{\phi})(\dot{\phi} - \dot{\hat{\phi}}) + \frac{1}{k_i}(b_\phi - \hat{b}_\phi)(\dot{b}_\phi - \dot{\hat{b}}_\phi) \\ &= (\phi - \hat{\phi})(p - (y_{gyro,x} - \hat{b}) - k_p(\hat{\phi}_{accel} - \hat{\phi})) - \frac{1}{k_i}(b_\phi - \hat{b}_\phi)(-k_i)(\hat{\phi}_{accel} - \hat{\phi}) \\ &= (\phi - \hat{\phi})((y_{gyro,x} - b) - (y_{gyro,x} - \hat{b}) - k_p(\phi + \nu_\phi - \hat{\phi})) + (b_\phi - \hat{b}_\phi)(\phi + \nu_\phi - \hat{\phi}) \\ &\leq -k_p(\phi - \hat{\phi})^2 + \gamma|\nu_\phi| \left\| \begin{pmatrix} \phi - \hat{\phi} \\ b - \hat{b} \end{pmatrix} \right\|.\end{aligned}$$

If $\nu_\phi = 0$ then LaSalle's invariance principle can be used to show asymptotic convergence of the complementary filter. For non-zero ν_ϕ , the filter error is bounded by a function of the size of ν_ϕ .

We note also that for Euler angle representation for pitch and yaw, a similar derivation results in the complementary filters

$$\begin{aligned}\dot{\hat{\theta}} &= (y_{gyro,y} - \hat{b}_\theta) + k_p e_\theta \\ \dot{\hat{b}}_\theta &= -k_i e_\theta \\ e_\theta &= \hat{\theta}_{accel} - \hat{\theta},\end{aligned}\tag{11.11}$$

$$\begin{aligned}\dot{\hat{\psi}} &= (y_{gyro,z} - \hat{b}_\psi) + k_p e_\psi \\ \dot{\hat{b}}_\psi &= -k_i e_\psi\end{aligned}\tag{11.12}$$

$$e_\psi = \hat{\psi}_{mag} - \hat{\psi}.\tag{11.13}$$

Digital Implementation of the Simple Complementary Filter Using the Euler approximation

$$\dot{z}(t) \approx \frac{z(t) - z(t - T_s)}{T_s}$$

where T_s is the sample rate, the simple complementary filter can be implemented using the following pseudo-code.

Inputs:

- The rate gyro measurements $y_{gyro,x}$, $y_{gyro,y}$, $y_{gyro,z}$.
- The accelerometer measurements $y_{accel,x}$, $y_{accel,y}$, and $y_{accel,z}$.
- The (processed) magnetometer measurement $y_{mag,\psi}$.
- Accelerometer and magnetometer biases b_x , b_y , b_z , b_ψ .
- The sample rate T_s .

Initialization:

- Initialize the estimate of the biases $\hat{b}_\phi[0]$, $\hat{b}_\theta[0]$, $\hat{b}_\psi[0]$.
- Initialize the estimate of the Euler angles $\hat{\phi}[0]$, $\hat{\theta}[0]$, $\hat{\psi}[0]$.

Step 1: Process the accelerometers and magnetometer:

- $\hat{\phi}_{accel}[n] = \tan^{-1} \left(\frac{y_{accel,y}[n] - b_y}{y_{accel,z}[n] - b_z} \right)$
- $\hat{\theta}_{accel}[n] = \sin^{-1} \left(\frac{y_{accel,x}[n] - b_x}{g} \right)$
- $\hat{\psi}_{mag}[n] = y_{mag,\psi}[n] - b_\psi$.

Step 2: Compute the errors

- $e_\phi[n] = \hat{\phi}_{accel}[n] - \hat{\phi}[n-1]$
- $e_\theta[n] = \hat{\theta}_{accel}[n] - \hat{\theta}[n-1]$
- $e_\psi[n] = \hat{\psi}_{mag}[n] - \hat{\psi}[n-1]$

Step 3: Update the bias estimates:

- $\hat{b}_\phi[n] = \hat{b}_\phi[n-1] - T_s k_i e_\phi[n]$
- $\hat{b}_\theta[n] = \hat{b}_\theta[n-1] - T_s k_i e_\theta[n]$
- $\hat{b}_\psi[n] = \hat{b}_\psi[n-1] - T_s k_i e_\psi[n]$

Step 4: Update Euler angle estimates:

- $\hat{\phi}[n] = \hat{\phi}[n-1] = T_s ((y_{gyro,x}[n] - \hat{b}_\phi[n]) + k_p e_\phi[n])$
- $\hat{\theta}[n] = \hat{\theta}[n-1] = T_s ((y_{gyro,y}[n] - \hat{b}_\theta[n]) + k_p e_\theta[n])$
- $\hat{\psi}[n] = \hat{\psi}[n-1] = T_s ((y_{gyro,z}[n] - \hat{b}_\psi[n]) + k_p e_\psi[n])$

Simulation Results <Need some simulation results here.>

11.4 Old Stuff that might be useful

11.5 Sensors

11.5.1 Camera

The control objective is to hold the position of the quadrotor over a ground based target that is detected using the vision sensor. In this section we will briefly describe how to estimate p_x and p_y in the vehicle 1-frame.

We will assume that the camera is mounted so that the optical axis of the camera is aligned with the body frame z -axis and so that the x -axis of the camera points out the right of the quadrotor and the y -axis of the camera points to the back of the quadrotor.

The camera model is shown in Figure ???. The position of the target in the vehicle-1 frame is (p_x, p_y, p_z) . The pixel location of the target in the image is (ϵ_x, ϵ_y) .

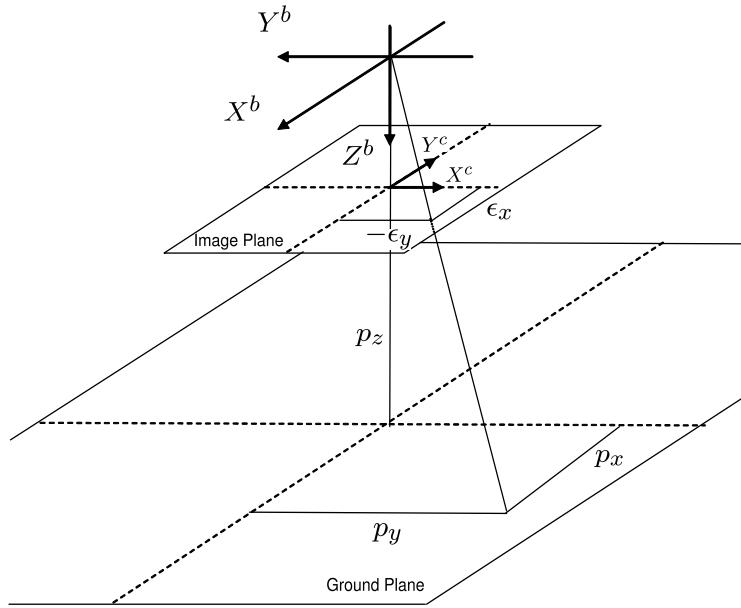


Figure 11.10: Camera model for the quadrotor.

The geometry for p_y is shown in Figure ???. From the geometry shown in Figure ???, we can see that

$$p_y = p_z \tan \left(\phi - \epsilon_x \frac{\eta}{M_y} \right), \quad (11.14)$$

where η is the camera field-of-view, and M_y is the number of pixels along the camera y -axis. In Figure ???, both p_y and ϵ_x are negative. Positive values are toward the right rotor. A similar equation can be

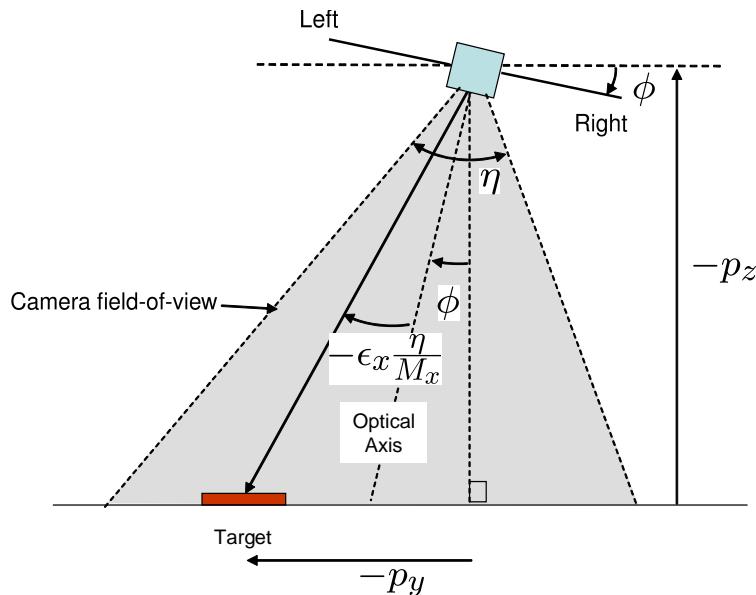


Figure 11.11: The geometry introduced by the vision system. The height above ground is given by $-p_z$, the lateral position error is p_y , the roll angle is ϕ , the field-of-view of the camera is η , the lateral pixel location of the target in the image is ϵ_x , and the total number of pixels along the lateral axis of the camera is M_x .

derived for p_x as

$$p_x = -p_z \tan\left(\theta - \epsilon_y \frac{\eta}{M_y}\right). \quad (11.15)$$

11.6 State Estimation

The objective of this section is to describe techniques for estimating the state of the quadrotor from sensor measurements. We need to estimate the following states: $p_x, p_y, p_z, u, v, w, \phi, \theta, \psi, p, q, r$.

The angular rates p, q , and r can be obtained by low pass filtering the rate gyros. The remain states require a Kalman filter. Both are discussed below.

11.6.1 Low Pass Filters

The Laplace transforms representation of a simple low-pass filter is given by

$$Y(s) = \frac{a}{s+a} U(s),$$

where $u(t)$ is the input of the filter and $y(t)$ is the output. Inverse Laplace transforming we get

$$\dot{y} = -ay + au. \quad (11.16)$$

Using a zeroth order approximation of the derivative we get

$$\frac{y(t+T) - y(t)}{T} = -ay(t) + au(t),$$

where T is the sample rate. Solving for $y(t+T)$ we get

$$y(t+T) = (1 - aT)y(t) + aTu(t).$$

For the zeroth order approximation to be valid we need $aT \ll 1$. If we let $\alpha = aT$ then we get the simple form

$$y(t+T) = (1 - \alpha)y(t) + \alpha u(t).$$

Note that this equation has a nice physical interpretation: the new value of y (filtered value) is a weighted average of the old value of y and u (unfiltered value). If u is noisy, then $\alpha \in [0, 1]$ should be set to a small value. However, if u is relatively noise free, then α should be close to unity.

In the derivation of the discrete-time implementation of the low-pass filter, it is possible to be more precise. In particular, returning to Equation (11.16), from linear systems theory, it is well known that the solution is given by

$$y(t+T) = e^{-aT}y(t) + a \int_0^T e^{-a(T-\tau)}u(\tau) d\tau.$$

Assuming that $u(t)$ is constant between sample periods results in the expression

$$\begin{aligned} y(t+T) &= e^{-aT}y(t) + a \int_0^T e^{-a(T-\tau)} d\tau u(t)y(t+T) \\ &= e^{-aT}y(t) + (1 - e^{-aT})u(t). \end{aligned} \quad (11.17)$$

Note that since $e^x = 1 + x + \frac{x^2}{2!} + \dots$ we have that $e^{-aT} \approx 1 - aT$ and $1 - e^{-aT} \approx aT$.

Therefore, if the sample rate is not fixed (common for micro controllers), and it is desired to have a fixed cut-off frequency, then Equation (11.17) is the preferable way to implement a low-pass filter in digital hardware.

We will use the notation $LPF(\cdot)$ to represent the low-pass filter operator. Therefore $\hat{x} = LPF(x)$ is the low-pass filtered version of x .

11.6.2 Angular Rates p , q , and r .

The angular rates p , q , and r can be estimated by low-pass filtering the rate gyro signals:

$$\hat{p} = LPF(y_{\text{gyro},x}) \quad (11.18)$$

$$\hat{q} = LPF(y_{\text{gyro},y}) \quad (11.19)$$

$$\hat{r} = LPF(y_{\text{gyro},z}). \quad (11.20)$$

11.6.3 Dynamic Observer Theory

The objective of this section is to briefly review observer theory.

Suppose that we have a linear time-invariant system modeled by the equations

$$\dot{x} = Ax + Bu$$

$$y = Cx.$$

A continuous-time observer for this system is given by the equation

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{copy of the model}} + \underbrace{L(y - C\hat{x})}_{\text{correction due to sensor reading}}, \quad (11.21)$$

copy of the model correction due to sensor reading

where \hat{x} is the estimated value of x . Letting $\tilde{x} = x - \hat{x}$ we observe that

$$\dot{\tilde{x}} = (A - LC)\tilde{x}$$

which implies that the observation error decays exponentially to zero if L is chosen such that the eigenvalues of $A - LC$ are in the open left half of the complex plane.

In practice, the sensors are usually sampled and processed in digital hardware at some sample rate T_s . How do we modify the observer equation shown in Equation (11.21) to account for sampled sensor readings?

The typical approach is to propagate the system model between samples using the equation

$$\dot{\hat{x}} = A\hat{x} + Bu \quad (11.22)$$

and then to update the estimate when a measurement is received using the equation

$$\hat{x}^+ = \hat{x}^- + L(y(t_k) - C\hat{x}^-),$$

where t_k is the instant in time that the measurement is received and \hat{x}^- is the state estimate produced by Equation (11.22) at time t_k .

Equation (11.22) is then re-instantiated with initial conditions given by \hat{x}^+ . The continuous-discrete observer is summarized in Table ??.

System model:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y(t_k) &= Cx(t_k) \\ \text{Initial Condition } x(0).\end{aligned}$$

Assumptions:

- Knowledge of $A, B, C, u(t)$.
- No measurement noise.

Prediction: In between measurements ($t \in [t_{k-1}, t_k]$):

- Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.
- Initial condition is $\hat{x}^+(t_{k-1})$.
- Label the estimate at time t_k as $\hat{x}^-(t_k)$.

Correction: At sensor measurement ($t = t_k$):

$$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L(y(t_k) - C\hat{x}^-(t_k)).$$

Table 11.1: Continuous-Discrete Linear Observer.

The observation process is shown graphically in Figure ???. Note

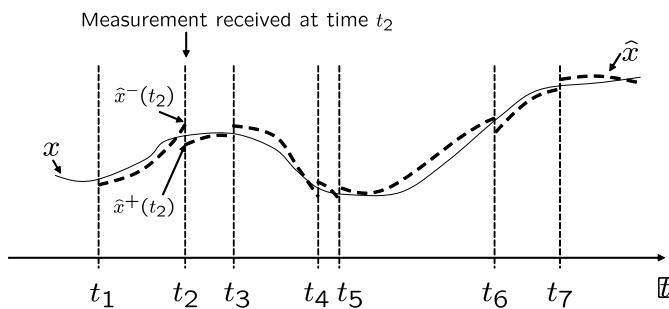


Figure 11.12: sdf

that it is not necessary to have a fixed sample rate. The continuous-discrete observer can be implemented using Algorithm 1.

Algorithm 1: Continuous-Discrete Observer

```

1: Initialize:  $\hat{x} = 0$ .
2: Pick an output sample rate  $T_{out}$  which is much less than the sam-
   ple rates of the sensors.
3: At each sample time  $T_{out}$ :
4: for  $i = 1$  to  $N$  do {Prediction: Propagate the state equation.}
5:    $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) (A\hat{x} + Bu)$ 
6: end for
7: if A measurement has been received from sensor  $i$  then {Correc-
   tion: Measurement Update}
8:    $\hat{x} = \hat{x} + L_i (y_i - C_i \hat{x})$ 
9: end if
```

Note that we did not use the fact that the process was linear. Suppose instead that we have a nonlinear system of the form

$$\dot{x} = f(x, u) \quad (11.23)$$

$$y = c(x) \quad (11.24)$$

, then the continuous discrete observer is given in table ??.

System model:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y(t_k) &= c(x(t_k)) \\ \text{Initial Condition } x(0). \end{aligned}$$

Assumptions:

- Knowledge of f , c , $u(t)$.
- No measurement noise.

Prediction: In between measurements ($t \in [t_{k-1}, t_k]$):

- Propagate $\dot{\hat{x}} = f(\hat{x}, u)$.
- Initial condition is $\hat{x}^+(t_{k-1})$.
- Label the estimate at time t_k as $\hat{x}^-(t_k)$.

Correction: At sensor measurement ($t = t_k$):

$$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L(y(t_k) - c(\hat{x}^-(t_k)))$$

Table 11.2: Continuous-Discrete Nonlinear Observer.

The real question is how to pick the observer gain L .

11.6.4 Essentials from Probability Theory

Let $X = (x_1, \dots, x_n)^T$ be a random vector whose elements are random variables. The mean, or expected value of X is denoted by

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} E\{x_1\} \\ \vdots \\ E\{x_n\} \end{pmatrix} = E\{X\},$$

where

$$E\{x_i\} = \int \xi f_i(\xi) d\xi,$$

and $f(\cdot)$ is the probability density function for x_i . Given any pair of components x_i and x_j of X , we denote their covariance as

$$\text{cov}(x_i, x_j) = \Sigma_{ij} = E\{(x_i - \mu_i)(x_j - \mu_j)\}.$$

The covariance of any component with itself is the variance, i.e.,

$$\text{var}(x_i) = \text{cov}(x_i, x_i) = \Sigma_{ii} = E\{(x_i - \mu_i)^2\}.$$

The standard deviation of x_i is the square root of the variance:

$$\text{stdev}(x_i) = \sigma_i = \sqrt{\Sigma_{ii}}.$$

The covariances associated with a random vector X can be grouped into a matrix known as the covariance matrix:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{pmatrix} = E\{(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^T\} = E\{XX^T\} - \boldsymbol{\mu}\boldsymbol{\mu}^T.$$

Note that $\Sigma = \Sigma^T$ so that Σ is both symmetric and positive semi-definite, which implies that its eigenvalues are real and nonnegative.

The probability density function for a Gaussian random variable is given by

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-\mu_x)^2}{\sigma_x^2}},$$

where μ_x is the mean of x and σ_x is the standard deviation. The vector equivalent is given by

$$f_X(X) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp \left[-\frac{1}{2}(X - \boldsymbol{\mu})^T \Sigma^{-1} (X - \boldsymbol{\mu}) \right],$$

in which case we write

$$X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma),$$

and say that X is normally distributed with mean $\boldsymbol{\mu}$ and covariance Σ .

Figure ?? shows the level curves for a 2D Gaussian random variable with different covariance matrices.

11.6.5 Derivation of the Kalman Filter

In this section we assume the following state model:

$$\begin{aligned} \dot{x} &= Ax + Bu + G\xi \\ y_k &= Cx_k + \eta_k, \end{aligned}$$

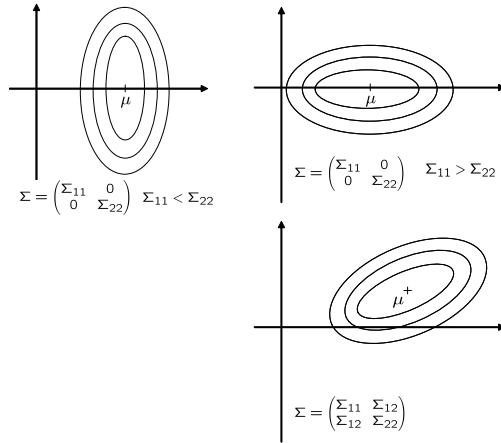


Figure 11.13: Level curves for the pdf of a 2D Gaussian random variable.

where $y_k = y(t_k)$ is the k^{th} sample of y , $x_k = x(t_k)$ is the k^{th} sample of x , η_k is the measurement noise at time t_k , ξ is a zero-mean Gaussian random process with covariance Q , and η_k is a zero-mean Gaussian random variable with covariance R . Note that the sample rate does not need to be fixed.

The observer will therefore have the following form:

Prediction: In between measurements: ($t \in [t_{k-1}, t_k]$)

Propagate $\hat{x} = A\hat{x} + Bu$.

Initial condition is $\hat{x}^+(t_{k-1})$.

Label the estimate at time t_k as $\hat{x}^-(t_k)$.

Correction: At sensor measurement ($t = t_k$):

$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L(y(t_k) - C\hat{x}^-(t_k))$.

Our objective is to pick L to minimize $\text{tr}(P(t))$.

Between Measurements. Differentiating \tilde{x} we get

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu + G\xi - A\hat{x} - Bu \\ &= A\tilde{x} + G\xi.\end{aligned}$$

Therefore we have that

$$\tilde{x}(t) = e^{At}\tilde{x}_0 + \int_0^t e^{A(t-\tau)}G\xi(\tau) d\tau.$$

We can therefore compute the evolution for P as

$$\begin{aligned}\dot{P} &= \frac{d}{dt} E\{\tilde{x}\tilde{x}^T\} \\ &= E\{\dot{\tilde{x}}\tilde{x}^T + \tilde{x}\dot{\tilde{x}}^T\} \\ &= E\left\{A\tilde{x}\tilde{x}^T + G\xi\tilde{x}^T + \tilde{x}\tilde{x}^T A^T + \tilde{x}\xi^T G^T\right\} \\ &= AP + PA^T + GE\{\xi\tilde{x}^T\}^T + E\{\tilde{x}\xi^T\}G^T.\end{aligned}$$

As in the previous section we get

$$\begin{aligned} E\{\xi \tilde{x}^T\} &= E\left\{\xi(t) \tilde{x}_0 e^{A^T t} + \int_0^t \xi(\tau) \xi^T(\tau) G^T e^{A^T(t-\tau)} d\tau\right\} \\ &= \frac{1}{2} Q G^T, \end{aligned}$$

which implies that

$$\dot{P} = AP + PA^T + GQG^T.$$

At Measurements. At a measurement we have that

$$\begin{aligned} \tilde{x}^+ &= x - \hat{x}^+ \\ &= x - \hat{x}^- - L(Cx + \eta - C\hat{x}^-) \\ &= \tilde{x}^- - LC\tilde{x}^- - L\eta. \end{aligned}$$

Therefore

$$\begin{aligned} P^+ &= E\{\tilde{x}^+ \tilde{x}^{+T}\} \\ &= E\left\{(\tilde{x}^- - LC\tilde{x}^- - L\eta)(\tilde{x}^- - LC\tilde{x}^- - L\eta)^T\right\} \\ &= E\left\{\tilde{x}^- \tilde{x}^{-T} - \tilde{x}^- \tilde{x}^{-T} C^T L^T - \tilde{x}^- \eta^T L^T \right. \\ &\quad \left. - LC\tilde{x}^- \tilde{x}^{-T} + LC\tilde{x}^- \tilde{x}^{-T} C^T L^T + LC\tilde{x}^- \eta^T L^T \right. \\ &\quad \left. - L\eta \tilde{x}^{-T} + L\eta \tilde{x}^{-T} C^T L^T + L\eta \eta^T L^T\right\} \\ &= P^- - P^- C^T L^T - LCP^- + LCP^- C^T L^T + LRL^T. \quad (11.25) \end{aligned}$$

Our objective is to pick L to minimize $\text{tr}(P^+)$. A necessary condition is

$$\begin{aligned} \frac{\partial}{\partial L} \text{tr}(P^+) &= -P^- C^T - P^- C^T + 2LCP^- C^T + 2LR = 0 \\ &\implies 2L(R + CP^- C^T) = 2P^- C^T \\ &\implies L = P^- C^T (R + CP^- C^T)^{-1}. \end{aligned}$$

Plugging back into Equation (11.25) give

$$\begin{aligned} P^+ &= P^- + P^- C^T (R + CP^- C^T)^{-1} CP^- - P^- C^T (R + CP^- C^T)^{-1} CP^- \\ &\quad + P^- C^T (R + CP^- C^T)^{-1} (CP^- C^T + R) (R + CP^- C^T)^{-1} CP^- \\ &= P^- - P^- C^T (R + CP^- C^T)^{-1} CP^- \\ &= (I - P^- C^T (R + CP^- C^T)^{-1} C) P^- \\ &= (I - LC) P^-. \end{aligned}$$

For linear systems, the continuous-discrete Kalman filter is summarized in Table ??

If the system is nonlinear, then the Kalman filter can still be applied but we need to linearize the nonlinear equations in order to

System model:

$$\begin{aligned}\dot{x} &= Ax + Bu + \xi \\ y_i(t_k) &= C_i x(t_k) + \eta_k \\ \text{Initial Condition } x(0).\end{aligned}$$

Assumptions:

- Knowledge of $A, B, C_i, u(t)$.
- Process noise satisfies $\xi \sim \mathcal{N}(0, Q)$.
- Measurement noise satisfies $\eta_k \sim \mathcal{N}(0, R)$.

Prediction: In between measurements ($t \in [t_{k-1}, t_k]$):

$$\begin{aligned}\text{Propagate } \dot{\hat{x}} &= A\hat{x} + Bu. \\ \text{Propagate } \dot{P} &= AP + PA^T + Q.\end{aligned}$$

Correction: At the i^{th} sensor measurement ($t = t_k$):

$$\begin{aligned}L_i &= P^- C_i^T (R_i + C_i P C_i^T)^{-1}, \\ P^+ &= (I - L_i C_i) P^-, \\ \hat{x}^+(t_k) &= \hat{x}^-(t_k) + L_i (y(t_k) - C_i \hat{x}^-(t_k)).\end{aligned}$$

Table 11.3: Continuous-Discrete Kalman Filter.

compute the error covariance matrix P and the Kalman gain L . The extended Kalman filter (EKF) is given in Table ??, and an algorithm to implement the EKF is given in Algorithm 2.

11.6.6 Application to the quadrotor

In this section we will discuss the application of Algorithm 2 to the quadrotor.

We would like to estimate the state

$$\hat{x} = \begin{pmatrix} \hat{p}_x \\ \hat{p}_y \\ \hat{p}_z \\ \dot{\hat{p}}_x \\ \dot{\hat{p}}_y \\ \dot{\hat{p}}_z \\ \hat{\phi} \\ \hat{\theta} \\ \hat{\psi} \end{pmatrix},$$

where the rate gyros and accelerometers will be used to drive the prediction step, and an ultrasonic altimeter and camera will be used in the correction step.

The propagation model is obtained from Equations (??)–(??), and

System model:

$$\begin{aligned}\dot{x} &= f(x, u) + \xi \\ y_i(t_k) &= c_i(x(t_k)) + \eta_k \\ \text{Initial Condition } x(0). &\end{aligned}$$

Assumptions:

- Knowledge of f , c_i , $u(t)$.
- Process noise satisfies $\xi \sim \mathcal{N}(0, Q)$.
- Measurement noise satisfies $\eta_k \sim \mathcal{N}(0, R)$.

Prediction: In between measurements ($t \in [t_{k-1}, t_k]$):

$$\begin{aligned}\text{Propagate } \dot{\hat{x}} &= f(\hat{x}, u), \\ \text{Compute } A &= \left. \frac{\partial f}{\partial x} \right|_{x=\hat{x}(t)}, \\ \text{Propagate } \dot{P} &= AP + PA^T + Q.\end{aligned}$$

Correction: At the i^{th} sensor measurement ($t = t_k$):

$$\begin{aligned}C_i &= \left. \frac{\partial c_i}{\partial x} \right|_{x=\hat{x}^-} \\ L_i &= P^- C_i^T (R_i + C_i P C_i^T)^{-1}, \\ P^+ &= (I - L_i C_i) P^-, \\ \hat{x}^+(t_k) &= \hat{x}^-(t_k) + L_i (y_i(t_k) - c_i(\hat{x}^-(t_k))).\end{aligned}$$

(??) as

$$f(x, u) = \begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \cos \phi \sin \theta a_z \\ -\sin \phi a_z \\ g + \cos \phi \cos \theta a_z \\ p + q \sin \phi \tan \theta + r \cos \phi \tan \theta \\ q \cos \phi - r \sin \phi \\ q \frac{\sin \phi}{\cos \theta} + r \frac{\cos \phi}{\cos \theta} \end{pmatrix},$$

where we have used the fact that the z -axis of the accelerometer measures $a_z = -F/m$. Differentiating we obtain

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -s_\phi s_\theta a_z & c_\phi c_\theta a_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -c_\phi a_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -s_\phi c_\theta a_z & -c_\phi s_\theta a_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & qc\phi t\theta - rs\phi t\theta & (qs\phi + rc\phi)/c^2\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -qs\phi - rc\phi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{qc\phi - rs\phi}{c\theta} & -(qs\phi + rc\phi)\frac{t\theta}{c\theta} & 0 \end{pmatrix}$$

Note that it may be adequate (not sure) to use a small angle ap-

Table 11.4: Continuous-Discrete Extended Kalman Filter.

Algorithm 2: Continuous-Discrete Extended Kalman Filter

```

1: Initialize:  $\hat{x} = 0$ .
2: Pick an output sample rate  $T_{out}$  which is much less than the sam-
   ple rates of the sensors.
3: At each sample time  $T_{out}$ :
4: for  $i = 1$  to  $N$  do {Prediction: Propagate the equations.}
5:    $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) (f(\hat{x}, u))$ 
6:    $A = \frac{\partial f}{\partial x}$ 
7:    $P = P + \left(\frac{T_{out}}{N}\right) (AP + PA^T + GQG^T)$ 
8: end for
9: if A measurement has been received from sensor  $i$  then {Correc-
   tion: Measurement Update}
10:   $C_i = \frac{\partial c_i}{\partial x}$ 
11:   $L_i = PC_i^T(R_i + C_i P C_i^T)^{-1}$ 
12:   $P = (I - L_i C_i)P$ 
13:   $\hat{x} = \hat{x} + L_i(y_i - c_i(\hat{x}))$ .
14: end if
```

proximation in the model resulting in

$$f(x, u) = \begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \theta a_z \\ -\phi a_z \\ g + a_z \\ p \\ q \\ r \end{pmatrix},$$

and

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -a_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

If this form works, then the update equation for P can be coded by hand, significantly reducing the computational burden. Note also that $f(x, u)$ does not take into account the motion of the target. A feedforward term can be added to f to account for the target motion

as

$$f(x, u) = \begin{pmatrix} \dot{p}_x - \dot{m}_x \\ \dot{p}_y - \dot{m}_y \\ \dot{p}_z \\ \theta a_z - \ddot{m}_x \\ -\phi a_z - \ddot{m}_y \\ g + a_z \\ p \\ q \\ r \end{pmatrix},$$

where (\dot{m}_x, \dot{m}_y) is the velocity of the target in the vehicle-1 frame, and (\ddot{m}_x, \ddot{m}_y) is the acceleration of the target in the vehicle-1 frame.

Let the relative position between the quadrotor and the target be denoted by $\mathbf{p}^{v1} = (p_x, p_y, p_z)^T$. We can transform to the camera frame as

$$\mathbf{p}^c = R_g^c R_b^g R_{v1}^b \mathbf{p}^{v1},$$

where

$$R_g^c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

is the rotation matrix from gimbal coordinates to camera coordinates,

$$R_b^g = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

transforms body coordinates to gimbal coordinates, and

$$R_{v1}^b = \begin{pmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{pmatrix}$$

transforms vehicle-1 coordinates to body coordinates. Using the small angle approximation for ϕ and θ we get

$$\mathbf{p}^c = \begin{pmatrix} \phi\theta p_x + p_y + \phi p_z \\ -p_x + \theta p_z \\ \theta p_x - \phi p_y + p_z \end{pmatrix}.$$

The model for the pixel coordinates are therefore computed as

$$\begin{aligned} \epsilon_x &= c_{\epsilon_x}(x) \triangleq f \frac{\phi\theta p_x + p_y + \phi p_z}{\theta p_x - \phi p_y + p_z} \\ \epsilon_2 &= c_{\epsilon_y}(x) \triangleq f \frac{-p_x + \theta p_z}{\theta p_x - \phi p_y + p_z}. \end{aligned}$$

We therefore have the following sensors available to correct the state estimate:

$$\text{sensor 1: altimeter } y_1 = c_{alt}(x) \stackrel{\Delta}{=} -p_z + \eta_1[k] \quad (11.26)$$

$$\text{sensor 2: vision x-pixel } y_2 = c_{\epsilon_x}(x) + \eta_2[k]. \quad (11.27)$$

$$\text{sensor 3: vision y-pixel } y_3 = c_{\epsilon_y}(x) + \eta_3[k] \quad (11.28)$$

$$\text{sensor 4: vision heading } y_4 = c_{\epsilon_\psi}(x) \stackrel{\Delta}{=} \pi/2 + \psi + \eta_4[k]. \quad (11.29)$$

The linearization of the first and fourth output functions are given by

$$C_1 = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (11.30)$$

$$C_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (11.31)$$

The linearization of the expression for the pixel coordinates is messy and can easily be computed numerically using the approximation

$$\frac{\partial f(x_1, \dots, x_i, \dots, x_n)}{\partial x_i} \approx \frac{f(x_1, \dots, x_i + \Delta, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{\Delta},$$

where Δ is a small constant.

12

Trajectory Estimation

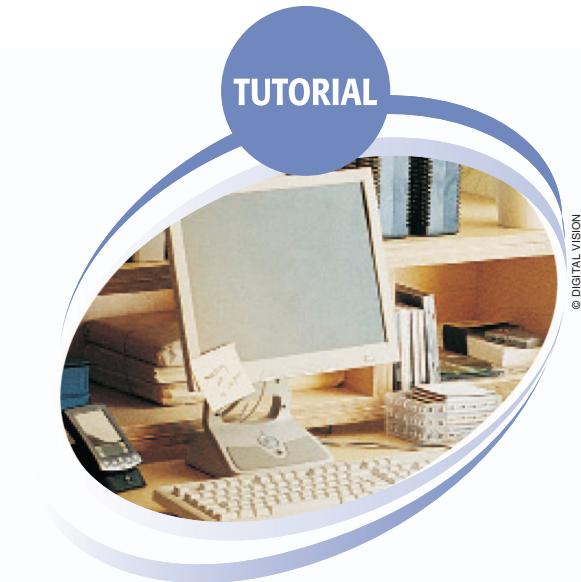
- overview of other sensors: GPS, magnetometer, barometer, laser altimeter, sonar altimeter, optic flow
 - EKF assuming attitude is known
 - MEKF for full state using quaternions

13

Visual Odometry

Stuff from paper with Dan.

13.1 *Visual Odometry Tutorial PI, RAM, 2011*



© DIGITAL VISION

Visual Odometry

Part I: The First 30 Years and Fundamentals

By Davide Scaramuzza and Friedrich Fraundorfer

Visual odometry (VO) is the process of estimating the egomotion of an agent (e.g., vehicle, human, and robot) using only the input of a single or multiple cameras attached to it. Application domains include robotics, wearable computing, augmented reality, and automotive. The term VO was coined in 2004 by Nister in his landmark paper [1]. The term was chosen for its similarity to wheel odometry, which incrementally estimates the motion of a vehicle by integrating the number of turns of its wheels over time. Likewise, VO operates by incrementally estimating the pose of the vehicle through examination of the changes that motion induces on the images of its onboard cameras. For VO to work effectively, there should be sufficient illumination in the environment and a static scene with enough texture to allow apparent motion to be extracted. Furthermore, consecutive frames should be captured by ensuring that they have sufficient scene overlap.

Digital Object Identifier 10.1109/MRA.2011.943233
Date of publication: 8 December 2011

The advantage of VO with respect to wheel odometry is that VO is not affected by wheel slip in uneven terrain or other adverse conditions. It has been demonstrated that compared to wheel odometry, VO provides more accurate trajectory estimates, with relative position error ranging from 0.1 to 2%. This capability makes VO an interesting supplement to wheel odometry and, additionally, to other navigation systems such as global positioning system (GPS), inertial measurement units (IMUs), and laser odometry (similar to VO, laser odometry estimates the egomotion of a vehicle by scan-matching of consecutive laser scans). In GPS-denied environments, such as underwater and aerial, VO has utmost importance.

This two-part tutorial and survey provides a broad introduction to VO and the research that has been undertaken from 1980 to 2011. Although the first two decades witnessed many offline implementations, only in the third decade did real-time working systems flourish, which has led VO to be used on another planet by two Mars-exploration rovers for the first time. Part I (this tutorial) presents a historical review of the first 30 years of research in this field and its fundamentals. After a brief discussion on camera

modeling and calibration, it describes the main motion-estimation pipelines for both monocular and binocular scheme, outlining pros and cons of each implementation. Part II will deal with feature matching, robustness, and applications. It will review the main point-feature detectors used in VO and the different outlier-rejection schemes. Particular emphasis will be given to the random sample consensus (RANSAC), and the distinct tricks devised to speed it up will be discussed. Other topics covered will be error modeling, location recognition (or loop-closure detection), and bundle adjustment.

This tutorial provides both the experienced and non-expert user with guidelines and references to algorithms to build a complete VO system. Since an ideal and unique VO solution for every possible working environment does not exist, the optimal solution should be chosen carefully according to the specific navigation environment and the given computational resources.

History of Visual Odometry

The problem of recovering relative camera poses and three-dimensional (3-D) structure from a set of camera images (calibrated or noncalibrated) is known in the computer vision community as *structure from motion* (SFM). Its origins can be dated back to works such as [2] and [3]. VO is a particular case of SFM. SFM is more general and tackles the problem of 3-D reconstruction of both the structure and camera poses from sequentially ordered or unordered image sets. The final structure and camera poses are typically refined with an offline optimization (i.e., bundle adjustment), whose computation time grows with the number of images [4]. Conversely, VO focuses on estimating the 3-D motion of the camera sequentially—as a new frame arrives—and in real time. Bundle adjustment can be used to refine the local estimate of the trajectory.

The problem of estimating a vehicle's egomotion from visual input alone started in the early 1980s and was described by Moravec [5]. It is interesting to observe that most of the early research in VO [5]–[9] was done for planetary rovers and was motivated by the NASA Mars exploration program in the endeavor to provide all-terrain rovers with the capability to measure their 6-degree-of-freedom (DoF) motion in the presence of wheel slippage in uneven and rough terrains.

The work of Moravec stands out not only for presenting the first motion-estimation pipeline—whose main functioning blocks are still used today—but also for describing one of the earliest corner detectors (after the first one proposed in 1974 by Hannah [10]) which is known today as the *Moravec corner detector* [11], a predecessor of the one proposed by Förstner [12] and Harris and Stephens [3], [82].

Moravec tested his work on a planetary rover equipped with what he termed a *slider stereo*: a single camera sliding on a rail. The robot moved in a stop-and-go fashion,

digitizing and analyzing images at every location. At each stop, the camera slid horizontally taking nine pictures at equidistant intervals. Corners were detected in an image using his operator and matched along the epipolar lines of the other eight frames using normalized cross correlation. Potential matches at the next robot locations were found again by correlation using a coarse-to-fine strategy to account for large-scale changes. Outliers were subsequently removed by checking for depth inconsistencies in the eight stereo pairs. Finally, motion was computed as the rigid body transformation to align the triangulated 3-D points seen at two consecutive robot positions. The system of equation was solved via a weighted least square, where the weights were inversely proportional to the distance from the 3-D point.

Although Moravec used a single sliding camera, his work belongs to the class of stereo VO algorithms. This terminology accounts for the fact that the relative 3-D position of the features is directly measured by triangulation at every robot location and used to derive the relative motion. Trinocular methods belong to the same class of algorithms. The alternative to stereo vision is to use a single camera. In this case, only bearing information is available. The disadvantage is that motion can only be recovered up to a scale factor. The absolute scale can then be determined from direct measurements (e.g., measuring the size of an element in the scene), motion constraints, or from the integration with other sensors, such as IMU, air-pressure, and range sensors. The interest in monocular methods is due to the observation that stereo VO can degenerate to the monocular case when the distance to the scene is much larger than the stereo baseline (i.e., the distance between the two cameras). In this case, stereo vision becomes ineffective and monocular methods must be used. Over the years, monocular and stereo VOs have almost progressed as two independent lines of research. In the remainder of this section, we have surveyed the related work in these fields.

Stereo VO

Most of the research done in VO has been produced using stereo cameras. Building upon Moravec's work, Matthies and Shafer [6], [7] used a binocular system and Moravec's procedure for detecting and tracking corners. Instead of using a scalar representation of the uncertainty as Moravec did, they took advantage of the error covariance matrix of the triangulated features and incorporated it into the motion estimation step. Compared to Moravec, they demonstrated superior results in trajectory recovery for a planetary rover, with 2% relative error on a 5.5-m path. Olson et al. [9], [13] later extended that work by

The advantage of VO with respect to wheel odometry is that VO is not affected by wheel slip in uneven terrain or other adverse conditions.

Keyframe selection is a very important step in VO and should always be done before updating the motion.

introducing an absolute orientation sensor (e.g., compass or omnidirectional camera) and using the Forstner corner detector, which is significantly faster to compute than Moravec's operator. They showed that the use of camera egomotion estimates alone results in accumulation errors with superlinear growth in the distance traveled, leading to increased orientation errors. Conversely, when an absolute orientation sensor is incorporated, the error growth can be reduced to a linear function of the distance traveled. This led them to a relative position error of 1.2% on a 20-m path.

Lacroix et al. [8] implemented a stereo VO approach for planetary rovers similar to those explained earlier. The difference lies in the selection of key points. Instead of using the Forstner detector, they used dense stereo and, then, selected the candidate key points by analyzing the correlation function around its peaks—an approach that was later exploited in [14], [15], and other works. This choice was based on the observation that there is a strong correlation between the shape of the correlation curve and the standard deviation of the feature depth. This observation was later used by Cheng et al. [16], [17] in their final VO implementation onboard the Mars rovers. They improved on the earlier implementation by Olson et al. [9], [13] in two areas. First, after using the Harris corner detector, they utilized the curvature of the correlation function around the feature—as proposed by Lacroix et al.—to define the error covariance matrix of the image point. Second, as proposed by Nister et al. [1], they used the random sample consensus (RANSAC) RANSAC [18] in the least-squares motion estimation step for outlier rejection.

A different approach to motion estimation and outlier removal for an all-terrain rover was proposed by Milella and Siegwart [14]. They used the Shi-Tomasi approach [19] for corner detection, and similar to Lacroix, they retained those points with high confidence in the stereo disparity map. Motion estimation was then solved by first using least squares, as in the methods earlier, and then the iterative closest point (ICP) algorithm [20]—an algorithm popular for 3-D registration of laser scans—for pose refinement. For robustness, an outlier removal stage was incorporated into the ICP.

The works mentioned so far have in common that the 3-D points are triangulated for every stereo pair, and the relative motion is solved as a 3-D-to-3-D point registration (alignment) problem. A completely different approach was proposed in 2004 by Nister et al. [1]. Their paper is known not only for coining the term VO but also for providing the first real-time long-run implementation with a robust outlier rejection scheme. Nister et al. improved the earlier implementations in several areas. First, contrary to all previous works, they did not track features among frames

but detected features (Harris corners) independently in all frames and only allowed matches between features. This has the benefit of avoiding feature drift during cross-correlation-based tracking. Second, they did not compute the relative motion as a 3-D-to-3-D point registration problem but as a 3-D-to-two-dimensional (2-D) camera-pose estimation problem (these methods are described in the “Motion Estimation” section). Finally, they incorporated RANSAC outlier rejection into the motion estimation step.

A different motion estimation scheme was introduced by Comport et al. [21]. Instead of using 3-D-to-3-D point registration or 3-D-to-2-D camera-pose estimation techniques, they relied on the quadrifocal tensor, which allows motion to be computed from 2-D-to-2-D image matches without having to triangulate 3-D points in any of the stereo pairs. The benefit of using directly raw 2-D points in lieu of triangulated 3-D points lies in a more accurate motion computation.

Monocular VO

The difference from the stereo scheme is that in the monocular VO, both the relative motion and 3-D structure must be computed from 2-D bearing data. Since the absolute scale is unknown, the distance between the first two camera poses is usually set to one. As a new image arrives, the relative scale and camera pose with respect to the first two frames are determined using either the knowledge of 3-D structure or the trifocal tensor [22].

Successful results with a single camera over long distances (up to several kilometers) have been obtained in the last decade using both perspective and omnidirectional cameras [23]–[29]. Related works can be divided into three categories: feature-based methods, appearance-based methods, and hybrid methods. Feature-based methods are based on salient and repeatable features that are tracked over the frames; appearance-based methods use the intensity information of all the pixels in the image or subregions of it; and hybrid methods use a combination of the previous two.

In the first category are the works by the authors in [1], [24], [25], [27], and [30]–[32]. The first real-time, large-scale VO with a single camera was presented by Nister et al. [1]. They used RANSAC for outlier rejection and 3-D-to-2-D camera-pose estimation to compute the new upcoming camera pose. The novelty of their paper is the use of a five-point minimal solver [33] to calculate the motion hypotheses in RANSAC. After that paper, five-point RANSAC became very popular in VO and was used in several other works [23], [25], [27]. Corke et al. [24] provided an approach for monocular VO based on omnidirectional imagery from a catadioptric camera and optical flow. Lhuillier [25] and Mouragnon et al. [30] presented an approach based on local windowed-bundle adjustment to recover both the motion and the 3-D map (this means that bundle adjustment is performed over a window of the last m frames). Again, they used the five-point RANSAC in [33] to remove the outliers. Tardif et al. [27] presented an

approach for VO on a car over a very long run (2.5 km) without bundle adjustment. Contrary to the previous work, they decoupled the rotation and translation estimation. The rotation was estimated by using points at infinity and the translation from the recovered 3-D map. Erroneous correspondences were removed with five-point RANSAC.

Among the appearance-based or hybrid approaches are the works by the authors in [26], [28], and [29]. Goecke et al. [26] used the Fourier–Mellin transform for registering perspective images of the ground plane taken from a car. Milford and Wyeth [28] presented a method to extract approximate rotational and translational velocity information from a single perspective camera mounted on a car, which was then used in a RatSLAM scheme [34]. They used template tracking on the center of the scene. A major drawback with appearance-based approaches is that they are not robust to occlusions. For this reason, Scaramuzza and Siegwart [29] used image appearance to estimate the rotation of the car and features from the ground plane to estimate the translation and the absolute scale. The feature-based approach was also used to detect failures of the appearance-based method.

All the approaches mentioned earlier are designed for unconstrained motion in 6 DoF. However, several VO works have been specifically designed for vehicles with motion constraints. The advantage is decreased computation time and improved motion accuracy. For instance, Liang and Pears [35], Ke and Kanade [36], Wang et al. [37], and Guerrero et al. [38] took advantage of homographies for estimating the egomotion on a dominant ground plane. Scaramuzza et al. [31], [39] introduced a one-point RANSAC outlier rejection based on the vehicle nonholonomic constraints to speed up egomotion estimation to 400 Hz. In the follow-up work, they showed that nonholonomic constraints allow the absolute scale to be recovered from a single camera whenever the vehicle makes a turn [40]. Following that work, vehicle nonholonomic constraints have also been used by Pretto et al. [32] for improving feature tracking and by Fraundorfer et al. [41] for windowed bundle adjustment (see the following section).

Reducing the Drift

Since VO works by computing the camera path incrementally (pose after pose), the errors introduced by each new frame-to-frame motion accumulate over time. This generates a drift of the estimated trajectory from the real path. For some applications, it is of utmost importance to keep drift as small as possible, which can be done through local optimization over the last m camera poses. This approach—called *sliding window bundle adjustment* or *windowed bundle adjustment*—has been used in several works, such as [41]–[44]. In particular, on a 10-km VO experiment, Konolige et al. [43] demonstrated that windowed-bundle adjustment can decrease the final position error by a factor of 2–5. Obviously, the VO drift can also be reduced through combination with other

sensors, such as GPS and laser, or even with only an IMU [43], [45], [46].

V-SLAM

Although this tutorial focuses on VO, it is worth mentioning the parallel line of research undertaken by visual simultaneous localization and mapping (V-SLAM). For an in-depth study of the SLAM problem, the reader is referred to two tutorials on this topic by Durrant-Whyte and Bailey [47], [48]. Two methodologies have become predominant in V-SLAM: 1) filtering methods fuse the information from all the images with a probability distribution [49] and 2) nonfiltering methods (also called *keyframe methods*) retain the optimization of global bundle adjustment to selected keyframes [50]. The main advantages of either approach have been evaluated and summarized in [51].

In the last few years, successful results have been obtained using both single and stereo cameras [49], [52]–[62]. Most of these works have been limited to small, indoor workspaces and only a few of them have recently been designed for large-scale areas [54], [60], [62]. Some of the early works in real-time V-SLAM were presented by Chiuso et al. [52], Deans [53], and Davison [49] using a full-covariance Kalman approach. The advantage of Davison's work was to account for repeatable localization after an arbitrary amount of time. Later, Handa et al. [59] improved on that work using an active matching technique based on a probabilistic framework. Civera et al. [60] built upon that work by proposing a combination of one-point RANSAC within the Kalman filter that uses the available prior probabilistic information from the filter in the RANSAC model-hypothesis stage. Finally, Strasdat et al. [61] presented a new framework for large-scale V-SLAM that takes advantage of the keyframe optimization approach [50] while taking into account the special character of SLAM.

**VO is only concerned
with the local
consistency of the
trajectory, whereas
SLAM with the global
consistency.**

VO Versus V-SLAM

In this section, the relationship of VO with V-SLAM is analyzed. The goal of SLAM in general (and V-SLAM in particular) is to obtain a global, consistent estimate of the robot path. This implies keeping a track of a map of the environment (even in the case where the map is not needed per se) because it is needed to realize when the robot returns to a previously visited area. (This is called *loop closure*. When a loop closure is detected, this information is used to reduce the drift in both the map and camera path. Understanding when a loop closure occurs and efficiently integrating this new constraint into the current map are two of the main issues in SLAM.) Conversely, VO aims at recovering the path incrementally, pose after pose, and potentially optimizing only over the last n poses of the

path (this is also called *windowed bundle adjustment*). This sliding window optimization can be considered equivalent to building a local map in SLAM; however, the philosophy is different: in VO, we only care about local consistency of the trajectory and the local map is used to obtain a more accurate estimate of the local trajectory (for example, in bundle adjustment), whereas SLAM is concerned with the global map consistency.

VO can be used as a building block for a complete SLAM algorithm to recover the incremental motion of the camera; however, to make a complete SLAM method, one must also add some way to detect loop closing and possibly a global optimization step to obtain a metrically consistent map (without this step, the map is still topologically consistent).

In the motion estimation step, the camera motion between the current and the previous image is computed.

If the user is only interested in the camera path and not in the environment map, there is still the possibility of using a complete V-SLAM method instead of one of the VO techniques described in this tutorial. A V-SLAM method is potentially much more precise, because it enforces many more constraints on the path, but not necessarily more robust (e.g., outliers in loop closing can severely affect the map consistency). In addition, it is more complex and computationally expensive.

In the end, the choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation. Although the global consistency of the camera path is sometimes desirable, VO trades off consistency for real-time performance, without the need to keep track of all the previous history of the camera.

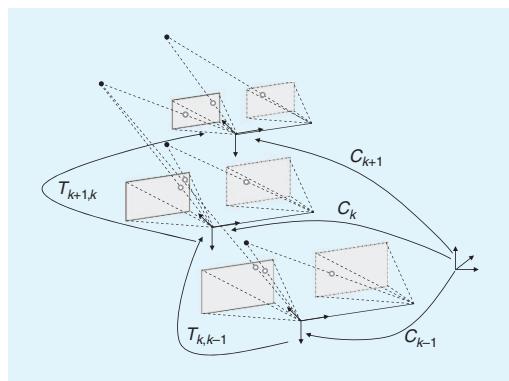


Figure 1. An illustration of the visual odometry problem. The relative poses $T_{k,k-1}$ of adjacent camera positions (or positions of a camera system) are computed from visual features and concatenated to get the absolute poses C_k with respect to the initial coordinate frame at $k = 0$.

Formulation of the VO Problem

An agent is moving through an environment and taking images with a rigidly attached camera system at discrete time instants k . In case of a monocular system, the set of images taken at times k is denoted by $I_{0:n} = \{I_0, \dots, I_n\}$. In case of a stereo system, there are a left and a right image at every time instant, denoted by $I_{l,0:n} = \{I_{l,0}, \dots, I_{l,n}\}$ and $I_{r,0:n} = \{I_{r,0}, \dots, I_{r,n}\}$. Figure 1 shows an illustration of this setting.

For simplicity, the camera coordinate frame is assumed to be also the agent's coordinate frame. In case of a stereo system, without loss of generality, the coordinate system of the left camera can be used as the origin.

Two camera positions at adjacent time instants $k - 1$ and k are related by the rigid body transformation $T_{k,k-1} \in \mathbb{R}^{4 \times 4}$ of the following form:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}, \quad (1)$$

where $R_{k,k-1} \in SO(3)$ is the rotation matrix, and $t_{k,k-1} \in \mathbb{R}^{3 \times 1}$ the translation vector. The set $T_{1:n} = \{T_{1,0}, \dots, T_{n,n-1}\}$ contains all subsequent motions. To simplify the notation, from now on, T_k will be used instead of $T_{k,k-1}$. Finally, the set of camera poses $C_{0:n} = \{C_0, \dots, C_n\}$ contains the transformations of the camera with respect to the initial coordinate frame at $k = 0$. The current pose C_n can be computed by concatenating all the transformations T_k ($k = 1 \dots n$), and, therefore, $C_n = C_{n-1} T_n$, with C_0 being the camera pose at the instant $k = 0$, which can be set arbitrarily by the user.

The main task in VO is to compute the relative transformations T_k from the images I_k and I_{k-1} and then to concatenate the transformations to recover the full trajectory $C_{0:n}$ of the camera. This means that VO recovers the path incrementally, pose after pose. An iterative refinement over the last m poses can be performed after this step to obtain a more accurate estimate of the local trajectory. This iterative refinement works by minimizing the sum of the squared reprojection errors of the reconstructed 3-D points (i.e., the 3-D map) over the last m images (this is called *windowed-bundle adjustment*, because it is performed on a window of m frames. Bundle adjustment will be described in Part II of this tutorial). The 3-D points are obtained by triangulation of the image points (see the “Triangulation and Keyframe Selection” section).

As mentioned in the “Monocular VO” section, there are two main approaches to compute the relative motion T_k : appearance-based (or global) methods, which use the intensity information of all the pixels in the two input images, and feature-based methods, which only use salient and repeatable features extracted (or tracked) across the images. Global methods are less accurate than feature-based methods and are computationally more expensive. (As observed in the “History of VO” section, most appearance-based methods have been applied to monocular VO. This is due to ease of implementation compared with the

stereo camera case.) Feature-based methods require the ability to robustly match (or track) features across frames but are faster and more accurate than global methods. Therefore, most VO implementations are feature based.

The VO pipeline is summarized in Figure 2. For every new image i_k (or image pair in the case of a stereo camera), the first two steps consist of detecting and matching 2-D features with those from the previous frames. Two-dimensional features that are the reprojection of the same 3-D feature across different frames are called *image correspondences*. (As will be explained in Part II of this tutorial, we distinguish between feature matching and feature tracking. The first one consists of detecting features independently in all the images and then matching them based on some similarity metrics; the second one consists of finding features in one image and then tracking them in the next images using a local search technique, such as correlation.) The third step consists of computing the relative motion T_k between the time instants $k - 1$ and k . Depending on whether the correspondences are specified in three or two dimensions, there are three distinct approaches to tackle this problem (see the “Motion Estimation” section). The camera pose C_k is then computed by concatenation of T_k with the previous pose. Finally, an iterative refinement (bundle adjustment) can be done over the last m frames to obtain a more accurate estimate of the local trajectory.

Motion estimation is explained in this tutorial (see “Motion Estimation” section). Feature detection and matching and bundle adjustment will be described in Part II. Also, notice that for an accurate motion computation, feature correspondences should not contain outliers (i.e., wrong data associations). Ensuring accurate motion estimation in the presence of outliers is the task of robust estimation, which will be described in Part II of this tutorial. Most VO implementations assume that the camera is calibrated. To this end, the next section reviews the standard models and calibration procedures for perspective and omnidirectional cameras.

Camera Modeling and Calibration

VO can be done using both perspective and omnidirectional cameras. In this section, we review the main models.

Perspective Camera Model

The most used model for perspective camera assumes a pin-hole projection system: the image is formed by the intersection of the light rays from the objects through the center of the lens (projection center), with the focal plane [Figure 3(a)]. Let $X = [x, y, z]^\top$ be a scene point in the camera reference frame and $p = [u, v]^\top$ its projection on the image plane measured in pixels. The mapping from the 3-D world to the 2-D image is given by the perspective projection equation:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = KX = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad (2)$$

where λ is the depth factor, α_u and α_v the focal lengths, and u_0, v_0 the image coordinates of the projection center. These parameters are called *intrinsic parameters*. When the field of view of the camera is larger than 45° , the effects of the radial distortion may become visible and can be modeled using a second- (or higher)-order polynomial. The derivation of the complete model can be found in computer vision textbooks, such as [22] and [63]. Let $\tilde{p} = [\tilde{u}, \tilde{v}, 1]^\top = K^{-1}[u, v, 1]^\top$ be the normalized image coordinates. Normalized coordinates will be used throughout in the following sections.

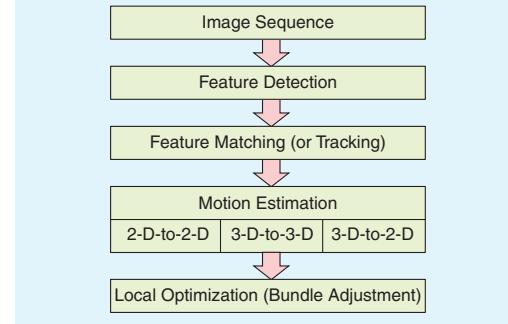


Figure 2. A block diagram showing the main components of a VO system.

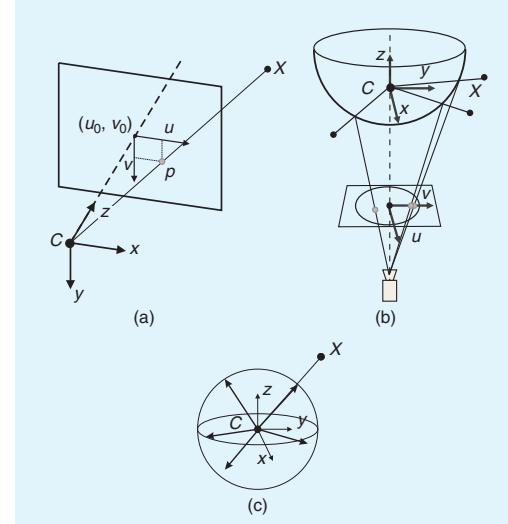


Figure 3. (a) Perspective projection, (b) catadioptric projection, and (c) a spherical model for perspective and omnidirectional cameras. Image points are represented as directions to the viewed points normalized on the unit sphere.

In GPS-denied environments, VO becomes of utmost importance.

Omnidirectional Camera Model

Omnidirectional cameras are cameras with wide field of view (even more than 180°) and can be built using fish-eye lenses or by combining standard cameras with mirrors [the latter are called *catadioptric cameras*, Figure 3(b)]. Typical mirror shapes in catadioptric cameras are quadratic surfaces of revolution (e.g., paraboloid or hyperboloid), because they guarantee a single projection center, which makes it possible to use the motion estimation theory presented in the “Motion Estimation” section.

Currently, there are two accepted models for omnidirectional cameras. The first one proposed by Geyer and Daniilidis [64] is for general catadioptric cameras (parabolic or hyperbolic), while the second one proposed by Scaramuzza et al. [65] is a unified model for both fish-eye and catadioptric cameras. A survey of these two models can be found in [66] and [67]. The projection equation of the unified model is as follows:

$$\lambda \begin{bmatrix} u \\ v \\ a_0 + a_1\rho + \dots + a_{n-1}\rho^{n-1} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad (3)$$

where $\rho = \sqrt{u^2 + v^2}$ and a_0, a_1, \dots, a_n are intrinsic parameters that depend on the type of mirror or fish-eye lens. As shown in [65], $n = 4$ is a reasonable choice for a large variety of mirrors and fish-eye lenses. Finally, this model assumes that the image plane satisfies the ideal property that the axes of symmetry of the camera and mirror are aligned. Although this assumption holds for most catadioptric and fish-eye cameras, misalignments can be modeled by introducing a perspective projection between the ideal and real-image plane [66].

Spherical Model

As mentioned earlier, it is desirable that the camera possesses a single projection center (also called *single effective viewpoint*). In a catadioptric camera, this happens when the rays reflected by the mirror intersect all in a single point (namely C). The existence of this point allows us to model any omnidirectional projection as a mapping from the single viewpoint to a sphere. For convenience, a unit sphere is usually adopted.

It is important to notice that the spherical model applies not only to omnidirectional cameras but also to perspective cameras. If the camera is calibrated, any point in the perspective or omnidirectional image can be mapped into a vector on the unit sphere. As can be observed in Figure 3(c), these unit vectors represent the directions to the viewed scene points. These vectors are called *normalized image points on the unit sphere*.

Camera Calibration

The goal of calibration is to accurately measure the intrinsic and extrinsic parameters of the camera system. In a multicamera system (e.g., stereo and trinocular), the extrinsic parameters describe the mutual position and orientation between each camera pair. The most popular method uses a planar checkerboard-like pattern. The position of the squares on the board is known. To compute the calibration parameters accurately, the user must take several pictures of the board shown at different positions and orientations by ensuring that the field of view of the camera is filled as much as possible. The intrinsic and extrinsic parameters are then found through a least-square minimization method. The input data are the 2-D positions of the corners of the squares of the board and their corresponding pixel coordinates in each image.

Many camera calibration toolboxes have been devised for MATLAB and C. An up-to-date list can be found in [68]. Among these, the most popular ones for MATLAB are given in [69] and [70]–[72]—for perspective and omnidirectional cameras, respectively. A C implementation of camera calibration for perspective cameras can be found in OpenCV [73], the open-source computer vision library.

Motion Estimation

Motion estimation is the core computation step performed for every image in a VO system. More precisely, in the motion estimation step, the camera motion between the current image and the previous image is computed. By concatenation of all these single movements, the full trajectory of the camera and the agent (assuming that the camera is rigidly mounted) can be recovered. This section explains how the transformation T_k between two images I_{k-1} and I_k can be computed from two sets of corresponding features f_{k-1}, f_k at time instants $k - 1$ and k , respectively. Depending on whether the feature correspondences are specified in two or three dimensions, there are three different methods.

- **2-D-to-2-D:** In this case, both f_{k-1} and f_k are specified in 2-D image coordinates.
- **3-D-to-3-D:** In this case, both f_{k-1} and f_k are specified in 3-D. To do this, it is necessary to triangulate 3-D points at each time instant; for instance, by using a stereo camera system.
- **3-D-to-2-D:** In this case, f_{k-1} are specified in 3-D and f_k are their corresponding 2-D reprojections on the image I_k . In the monocular case, the 3-D structure needs to be triangulated from two adjacent camera views (e.g., I_{k-2} and I_{k-1}) and then matched to 2-D image features in a third view (e.g., I_k). In the monocular scheme, matches over at least three views are necessary.

Notice that features can be points or lines. In general, due to the lack of lines in unstructured scenes, point features are used in VO. An in-depth review of these three approaches for both point and line features can be found in [74]. The formulation given in this tutorial is for point features only.

2-D to 2-D: Motion from Image Feature Correspondences

Estimating the Essential Matrix

The geometric relations between two images I_k and I_{k-1} of a calibrated camera are described by the so-called essential matrix E . E contains the camera motion parameters up to an unknown scale factor for the translation in the following form:

$$E_k \simeq \hat{t}_k R_k, \quad (4)$$

where $t_k = [t_x, t_y, t_z]^\top$ and

$$\hat{t}_k = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}. \quad (5)$$

The symbol \simeq is used to denote that the equivalence is valid up to a multiplicative scalar.

The essential matrix can be computed from 2-D-to-2-D feature correspondences, and rotation and translation can directly be extracted from E . The main property of 2-D-to-2-D-based motion estimation is the epipolar constraint, which determines the line on which the corresponding feature point \tilde{p}' of \tilde{p} lies in the other image (Figure 4). This constraint can be formulated by $\tilde{p}'^\top E \tilde{p} = 0$, where \tilde{p}' is a feature location in one image (e.g., I_k) and \tilde{p} is the location of its corresponding feature in another image (e.g., I_{k-1}). \tilde{p} and \tilde{p}' are normalized image coordinates. For the sake of simplicity, throughout the following sections, normalized coordinates in the form $\tilde{p} = [\tilde{u}, \tilde{v}, 1]^\top$ will be used (see the Perspective Camera Model" section). However, very similar equations can also be derived for normalized coordinates on the unit sphere (see the "Spherical Model" section).

The essential matrix can be computed from 2-D-to-2-D feature correspondences using the epipolar constraint. The minimal case solution involves five 2-D-to-2-D correspondences [75] and an efficient implementation proposed by Nister in [76]. Nister's five-point algorithm has become the standard for 2-D-to-2-D motion estimation in the presence of outliers (the problem of robust estimation will be tackled in Part II of this tutorial). A simple and straightforward solution for $n \geq 8$ noncoplanar points is the Longuet-Higgins' eight-point algorithm [2], which is summarized here. Each feature match gives a constraint of the following form:

$$[\tilde{u}\tilde{u}' \quad \tilde{u}'\tilde{v} \quad \tilde{u}' \quad \tilde{u}\tilde{v}' \quad \tilde{v}\tilde{v}' \quad \tilde{v}' \quad \tilde{u} \quad \tilde{v} \quad 1]E = 0, \quad (6)$$

where $E = [e_1 \ e_2 \ e_3 \ e_4 \ e_5 \ e_6 \ e_7 \ e_8 \ e_9]^\top$.

Stacking the constraints from eight points gives the linear equation system $AE = 0$, and by solving the system, the parameters of E can be computed. This homogeneous equation system can easily be solved using singular value decomposition (SVD) [2]. Having more than eight points leads to an overdetermined system to solve in the least-

squares sense and provides a degree of robustness to noise. The SVD of A has the form $A = USV^\top$, and the least-squares estimate of E with $\|E\| = 1$ can be found as the last column of V . However, this linear estimation of E does not fulfill the inner constraints of an essential matrix, which come from the multiplication of the rotation matrix R and the skew-symmetric translation matrix \hat{t} . These constraints are visible in the singular values of the essential matrix. A valid essential matrix after SVD is $E = USV^\top$ and has $\text{diag}(S) = \{s, s, 0\}$, which means that the first and second singular values are equal and the third one is zero. To get a valid E that fulfills the constraints, the solution needs to be projected onto the space of valid essential matrices. The projected essential matrix is $\bar{E} = U \text{diag}\{1, 1, 0\} V^\top$.

Observe that the solution of the eight-point algorithm is degenerate when the 3-D points are coplanar. Conversely, the five-point algorithm works also for coplanar points. Finally, observe that the eight-point algorithm works for both calibrated (perspective or omnidirectional) and uncalibrated (only perspective) cameras, whereas the five-point algorithm assumes that the camera (perspective or omnidirectional) is calibrated.

Extracting R and t from E

From the estimate of \bar{E} , the rotation and translation parts can be extracted. In general, there are four different solutions for R, t for one essential matrix; however, by triangulation of a single point, the correct R, t pair can be identified. The four solutions are

$$R = U(\pm W^\top)V^\top, \\ \hat{t} = U(\pm W)SU^\top,$$

where

$$W^\top = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (7)$$

An efficient decomposition of E into R and t is described in [76].

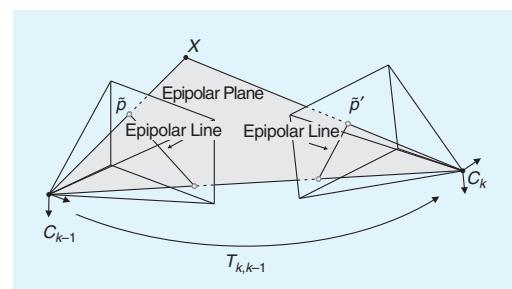


Figure 4. An illustration of the epipolar constraint.

After selecting the correct solution by triangulation of a point and choosing the solution where the point is in front of both cameras, a nonlinear optimization of the rotation and translation parameters should be performed using the estimate R , t as initial values. The function to minimize is the reprojection error defined in (10).

Computing the Relative Scale

To recover the trajectory of an image sequence, the different transformations $T_{0:n}$ have to be concatenated. To do this, the proper relative scales need to be computed as the absolute scale of the translation cannot be computed from two images. However, it is possible to compute relative scales for the subsequent transformations. One way of doing this is to triangulate 3-D points X_{k-1} and X_k from two subsequent image pairs. From the corresponding 3-D points, the relative distances between any combination of two 3-D points can be computed. The proper scale can then be determined from the distance ratio r between a point pair in X_{k-1} and a pair in X_k .

**Bundle adjustment
can be used to refine
the local estimate of
the trajectory.**

$$r = \frac{\|X_{k-1,i} - X_{k-1,j}\|}{\|X_{k,i} - X_{k,j}\|}. \quad (8)$$

For robustness, the scale ratios for many point pairs are computed and the mean (or in presence of outliers, the median) is used. The translation vector t is then scaled with this distance ratio. Observe that the relative-scale computation requires features to be matched (or tracked) over multiple frames (at least three). Instead of performing explicit triangulation of the 3-D points, the scale can also be recovered by exploiting the trifocal constraint between three-view matches of 2-D features [22].

The VO algorithm with the 2-D-to-2-D correspondences is summarized in Algorithm 1.

Algorithm 1. VO from 2-D-to-2-D correspondences.

- 1) Capture new frame I_k
- 2) Extract and match features between I_{k-1} and I_k
- 3) Compute essential matrix for image pair I_{k-1}, I_k
- 4) Decompose essential matrix into R_k and t_k , and form T_k
- 5) Compute relative scale and rescale t_k accordingly
- 6) Concatenate transformation by computing $C_k = C_{k-1} T_k$
- 7) Repeat from 1).

3D-to-3D: Motion from 3-D Structure Correspondences

For the case of corresponding 3-D-to-3-D features, the camera motion T_k can be computed by determining the aligning transformation of the two 3-D feature sets.

Corresponding 3-D-to-3-D features are available in the stereo vision case.

The general solution consists of finding the T_k that minimizes the L_2 distance between the two 3-D feature sets

$$\arg \min_{T_k} \sum_i \|\tilde{X}_k^i - T_k \tilde{X}_{k-1}^i\|, \quad (9)$$

where the superscript i denotes the i th feature, and \tilde{X}_k , \tilde{X}_{k-1} are the homogeneous coordinates of the 3-D points, i.e., $\tilde{X} = [x, y, z, 1]^\top$.

As shown in [77], the minimal case solution involves three 3-D-to-3-D noncollinear correspondences, which can be used for robust estimation in the presence of outliers (Part II of this tutorial). For the case of $n \geq 3$ correspondences, one possible solution (according to Arun et al. [78]) is to compute the translation part as the difference of the centroids of the 3-D feature sets and the rotation part using SVD. The translation is given by

$$t_k = \bar{X}_k - R\bar{X}_{k-1},$$

where $\bar{\cdot}$ stands for the arithmetic mean value.

The rotation can be efficiently computed using SVD as

$$R_k = VU^\top,$$

where $USV^\top = svd((X_{k-1} - \bar{X}_{k-1})(X_k - \bar{X}_k)^\top)$ and X_{k-1} and X_k are sets of corresponding 3-D points.

If the measurement uncertainties of the 3-D points are known, they can be added as weights into the estimation as described by Maimone et al. [17]. The computed transformations have absolute scale, and thus, the trajectory of a sequence can be computed by directly concatenating the transformations.

The VO algorithm with the 3-D-to-3-D correspondences is summarized in Algorithm 2.

Algorithm 2. VO from 3-D-to-3-D correspondences.

- 1) Capture two stereo image pairs $I_{l,k-1}, I_{r,k-1}$ and $I_{l,k}, I_{r,k}$
- 2) Extract and match features between $I_{l,k-1}$ and $I_{l,k}$
- 3) Triangulate matched features for each stereo pair
- 4) Compute T_k from 3-D features X_{k-1} and X_k
- 5) Concatenate transformation by computing $C_k = C_{k-1} T_k$
- 6) Repeat from 1).

To compute the transformation, it is also possible to avoid the triangulation of the 3-D points in the stereo camera and use quadrifocal constraints instead. This method was pointed out by Comport et al. [21]. The quadrifocal tensor allows computing the transformation directly from 2-D-to-2-D stereo correspondences.

3-D-to-2-D: Motion from 3-D Structure and Image Feature Correspondences

As pointed out by Nister et al. [1], motion estimation from 3-D-to-2-D correspondences is more accurate than from 3-D-to-3-D correspondences because it minimizes the image reprojection error (10) instead of the 3-D-to-3-D feature position error (9). The transformation T_k is computed from the 3-D-to-2-D correspondences X_{k-1} and p_k . X_{k-1} can be estimated from stereo data or, in the monocular case, from triangulation of the image measurements p_{k-1} and p_{k-2} . The latter, however, requires image correspondences across three views.

The general formulation in this case is to find T_k that minimizes the image reprojection error

$$\arg \min_{T_k} \sum_i \| p_k^i - \hat{p}_{k-1}^i \|^2, \quad (10)$$

where \hat{p}_{k-1}^i is the reprojection of the 3-D point X_{k-1}^i into image I_k according to the transformation T_k . This problem is known as *perspective from n points* (PnP) (or resection), and there are many different solutions to it in the literature [79]. As shown in [18], the minimal case involves three 3-D-to-2-D correspondences. This is called *perspective from three points* (P3P) and returns four solutions that can be disambiguated using one or more additional points. (A fast implementation of P3P is described in [80], and C code can be freely downloaded from the authors' Web page.) In the 3-D-to-2-D case, P3P is the standard method for robust motion estimation in the presence of outliers [18]. Robust estimation will be described in Part II of this tutorial.

A simple and straightforward solution to the PnP problem for $n \geq 6$ points is the direct linear transformation algorithm [22]. One 3-D-to-2-D point correspondence provides two constraints of the following form for the entries of $P_k = [R|t]$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -x & -y & -z & -1 & x\tilde{v} & y\tilde{v} & z\tilde{v} & \tilde{v} \\ x & y & z & 1 & 0 & 0 & 0 & 0 & -x\tilde{u} & -y\tilde{u} & -z\tilde{u} & -\tilde{u} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = 0, \quad (11)$$

where each P_j^\top is a four vector (the j th row of P_k) and x, y, z are the coordinates of the 3-D points X_{k-1} .

Stacking the constraints of six-point correspondences gives a linear system of equations of the form $AP = 0$. The entries of P can be computed from the nullvector of A , e.g., by using SVD. The rotation and translation parts can easily be extracted from $P_k = [R|t]$. The resulting rotation R is not necessarily orthonormal. However, this is not a problem since both R and t can be refined by nonlinear optimization of the reprojection error as defined in (10).

The 3-D-to-2-D motion estimation assumes that the 2-D image points only come from one camera. This means that for the case of a stereo camera, the 2-D image points are those of either the left or the right camera. Obviously, it is desirable to make use of the image points of both

cameras at the same time. A generalized version of the 3-D-to-2-D motion estimation algorithm for nonconcurrent rays (i.e., 2-D image points from multiple cameras) was proposed by Nister in [81] for extrinsically calibrated cameras (i.e., the mutual position and orientation between the cameras is known).

For the monocular case, it is necessary to triangulate 3-D points and estimate the pose from 3-D-to-2-D matches in an alternating fashion. This alternating scheme is often referred to as SFM. Starting from two views, the initial set of 3-D points and the first transformation are computed from 2-D-to-2-D feature matches. Subsequent transformations are then computed from 3-D-to-2-D feature matches. To do this, features need to be matched (or tracked) over multiple frames (at least three). New 3-D features are again triangulated when a new transformation is computed and added to the set of 3-D features. The main challenge of this method is to maintain a consistent and accurate set of triangulated 3-D features and to create 3-D-to-2-D feature matches for at least three adjacent frames.

The VO algorithm with 3-D-to-2-D correspondences is summarized in Algorithm 3.

**2-D-to-2-D and
3-D-to-2-D methods
are more accurate
than 3-D-to-3-D
methods.**

Algorithm 3. VO from 3-D-to-2-D Correspondences

- 1) Do only once:
 - 1.1) Capture two frames I_{k-2}, I_{k-1}
 - 1.2) Extract and match features between them
 - 1.3) Triangulate features from I_{k-2}, I_{k-1}
- 2) Do at each iteration:
 - 2.1) Capture new frame I_k
 - 2.2) Extract features and match with previous frame I_{k-1}
 - 2.3) Compute camera pose (PnP) from 3-D-to-2-D matches
 - 2.4) Triangulate all new feature matches between I_k and I_{k-1}
 - 2.5) Iterate from 2.1).

Triangulation and Keyframe Selection

Some of the previous motion estimation methods require triangulation of 3-D points (structure) from 2-D image correspondences. Structure computation is also needed by bundle adjustment (Part II of this tutorial) to compute a more accurate estimate of the local trajectory.

Triangulated 3-D points are determined by intersecting back-projected rays from 2-D image correspondences of at least two image frames. In perfect conditions, these rays would intersect in a single 3-D point. However, because of image noise, camera model and calibration errors, and

feature matching uncertainty, they never intersect. Therefore, the point at a minimal distance, in the least-squares sense, from all intersecting rays can be taken as an estimate of the 3-D point position. Notice that the standard deviation of the distances of the triangulated 3-D point from all rays gives an idea of the quality of the 3-D point. Three-dimensional points with large uncertainty will be thrown out. This happens especially when frames are taken at very nearby intervals compared with the distance to the scene points. When this occurs, 3-D points exhibit very large uncertainty. One way to avoid this consists of skipping frames until the average uncertainty of the 3-D points decreases below a certain threshold. The selected frames are called *keyframes*. Keyframe selection is a very important step in VO and should always be done before updating the motion.

Discussion

According to Nister et al. [1], there is an advantage in using the 2-D-to-2-D and 3-D-to-2-D methods compared to the 3-D-to-3-D method for motion computation. Nister compared the VO performance of the 3-D-to-3-D case to that of the 3-D-to-2-D case for a stereo camera system and found the latter being greatly superior to the former. The reason is due to the triangulated 3-D points being much more uncertain in the depth direction. When 3-D-to-3-D feature correspondences are used in motion computation, their uncertainty may have a devastating effect on the motion estimate. In fact, in the 3-D-to-3-D case, the 3-D position error, (9), is minimized whereas in the 3-D-to-2-D case the image reprojection error, (10).

In the monocular scheme, the 2-D-to-2-D method is preferable compared to the 3-D-to-2-D case since it avoids point triangulation. However, in practice, the 3-D-to-2-D method is used more often than the 2-D-to-2-D method. The reason lies in its faster data association. As will be described in Part II of this tutorial, for accurate motion computation, it is of utmost importance that the input data do not contain outliers. Outlier rejection is a very delicate step, and the computation time of this operation is strictly linked to the minimum number of points necessary to estimate the motion. As mentioned previously, the 2-D-to-2-D case requires a minimum of five-point correspondences (see the five-point algorithm); however, only three correspondences are necessary in the 3-D-to-2-D motion case (see P3P). As will be shown in Part II of this tutorial, this lower number of points results in a much faster motion estimation.

An advantage of the stereo camera scheme compared to the monocular one, besides the property that 3-D features are computed directly in the absolute scale, is that matches need to be computed only between two views instead of three views as in the monocular scheme. Additionally, since

the 3-D structure is computed directly from a single stereo pair rather than from adjacent frames as in the monocular case, the stereo scheme exhibits less drift than the monocular one in case of small motions. Monocular methods are interesting because stereo VO degenerates into the monocular case when the distance to the scene is much larger than the stereo baseline (i.e., the distance between the two cameras). In this case, stereo vision becomes ineffective and monocular methods must be used.

Regardless of the chosen motion computation method, local bundle adjustment (over the last m frames) should always be performed to compute a more accurate estimate of the trajectory. After bundle adjustment, the effects of the motion estimation method are much more alleviated.

Conclusions

This tutorial has described the history of VO, the problem formulation, and the distinct approaches to motion computation. VO is a well-understood and established part of robotics. Part II of this tutorial will summarize the remaining building blocks of the VO pipeline: how to detect and match salient and repeatable features across frames, robust estimation in the presence of outliers, and bundle adjustment. In addition, error propagation, applications, and links to free-to-download code will be included.

Acknowledgments

The authors thank Konstantinos Derpanis, Oleg Naroditsky, Carolin Baez, and Andrea Censi for their fruitful comments and suggestions.

References

- [1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2004, pp. 652–659.
- [2] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 10, pp. 133–135, 1981.
- [3] C. Harris and J. Pike, "3d positional integration from image sequences," in *Proc. Alvey Vision Conf.*, 1988, pp. 87–90.
- [4] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, "Building rome on a cloudless day," in *Proc. European Conf. Computer Vision*, 2010, pp. 368–381.
- [5] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1980.
- [6] L. Matthies and S. Shafer, "Error modeling in stereo navigation," *IEEE J. Robot. Automat.*, vol. 3, no. 3, pp. 239–248, 1987.
- [7] L. Matthies, "Dynamic stereo vision," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1989.
- [8] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo, "Rover self localization in planetary-like environments," in *Proc. Int. Symp. Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, 1999, pp. 433–440.
- [9] C. Olson, L. Matthies, M. Schoppers, and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, pp. 453–458.

-
- [10] M. Hannah, "Computer matching of areas in stereo images," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1974.
- [11] H. Moravec, "Towards automatic visual obstacle avoidance," in *Proc. 5th Int. Joint Conf. Artificial Intelligence*, Aug. 1977, p. 584.
- [12] W. Forstner, "A feature based correspondence algorithm for image matching," *Int. Arch. Photogrammetry*, vol. 26, no. 3, pp. 150–166, 1986.
- [13] C. Olson, L. Matthies, M. Schoppers, and M. Maimone, "Rover navigation using stereo ego-motion," *Robot. Autonom. Syst.*, vol. 43, no. 4, pp. 215–229, 2003.
- [14] A. Milella and R. Siegwart, "Stereo-based ego-motion estimation using pixel tracking and iterative closest point," in *Proc. IEEE Int. Conf. Vision Systems*, pp. 21–24, 2006.
- [15] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2008, pp. 3946–3952.
- [16] Y. Cheng, M. W. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," *IEEE Robot. Automat. Mag.*, vol. 13, no. 2, pp. 54–62, 2006.
- [17] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers: Field reports," *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.
- [18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [19] C. Tomasi and J. Shi, "Good features to track," in *Proc. Computer Vision and Pattern Recognition (CVPR '94)*, 1994, pp. 593–600.
- [20] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
- [21] A. Comport, E. Malis, and P. Rives, "Accurate quadrifocal tracking for robust 3d visual odometry," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2007, pp. 40–45.
- [22] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge U.K.: Cambridge Univ. Press, 2004.
- [23] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *J. Field Robot.*, vol. 23, no. 1, pp. 3–20, 2006.
- [24] P. I. Corke, D. Streleow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2005, pp. 4007–4012.
- [25] M. Lhuillier, "Automatic structure and motion using a catadioptric camera," in *Proc. IEEE Workshop Omnidirectional Vision*, 2005, pp. 1–8.
- [26] R. Goecke, A. Asthana, N. Pettersson, and L. Pettersson, "Visual vehicle egomotion estimation using the Fourier-Mellin transform," in *Proc. IEEE Intelligent Vehicles Symp.*, 2007, pp. 450–455.
- [27] J. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2008, pp. 2531–2538.
- [28] M. J. Milford and G. Wyeth, "Single camera vision-only SLAM on a suburban road network," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA '08)*, 2008, pp. 3684–3689.
- [29] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE Trans. Robot. (Special Issue on Visual SLAM)*, vol. 24, no. 5, pp. 1015–1026, Oct. 2008.
- [30] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2006, pp. 363–370.
- [31] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA '09)*, 2009, pp. 4293–4299.
- [32] A. Pretto, E. Menegatti, and E. Pagello, "Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2011, pp. 3289–3296.
- [33] D. Nister, "An efficient solution to the five-point relative pose problem," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2003, pp. 195–202.
- [34] M. Milford, G. Wyeth, and D. Prasser, "RatSLAM: A hippocampal model for simultaneous localization and mapping," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA '04)*, 2004, pp. 403–408.
- [35] B. Liang and N. Pears, "Visual navigation using planar homographies," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA '02)*, 2002, pp. 205–210.
- [36] Q. Ke and T. Kanade, "Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and ground-layer detection," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, June 2003, pp. 390–397.
- [37] H. Wang, K. Yuan, W. Zou, and Q. Zhou, "Visual odometry based on locally planar ground assumption," in *Proc. IEEE Int. Conf. Information Acquisition*, 2005, pp. 59–64.
- [38] J. Guerrero, R. Martinez-Cantin, and C. Sagues, "Visual map-less navigation based on homographies," *J. Robot. Syst.*, vol. 22, no. 10, pp. 569–581, 2005.
- [39] D. Scaramuzza, "1-point-RANSAC structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *Int. J. Comput. Vis.*, vol. 95, no. 1, pp. 74–85, 2011.
- [40] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart, "Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Kyoto, Oct. 2009, pp. 1413–1419.
- [41] F. Fraundorfer, D. Scaramuzza, and M. Pollefeys, "A constricted bundle adjustment parameterization for relative scale estimation in visual odometry," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 1899–1904.
- [42] N. Sunderhauf, K. Konolige, S. Lacroix, and P. Protzel, "Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle," in *Tagungsband Autonome Mobile Systeme, Reihe Informatik aktuell*. Levi, Schanz, Lafrenz, and Avrutin, Eds. Berlin, Springer-Verlag, 2005, pp. 157–163.
- [43] K. Konolige, M. Agrawal, and J. Sol, "Large scale visual odometry for rough terrain," in *Proc. Int. Symp. Robotics Research*, 2007.
- [44] J. Tardif, M. G. M. Laverne, A. Kelly, and M. Laverne, "A new approach to vision-aided inertial navigation," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010, pp. 4161–4168.
- [45] A. I. Mourikis and S. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2007, pp. 3565–3572.
- [46] E. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *Int. J. Robot. Res.*, vol. 30, no. 4, pp. 407–430, 2010.
- [47] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I. The essential algorithms," *Robot. Automat. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

- [48] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II. State of the art," *Robot. Automat. Mag.*, vol. 13, no. 3, pp. 108–117, 2006.
- [49] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. Int. Conf. Computer Vision*, 2003, pp. 1403–1410.
- [50] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proc. Int. Symp. Mixed and Augmented Reality*, 2007, pp. 225–234.
- [51] H. Strasdat, J. Montiel, and A. Davison, "Real time monocular SLAM: Why filter?" in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2657–2664.
- [52] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "3-D motion and structure from 2-D motion causally integrated over time: Implementation," in *Proc. European Conf. Computer Vision*, 2000, pp. 734–750.
- [53] M. C. Deans, "Bearing-only localization and mapping," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, 2002.
- [54] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardos, "Mapping large loops with a single hand-held camera," in *Proc. Robotics Science and Systems*, 2007.
- [55] T. Lemaire and S. Lacroix, "Vision-based SLAM: Stereo and monocular approaches," *Int. J. Computer Vision*, vol. 74, no. 3, pp. 343–364, 2006.
- [56] E. Eade and T. Drummond, "Monocular SLAM as a graph of coalesced observations," in *Proc. IEEE Int. Conf. Computer Vision*, 2007, pp. 1–8.
- [57] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. European Conf. Computer Vision*, 2008, pp. 802–815.
- [58] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1066–1077, 2008.
- [59] A. Handa, M. Chli, H. Strasdat, and A. J. Davison, "Scalable active matching," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2010, pp. 1546–1553.
- [60] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for ekf filtering: Application to real-time structure from motion and visual odometry," *J. Field Robot.*, vol. 27, no. 5, pp. 609–631, 2010.
- [61] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM," in *Proc. Robotics Science and Systems*, 2010.
- [62] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *Int. J. Computer Vision*, vol. 94, no. 2, pp. 198–214, 2010.
- [63] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3D Vision, from Images to Models*. Berlin: Springer-Verlag, 2003.
- [64] C. Geyer and K. Daniilidis, "A unifying theory for central panoramic systems and practical applications," in *Proc. European Conf. Computer Vision*, 2000, pp. 445–461.
- [65] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Proc. IEEE Int. Conf. Computer Vision Systems (ICVS) 2006*, Jan. 2006, pp. 45–53.
- [66] D. Scaramuzza, "Omnidirectional vision: From calibration to robot motion estimation" Ph.D. dissertation, ETH Zurich, 2008.
- [67] D. Scaramuzza, "Omnidirectional camera," in *Encyclopedia of Computer Vision*, K. Ikeuchi, Ed. Berlin: Springer-Verlag, 2012.
- [68] J. Bouguet (2011). List of camera calibration toolboxes. [Online]. Available: <http://www.vision.caltech.edu/bouguetj/calib/doc/htmls/links.html>
- [69] J. Bouguet. Camera calibration toolbox for Matlab. [Online]. Available: <http://www.vision.caltech.edu/bouguetj/calib/doc/index.html>
- [70] D. Scaramuzza. (2006). Ocamcalib toolbox: Omnidirectional camera calibration toolbox for Matlab (uses planar grids) google for "ocamcalib." [Online]. Available: <https://sites.google.com/site/scarabotix/ocamcalib-toolbox>
- [71] C. Mei. (2006). Omnidirectional camera calibration toolbox for Matlab (uses planar grids). [Online]. Available: <http://homepage-s.laas.fr/~cmei/index.php/Toolbox>
- [72] J. Barreto. Omnidirectional camera calibration toolbox for Matlab (uses lines). [Online]. Available: <http://www.isr.uc.pt/~jpbar/CatPack/pag1.htm>
- [73] OpenCV: Open-source computer vision library. [Online]. Available: <http://opencv.willowgarage.com/wiki/>
- [74] T. Huang and A. Netravalli, "Motion and structure from feature correspondences: A review," *Proc. IEEE*, vol. 82, no. 2, pp. 252–268, 1994.
- [75] E. Kruppa, "Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung," *Sitzungsberichte der Akademie der Wissenschaften, Wien, Mathematisch-Naturwissenschaftlichen Klasse, Abteilung IIa*, vol. 122, pp. 1939–1948, 1913.
- [76] D. Nister, "An efficient solution to the five-point relative pose problem," in *Proc. Computer Vision and Pattern Recognition (CVPR '03)*, 2003, pp. II: 195–202.
- [77] H. Goldstein, *Classical Mechanics*. Reading, MA: Addison-Wesley, 1981.
- [78] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 698–700, 1987.
- [79] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Accurate non-iterative O(n) solution to the PnP problem," in *Proc. IEEE Int. Conf. Computer Vision*, 2007, pp. 1–8.
- [80] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parameterization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011, pp. 2969–2976.
- [81] D. Nister, "A minimal solution to the generalised 3-point pose problem," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, pp. 560–567.
- [82] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vision Conf.*, 1988, pp. 147–151.

Davide Scaramuzza, GRASP Lab, Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia. E-mail: davide.scaramuzza@ieee.org.

Friedrich Fraundorfer, Computer Vision and Geometry Lab, Institute of Visual Computing, Department of Computer Science, ETH Zurich, Switzerland. E-mail: fraundorfer@inf.ethz.ch.

13.2 Visual Odometry Tutorial PII, RAM, 2012



© DIGITAL VISION

Visual Odometry

Part II: Matching, Robustness, Optimization, and Applications

By Friedrich Fraundorfer and Davide Scaramuzza

Visual odometry (VO) is the process of estimating the egomotion of an agent (e.g., vehicle, human, and robot) using the input of a single or multiple cameras attached to it. Application domains include robotics, wearable computing, augmented reality, and automotive. The term VO was popularized in 2004 by Nister in his landmark article [1], but already appeared earlier, e.g., [88], [89]. The term was chosen for its similarity to wheel odometry, which incrementally estimates the motion of a vehicle by integrating the number of turns of its wheels over time. Likewise, VO operates by incrementally estimating the pose of the vehicle through examination of the changes that movement induces on the images of its onboard cameras. For the VO to work effectively, there should be sufficient illumination in the environment and a static scene with sufficient texture to allow apparent motion to be extracted. Furthermore, consecutive frames should be captured by ensuring that they have sufficient scene overlap.

Digital Object Identifier 10.1109/MRA.2012.2182810
Date of publication: 16 February 2012

The advantage of VO with respect to wheel odometry is that VO is not affected by wheel slip in uneven terrain or other adverse conditions. It has been demonstrated that compared to wheel odometry, VO provides more accurate trajectory estimates, with the relative position error ranging from 0.1 to 2%. This capability makes VO an interesting supplement to wheel odometry and, additionally, other navigation systems such as global positioning system (GPS), inertial measurement units (IMUs), and laser odometry (similar to VO, laser odometry estimates the egomotion of a vehicle by scan matching of consecutive laser scans). In GPS-denied environments, such as underwater and aerial, VO has utmost importance.

This two-part tutorial and survey provides a broad introduction to VO and the research that has been undertaken from 1980 to 2011. Although the first two decades witnessed many offline implementations, only during the third decade did real-time working systems flourish, which has led VO to be used on another planet by two Mars-exploration rovers for the first time. Part I presented a historical review of the first 30 years of research in this field, a discussion on camera modeling and calibration,

and a description of the main motion-estimation pipelines for both monocular and binocular schemes, outlining the pros and cons of each implementation [87]. Part II (this tutorial) deals with feature matching, robustness, and applications. It reviews the main point-feature detectors used in VO and the different outlier-rejection schemes. Particular emphasis is given to the random sample consensus (RANSAC) and the strategies devised to speed it up are discussed. Other topics covered are error modeling, loop-closure detection (or location recognition), and bundle adjustment. Links to online, ready-to-use code are also given. The mathematical notation and concepts used in this article are defined in Part I of this tutorial and, therefore, are not repeated here.

Feature Selection and Matching

There are two main approaches to find feature points and their correspondences. The first one is to find features in one image and track them in the following images using local search techniques, such as correlation. The second one is to independently detect features in all the images and match them based on some similarity metric between their descriptors. The former approach is more suitable when the images are taken from nearby viewpoints, whereas the latter is more suitable when a large motion or viewpoint change is expected. Early research in VO is opted for the former approach [2]–[5] while the works in the last decade concentrated on the latter approach [1], [6]–[9]. The reason is that early works were conceived for small-scale environments, where images were taken from nearby viewpoints, while in the last few decades, the focus has shifted to large-scale environments, and so the images are taken as far apart as possible from each to limit the motion-drift-related issues.

Feature Detection

During the feature-detection step, the image is searched for salient keypoints that are likely to match well in other images. A local feature is an image pattern that differs from its immediate neighborhood in terms of intensity, color, and texture. For VO, point detectors, such as corners or blobs, are important because their position in the image can be measured accurately.

A corner is defined as a point at the intersection of two or more edges. A blob is an image pattern that differs from its immediate neighborhood in terms of intensity, color, and texture. It is not an edge, nor a corner. The appealing properties that a good feature detector should have are: localization accuracy (both in position and scale), repeatability (i.e., a large number of features should be redetected in the

next images), computational efficiency, robustness (to noise, compression artifacts, blur), distinctiveness (so that features can be accurately matched across different images), and invariance {to both photometric (e.g., illumination) and geometric changes [rotation, scale (zoom), perspective distortion]}.

The VO literature is characterized by many point-feature detectors, such as corner detectors (e.g., Moravec [2], Förstner [10], Harris [11], Shi-Tomasi [12], and FAST [13]) and blob detectors (SIFT [14], SURF [15], and CENSURE [16]). An overview of these detectors can be found in [17]. Each detector has its own pros and cons. Corner detectors are fast to compute but are less distinctive, whereas blob detectors are more distinctive but slower to detect. Additionally, corners are better localized in image position than blobs but are less localized in scale. This means that corners cannot be redetected as often as blobs after large changes in scale and viewpoint. However, blobs are not always the right choice in some environments—for instance, SIFT automatically neglects corners that urban environments are extremely rich of. For these reasons, the choice of the appropriate feature detector should be carefully considered, depending on the computational constraints, real-time requirements, environment type, and motion baseline (i.e., how nearby images are taken). An approximate comparison of properties and performance of different corner and blob detectors is given in Figure 1. Notice that SIFT, SURF, and CENSURE are not true affine invariant detectors but were empirically found to be invariant up to certain changes of the viewpoint. A performance evaluation of feature detectors and descriptors for indoor VO has been given in [18] and for outdoor environments in [9] and [19].

Every feature detector consists of two stages. The first is to apply a feature-response function on the entire image [such as the corner response function in the Harris detector or the difference-of-Gaussian (DoG) operator of the SIFT]. The second step is to apply nonmaxima suppression on the output of the first step. The goal is to identify all local minima (or maxima) of the feature-response function. The output of the nonmaxima suppression represents detected features. The trick to make a detector invariant to scale

	Corner Detector	Blob Detector	Rotation Invariant	Scale Invariant	Affine Invariant	Repeatability	Localization Accuracy	Robustness	Efficiency
Haris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
FAST	x		x	x		++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
SURF		x	x	x	x	+++	++	++	++
CENSURE		x	x	x	x	+++	++	+++	+++

Figure 1. Comparison of feature detectors: properties and performance.

changes consists in applying the detector at lower-scale and upper-scale versions of the same image [Figure 2(a)]. Invariance to perspective changes is instead attained by approximating the perspective distortion as an affine one.

SIFT is a feature devised for object and place recognition and found to give outstanding results for VO. The SIFT detector starts by convolving the upper and lower scales of the image with a DoG operator and then takes the local minima or maxima of the output across scales and space (Figure 2). The power of SIFT is in its robust descriptor, which will be explained in the following section. The SURF detector builds upon the SIFT but uses box filters to approximate the Gaussian, resulting in a faster computation compared to SIFT, which is achieved with integral images [90].

Feature Descriptor

In the feature description step, the region around each detected feature is converted into a compact descriptor that can be matched against other descriptors. The simplest descriptor of a feature is its appearance, that is, the intensity of the pixels in a patch around the feature point. In this case, error metrics such as the sum of squared differences (SSDs) or the normalized cross correlation (NCC) can be used to compare intensities [20]. Contrary to SSD, NCC compensates well for slight brightness changes. An alternative and more robust image similarity measure is the Census transform [21], which converts each image patch into a binary vector representing which neighbors have their intensity above or below the intensity of the central pixel. The patch similarity is then measured through Hamming distance.

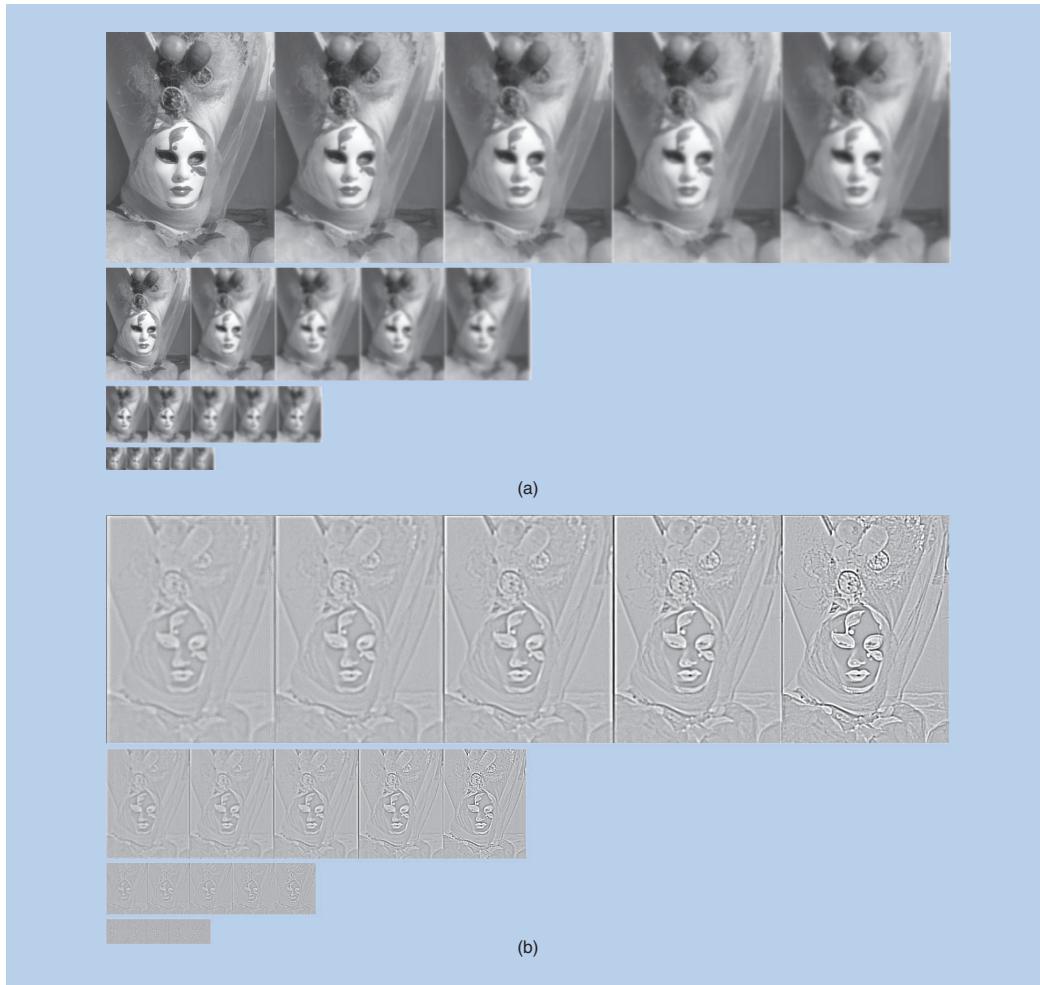


Figure 2. The original image (a, left) is smoothed with four Gaussian filters with different sigmas, and this is repeated after downsampling the image of a factor two. Finally, (b) DoG images are computed by taking the difference between successive Gaussian-smoothed images. SIFT features are found as local minima or maxima of DoG images across scales and space.

In many cases, the local appearance of the feature is not a good descriptor of the information carried by the feature because its appearance will change with orientation, scale, and viewpoint changes. In fact, SSD and NCC are not invariant to any of these changes, and, therefore, their use is limited to images taken at nearby positions. One of the most popular descriptors for point features is the SIFT. The SIFT descriptor is basically a histogram of local gradient orientations. The patch around the feature is decomposed into a 4×4 grid. For each quadrant, a histogram of eight gradient orientations is built. All these histograms are then concatenated together, forming a 128-element descriptor vector. To reduce the effect of illumination changes, the descriptor is then normalized to unit length.

The SIFT descriptor proved to be stable against changes in illumination, rotation, and scale, and even up to 60° changes in viewpoint. Example of SIFT features are shown in Figure 3. The orientation and scale of each feature is shown. The SIFT descriptor can, in general, be computed for corner or blob features; however, its performance will decrease on corners because, by definition, corners occur at the intersection of edges. Therefore, its descriptor won't be as distinctive as for blobs, which, conversely, lie in highly textured regions of the image.

Between 2010 and 2011, three new descriptors have been devised, which are much faster to compute than SIFT and SURF. A simple binary descriptor named BRIEF [22] became popular: it uses pairwise brightness comparisons sampled from a patch around the keypoint. While extremely fast to extract and compare, it still exhibits high discriminative power in the absence of rotation and scale change. Inspired by its success, ORB [23] was developed, which tackles orientation invariance and an optimization of the sampling scheme for the brightness value pairs. Along the same lines, BRISK [24] provides a keypoint detector based on FAST, which allows scale and rotation invariance, and a binary descriptor that uses a configurable sampling pattern.

Feature Matching

The feature-matching step searches for corresponding features in other images. Figure 4 shows the SIFT features matched across multiple frames overlaid on the first image. The set of matches corresponding to the same feature is called *feature track*. The simplest way for matching features between two images is to compare all feature descriptors in the first image to all other feature descriptors in the second image. Descriptors are

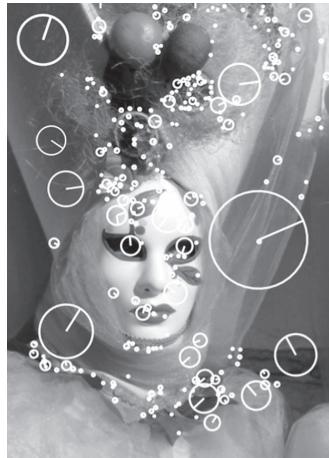


Figure 3. SIFT features shown with orientation and scale.

compared using a similarity measure. If the descriptor is the local appearance of the feature, then a good measure is the SSD or NCC. For SIFT descriptors, this is the Euclidean distance.

Mutual Consistency Check

After comparing all feature descriptors between two images, the best correspondence of a feature in the second image is chosen as that with the closest descriptor (in terms of distance or similarity). However, this stage may result with features in the second image matching with more than one feature in the first image. To decide which match to accept, the mutual consistency check can be used. This consists in pairing every feature in the second image with features in the first image. Only pairs of corresponding

features that mutually have each other as a preferred match are accepted as correct.

Constrained Matching

A disadvantage of this exhaustive matching is that it is quadratic in the number of features, which can become impractical when the number of features is large (e.g., several thousands). A better approach is to use an indexing structure, such as a multidimensional search tree or a hash table, to rapidly search for features near a given feature. A faster feature matching is to search for potential correspondences in regions of the second image where they are expected to be. These regions can be predicted using a motion model and the three-dimensional (3-D) feature position (if available). For instance, this is the case in the 3-D-to-2-D-based motion estimation described in Part I of this tutorial. The motion can be given by an additional sensor like IMU, wheel odometry [25], laser, and GPS, or can be inferred from the previous position assuming a constant velocity model, as proposed in [26]. The



Figure 4. SIFT-feature tracks.

predicted region is then calculated as an error ellipse from the uncertainty of the motion and that of the 3-D point.

Alternatively, if only the motion model is known but not the 3-D feature position, the corresponding match can be searched along the epipolar line in the second image. This process is called *epipolar matching*. As can be observed in Figure 5, a single 2-D feature and the two camera centers define a plane in the 3-D space that intersect both images into two lines, called *epipolar lines*. An epipolar line can be computed directly from a 2-D feature and the relative motion of the camera, as explained in Part I of this tutorial. Each feature in the first image has a different epipolar line in the second image.

In stereovision, instead of computing the epipolar line for each candidate feature, the images are usually rectified. Image rectification is a remapping of an image pair into a new image pair where epipolar lines of the left and right images are horizontal and aligned to each other. This has the advantage of facilitating image-correspondence search since epipolar lines no longer have to be computed for each feature: the correspondent of one feature in the left (right) image can be searched across those features in the right (left) image, which lie on the same row. Image rectification can be executed efficiently on graphics processing units (GPUs). In stereovision, the relative position between the two cameras is known precisely. However, if the motion is affected by uncertainty, the epipolar search is usually expanded to a rectangular area within a certain distance from the epipolar line. In stereovision, SSD, NCC, and Census transform are the widely used similarity metrics for epipolar matching [91].

Feature Tracking

An alternative to independently finding features in all candidate images and then matching them is to detect features in the first image and, then, search for their corresponding matches in the following images. This detect-then-track approach is suitable for VO applications where images are taken at nearby locations, where the amount of motion and appearance deformation between adjacent frames is small. For this particular application, SSD and NCC can work well.

However, if features are tracked over long image sequences, their appearance can undergo larger changes.

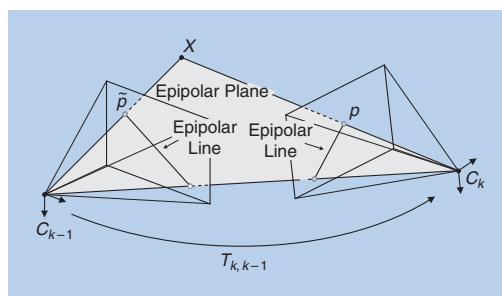


Figure 5. Illustration of the epipolar constraint.

In this case, a solution is to apply an affine-distortion model to each feature. The resulting tracker is often called *KanadeLucasTomasi* (KLT) tracker [12].

Discussion

SIFT Matching

For SIFT feature matching, a distance-ratio test was proposed by the authors initially, for use in place and object detection [14]. This distance-ratio test accepts the closest match (the one with minimum Euclidean distance) only if the ratio between the closest and the second closest match is smaller than a user-specified threshold. The idea behind this test is to remove matches that might be ambiguous, e.g., due to repetitive structure. The threshold for the test can only be set heuristically and an unlucky guess might remove correct matches as well. Therefore, in many cases, it might be beneficial to skip the ratio test and let RANSAC take care of the outliers as explained in the “Outlier Removal” section.

Lines and Edgelets

An alternative to point features for VO is to use lines or edgelets, as proposed in [27] and [28]. They can be used in addition to points in structured environments and may provide additional cues, such as direction (of the line or edgelet), and planarity and orthogonality constraints. Contrary to points, lines are more difficult to match because lines are more likely to be occluded than points. Furthermore, the origin and end of a line segment of edgelet may not exist (e.g., occlusions and horizon line).

Number of Features and Distribution

The distribution of the features in the image has been found to affect the VO results remarkably [1], [9], [29]. In particular, more features provide more stable motion-estimation results than with fewer features, but at the same time, the keypoints should cover the image as evenly as possible. To do this, the image can be partitioned into a grid, and the feature detector is applied to each cell by tuning the detection thresholds until a minimum number of features are found in each subimage [1]. As a rule of the thumb, 1,000 features is a good number for a 640×480 -pixel image.

Dense and Correspondence-Free Methods

An alternative to sparse-feature extraction is to use dense methods, such as optical flow [30], or feature-less methods [31]. Optical flow aims at tracking, ideally, each individual pixel or a subset of the whole image (e.g., all pixels on a grid specified by the user). However, similar to feature tracking, it assumes small motion between frames and, therefore, is not suitable for VO applications since motion error accumulates quickly. Another alternative is feature-less motion-estimation methods, such as [31]: all the pixels in the two images are used to compute the relative motion using a harmonic Fourier transform. This method has the advantage to work especially with low-texture images but

is computationally extremely expensive (can take up to several minutes), and the recovered motion is less accurate than with feature-based methods.

Outlier Removal

Matched points are usually contaminated by outliers, that is, wrong data associations. Possible causes of outliers are image noise, occlusions, blur, and changes in viewpoint and illumination for which the mathematical model of the feature detector or descriptor does not account for. For instance, most of the feature-matching techniques assume linear illumination changes, pure camera rotation and scaling (zoom), or affine distortion. However, these are just mathematical models that approximate the more complex reality (image saturation, perspective distortion, and motion blur). For the camera motion to be estimated accurately, it is important that outliers be removed. Outlier rejection is the most delicate task in VO. An example VO result before and after removing the outliers is shown in Figure 6.

RANSAC

The solution to outlier removal consists in taking advantage of the geometric constraints introduced by the motion model. Robust estimation methods, such as M-estimation [32], case deletion, and explicitly fitting and removing outliers [33], can be used but these often work only if there are relatively few outliers. RANSAC [34] has been established as the standard method for model estimation in the presence of outliers.

The idea behind RANSAC is to compute model hypotheses from randomly sampled sets of data points and then verify these hypotheses on the other data points. The hypothesis that shows the highest consensus with the other data is selected as a solution. For two-view motion estimation as used in VO, the estimated model is the relative motion (R, t) between the two camera positions, and the data points are the candidate feature correspondences.

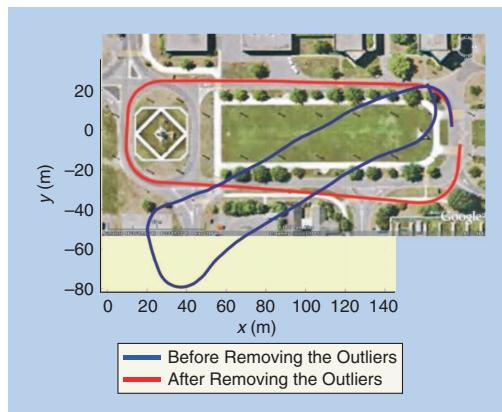


Figure 6. Comparison between VO trajectories estimated before and after removing the outliers. (Photo courtesy of Google Maps © 2007 Google, © 2007 Tele Atlas.)

Inlier points to a hypothesis are found by computing the point-to-epipolar line distance [35]. The point-to-epipolar line distance is usually computed as a first-order approximation—called *Sampson distance*—for efficiency reasons [35]. An alternative to the point-to-epipolar line distance is the directional error proposed by Oliensis [36]. The directional error measures the angle between the ray of the image feature and the epipolar plane. The authors claim that the use of the directional error is advantageous for the case of omnidirectional and wide-angle cameras but also beneficial for the standard camera case.

The outline of RANSAC is given in Algorithm 1.

Algorithm 1. VO from 2-D-to-2-D correspondences.

- 1) Initial: let A be a set of N feature correspondences
- 2) Repeat
- 2.1) Randomly select a sample of s points from A
- 2.2) Fit a model to these points
- 2.3) Compute the distance of all other points to this model
- 2.4) Construct the inlier set (i.e. count the number of points whose distance from the model $< d$)
- 2.5) Store these inliers
- 2.6) Until maximum number of iterations reached
- 3) The set with the maximum number of inliers is chosen as a solution to the problem
- 4) Estimate the model using all the inliers.

The number of subsets (iterations) N that is necessary to guarantee that a correct solution is found can be computed by

$$N = \frac{\log(1 - p)}{\log(1 - (1 - \epsilon)^s)}, \quad (1)$$

where s is the number of data points from which the model can be instantiated, ϵ is the percentage of outliers in the data points, and P is the requested probability of success [34]. For the sake of robustness, in many practical implementations, N is usually multiplied by a factor of ten. More advanced implementations of RANSAC estimate the fraction of inliers adaptively, iteration after iteration.

As observed, RANSAC is a probabilistic method and is nondeterministic in that it exhibits a different solution on different runs; however, the solution tends to be stable when the number of iterations grows.

Minimal Model Parameterizations

As can be observed in Figure 7, N is exponential in the number of data point s necessary to estimate the model. Therefore, there is a high interest in using a minimal parameterization of the model. In Part I of this tutorial, an eight-point minimal solver for uncalibrated cameras was described. Although it works also for calibrated cameras, the eight-point algorithm fails when the scene points are

coplanar. However, when the camera is calibrated, its six degrees of freedom (DoF) motion can be inferred from a minimum of five-point correspondences, and the first solution to this problem was given in 1913 by Kruppa [37]. Several five-point minimal solvers were proposed later in [38]–[40], but an efficient implementation, based on [39], was found only in 2003 by Nister [41] and later revised in [42]. Before that, the six-, [43], seven-, [44], or eight- solvers were commonly used. However, the five-point solver has the advantage that it works also for planar scenes. (Observe that eight- and seven-point solvers work for uncalibrated, perspective cameras. To use them also with omnidirectional cameras, the camera needs to be calibrated. Alternatively, n -point solvers for uncalibrated omnidirectional cameras have also been proposed [45]–[47], where n depends on the type of mirror or fish eye used. Lim et al. [48] showed that, for calibrated omnidirectional cameras, 6 DoF motion can be recovered using only two pairs of antipodal image points. Antipodal image points are points whose rays are aligned but which correspond to opposite viewing directions. They also showed that antipodal points allow us to independently estimate translation and rotation.)

Despite the five-point algorithm represents the minimal solver for 6 DoF motion of calibrated cameras, in the last few decades, there have been several attempts to exploit different cues to reduce the number of motion parameters. In [49], Fraundorfer et al. proposed a three-point minimal solver for the case of two known camera-orientation

angles. For instance, this can be used when the camera is rigidly attached to a gravity sensor (in fact, the gravity vector fixes two camera-orientation angles). Later, Naroditsky et al. [50] improved on that work by showing that the three-point minimal solver can be used in a four-point (three-plus-one) RANSAC scheme. The three-plus-one stands for the fact that an additional far scene point (ideally, a point at infinity) is used to fix the two orientation angles. Using their four-point RANSAC, they also show a successful 6 DoF VO. A two-point minimal solver for 6-DoF VO was proposed by Kneip et al. [51], which uses the full rotation matrix from an IMU rigidly attached to the camera.

In the case of planar motion, the motion model complexity is reduced to 3 DoF and can be parameterized with two points as described in [52]. For wheeled vehicles, Scaramuzza et al. [9], [53] showed that the motion can be locally described as planar and circular, and, therefore, the motion model complexity is reduced to 2 DoF, leading to a one-point minimal solver. Using a single point for motion estimation is the lowest motion parameterization possible and results in the most efficient RANSAC algorithm. Additionally, they show that, by using histogram, voting outliers can be found in a small, single iteration. A performance evaluation of five-, two-, and one-point RANSAC algorithms for VO was finally presented in [54].

To recap, the reader should remember that, if the camera motion is unconstrained, the minimum number of points to estimate the motion is five, and, therefore, the five-point RANSAC (or the six-, seven-, or eight-point one) should be used. Of course, using the five-point RANSAC will require less iterations (and thus less time) than the six-, seven-, or eight-point RANSAC. A summary of the number of minimum RANSAC iterations as a function of the number of model parameters s is shown in Table 1 for the eight-, seven-, five-, four-, two-, one-point minimal solvers. These values were obtained from (1), assuming a probability of success $P = 99\%$ and a percentage of outliers $\epsilon = 50\%$.

Reducing the Iterations of RANSAC

As can be observed in Table 1, with $P = 99\%$ and $\epsilon = 50\%$, the five-point RANSAC requires a minimum of 145 iterations. However, in reality, the things are not always so straightforward. Sometimes, the number of outliers is underestimated and using more iterations increases the chances to find more inliers. In some cases, it can even be necessary to allow for thousands of iterations. Because of this, several works have been produced in the endeavor of increasing the speed of RANSAC.

The maximum likelihood estimation sample consensus [55] makes the measurement of correspondences more reliable and improves the estimate of the hypotheses. The progressive sample consensus [56]

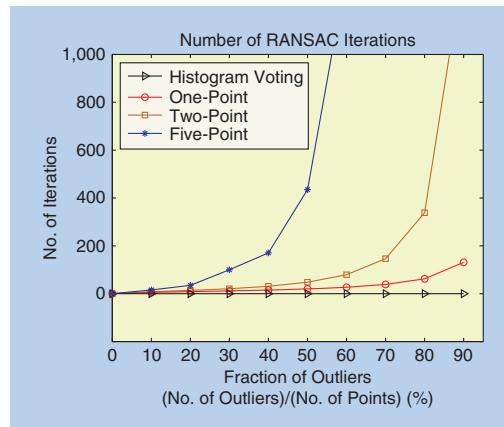


Figure 7. Number of RANSAC iterations versus fraction of outliers.

Table 1. Number of RANSAC iterations.

Number of points (s):	8	7	6	5	4	2	1
Number of iterations (N):	1,177	587	292	145	71	16	7

ranks the correspondences based on their similarity and generates motion hypotheses starting from points with higher rank. Preemptive RANSAC [57] uses preemptive scoring of the motion hypotheses and a fixed number of iterations. Uncertainty RANSAC [58] incorporates feature uncertainty and shows that this determines a decrease in the number of potential outliers, thus enforcing a reduction in the number of iterations. In [59], a deterministic RANSAC approach is proposed, which also estimates the probability that a match is correct.

What all the mentioned algorithms have in common is that the motion hypotheses are directly generated from the points. Conversely, other algorithms operate by sampling the hypotheses from a proposal distribution of the vehicle motion model [60], [61].

Among all these algorithms, preemptive RANSAC has been the most popular one because the number of iterations can be fixed a priori, which has several advantages when real-time operation is necessary.

Is It Really Better to Use a Minimal Set in RANSAC?

If one is concerned with certain speed requirements, using a minimal point set is definitely better than using a nonminimal set. However, even the five-point RANSAC might not be the best idea if the image correspondences are very noisy. In this case, using more points than a minimal set is proved to give better performance (in terms of accuracy and number of inliers) [62], [63]. To understand it, consider a single iteration of the five-point RANSAC: at first, five random points are selected and used to estimate the motion model; second, this motion hypothesis is tested on all other points. If the selected five points are inliers with large image noise, the motion estimated from them will be inaccurate and will exhibit fewer inliers when tested on all the other points. Conversely, if the motion is estimated from more than five points using the five-point solver, the effects of noise are averaged and the estimated model will be more accurate, with the effect that more inliers will be identified. Therefore, when the computational time is not a real concern and one deals with noisy features, using a nonminimal set may be better than using a minimal set [62].

Error Propagation

In VO, individual transformations $T_{k,k-1}$ are concatenated to form the current pose of the robot C_k (see Part I of this tutorial). Each of these transformations $T_{k,k-1}$ has an uncertainty, and the uncertainty of the camera pose C_k depends on the uncertainty of past transformations. This is illustrated in Figure 8. The uncertainty of the transformation $T_{k+1,k}$ computed by VO depends on camera geometry and the image features. A derivation for the stereo case can be found in [3].

In the following, the uncertainty propagation is discussed. Each camera pose C_k and each transformation $T_{k,k-1}$ can be represented by a six-element vector containing

the position (x, y, z) and orientation (in Euler angles ϕ, θ, ψ). These six-element vectors are denoted by \vec{C}_k and $\vec{T}_{k,k-1}$, respectively, e.g., $\vec{C}_k = (x, y, z, \phi, \theta, \psi)^\top$. Each transformation $\vec{T}_{k,k-1}$ is represented by its mean and covariance $\Sigma_{k,k-1}$. The covariance matrix $\Sigma_{k,k-1}$ is a 6×6 matrix. The camera pose C_k is written as $\vec{C}_k = f(\vec{C}_{k-1}, \vec{T}_{k,k-1})$, that is a function of the previous pose \vec{C}_{k-1} and the transformation $\vec{T}_{k,k-1}$ with their covariances Σ_{k-1} and $\Sigma_{k,k-1}$, respectively. The combined covariance matrix \vec{C}_k is a 12×12 matrix and a compound of the covariance matrices $\Sigma_{k,k-1}$ and Σ_{k-1} . \vec{C}_k can be computed by using the error propagation law [64], which uses a first-order Taylor approximation; therefore,

$$\Sigma_k = J \begin{bmatrix} \Sigma_{k-1} & 0 \\ 0 & \Sigma_{k,k-1} \end{bmatrix} J^\top \quad (2)$$

$$= J_{\vec{C}_{k-1}} \Sigma_{k-1} J_{\vec{C}_{k-1}}^\top + J_{\vec{T}_{k,k-1}} \Sigma_{k,k-1} J_{\vec{T}_{k,k-1}}^\top, \quad (3)$$

where $J_{\vec{C}_{k-1}}$, $J_{\vec{T}_{k,k-1}}$ are the Jacobians of f with respect to \vec{C}_{k-1} and $\vec{T}_{k,k-1}$, respectively. As can be observed from this equation, the camera-pose uncertainty is always increasing when concatenating transformations. Thus, it is important to keep the uncertainties of the individual transformations small to reduce the drift.

Camera Pose Optimization

VO computes the camera poses by concatenating the transformations, in most cases from two subsequent views at times k and $k-1$ (see Part I of this tutorial). However, it might also be possible to compute transformations between the current time k and the n last time steps $T_{k,k-2}, \dots, T_{k,k-n}$, or even for any time step $T_{i,j}$. If these transformations are known, they can be used to improve the camera poses by using them as additional constraints in a pose-graph optimization.

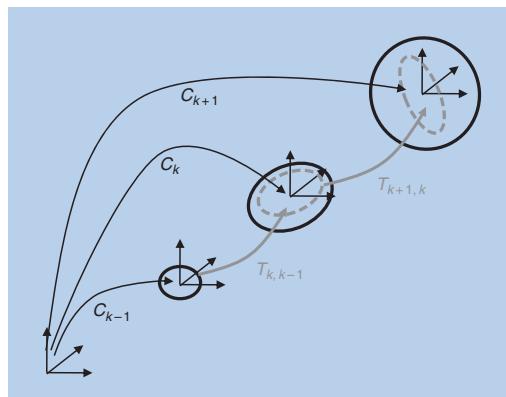


Figure 8. The uncertainty of the camera pose at C_k is a combination of the uncertainty at C_{k-1} (black solid ellipse) and the uncertainty of the transformation $T_{k,k-1}$ (gray dashed ellipse).

Pose-Graph Optimization

The camera poses computed from VO can be represented as a pose graph, which is a graph where the camera poses are the nodes and the rigid-body transformations between the camera poses are the edges between nodes [65]. Each additional transformation that is known can be added as an edge into the pose graph. The edge constraints e_{ij} define the following cost function:

$$\sum_{e_{ij}} \|C_i - T_{e_{ij}} C_j\|^2, \quad (4)$$

where $T_{e_{ij}}$ is the transformation between the poses i and j . Pose graph optimization seeks the camera pose parameters that minimize this cost function. The rotation part of the transformation makes the cost function nonlinear, and a nonlinear optimization algorithm (e.g., Levenberg-Marquardt) has to be used.

Loop Constraints for Pose-Graph Optimization

Loop constraints are valuable constraints for pose graph optimization. These constraints form graph edges between nodes that are usually far apart and between which large drift might have been accumulated. Commonly, events like reobserving a landmark after not seeing it for a long time or coming back to a previously mapped area are called *loop detections* [66]. Loop constraints can be found by evaluating visual similarity between the current camera images and past camera images. Visual similarity can be computed using global image descriptors (e.g., [67] and [68]) or local image descriptors (e.g., [69]). Recently, loop detection by visual similarity using local image descriptors got a lot of attention and one of the most successful methods are based on the so-called visual words [70]–[73]. In these approaches, an image is represented by a bag of visual words. The visual similarity between two images is then computed as the distance of the visual word histograms of the two images. The visual word-based approach is extremely efficient to compute visual similarities between large sets of image data, a property important for loop detection. A visual word represents a high-dimensional feature descriptor (e.g., SIFT or SURF) with a single integer number. For this quantization, the original high-dimensional descriptor space is divided into nonoverlapping cells by k -means clustering [74], which is called the *visual vocabulary*. All feature descriptors that fall within the same cell will get the cell number assigned, which represents the visual word. Visual-word-based similarity computation is often accelerated by organizing the visual-word database as an inverted-file data structure [75] that makes use of the finite range of visual vocabulary. Visual similarity computation is the first step of loop detection. After finding the top- n similar images, usually a geometric verification using the epipolar constraint is performed and, for confirmed matches, a rigid-body transformation is computed using wide-baseline feature matches between the

two images. This rigid-body transformation is added to the pose graph as an additional loop constraint.

Windowed (or Local) Bundle Adjustment

Windowed bundle adjustment [76] is similar to pose-graph optimization as it tries to optimize the camera parameters but, in addition, it also optimizes the 3-D-landmark parameters at the same time. It is applicable to the cases where image features are tracked over more than two frames. Windowed bundle adjustment considers a so-called window of n image frames and then performs a parameter optimization of camera poses and 3-D landmarks for this set of image frames. In bundle adjustment, the error function to minimize is the image reprojection error:

$$\arg \min_{X^i, C_k} \sum_{i,k} \|p_k^i - g(X^i, C_k)\|^2, \quad (5)$$

where p_k^i is the i th image point of the 3-D landmark X^i measured in the k th image and $g(X^i, C_k)$ is its image reprojection according to the current camera pose C_k .

The reprojection error is a nonlinear function, and the optimization is usually carried out using Levenberg-Marquardt. This requires an initialization that is close to the minimum. Usually, a standard two-view VO solution serves as initialization. The Jacobian for this optimization problem has a specific structure that can be exploited for efficient computation [76].

Windowed bundle adjustment reduces the drift compared to two-view VO because it uses feature measurements over more than two image frames. The current camera pose is linked via the 3-D landmark, and the image feature tracks not only the previous camera pose but also the camera poses further back. The current and $n - 1$ previous camera poses need to be consistent with the measurements over n image frames. The choice of the window size n is mostly governed by computational reasons. The computational complexity of bundle adjustment in general is $O((qM + lN)^3)$ with M and N being the number of points and cameras poses and q and l the number of parameters for points and camera poses. A small window size limits the number of parameters for the optimization and thus makes real-time bundle adjustment possible. It is possible to reduce the computational complexity by just optimizing over the camera parameters and keeping the 3-D landmarks fixed, e.g., if the 3-D landmarks are accurately triangulated from a stereo setup.

Applications

VO has been successfully applied within various fields. It is used for egomotion estimation for space exploration (e.g., computing the egomotion of Mars Rovers [25] and that of a planetary lander in the decent phase [77]) and can also be found in consumer hardware, e.g., the Dacuda scanner mouse [78].

Table 2. Software and data sets.

Author	Description	Link
Willow Garage	OpenCV: A computer vision library maintained by Willow Garage. The library includes many of the feature detectors mentioned in this tutorial (e.g., Harris, KLT, SIFT, SURF, FAST, BRIEF, ORB). In addition, the library contains the basic motion-estimation algorithms as well as stereo-matching algorithms.	http://opencv.willowgarage.com
Willow Garage	Robot operating system (ROS): A huge library and middleware maintained by Willow Garage for developing robot applications. Contains a VO package and many other computer-vision-related packages.	http://www.ros.org
Willow Garage	Point cloud library (PCL): A 3-D-data-processing library maintained from Willow Garage, which includes useful algorithms to compute transformations between 3-D-point clouds.	http://pointclouds.org
Henrik Stewenius et al.	Five-point algorithm: An implementation of the five-point algorithm for computing the essential matrix.	http://www.vis.uky.edu/~stewe/FIVEPOINT/
Changchang Wu et al.	SiftGPU: Real-time implementation of SIFT.	http://cs.unc.edu/~ccwu/siftgpu
Nico Cornelis et al.	GPUSurf: Real-time implementation of SURF.	http://homes.esat.kuleuven.be/~ncorneli/gpusurf
Christopher Zach	GPU-KLT: Real-time implementation of the KLT tracker.	http://www.inf.ethz.ch/personal/chzachopensource.html
Edward Rosten	Original implementation of the FAST detector.	http://www.edwardrosten.com/work/fast.html
Michael Calonder	Original implementation of the BRIEF descriptor.	http://cvlab.epfl.ch/software/brief/
Leutenegger et al.	BRISK feature detector.	http://www.asl.ethz.ch/people/lestefan/personal/BRISK
Jean-Yves Bouguet	Camera Calibration Toolbox for MATLAB.	http://www.vision.caltech.edu/bouguetj/calib_doc
Davide Scaramuzza	OCamCalib: Omnidirectional Camera Calibration Toolbox for MATLAB.	https://sites.google.com/site/scarabotix/ocamcalib-toolbox
Christopher Mei	Omnidirectional camera calibration toolbox for MATLAB	http://homepages.laas.fr/~cmei/index.php/Toolbox
Mark Cummins	Fast appearance-based mapping: Visual-word-based loop detection.	http://www.robots.ox.ac.uk/~mjc/Software.htm
Friedrich Fraundorfer	Vocsearch: Visual-word-based place recognition and image search.	http://www.inf.ethz.ch/personal/fraundof/page2.html
Manolis Lourakis	Sparse bundle adjustment (SBA)	http://www.ics.forth.gr/~lourakis/sba
Christopher Zach	Simple sparse bundle adjustment (SSBA)	http://www.inf.ethz.ch/personal/chzachopensource.html
Rainer Kuemmerle et al.	G2O: Library for graph-based nonlinear function optimization. Contains several variants of SLAM and bundle adjustment.	http://openslam.org/g2o
RAWSEEDS EU Project	RAWSEEDS: Collection of data sets with different sensors (lidars, cameras, and IMUs) with ground truth.	http://www.rawseeds.org
SFLY EU Project	SFLY-MAV data set: Camera-IMU data set captured from an aerial vehicle with Vicon data for ground truth.	http://www.sfly.org
Davide Scaramuzza	ETH OMNI-VO: An omnidirectional-image data set captured from the roof of a car for several kilometers in a urban environment. MATLAB code for VO is provided.	http://sites.google.com/site/scarabotix

VO is applied in all kinds of mobile-robotics systems, such as space robots, ground robots, aerial robots, and underwater robots. But probably, the most popular application of VO has been on NASA Mars exploration rovers [25], [79]. NASA's VO has been used since January 2004 to track the motion of the two NASA rovers Spirit and

Opportunity as a supplement to dead reckoning. Their stereo VO system was implemented on a 20-MHz central processing unit and took up to three minutes for a two-view structure-from-motion step. VO was mainly used to approach targets efficiently as well as to maintain vehicle safety while driving near obstacles on slopes, achieving

difficult drive approaches, performing slip checks to ensure that the vehicle is still making progress.

VO is also applied onboard of unmanned aerial vehicles of all kinds of sizes, e.g., within the Autonomous Vehicle Aerial Tracking and Reconnaissance [80] and Swarm of Micro Flying Robots (SFLY) [81] projects. Within the SFLY project, VO was used to perform autonomous take-off, point-to-point navigation, and landing of small-scale quadrocopters.

Autonomous underwater vehicle is also a domain where VO plays a big role. Underwater vehicles cannot rely on GPS for position estimation; thus, onboard sensors need to be used. Cameras provide a cost-effective solution; in addition, the ocean floor often provides a texture-rich environment [82], which is ideal for computer vision methods. Applications range from coral-reef inspection (e.g., the Starbug system [82] to archaeological surveys [83].

VO also plays a big role in the automotive industry. Driver assistance systems (e.g., assisted braking) already rely on computer vision and digital cameras. VO for automotive market is in development, and its first demonstrations have been successfully shown, e.g., within the Daimler 6-D-Vision system [84] or as part of the VisLab autonomous vehicle [85]. Driving the development of this technology is the low cost of vision sensors as compared to Lidar sensors, which is an important factor for the automotive industry.

Available Code

Some algorithms that can be used to build a VO system are made publicly available by their authors. Table 2 points the reader to a selection of these resources.

Conclusions

Part II of the tutorial has summarized the remaining building blocks of the VO pipeline: specifically, how to detect and match salient and repeatable features across frames and robust estimation in the presence of outliers and bundle adjustment. In addition, error propagation, applications, and links to publicly available code are included. VO is a well understood and established part of robotics.

VO has reached a maturity that has allowed us to successfully use it for certain classes of applications: space, ground, aerial, and underwater. In the presence of loop closures, VO can be used as a building block for a complete SLAM algorithm to reduce motion drift. Challenges that still remain are to develop and demonstrate large-scale and long-term implementations, such as driving autonomous cars for hundreds of miles. Such systems have recently been demonstrated using Lidar and Radar sensors [86]. However, for VO to be used in such systems, technical issues regarding robustness and, especially, long-term stability have to be resolved. Eventually, VO has the potential to replace Lidar-based systems for egomotion estimation, which are currently leading the state of the art in accuracy, robustness, and reliability. VO offers a cheaper and mechanically easier-

to-manufacture solution for egomotion estimation, while, additionally, being fully passive. Furthermore, the ongoing miniaturization of digital cameras offers the possibility to develop smaller and smaller robotic systems capable of ego-motion estimation.

Acknowledgments

The authors thank Konstantinos Derpanis for his fruitful comments and suggestions.

References

- [1] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2004, pp. 652–659.
- [2] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford University, Stanford, CA, 1980.
- [3] L. Matthies and S. Shafer, "Error modeling in stereo navigation," *IEEE J. Robot. Automat.*, vol. 3, no. 3, pp. 239–248, 1987.
- [4] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo, "Rover self localization in planetary-like environments," in *Proc. Int. Symp. Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, 1999, pp. 433–440.
- [5] C. Olson, L. Matthies, M. Schoppers, and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, pp. 453–458.
- [6] M. Lhuillier, "Automatic structure and motion using a catadioptric camera," in *Proc. IEEE Workshop Omnidirectional Vision*, 2005, pp. 1–8.
- [7] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2006, pp. 363–370.
- [8] J. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2008, pp. 2531–2538.
- [9] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 4293–4299.
- [10] W. Forstner, "A feature based correspondence algorithm for image matching," *Int. Archiv. Photogram.*, vol. 26, no. 3, pp. 150–166, 1986.
- [11] C. Harris and J. Pike, "3d positional integration from image sequences," in *Proc. Alvey Vision Conf.*, 1987, pp. 233–236.
- [12] C. Tomasi and J. Shi, "Good features to track," in *Proc. CVPR*, 1994, pp. 593–600.
- [13] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conf. Computer Vision*, 2006, vol. 1, pp. 430–443.
- [14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 20, no. 2, pp. 91–110, 2003.
- [15] H. Bay, T.uytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. ECCV*, 2006, pp. 404–417.
- [16] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extrema for realtime feature detection and matching," in *Proc. European Conf. Computer Vision*, 2008, pp. 102–115.
- [17] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, MIT Press, 2011.
- [18] A. Schmidt, M. Kraft, and A. Kasinski, "An evaluation of image feature detectors and descriptors for robot navigation," in *Proc. Int. Conf. Computer Vision and Graphics*, 2010, pp. 251–259.

- [19] N. Govender, "Evaluation of feature detection algorithms for structure from motion," Council for Scientific and Industrial Research, Pretoria, Technical Report, 2009.
- [20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Englewood Cliffs, NJ, Prentice Hall, 2007.
- [21] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proc. European Conf. Computer Vision*, 1994, pp. 151–158.
- [22] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. European Conf. Computer Vision*, 2010, pp. 778–792.
- [23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf (pdf)," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Barcelona, Nov. 2011, pp. 2564–2571.
- [24] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proc. Int. Conf. Computer Vision*, 2011, pp. 2548–2555.
- [25] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers: Field reports," *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.
- [26] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. Int. Conf. Computer Vision*, 2003, pp. 1403–1410.
- [27] T. Lemaire and S. Lacroix, "Vision-based SLAM: Stereo and monocular approaches," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 343–364, 2006.
- [28] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. European Conf. Computer Vision*, 2008, pp. 802–815.
- [29] H. Strasdat, J. Montiel, and A. Davison, "Real time monocular SLAM: Why filter?" in *Proc. IEEE Int. Conf. Robotics and Automation*, 2010, pp. 2657–2664.
- [30] B. Horn and B. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, 1981.
- [31] A. Makadia, C. Geyer, and K. Daniilidis, "Correspondence-free structure from motion," *Int. J. Comput. Vis.*, vol. 75, no. 3, pp. 311–327, 2007.
- [32] P. Torr and D. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," *Int. J. Comput. Vis.*, vol. 24, no. 3, pp. 271–300, 1997.
- [33] K. Sim and R. Hartley, "Recovering camera motion using l_∞ minimization," *IEEE Conf. Computer Vision and Pattern Recognition*, 2006, pp. 1230–1237.
- [34] M. A. Fischler and R. C. Bolles, "RANSAC sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [35] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, MA, Cambridge Univ. Press, 2004.
- [36] J. Oliensis, "Exact two-image structure from motion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 12, pp. 1618–1633, 2002.
- [37] E. Kruppa, "Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung," in *Proc. Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. IIa*, 1913, vol. 122, pp. 1939–1948.
- [38] O. Faugeras and S. Maybank, "Motion from point matches: Multiplicity of solutions," *Int. J. Comput. Vis.*, vol. 4, no. 3, pp. 225–246, 1990.
- [39] J. Philip, "A non-iterative algorithm for determining all essential matrices corresponding to five point pairs," *Photogram. Rec.*, vol. 15, no. 88, pp. 589–599, 1996.
- [40] B. Triggs, "Routines for relative pose of two calibrated cameras from 5 points," INRIA Rhone-Alpes, Tech. Rep., 2000.
- [41] D. Nister, "An efficient solution to the five-point relative pose problem," *Proc. CVPR03*, 2003, pp. II: 195–202.
- [42] H. Stewenius, C. Engels, and D. Nister, "Recent developments on direct relative orientation," *ISPRS J. Photogram. Remote Sens.*, vol. 60, no. 4, pp. 284–294, 2006.
- [43] O. Pizarro, R. Eustice, and H. Singh, "Relative pose estimation for instrumented, calibrated imaging platforms," in *Proc. DICTA*, 2003, pp. 601–612.
- [44] R. Sturm, "Das problem der projektivitaet und seine anwendung auf die flaechen zweiten grades," *Math. Annal.*, vol. 1, no. 4, pp. 533–573, 1869.
- [45] C. Geyer and H. Stewenius, "A nine-point algorithm for estimating paracatadioptric fundamental matrices," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR'07)*, June 2007, pp. 1–8, 17–22.
- [46] P. Sturm and J. Barreto, "General imaging geometry for central catadioptric cameras," in *Proc. 10th European Conf. Computer Vision*, Marseille, France, 2008, pp. 609–622.
- [47] P. Sturm, S. Ramalingam, J. Tardif, S. Gasparini, and J. Barreto, "Camera models and fundamental concepts used in geometric computer vision," *Foundat. Trends Comput. Graph. Vis.*, vol. 6, no. 1–2, pp. 1–183, 2010.
- [48] J. Lim, N. Barnes, and H. Li, "Estimating relative camera motion from the antipodal-epipolar constraint," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, no. 10, pp. 1907–1914, 2010.
- [49] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, "A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles," in *Proc. European Conf. Computer Vision*, 2010, pp. 269–282.
- [50] O. Naroditsky, X. S. Zhou, J. Gallier, S. I. Roumeliotis, and K. Daniilidis, "Two efficient solutions for visual odometry using directional correspondence," *IEEE Trans. Pattern Anal. Machine Intell.*, no. 99, p. 1, 2011.
- [51] L. Kneip, M. Chli, and R. Siegwart, "Robust real-time visual odometry with a single camera and an imu," in *Proc. British Machine Vision Conf.*, 2011.
- [52] D. Ortin and J. M. M. Montiel, "Indoor robot motion based on monocular images," *Robotica*, vol. 19, no. 3, pp. 331–342, 2001.
- [53] D. Scaramuzza, "1-point-RANSAC structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *Int. J. Comput. Vis.*, vol. 95, no. 1, pp. 74–85, 2011.
- [54] D. Scaramuzza, "Performance evaluation of 1-point ransac visual odometry," *J. Field Robot.*, vol. 28, no. 5, pp. 792–811, 2011.
- [55] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 138–156, 2000.
- [56] O. Chum and J. Matas, "Matching with prosac—Progressive sample consensus," in *Proc. CVPR*, 2005, pp. 220–226.
- [57] D. Nister, "Preemptive ransac for live structure and motion estimation," *Machine Vis. Applicat.*, vol. 16, no. 5, pp. 321–329, 2005.
- [58] R. Raguram, J. Frahm, and M. Pollefeys, "Exploiting uncertainty in random sample consensus," in *Proc. ICCV*, 2009, pp. 2074–2081.
- [59] P. McIlroy, E. Rosten, S. Taylor, and T. Drummond, "Deterministic sample consensus with multiple match hypotheses," in *Proc. British Machine Vision Conf.*, 2010, pp. 1–11.

- [60] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for ekf filtering: Application to real-time structure from motion and visual odometry," *J. Field Robot.*, vol. 27, no. 5, pp. 609–631, 2010.
- [61] D. Scaramuzza, A. Censi, and K. Daniilidis, "Exploiting motion priors in visual odometry for vehicle-mounted cameras with non-holonomic constraints," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2011, pp. 4469–4476.
- [62] E. Rosten, G. Reitmayr, and T. Drummond, "Improved ransac performance using simple, iterative minimal-set solvers," University of Cambridge, Tech Rep., arXiv:1007.1432v1, 2010.
- [63] O. Chum, J. Matas, and J. Kittler, "Locally optimized ransac," in *Proc. DAGM-Symp.*, 2003, pp. 236–243.
- [64] R. C. Smith, P. Cheeseman. (1986). On the representation, and estimation of spatial uncertainty, *Int. J. Robot. Res.*, [Online], vol. 5, no. 4, pp. 56–68. Available: <http://ijr.sagepub.com/content/5/4/56.abstract>
- [65] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. ICRA*, 2006, pp. 2262–2269.
- [66] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II. State of the art," *IEEE Robot. Automat. Mag.*, vol. 13, no. 3, pp. 108–117, 2006.
- [67] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization," in *Proc. IEEE Int. Conf. Robotics and Automation*, Apr. 2000, pp. 1023–1029.
- [68] M. Jogan and A. Leonardis, "Robust localization using panoramic view-based recognition," in *Proc. ICPR*, 2000, vol. 4, pp. 136–139.
- [69] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1–2, pp. 43–72, 2005.
- [70] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006, pp. 1180–1187.
- [71] M. Cummins and P. Newman, (2008). "FAB-MAP: Probabilistic localization mapping in the space of appearance," *Int. J. Robot. Res.*, [Online], vol. 27, no. 6, pp. 647–665. Available: <http://ijr.sagepub.com/cgi/content/abstract/27/6/647>
- [72] F. Fraundorfer, C. Engels, and D. Nistér, "Topological mapping, localization and navigation using image collections," in *Proc. IEEE/RSJ Conf. Intelligent Robots and Systems*, 2007, pp. 3872–3877.
- [73] F. Fraundorfer, C. Wu, J.-M. Frahm, and M. Pollefeys, "Visual word based location recognition in 3d models using distance augmented weighting," in *Proc. 4th Int. Symp. 3D Data Processing, Visualization, and Transmission*, 2008, pp. 1–8.
- [74] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, New York, Wiley, 2001.
- [75] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, New York, 2006, pp. 2161–2168.
- [76] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment a modern synthesis," in *Proc. Int. Workshop Vision Algorithms: Theory and Practice (ICCV'99)*, 2000, pp. 298–372.
- [77] G. Sibley, L. Matthies, and G. Sukhatme. (2010). Sliding window filter with application to planetary landing. *J. Field Robot.*, [Online], vol. 27, no. 5, pp. 587–608. Available: <http://dx.doi.org/10.1002/rob.20360>
- [78] Dacuda AG. (2011). Dacuda scanner mouse. [Online]. Available: <http://www.dacuda.com/>
- [79] Y. Cheng, M. W. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers," *IEEE Robot. Automat. Mag.*, vol. 13, no. 2, pp. 54–62, 2006.
- [80] J. Kelly and G. S. Sukhatme. (2007. Sept.). An experimental study of aerial stereo visual odometry. In *Proc. IFAC –Int. Federation of Automatic Control Symp. Intelligent Autonomous Vehicles*, Toulouse, France. [Online]. Available: <http://cres.usc.edu/cgi-bin/print.pub.details.pl?pubid=543>
- [81] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments," *J. Field Robot.*, vol. 28, no. 6, pp. 854–874, 2011.
- [82] M. Dunbabin, J. Roberts, K. Usher, G. Winstanley, and P. Corke, "A hybrid AUV design for shallow water reef navigation," in *Proc. IEEE Int. Conf. Robotics and Automation, ICRA*, Apr. 2005, pp. 2105–2110.
- [83] B. P. Foley, K. DellaPorta, D. Sakellarou, B. S. Bingham, R. Camilli, R. M. Eustice, D. Evangelista, V. L. Ferrini, K. Katsaros, D. Kourkoumelis, A. Mallios, P. Michal, D. A. Mindell, C. Roman, H. Singh, D. S. Switzer, and T. Theodoulou, "The 2005 chios ancient shipwreck survey: New methods for underwater archaeology," *Hesperia*, vol. 78, no. 2, pp. 269–305, 2009.
- [84] A. G. Daimler. (2011). 6d vision. [Online]. Available: <http://www.6d-vision.com/>
- [85] M. Bertozzi, A. Broggi, E. Cardarelli, R. Fedriga, L. Mazzei, and P. Porta, "Viac expedition toward autonomous mobility [from the field]," *IEEE Robot. Automat. Mag.*, vol. 18, no. 3, pp. 120–124, Sept. 2011.
- [86] E. Guizzo. (2011). How Google's self-driving car works. [Online]. Available: <http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works>
- [87] D. Scaramuzza and F. Fraundorfer, "Visual odometry," *IEEE Robotics Automat. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.
- [88] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Stereo ego-motion improvements for robust rover navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA 2001)*, 2001, vol. 2, pp. 1099–1104.
- [89] M. V. Srinivasan, S. W. Zhang, M. Lehrer, and T. S. Collett, "Honeybee navigation en route to the goal: Visual flight control and odometry," *J. Exp. Biol.*, vol. 199, 1996, pp. 237–244.
- [90] P. Viola and M. Jones, "Robust real time object detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2001.
- [91] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, no. 1–3, pp. 7–42, Apr.–June 2002.

Friedrich Fraundorfer, Institute of Visual Computing, Department of Computer Science, ETH Zurich, Switzerland. E-mail: fraundorfer@inf.ethz.ch.

Davide Scaramuzza, Robotics and Perception Lab, Department of Informatics, University of Zurich, Switzerland. E-mail: davide.scaramuzza@ieee.org.



13.3 *SVO: Fast Semi-Direct Monocular Visual Odometry, ICRA,
2014*

2014 IEEE International Conference on Robotics & Automation (ICRA)
 Hong Kong Convention and Exhibition Center
 May 31 - June 7, 2014. Hong Kong, China

SVO: Fast Semi-Direct Monocular Visual Odometry

Christian Forster, Matia Pizzoli, Davide Scaramuzza*

Abstract—We propose a semi-direct monocular visual odometry algorithm that is precise, robust, and faster than current state-of-the-art methods. The semi-direct approach eliminates the need of costly feature extraction and robust matching techniques for motion estimation. Our algorithm operates directly on pixel intensities, which results in subpixel precision at high frame-rates. A probabilistic mapping method that explicitly models outlier measurements is used to estimate 3D points, which results in fewer outliers and more reliable points. Precise and high frame-rate motion estimation brings increased robustness in scenes of little, repetitive, and high-frequency texture. The algorithm is applied to micro-aerial-vehicle state-estimation in GPS-denied environments and runs at 55 frames per second on the onboard embedded computer and at more than 300 frames per second on a consumer laptop. We call our approach SVO (Semi-direct Visual Odometry) and release our implementation as open-source software.

I. INTRODUCTION

Micro Aerial Vehicles (MAVs) will soon play a major role in disaster management, industrial inspection and environment conservation. For such operations, navigating based on GPS information only is not sufficient. Precise fully autonomous operation requires MAVs to rely on alternative localization systems. For minimal weight and power-consumption it was therefore proposed [1]–[5] to use only a single downward-looking camera in combination with an Inertial Measurement Unit. This setup allowed fully autonomous way-point following in outdoor areas [1]–[3] and collaboration between MAVs and ground robots [4], [5].

To our knowledge, all monocular Visual Odometry (VO) systems for MAVs [1], [2], [6], [7] are feature-based. In RGB-D and stereo-based SLAM systems however, direct methods [8]–[11]—based on photometric error minimization—are becoming increasingly popular.

In this work, we propose a semi-direct VO that combines the success-factors of feature-based methods (tracking many features, parallel tracking and mapping, keyframe selection) with the accuracy and speed of direct methods. High frame-rate VO for MAVs promises increased robustness and faster flight maneuvers.

An open-source implementation and videos of this work are available at: <http://rpg.ifi.uzh.ch/software>

A. Taxonomy of Visual Motion Estimation Methods

Methods that simultaneously recover camera pose and scene structure from video can be divided into two classes:

*The authors are with the Robotics and Perception Group, University of Zurich, Switzerland—<http://rpg.ifi.uzh.ch>. This research was supported by the Swiss National Science Foundation through project number 200021-143607 (“Swarm of Flying Cameras”), the National Centre of Competence in Research Robotics, and the CTI project number 14652.1.

a) Feature-Based Methods: The standard approach is to extract a sparse set of salient image features (e.g. points, lines) in each image; match them in successive frames using invariant feature descriptors; robustly recover both camera motion and structure using epipolar geometry; finally, refine the pose and structure through reprojection error minimization. The majority of VO algorithms [12] follows this procedure, independent of the applied optimization framework. A reason for the success of these methods is the availability of robust feature detectors and descriptors that allow matching between images even at large inter-frame movement. The disadvantage of feature-based approaches is the reliance on detection and matching thresholds, the necessity for robust estimation techniques to deal with wrong correspondences, and the fact that most feature detectors are optimized for speed rather than precision, such that drift in the motion estimate must be compensated by averaging over many feature-measurements.

b) Direct Methods: Direct methods [13] estimate structure and motion directly from intensity values in the image. The local intensity gradient magnitude and direction is used in the optimisation compared to feature-based methods that consider only the distance to some feature-location. Direct methods that exploit all the information in the image, even from areas where gradients are small, have been shown to outperform feature-based methods in terms of robustness in scenes with little texture [14] or in the case of camera-defocus and motion blur [15]. The computation of the photometric error is more intensive than the reprojection error, as it involves warping and integrating large image regions. However, since direct methods operate directly on the intensity values of the image, the time for feature detection and invariant descriptor computation can be saved.

B. Related Work

Most monocular VO algorithms for MAVs [1], [2], [7] rely on PTAM [16]. PTAM is a feature-based SLAM algorithm that achieves robustness through tracking and mapping many (hundreds) of features. Simultaneously, it runs in real-time by parallelizing the motion estimation and mapping tasks and by relying on efficient keyframe-based Bundle Adjustment (BA) [17]. However, PTAM was designed for augmented reality applications in small desktop scenes and multiple modifications (e.g., limiting the number of keyframes) were necessary to allow operation in large-scale outdoor environments [2].

Early direct monocular SLAM methods tracked and mapped few—sometimes manually selected—planar patches [18]–[21]. While the first approaches [18], [19] used filtering algorithms to estimate structure and motion, later methods

[20]–[22] used nonlinear least squares optimization. All these methods estimate the surface normals of the patches, which allows tracking a patch over a wide range of viewpoints, thus, greatly reducing drift in the estimation. The authors of [19]–[21] reported real-time performance, however, only with few selected planar regions and on small datasets. A VO algorithm for omnidirectional cameras on cars was proposed in [22]. In [8], the local planarity assumption was relaxed and direct tracking with respect to arbitrary 3D structures computed from stereo cameras was proposed. In [9]–[11], the same approach was also applied to RGB-D sensors. With DTAM [15], a novel direct method was introduced that computes a dense depthmap for each keyframe through minimisation of a global, spatially-regularised energy functional. The camera pose is found through direct whole image alignment using the depth-map. This approach is computationally very intensive and only possible through heavy GPU parallelization. To reduce the computational demand, the method described in [23], which was published during the review process of this work, uses only pixels characterized by strong gradient.

C. Contributions and Outline

The proposed *Semi-Direct Visual Odometry* (SVO) algorithm uses feature-correspondence; however, feature-correspondence is an implicit result of direct motion estimation rather than of explicit feature extraction and matching. Thus, feature extraction is only required when a keyframe is selected to initialize new 3D points (see Figure 1). The advantage is increased speed due to the lack of feature-extraction at every frame and increased accuracy through subpixel feature correspondence. In contrast to previous direct methods, we use many (hundreds) of small patches rather than few (tens) large planar patches [18]–[21]. Using many small patches increases robustness and allows neglecting the patch normals. The proposed sparse model-based image alignment algorithm for motion estimation is related to model-based dense image alignment [8]–[10], [24]. However, we demonstrate that sparse information of depth is sufficient to get a rough estimate of the motion and to find feature-correspondences. As soon as feature correspondences and an initial estimate of the camera pose are established, the algorithm continues using only point-features; hence, the name “*semi-direct*”. This switch allows us to rely on fast and established frameworks for bundle adjustment (e.g., [25]).

A Bayesian filter that explicitly models outlier measurements is used to estimate the depth at feature locations. A 3D point is only inserted in the map when the corresponding depth-filter has converged, which requires multiple measurements. The result is a map with few outliers and points that can be tracked reliably.

The contributions of this paper are: (1) a novel semi-direct VO pipeline that is faster and more accurate than the current state-of-the-art for MAVs, (2) the integration of a probabilistic mapping method that is robust to outlier measurements.

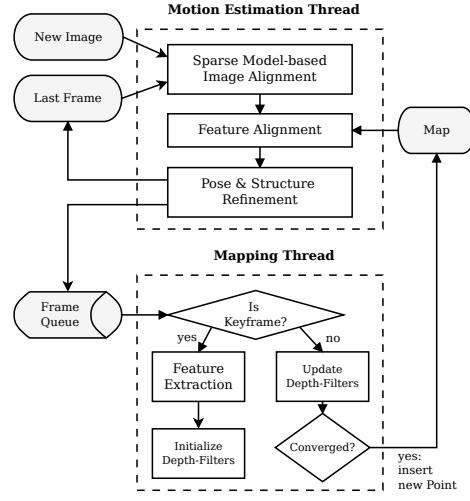


Fig. 1: Tracking and mapping pipeline

Section II provides an overview of the pipeline and Section III, thereafter, introduces some required notation. Section IV and V explain the proposed motion-estimation and mapping algorithms. Section VII provides experimental results and comparisons.

II. SYSTEM OVERVIEW

Figure 1 provides an overview of SVO. The algorithm uses two parallel threads (as in [16]), one for estimating the camera motion, and a second one for mapping as the environment is being explored. This separation allows fast and constant-time tracking in one thread, while the second thread extends the map, decoupled from hard real-time constraints.

The motion estimation thread implements the proposed semi-direct approach to relative-pose estimation. The first step is pose initialisation through *sparse model-based image alignment*: the camera pose relative to the previous frame is found through minimizing the *photometric error* between pixels corresponding to the projected location of the same 3D points (see Figure 2). The 2D coordinates corresponding to the reprojected points are refined in the next step through alignment of the corresponding feature-patches (see Figure 3). Motion estimation concludes by refining the pose and the structure through minimizing the reprojection error introduced in the previous feature-alignment step.

In the mapping thread, a probabilistic depth-filter is initialised for each 2D feature for which the corresponding 3D point is to be estimated. New depth-filters are initialised whenever a new keyframe is selected in regions of the image where few 3D-to-2D correspondences are found. The filters are initialised with a large uncertainty in depth. At every subsequent frame the depth estimate is updated in a Bayesian fashion (see Figure 5). When a depth filter’s uncertainty becomes small enough, a new 3D point is inserted in the map and is immediately used for motion estimation.

III. NOTATION

Before the algorithm is detailed, we briefly define the notation that is used throughout the paper.

The intensity image collected at timestep k is denoted with $I_k : \Omega \subset \mathbb{R}^2 \mapsto \mathbb{R}$, where Ω is the image domain. Any 3D point $\mathbf{p} = (x, y, z)^\top \in \mathcal{S}$ on the visible scene surface $\mathcal{S} \subset \mathbb{R}^3$ maps to the image coordinates $\mathbf{u} = (u, v)^\top \in \Omega$ through the camera projection model $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$:

$$\mathbf{u} = \pi(k\mathbf{p}), \quad (1)$$

where the subscript k denotes that the point coordinates are expressed in the camera frame of reference k . The projection π is determined by the intrinsic camera parameters which are known from calibration. The 3D point corresponding to an image coordinate \mathbf{u} can be recovered, given the inverse projection function π^{-1} and the depth $d_{\mathbf{u}} \in \mathcal{R}$:

$$k\mathbf{p} = \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}), \quad (2)$$

where $\mathcal{R} \subseteq \Omega$ is the domain for which the depth is known.

The camera position and orientation at timestep k is expressed with the *rigid-body transformation* $\mathbf{T}_{k,w} \in SE(3)$. It allows us to map a 3D point from the world coordinate frame to the camera frame of reference: $k\mathbf{p} = \mathbf{T}_{k,w} \cdot {}_w\mathbf{p}$. The relative transformation between two consecutive frames can be computed with $\mathbf{T}_{k,k-1} = \mathbf{T}_{k,w} \cdot \mathbf{T}_{k-1,w}^{-1}$. During the optimization, we need a minimal representation of the transformation and, therefore, use the Lie algebra $\mathfrak{se}(3)$ corresponding to the tangent space of $SE(3)$ at the identity. We denote the algebra elements—also named *twist coordinates*—with $\xi = (\omega, v)^T \in \mathbb{R}^6$, where ω is called the angular velocity and v the linear velocity. The twist coordinates ξ are mapped to $SE(3)$ by the exponential map [26]:

$$\mathbf{T}(\xi) = \exp(\hat{\xi}). \quad (3)$$

IV. MOTION ESTIMATION

SVO computes an initial guess of the relative camera motion and the feature correspondences using direct methods and concludes with a feature-based nonlinear reprojection-error refinement. Each step is detailed in the following sections and illustrated in Figures 2 to 4.

A. Sparse Model-based Image Alignment

The maximum likelihood estimate of the rigid body transformation $\mathbf{T}_{k,k-1}$ between two consecutive camera poses minimizes the negative log-likelihood of the intensity residuals:

$$\mathbf{T}_{k,k-1} = \arg \min_{\mathbf{T}} \iint_{\bar{\mathcal{R}}} \rho[\delta I(\mathbf{T}, \mathbf{u})] d\mathbf{u}. \quad (4)$$

The intensity residual δI is defined by the photometric difference between pixels observing the same 3D point. It can be computed by back-projecting a 2D point \mathbf{u} from the previous image I_{k-1} and subsequently projecting it into the current camera view:

$$\delta I(\mathbf{T}, \mathbf{u}) = I_k(\pi(\mathbf{T} \cdot \pi^{-1}(\mathbf{u}, d_{\mathbf{u}}))) - I_{k-1}(\mathbf{u}) \quad \forall \mathbf{u} \in \bar{\mathcal{R}}, \quad (5)$$

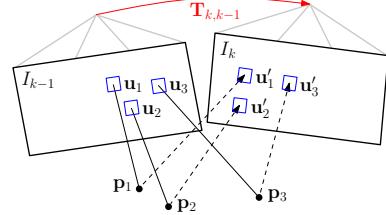


Fig. 2: Changing the relative pose $\mathbf{T}_{k,k-1}$ between the current and the previous frame implicitly moves the position of the reprojected points in the new image \mathbf{u}'_i . Sparse image alignment seeks to find $\mathbf{T}_{k,k-1}$ that minimizes the photometric difference between image patches corresponding to the same 3D point (blue squares). Note, in all figures, the parameters to optimize are drawn in red and the optimization cost is highlighted in blue.

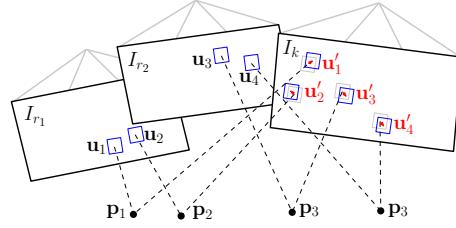


Fig. 3: Due to inaccuracies in the 3D point and camera pose estimation, the photometric error between corresponding patches (blue squares) in the current frame and previous keyframes r_i can further be minimized by optimising the 2D position of each patch individually.

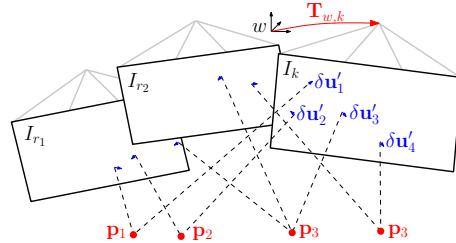


Fig. 4: In the last motion estimation step, the camera pose and the structure (3D points) are optimized to minimize the reprojection error that has been established during the previous feature-alignment step.

where $\bar{\mathcal{R}}$ is the image region for which the depth $d_{\mathbf{u}}$ is known at time $k-1$ and for which the back-projected points are visible in the current image domain:

$$\bar{\mathcal{R}} = \{\mathbf{u} \mid \mathbf{u} \in \mathcal{R}_{k-1} \wedge \pi(\mathbf{T} \cdot \pi^{-1}(\mathbf{u}, d_{\mathbf{u}})) \in \Omega_k\}. \quad (6)$$

For the sake of simplicity, we assume in the following that the intensity residuals are normally distributed with unit variance. The negative log likelihood minimizer then corresponds to the least squares problem: $\rho[\cdot] \triangleq \frac{1}{2} \|\cdot\|^2$. In practice, the distribution has heavier tails due to occlusions and thus, a robust cost function must be applied [10].

In contrast to previous works, where the depth is known for large regions in the image [8]–[10], [24], we only know the depth $d_{\mathbf{u}_i}$ at sparse feature locations \mathbf{u}_i . We denote small patches of 4×4 pixels around the feature point with the vector $\mathbf{I}(\mathbf{u}_i)$. We seek to find the camera pose that minimizes

the photometric error of all patches (see Figure 2):

$$\mathbf{T}_{k,k-1} = \arg \min_{\mathbf{T}_{k,k-1}} \frac{1}{2} \sum_{i \in \mathcal{R}} \| \delta \mathbf{I}(\mathbf{T}_{k,k-1}, \mathbf{u}_i) \|^2. \quad (7)$$

Since Equation (7) is nonlinear in $\mathbf{T}_{k,k-1}$, we solve it in an iterative Gauss-Newton procedure. Given an estimate of the relative transformation $\hat{\mathbf{T}}_{k,k-1}$, an incremental update $\mathbf{T}(\xi)$ to the estimate can be parametrised with a twist $\xi \in \mathfrak{se}(3)$. We use the *inverse compositional* formulation [27] of the intensity residual, which computes the update step $\mathbf{T}(\xi)$ for the reference image at time $k-1$:

$$\delta \mathbf{I}(\xi, \mathbf{u}_i) = \mathbf{I}_k \left(\pi(\hat{\mathbf{T}}_{k,k-1} \cdot \mathbf{p}_i) \right) - \mathbf{I}_{k-1} \left(\pi(\mathbf{T}(\xi) \cdot \mathbf{p}_i) \right), \quad (8)$$

with $\mathbf{p}_i = \pi^{-1}(\mathbf{u}_i, d_{\mathbf{u}_i})$. The inverse of the update step is then applied to the current estimate using Equation (3):

$$\hat{\mathbf{T}}_{k,k-1} \leftarrow \hat{\mathbf{T}}_{k,k-1} \cdot \mathbf{T}(\xi)^{-1}. \quad (9)$$

Note that we do not warp the patches for computing speed-reasons. This assumption is valid in case of small frame-to-frame motions and for small patch-sizes.

To find the optimal update step $\mathbf{T}(\xi)$, we compute the derivative of (7) and set it to zero:

$$\sum_{i \in \mathcal{R}} \nabla \delta \mathbf{I}(\xi, \mathbf{u}_i)^\top \delta \mathbf{I}(\xi, \mathbf{u}_i) = 0. \quad (10)$$

To solve this system, we linearize around the current state:

$$\delta \mathbf{I}(\xi, \mathbf{u}_i) \approx \delta \mathbf{I}(0, \mathbf{u}_i) + \nabla \delta \mathbf{I}(0, \mathbf{u}_i) \cdot \xi \quad (11)$$

The Jacobian $\mathbf{J}_i := \nabla \delta \mathbf{I}(0, \mathbf{u}_i)$ has the dimension 16×6 because of the 4×4 patch-size and is computed with the chain-rule:

$$\frac{\partial \delta \mathbf{I}(\xi, \mathbf{u}_i)}{\partial \xi} = \frac{\partial \mathbf{I}_{k-1}(\mathbf{a})}{\partial \mathbf{a}} \Big|_{\mathbf{a}=\mathbf{u}_i} \cdot \frac{\partial \pi(\mathbf{b})}{\partial \mathbf{b}} \Big|_{\mathbf{b}=\mathbf{p}_i} \cdot \frac{\partial \mathbf{T}(\xi)}{\partial \xi} \Big|_{\xi=0} \cdot \mathbf{p}_i$$

By inserting (11) into (10) and by stacking the Jacobians in a matrix \mathbf{J} , we obtain the normal equations:

$$\mathbf{J}^T \mathbf{J} \xi = -\mathbf{J}^T \delta \mathbf{I}(0), \quad (12)$$

which can be solved for the update twist ξ . Note that by using the inverse compositional approach, the Jacobian can be precomputed as it remains constant over all iterations (the reference patch $\mathbf{I}_{k-1}(\mathbf{u}_i)$ and the point \mathbf{p}_i do not change), which results in a significant speedup [27].

B. Relaxation Through Feature Alignment

The last step aligned the camera with respect to the previous frame. Through back-projection, the found relative pose $\mathbf{T}_{k,k-1}$ implicitly defines an initial guess for the feature positions of all visible 3D points in the new image. Due to inaccuracies in the 3D points' positions and, thus, the camera pose, this initial guess can be improved. To reduce the drift, the camera pose should be aligned with respect to the map, rather than to the previous frame.

All 3D points of the map that are visible from the estimated camera pose are projected into the image, resulting in an estimate of the corresponding 2D feature positions \mathbf{u}'_i (see Figure 3). For each reprojected point, the keyframe r

that observes the point with the closest observation angle is identified. The feature alignment step then optimizes all 2D feature-positions \mathbf{u}_i in the new image individually by minimizing the photometric error of the patch in the current image with respect to the reference patch in the keyframe r :

$$\mathbf{u}'_i = \arg \min_{\mathbf{u}'_i} \frac{1}{2} \| \mathbf{I}_k(\mathbf{u}'_i) - \mathbf{A}_i \cdot \mathbf{I}_r(\mathbf{u}_i) \|^2, \quad \forall i. \quad (13)$$

This alignment is solved using the inverse compositional Lucas-Kanade algorithm [27]. Contrary to the previous step, we apply an affine warping \mathbf{A}_i to the reference patch, since a larger patch size is used (8×8 pixels) and the closest keyframe is typically farther away than the previous image.

This step can be understood as a relaxation step that violates the epipolar constraints to achieve a higher correlation between the feature-patches.

C. Pose and Structure Refinement

In the previous step, we have established feature correspondence with subpixel accuracy at the cost of violating the epipolar constraints. In particular, we have generated a reprojection residual $\|\delta \mathbf{u}_i\| = \|\mathbf{u}_i - \pi(\mathbf{T}_{k,w} \cdot \mathbf{p}_i)\| \neq 0$, which on average is around 0.3 pixels (see Figure 11). In this final step, we again optimize the camera pose $\mathbf{T}_{k,w}$ to minimize the reprojection residuals (see Figure 4):

$$\mathbf{T}_{k,w} = \arg \min_{\mathbf{T}_{k,w}} \frac{1}{2} \sum_i \| \mathbf{u}_i - \pi(\mathbf{T}_{k,w} \cdot \mathbf{p}_i) \|^2. \quad (14)$$

This is the well known problem of *motion-only* BA [17] and can efficiently be solved using an iterative non-linear least squares minimization algorithm such as Gauss Newton.

Subsequently, we optimize the position of the observed 3D points through reprojection error minimization (*structure-only* BA). Finally, it is possible to apply local BA, in which both the pose of all close keyframes as well as the observed 3D points are jointly optimized. The BA step is omitted in the fast parameter settings of the algorithm (Section VII).

D. Discussion

The first (Section IV-A) and the last (Section IV-C) optimization of the algorithm seem to be redundant as both optimize the 6 DoF pose of the camera. Indeed, one could directly start with the second step and establish feature-correspondence through Lucas-Kanade tracking [27] of all feature-patches, followed by nonlinear pose refinement (Section IV-C). While this would work, the processing time would be higher. Tracking all features over large distances (e.g., 30 pixels) requires a larger patch and a pyramidal implementation. Furthermore, some features might be tracked inaccurately, which would require outlier detection. In SVO however, feature alignment is efficiently initialized by only optimizing six parameters—the camera pose—in the sparse image alignment step. The sparse image alignment step satisfies implicitly the epipolar constraint and ensures that there are no outliers.

One may also argue that the first step (sparse image alignment) would be sufficient to estimate the camera motion. In

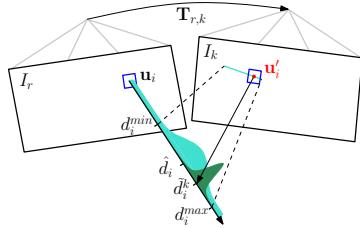


Fig. 5: Probabilistic depth estimate \hat{d}_i for feature i in the reference frame r . The point at the true depth projects to similar image regions in both images (blue squares). Thus, the depth estimate is updated with the triangulated depth \tilde{d}_i^k computed from the point \mathbf{u}'_i of highest correlation with the reference patch. The point of highest correlation lies always on the epipolar line in the new image.

fact, this is what recent algorithms developed for RGB-D cameras do [10], however, by aligning the full depth-map rather than sparse patches. We found empirically that using the first step only results in significantly more drift compared to using all three steps together. The improved accuracy is due to the alignment of the new image with respect to the keyframes and the map, whereas sparse image alignment aligns the new frame only with respect to the previous frame.

V. MAPPING

Given an image and its pose $\{I_k, \mathbf{T}_{k,w}\}$, the mapping thread estimates the depth of 2D features for which the corresponding 3D point is not yet known. The depth estimate of a feature is modeled with a probability distribution. Every subsequent observation $\{I_k, \mathbf{T}_{k,w}\}$ is used to update the distribution in a Bayesian framework (see Figure 5) as in [28]. When the variance of the distribution becomes small enough, the depth-estimate is converted to a 3D point using (2), the point is inserted in the map and immediately used for motion estimation (see Figure 1). In the following we report the basic results and our modifications to the original implementation in [28].

Every *depth-filter* is associated to a reference keyframe r . The filter is initialized with a high uncertainty in depth and the mean is set to the average scene depth in the reference frame. For every subsequent observation $\{I_k, \mathbf{T}_{k,w}\}$, we search for a patch on the epipolar line in the new image I_k that has the highest correlation with the reference patch. The epipolar line can be computed from the relative pose between the frames $\mathbf{T}_{r,k}$ and the optical ray that passes through \mathbf{u}_i . The point of highest correlation \mathbf{u}'_i corresponds to the depth \tilde{d}_i^k that can be found by triangulation (see Figure 5).

The measurement \tilde{d}_i^k is modeled with a *Gaussian + Uniform* mixture model distribution [28]: a good measurement is normally distributed around the true depth d_i while an outlier measurement arises from a uniform distribution in the interval $[d_i^{\min}, d_i^{\max}]$:

$$p(\tilde{d}_i^k | d_i, \rho_i) = \rho_i \mathcal{N}(\tilde{d}_i^k | d_i, \tau_i^2) + (1 - \rho_i) \mathcal{U}(\tilde{d}_i^k | d_i^{\min}, d_i^{\max}),$$

where ρ_i is the inlier probability and τ_i^2 the variance of a good measurement that can be computed geometrically by assuming a photometric disparity variance of one pixel in the image plane [29].

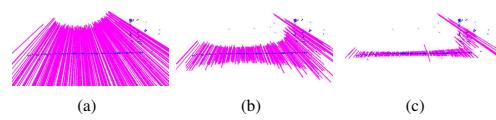


Fig. 6: Very little motion is required by the MAV (seen from the side at the top) for the uncertainty of the depth-filters (shown as magenta lines) to converge.

The recursive Bayesian update step for this model is described in detail in [28]. In contrast to [28], we use inverse depth coordinates to deal with large scene depths.

The proposed depth estimation is very efficient when only a small range around the current depth estimate on the epipolar line is searched; in our case the range corresponds to twice the standard deviation of the current depth estimate. Figure 6 demonstrates how little motion is required to significantly reduce the uncertainty in depth. The main advantage of the proposed methods over the standard approach of triangulating points from two views is that we observe far fewer outliers as every filter undergoes many measurements until convergence. Furthermore, erroneous measurements are explicitly modeled, which allows the depth to converge even in highly-similar environments. In [29] we demonstrate how the same approach can be used for *dense* mapping.

VI. IMPLEMENTATION DETAILS

The algorithm is bootstrapped to obtain the pose of the first two keyframes and the initial map. Like in [16], we assume a locally planar scene and estimate a homography. The initial map is triangulated from the first two views.

In order to cope with large motions, we apply the sparse image alignment algorithm in a coarse-to-fine scheme. The image is halfsampled to create an image pyramid of five levels. The intensity residual is then optimized at the coarsest level until convergence. Subsequently, the optimization is initialized at the next finer level. To save processing time, we stop after convergence on the third level, at which stage the estimate is accurate enough to initialize feature alignment.

The algorithm keeps for efficiency reasons a fixed number of *keyframes* in the map, which are used as reference for feature-alignment and for structure refinement. A keyframe is selected if the Euclidean distance of the new frame relative to all keyframes exceeds 12% of the average scene depth. When a new keyframe is inserted in the map, the keyframe farthest apart from the current position of the camera is removed.

In the mapping thread, we divide the image in cells of fixed size (e.g., 30×30 pixels). A new depth-filter is initialized at the FAST corner [30] with highest Shi-Tomasi score in the cell unless there is already a 2D-to-3D correspondence present. This results in evenly distributed features in the image. The same grid is also used for reprojecting the map before feature alignment. Note that we extract FAST corners at every level of the image pyramid to find the best corners independent of the scale.

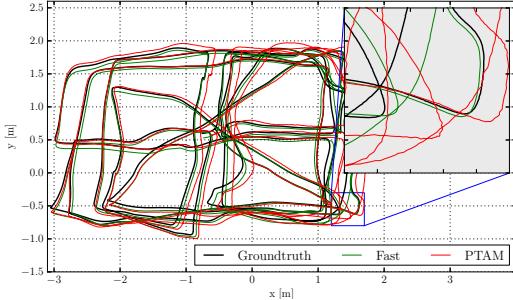


Fig. 7: Comparison against the ground-truth of SVO with the *fast* parameter setting (see Table I) and of PTAM. Zooming-in reveals that the proposed algorithm generates a smoother trajectory than PTAM.

VII. EXPERIMENTAL RESULTS

Experiments were performed on datasets recorded from a downward-looking camera¹ attached to a MAV and sequences from a handheld camera. The video was processed on both a laptop² and on an embedded platform³ that is mounted on the MAV (see Figure 17). Note that at maximum 2 CPU cores are used for the algorithm. The experiments on the consumer laptop were run with two different parameters' settings, one optimised for speed and one for accuracy (Table I). On the embedded platform only the *fast* parameters' setting is used.

	Fast	Accurate
Max number of features per image	120	200
Max number of keyframes	10	50
Local Bundle Adjustment	no	yes

TABLE I: Two different parameter settings of SVO.

We compare the performance of SVO with the modified PTAM algorithm of [2]. The reason we do not compare with the original version of PTAM [16] is because it does not handle large environments and is not robust enough in scenes of high-frequency texture [2]. The version of [2] solves these problems and constitutes to our knowledge the best performing monocular SLAM algorithm for MAVs.

A. Accuracy

We evaluate the accuracy on a dataset that has also been used in [2] and is illustrated in Figure 7. The ground-truth for the trajectory originates from a motion capture system. The trajectory is 84 meters long and the MAV flew on average 1.2 meters above the flat ground.

Figures 8 and 9 illustrate the position and attitude error over time. In order to generate the plots, we aligned the first 10 frames with the ground-truth using [31]. The results of PTAM are in a similar range as reported in [2]. Since the plots are highly dependent on the accuracy of alignment of the first 10 frames, we also report the drift in meters

¹Matrix Vision BlueFox, global shutter, 752×480 pixel resolution.

²Intel i7, 8 cores, 2.8 GHz

³Odroid-U2, ARM Cortex A-9, 4 cores, 1.6 GHz

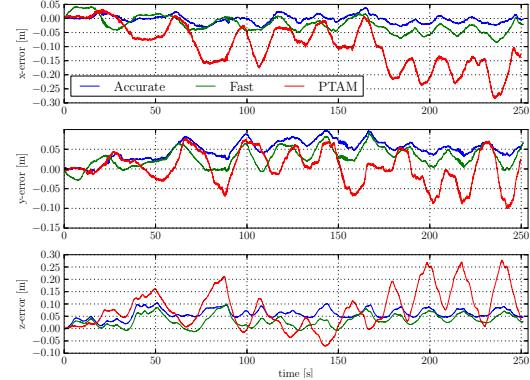


Fig. 8: Position drift of SVO with *fast* and *accurate* parameter setting and comparison against PTAM.

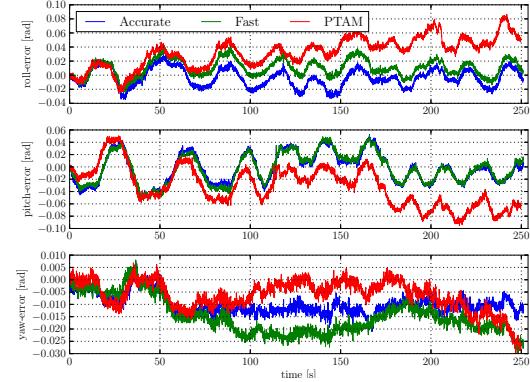


Fig. 9: Attitude drifts of SVO with *fast* and *accurate* parameter setting and comparison against PTAM.

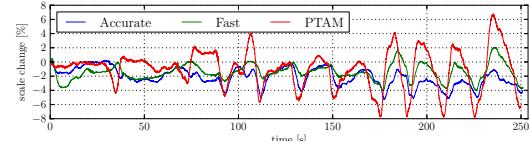


Fig. 10: Scale-drift over time of the trajectory shown in Figure 7

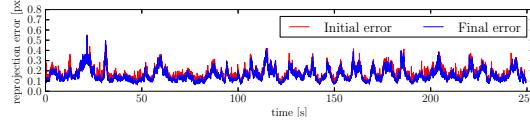


Fig. 11: Average reprojection error over time of the trajectory shown in Figure 7. The initial error is after sparse image alignment (Section IV-A) and the final error after pose refinement (Section IV-C).

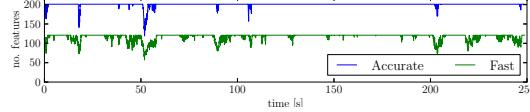


Fig. 12: Number of tracked features over time for two different parameter settings. For the *accurate* parameter setting, the number of features is limited to 200 and for the *fast* setting to 120.

	Pos-RMSE [m/s]	Pos-Median [m/s]	Rot-RMSE [deg/s]	Rot-Median [deg/s]
<i>fast</i>	0.0059	0.0047	0.4295	0.3686
<i>accurate</i>	0.0051	0.0038	0.4519	0.3858
PTAM	0.0164	0.0142	0.4585	0.3808

TABLE II: Relative pose and rotation error of the trajectory in Figure 7

per second in Table II as proposed and motivated in [32]. Overall, both versions of SVO are more accurate than PTAM. We suspect the main reason for this result to originate from the fact that the PTAM version of [2] does not extract features on the pyramid level of highest resolution and subpixel refinement is not performed for all features in PTAM. Neglecting the highest resolution image inevitably results in less accuracy which is clearly visible in the close-up of Figure 7. In [2], the use of lower resolution images is motivated by the fact that high-frequency self-similar texture in the image results in too many outlier 3D points. SVO efficiently copes with this problem by using the depth-filters which results in very few outliers.

Since a camera is only an angle-sensor, it is impossible to obtain the scale of the map through a Structure from Motion pipeline. Hence, in the above evaluation we also align the scale of the first 10 measurements with the ground-truth. The proposed pipeline propagates the scale, however with some drift that is shown in Figure 10. The scale drift is computed by comparing the euclidean norm of the relative translation against the ground-truth. The unknown scale and the scale drift motivate the need for a camera-IMU state estimation system for MAV control, as described in [33].

Figure 11 illustrates the average reprojection error. The sparse image alignment step brings the frame very close to the final pose, as the refinement step reduces the error only marginally. The reprojection error is “generated” in the feature-alignment step; hence, this plot also shows that patches move only a fraction of a pixel during this step.

The difference in accuracy between the *fast* and *accurate* parameter setting is not significant. Optimizing the pose and the observed 3D points separately at every iteration (*fast* parameter setting) is accurate enough for MAV motion estimation.

B. Runtime Evaluation

Figures 13 and 14 show a break-up of the time required to compute the camera motion on the specified laptop and embedded platform respectively with the *fast*-parameter setting. The laptop is capable to process the frames faster than 300 frames per second (fps) while the embedded platform runs at 55 fps. The corresponding time for PTAM is 91 fps and 27 fps respectively. The main difference is that SVO does not require feature extraction during motion estimation which constitutes the bulk of time in PTAM (7 ms on the laptop, 16 ms on the embedded computer). Additionally, PTAM tracks between 160 and 220 features while in the *fast* parameter setting, this value is limited to 120. The reason why we can reliably track the camera with less features is the use of depth-filters, which assures that the features being tracked

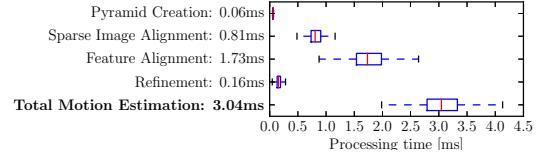


Fig. 13: Timing results on a laptop computer.

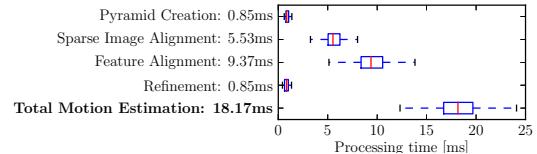


Fig. 14: Timing results on the embedded platform.

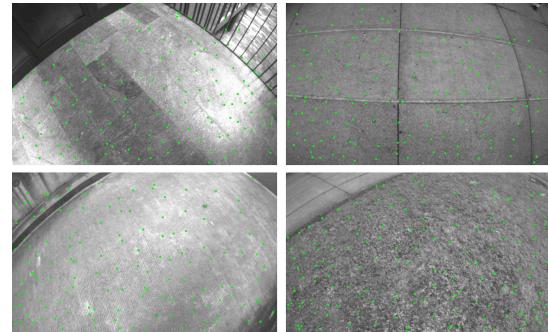


Fig. 15: Successful tracking in scenes of high-frequency texture.

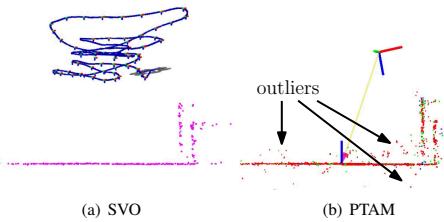


Fig. 16: Sideview of a piecewise-planar map created by SVO and PTAM. The proposed method has fewer outliers due to the depth-filter.

are reliable. Motion estimation for the *accurate* parameter setting takes on average 6ms on the laptop. The increase in time is mainly due to local BA, which is run at every keyframe and takes 14ms. The time required by the mapping thread to update all depth-filters with the new frame is highly dependent on the number of filters. The number of filters is high after a keyframe is selected and reduces quickly as filters converge. On average, the mapping thread is faster than the motion estimation thread, thus it is not a limiting factor.

C. Robustness

The speed and accuracy of SVO is partially due to the depth-filter, which produces only a minimal number of outlier 3D points. Also the robustness is due to the depth-

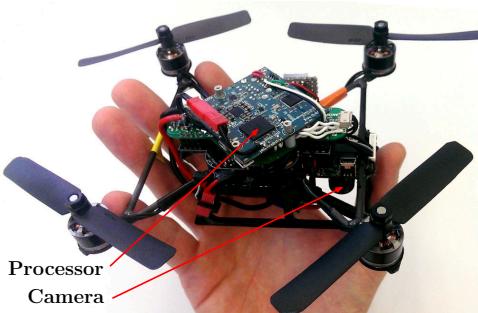


Fig. 17: “Nano+” by KMeI Robotics, customized with embedded processor and downward-looking camera. SVO runs at 55 frames per second on the platform and is used for stabilization and control.

filter: precise, high frame-rate tracking allows the filter to converge even in scenes of repetitive and high-frequency texture (e.g., asphalt, grass), as it is best demonstrated in the video accompanying this paper. Screenshots of the video are shown in Figure 15. Figure 16 shows a comparison of the map generated with PTAM and SVO in the same scene. While PTAM generates outlier 3D points, by contrast SVO has almost no outliers thanks to the use of the depth-filter.

VIII. CONCLUSION

In this paper, we proposed the semi-direct VO pipeline “SVO” that is precise and faster than the current state-of-the-art. The gain in speed is due to the fact that feature-extraction and matching is not required for motion estimation. Instead, a direct method is used, which is based directly on the image intensities. The algorithm is particularly useful for state-estimation onboard MAVs as it runs at more than 50 frames per second on current embedded computers. High frame-rate motion estimation, combined with an outlier resistant probabilistic mapping method, provides increased robustness in scenes of little, repetitive, and high frequency-texture.

REFERENCES

- [1] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, “Vision based MAV navigation in unknown and unstructured environments,” *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010.
- [2] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, “Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium,” *Journal of Field Robotics*, vol. 30, no. 5, 2013.
- [3] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, “Vision-Controlled Micro Flying Robots: from System Design to Autonomous Navigation and Mapping in GPS-denied Environments,” *IEEE Robotics and Automation Magazine*, 2014.
- [4] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, “Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.
- [5] C. Forster, M. Pizzoli, and D. Scaramuzza, “Air-Ground Localization and Map Augmentation Using Monocular Dense Reconstruction,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.
- [6] L. Kneip, M. Chli, and R. Siegwart, “Robust Real-Time Visual Odometry with a Single Camera and an IMU,” *Proc. British Machine Vision Conference*, 2011.
- [7] J. Engel, J. Sturm, and D. Cremers, “Accurate Figure Flying with a Quadrocopter Using Onboard Visual and Inertial Sensing,” in *Proc. ViCoMoR Workshop at IEEE/RSJ IROS*, 2012.
- [8] A. Comport, E. Malis, and P. Rives, “Real-time Quadrifocal Visual Odometry,” *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 245–266, Jan. 2010.
- [9] T. Tykkälä, C. Audras, and A. I. Comport, “Direct Iterative Closest Point for Real-time Visual Odometry,” in *Int. Conf. on Computer Vision*, 2011.
- [10] C. Kerl, J. Sturm, and D. Cremers, “Robust Odometry Estimation for RGB-D Cameras,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2013.
- [11] M. Meißland and A. I. Comport, “On unifying key-frame and voxel-based dense visual SLAM at large scales,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.
- [12] D. Scaramuzza and F. Fraundorfer, “Visual Odometry, Part I: The First 30 Years and Fundamentals [Tutorial],” *IEEE RAM*, 2011.
- [13] M. Irani and P. Anandan, “All About Direct Methods,” in *Proc. Workshop Vis. Algorithms: Theory Pract.*, 1999, pp. 267–277.
- [14] S. Lovegrove, A. J. Davison, and J. Ibanez-Guzman, “Accurate visual odometry from a rear parking camera,” in *Intelligent Vehicle, IEEE Symposium*, 2011.
- [15] R. a. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense Tracking and Mapping in Real-Time,” *IEEE Int. Conf. on Computer Vision*, pp. 2320–2327, Nov. 2011.
- [16] G. Klein and D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, Nov. 2007.
- [17] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Real-time Monocular SLAM: Why Filter?” *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2657 – 2664, 2010.
- [18] H. Jin, P. Favaro, and S. Soatto, “A semi-direct approach to structure from motion,” *The Visual Computer*, vol. 19, no. 6, pp. 377–394, 2003.
- [19] N. D. Molton, A. J. Davison, and I. Reid, “Locally Planar Patch Features for Real-Time Structure from Motion,” in *Proc. British Machine Vision Conference*, 2004.
- [20] G. Silveira, E. Malis, and P. Rives, “An Efficient Direct Approach to Visual SLAM,” *IEEE Transactions on Robotics*, 2008.
- [21] C. Mei, S. Benhimane, E. Malis, and P. Rives, “Efficient Homography-based Tracking and 3-D Reconstruction for Single Viewpoint Sensors,” *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1352 – 1364, 2008.
- [22] A. Pretto, E. Menegatti, and E. Pagello, “Omnidirectional Dense Large-Scale Mapping and Navigation Based on Meaningful Triangulation,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2011.
- [23] J. Engel, J. Sturm, and D. Cremers, “Semi-Dense Visual Odometry for a Monocular Camera,” in *Proc. IEEE Int. Conf. on Computer Vision*.
- [24] S. Benhimane and E. Malis, “Integration of Euclidean constraints in template based visual tracking of piecewise-planar scenes,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.
- [25] R. Kümmeler, G. Grisetti, and K. Konolige, “g2o: A General Framework for Graph Optimization,” *Proc. IEEE Int. Conf. on Robotics and Automation*, 2011.
- [26] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2005.
- [27] S. Baker and I. Matthews, “Lucas-Kanade 20 Years On: A Unifying Framework: Part 1,” *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2002.
- [28] G. Vogiatzis and C. Hernández, “Video-based, Real-Time Multi View Stereo,” *Image and Vision Computing*, vol. 29, no. 7, 2011.
- [29] M. Pizzoli, C. Forster, and D. Scaramuzza, “REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2014.
- [30] E. Rosten, R. Porter, and T. Drummond, “FASTER and better: A machine learning approach to corner detection,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.
- [31] S. Umeyama, “Least-Squares Estimation of Transformation Parameters Between Two Point Patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, 1991.
- [32] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A Benchmark for the Evaluation of RGB-D SLAM Systems,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012.
- [33] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, “A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation,” in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

13.4 Robust Visual Inertial Odometry Using a Direct EKF-Based Approach, IROS, 2015

2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
 Congress Center Hamburg
 Sept 28 - Oct 2, 2015. Hamburg, Germany

Robust Visual Inertial Odometry Using a Direct EKF-Based Approach

Michael Bloesch, Sammy Omari, Marco Hutter, Roland Siegwart
 Autonomous Systems Lab, ETH Zürich, Switzerland, bloeschm@ethz.ch

Abstract—In this paper, we present a monocular visual-inertial odometry algorithm which, by directly using pixel intensity errors of image patches, achieves accurate tracking performance while exhibiting a very high level of robustness. After detection, the tracking of the multilevel patch features is closely coupled to the underlying extended Kalman filter (EKF) by directly using the intensity errors as innovation term during the update step. We follow a purely robocentric approach where the location of 3D landmarks are always estimated with respect to the current camera pose. Furthermore, we decompose landmark positions into a bearing vector and a distance parametrization whereby we employ a minimal representation of differences on a corresponding σ -Algebra in order to achieve better consistency and to improve the computational performance. Due to the robocentric, inverse-distance landmark parametrization, the framework does not require any initialization procedure, leading to a truly power-up-and-go state estimation system. The presented approach is successfully evaluated in a set of highly dynamic hand-held experiments as well as directly employed in the control loop of a multirotor unmanned aerial vehicle (UAV).

I. INTRODUCTION

Navigation and control of autonomous robots in rough and highly unstructured environments requires high-bandwidth and precise knowledge of position and orientation. Especially in dynamic operation of robots, the underlying state estimation can quickly become the bottleneck in terms of achievable bandwidth, robustness and speed. To enable the required performance for highly dynamic operation of robots, we combine complementary information from vision- and inertial sensors. This approach has a long history and has been successfully applied to navigate unmanned aerial robots [24], [21], walking robots [23], [25] or cars [8].

Within the field of computer vision, Davison et al. [5] proposed one of the first real-time 3D monocular localization and mapping frameworks. Since then, a lot of improvements have been contributed from various research groups and further approaches have been proposed. A key issue is to improve the consistency of the estimation framework which is affected by its inherent nonlinearity [13], [3]. One approach is to make use of a robocentric representation for the tracked features and thereby significantly reduce the effect of nonlinearities [3], [4]. As alternative, Huang et al. [11] propose the use of a so-called observability constrained extended Kalman filter, whereby the inconsistencies can be avoided by using special linearization points while evaluating the system Jacobians.

This research was supported in part by the Swiss National Science Foundation (SNF) through project 200021_149427/1 and the National Centre of Competence in Research Robotics.

A somewhat related problem is the choice of the specific representation of the features. Since for monocular setups, the depth of a newly detected feature is unknown the initial 3D location estimate of the feature exhibits a high (infinite) uncertainty along the corresponding axis. In order to integrate this feature from the beginning into the estimation framework, Montiel et al. [18] proposed the use of an inverse-depth parametrization (IDP). With this parametrization, each feature location is represented by the camera position where the feature was initially detected, by a bearing vector (parametrized with azimuth and elevation angle), as well as the inverse depth of the feature. The resulting increase in consistency was analyzed in more detail for the IDP and other feature parametrization in [22].

While most standard visual odometry approaches are based on detected and tracked point features as source of visual information, so-called *direct* approaches directly use the image intensities in their estimation framework. Especially with the recent advent of RGBD cameras, so called *dense* approaches, where the intensity error over the full image is considered, have gained a lot of attention [1], [15]. In comparison to traditional vision-based state estimators, dense approaches have a significantly larger error term count and require appropriate methods in order to tackle the additional computational load. By employing highly optimized SSE-vectorized implementations, first real-time, CPU-based approaches for dense- or semi-dense motion estimation using a RGBD [15] or a monocular RGB camera [6], [7] have recently been proposed.

Incorporating inertial measurements in the estimation can significantly improve the robustness of the system, provides the estimation process with the notion of gravity, and allows for a more accurate and high bandwidth estimation of the velocities and rotational rates. By adapting the original EKF proposed by Davison et al. [5], additional IMU measurements can be relatively simply integrated into the ego-motion estimation, whereby calibration parameters can be co-estimated online [14], [12]. Leutenegger et al. [16] describe a *tightly* coupled approach in which the robot trajectory and sparse 3D landmarks are estimated in a joint optimization problem using inertial error terms as well as the reprojection error of the landmarks in the camera image. This is done in a windowed bundle adjustment approach over a set of keyframe images and a temporal inertial measurement window. Similarly, in [19] the authors estimate the trajectory in an IMU-driven filtering framework using the reprojection error of 3D landmarks as measurement updates. Instead of adding the landmarks to the filter state, they immediately

marginalize them out using a nullspace decomposition, thus leading to a small filter state size.

In the present paper we propose a visual-inertial odometry framework which combines and extends several of the above mentioned approaches. While targeting a simple and consistent approach and avoiding ad-hoc solutions, we adapt the structure of the standard visual-inertial EKF-SLAM formulation [14], [12]. The following keypoints are integrated into the proposed framework:

- Point features are parametrized by a bearing vector and a distance parameter with respect to the current frame. A suitable σ -Algebra is used for deriving the corresponding dynamics and performing filtering operations.
- Multilevel patch features are directly tracked within the EKF, whereby the intensity errors are used as innovation terms during the update step.
- A QR-decomposition is employed in order to reduce the high dimensional error terms and thus keep the Kalman update computationally tractable.
- A purely robocentric representation of the *full* filter state is employed. The camera extrinsics as well as the additive IMU biases are also co-estimated.

Together this yields a *fully robocentric* and *direct* monocular visual-inertial odometry framework which can be run real-time on a single standard CPU core. In several experiments on real data we show its reliable and accurate tracking performance while exhibiting a high robustness against fast motions and various disturbances. The framework is implemented in c++ and is available as open-source software [2].

II. FILTER SETUP

A. Overall Filter Structure and State Parametrization

The overall structure of the filter is derived from the one employed in [14], [12]: The inertial measurements are used to propagate the state of the filter, while the visual information is taken into account during the filter update steps. As a fundamental difference we make use of a fully robocentric representation of the filter state which can be seen as an adaptation of [4] (which is vision-only). One advantage of this formulation is that problems with unobservable states can inherently be avoided and thus the consistency of the estimates can be improved. On the other hand noise from the gyroscope will affect all states that need to be rotated during the state propagation (see section II-B). However, since the gyroscope noise is relatively small and because most states are observable this does not represent a significant issue.

Three different coordinate frames are used throughout the paper: the inertial world coordinate frame, \mathcal{I} , the IMU fixed coordinate frame, \mathcal{B} , as well as the camera fixed coordinate frame, \mathcal{V} . For tracking N visual features, we use the following filter state:

$$\mathbf{x} := (\mathbf{r}, \mathbf{v}, \mathbf{q}, \mathbf{b}_f, \mathbf{b}_\omega, \mathbf{c}, \mathbf{z}, \boldsymbol{\mu}_0, \dots, \boldsymbol{\mu}_N, \rho_0, \dots, \rho_N), \quad (1)$$

with:

- \mathbf{r} : robocentric position of IMU (expressed in \mathcal{B}),
- \mathbf{v} : robocentric velocity of IMU (expressed in \mathcal{B}),

- \mathbf{q} : attitude of IMU (map from \mathcal{B} to \mathcal{I}),
- \mathbf{b}_f : additive bias on accelerometer (expressed in \mathcal{B}),
- \mathbf{b}_ω : additive bias on gyroscope (expressed in \mathcal{B}),
- \mathbf{c} : translational part of IMU-camera extrinsics (expressed in \mathcal{B}),
- \mathbf{z} : rotational part of IMU-camera extrinsics (map from \mathcal{B} to \mathcal{V}),
- $\boldsymbol{\mu}_i$: bearing vector to feature i (expressed in \mathcal{V}),
- ρ_i : distance parameter of feature i .

The generic parametrization for the distance d_i of a feature i is given by the mapping $d_i = d(\rho_i)$ (with derivative $d'(\rho_i)$). In the context of this work we mainly tested the inverse distance parametrization, $d(\rho_i) = 1/\rho_i$. The investigation of further parametrization will be part of future work.

Rotations ($\mathbf{q}, \mathbf{z} \in SO(3)$) and unit vectors ($\boldsymbol{\mu}_i \in S^2$) are parametrized by following the approach of Hertzberg et al. [10]. This is required in order to perform operations like computing differences or derivatives as well as representing the uncertainty of the state in a minimal manner. For parametrizing unit vectors we employ rotations as underlying representation, whereby we define a \boxminus -operator which returns a difference between two unit vectors within a 2D linear subspace. The advantage of this parametrization is that the tangent space can be easily computed (which is used for defining the \boxminus -operator).

By using the combined bearing vector and distance parameterization, features can be initialized in an *undelayed* manner, i.e., the features are integrated into the filter at detection. The distance of a feature is initialized with a fixed value or, if sufficiently converged, with an estimate of the current average scene distance. The corresponding covariance is set to a very large value. In comparison to other parameterizations we do not over-parametrize the 3D feature location estimates, whereby each feature corresponds to 3 columns in the covariance matrix of the state (2 for the bearing vector and 1 for the distance parameter). This also avoids the need for re-parameterization [22].

B. State Propagation

Based on the proper acceleration measurement, $\tilde{\mathbf{f}}$, and the rotational rate measurement, $\tilde{\boldsymbol{\omega}}$, the evaluation of the IMU driven state propagation results in the following set of continuous differential equations (the superscript \times denotes the skew symmetric matrix of a vector):

$$\dot{\mathbf{r}} = -\hat{\boldsymbol{\omega}}^\times \mathbf{r} + \mathbf{v} + \mathbf{w}_r, \quad (2)$$

$$\dot{\mathbf{v}} = -\hat{\boldsymbol{\omega}}^\times \mathbf{v} + \hat{\mathbf{f}} + \mathbf{q}^{-1}(\mathbf{g}), \quad (3)$$

$$\dot{\mathbf{q}} = -\mathbf{q}(\hat{\boldsymbol{\omega}}), \quad (4)$$

$$\dot{\mathbf{b}}_f = \mathbf{w}_{bf}, \quad (5)$$

$$\dot{\mathbf{b}}_\omega = \mathbf{w}_{b\omega}, \quad (6)$$

$$\dot{\mathbf{c}} = \mathbf{w}_c, \quad (7)$$

$$\dot{\mathbf{z}} = \mathbf{w}_z, \quad (8)$$

$$\dot{\boldsymbol{\mu}}_i = \mathbf{N}^T(\boldsymbol{\mu}_i) \hat{\boldsymbol{\omega}}_{\mathcal{V}} - \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{N}^T(\boldsymbol{\mu}_i) \frac{\dot{\mathbf{v}}_{\mathcal{V}}}{d(\rho_i)} + \mathbf{w}_{\mu,i}, \quad (9)$$

$$\dot{\rho}_i = -\boldsymbol{\mu}_i^T \dot{\mathbf{v}}_{\mathcal{V}} / d'(\rho_i) + w_{\rho,i}, \quad (10)$$

where $N^T(\mu)$ linearly projects a 3D vector onto the 2D tangent space around the bearing vector μ , with the bias corrected and noise affected IMU measurements:

$$\hat{f} = \tilde{f} - b_f - w_f, \quad (11)$$

$$\hat{\omega} = \tilde{\omega} - b_\omega - w_\omega, \quad (12)$$

and with the camera linear velocity and rotational rate:

$$\hat{v}_V = z(v + \hat{\omega}^\times c), \quad (13)$$

$$\hat{\omega}_V = z(\hat{\omega}). \quad (14)$$

Furthermore, g is the gravity vector expressed in the world coordinate frame, and the terms of the form w_* are white Gaussian noise processes. The corresponding covariance parameters can either be taken from the IMU specifications or have to be tuned manually. Using an appropriate Euler forward integration scheme, i.e., using the \boxplus -operator where appropriate, the above time continuous equation can be transformed into a set of discrete prediction equations which are used during the prediction of the filter state [10].

Please note that the derivatives of bearing vectors and rotations lie within 2D and 3D vector spaces, respectively. This is required for achieving a minimal and consistent representation of the filter state and covariance.

C. Filter Update

For every captured image we perform a state update. We assume that we know the intrinsic calibration of the camera and can therefore compute the projection of a bearing μ to the corresponding pixel coordinate $p = \pi(\mu)$. As will be described in section III-B, we derive a 2D linear constraint, $b_i(\pi(\hat{\mu}_i))$, for each feature i which is predicted to be visible in the current frame with bearing vector $\hat{\mu}_i$. This linear constraint encompasses the intensity errors associated with a specific feature and can be directly employed as innovation term within the Kalman update (affected by additive discrete Gaussian pixel intensity noise n_i):

$$y_i = b_i(\pi(\hat{\mu}_i)) + n_i, \quad (15)$$

together with the Jacobian:

$$H_i = A_i(\pi(\hat{\mu}_i)) \frac{d\pi}{d\mu}(\hat{\mu}_i). \quad (16)$$

By stacking the above terms for all visible features we can directly perform a standard EKF update. However, if the initial guess for a certain bearing vector $\hat{\mu}_i$ has a large uncertainty the update will potentially fail. This typically occurs if features get newly initialized and exhibit a large distance uncertainty. In order to avoid this issue we improve the initial guess for a bearing vector with large uncertainty by performing a patch based search of the feature (section III-B). This basically improves the linearization point of the EKF by using the bearing vector obtained from the patch search $\bar{\mu}_i$ for evaluating the terms in eqs. (15) and (16). Please note that the EKF update equations have to be slightly adapted in order to account for the altered linearization point. A similar alternative would be to directly employ an iterative EKF.

In order to account for moving objects or other disturbances, a simple Mahalanobis based outlier detection is implemented within the update step. It compares the obtained innovation with the predicted innovation covariance and rejects the measurement whenever the weighted norm exceeds a certain threshold. This method inherently takes into account the covariance of the state and measurements. For instance it also considers the image gradients and thereby tends to reject gradient-less image patches easier.

III. MULTILEVEL PATCH FEATURE HANDLING

Along the lines of other visual-inertial EKF approaches ([14], [12]) we fully integrate visual features into the state of the Kalman filter (see also section II-A). Within the prediction step the new locations of the multilevel patch features are estimated by considering the IMU-driven motion model (eq. (9)). Especially if the calibration of the extrinsics and the feature distance parameters have converged, this yields high quality predictions for the feature locations. Additionally, the covariance of the predicted pixel location can be easily computed and the computational effort of a possible pre-alignment strategy can be adapted accordingly. The subsequent update step computes an innovation term by evaluating the discrepancy between the projection of the multilevel patch into the image frame and the image itself. Considering the cross-correlation between the states the EKF spreads the resulting corrections throughout the filter state. In the following the different steps and algorithms involving feature handling are discussed in more details. The overall workflow for a single feature is depicted in fig. 1.

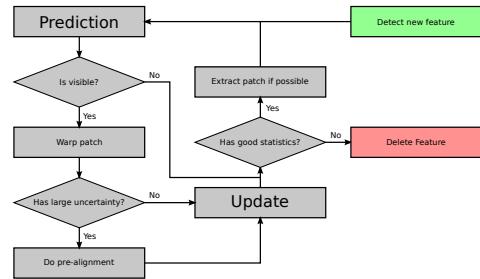


Fig. 1. Overview on the workflow of a feature in the filter state. The heuristics for adding and removing features are adapted to the total number of possible features.

A. Structure and Warping

For a given image pyramid (factor 2 down-sampling) and a given bearing vector μ a multilevel patch is obtained by extracting constant size (here 8x8 pixels) patches, P_l , for each image level l at the corresponding pixel coordinate $p = \pi(\mu)$. The advantage is that tracking such features is robust against bad initial guesses and image blur. Furthermore such patch features allow a *direct* intensity error feedback into the filter. In comparison to reprojection error based algorithms this allows to formulate a more accurate error model which

inherently takes into account the texture of the tracked image patch. For instance it also enables the use of edge features, whereby the gained information would be along the perpendicular to the edge.

By tracking two additional bearing vectors within the patch, we can compute an affine warping matrix $\mathbf{W} \in \mathbb{R}^{2 \times 2}$ in order to account for the local distortion of the patches between subsequent images. We assume that the distance of the feature is large w.r.t. the size of the patch and can thus choose the normal of the patches to point towards the center of the camera. Also, when a feature was successfully tracked within a frame, the multilevel patch is re-extracted in order to avoid the accumulation of errors.

B. Alignment Equations and QR-decomposition

Throughout the framework we make use of intensity errors in order to pre-align features or update the filter state. For a given image pyramid with images I_l and a given multilevel patch feature (with coordinates \mathbf{p} and patches P_l) the following intensity errors can be evaluated for image level l and patch pixel p_j :

$$e_{l,j} = P_l(\mathbf{p}_j) - I_l(\mathbf{p}_j s_l + \mathbf{W}\mathbf{p}_j) - m, \quad (17)$$

where the scalar $s_l = 0.5^l$ accounts for the down-sampling between the images of the image pyramid. Furthermore, by subtracting the mean intensity error m we can account for inter-frame illumination changes.

For regular patch alignment, the squared error terms of eq. (17) can be summed over all image levels and patch pixels and combined into a single Gauss-Newton optimization in order to find the optimal patch coordinates. However, the direct use of such a large number of error terms within an EKF would make it computationally intractable. In order to tackle this issue we apply a QR-decomposition on the linear equation system resulting from stacking all error terms in eq. (17) together for given estimated coordinates $\hat{\mathbf{p}}$:

$$\bar{\mathbf{b}}(\hat{\mathbf{p}}) = \bar{\mathbf{A}}(\hat{\mathbf{p}})\delta\mathbf{p}, \quad (18)$$

where $\bar{\mathbf{A}}(\hat{\mathbf{p}})$ can be computed based on the patch intensity gradients. Independent of the rank of the matrix $\bar{\mathbf{A}}(\hat{\mathbf{p}})$, the QR-decomposition of $\bar{\mathbf{A}}(\hat{\mathbf{p}})$ can be used to obtain an equivalent reduced linear equation system:

$$\mathbf{b}(\hat{\mathbf{p}}) = \mathbf{A}(\hat{\mathbf{p}})\delta\mathbf{p}, \quad (19)$$

with $\mathbf{A}(\hat{\mathbf{p}}) \in \mathbb{R}^{2 \times 2}$ and $\mathbf{b}(\hat{\mathbf{p}}) \in \mathbb{R}^2$. Since we assume that the additive noise magnitude on the intensities is equal for every patch pixel we can leave it out of the above derivations (it will remain constant for every entry).

One interesting remark is, that due to the scaling factor s_l in eq. (17), error terms for higher image levels will have a weaker corrective influence on the filter state or the patch alignment. On the other hand, their increased robustness w.r.t. image blur or bad initial alignment strongly increases the robustness of the overall alignment method for multilevel patch features.

C. Feature Detection and Removal

The detection of new features is based on a standard fast corner detector which provides a large amount of candidate feature locations. After removing candidates which are close to current tracked features, we compute an adapted Shi-Tomasi score for selecting new features which will be added to the state. The adapted Shi-Tomasi score basically considers the combined Hessian on multiple image levels, instead of only a single level. It directly approximates the Hessian of the above gradient matrix with $\mathbf{H} = \bar{\mathbf{A}}^T(\hat{\mathbf{p}})\bar{\mathbf{A}}(\hat{\mathbf{p}})$ and extracts the minimal eigenvalue. The advantage is that a high score is directly correlated with the alignment accuracy of the corresponding multilevel patch feature. Instead of returning the minimal eigenvalue, the method can return other eigenvalue based scores like the 1- or 2-norm. This could be useful in environments with scarce corner data, whereby the presented filter could be complemented by available edge-shaped features. Finally, the detection process is also coupled to a bucketing technique in order to achieve a good distribution of the features within the image frame.

Due to the fact that we can only track a limited number of features in the EKF, we have to implement a landmark management system to ensure that only reliable landmarks are inserted and kept in the filter state. Here, we fall back to heuristic methods, where we compute quality scores in order to decide whether a feature should be kept or not. The overall idea is to evaluate a local (only last few frames) and a global (how good was the feature tracked since it has been detected) quality score and remove the features below a certain threshold. Using an adaptive threshold we can control the total amount of features which are currently in the frame.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

The data for the experiments were recorded with the VI-Sensor [20], equipped with two time-synchronized, global-shutter, wide-VGA 1/3 inch imagers in a fronto-parallel stereo configuration. The cameras are equipped with lenses with a diagonal field of view of 120 degrees and are factory-calibrated by the manufacturer for a standard pinhole projection model and a radial-tangential distortion model. The imagers are hardware time-synchronized to the IMU to ensure mid-exposure IMU triggering. In the context of this work only the image stream from one camera is required.

Ground truth is provided through an external motion capture system for the pose of the sensor. The rate of the IMU measurements is 200 Hz and the image frame rate is 20 Hz. The employed IMU is an industrial-grade ADIS 16448, with an angular random walk of 0.66 deg/√Hz and a velocity random walk of 0.11 m/s/√Hz. The maximal number of features in the state is set to 50 and the algorithm is run using image pyramids with 4 levels. Whenever possible, covariance parameters are selected based on hardware specifications. Strong tuning was not necessary, and the framework works well for a large range of parameters. The initial IMU-camera extrinsics are only roughly guessed (the translation is set to

zero), and the initial inverse distance parameter for a feature is set to 0.5 m^{-1} with a standard deviation of 1 m^{-1} . A screenshot of the running framework is depicted in fig. 2.



Fig. 2. Screenshot of the running visual-inertial odometry framework. The 2σ uncertainty ellipses of the predicted feature locations are in yellow, whereby only features which are newly initialized (stretched ellipses) and features which re-enter the frame have a significant uncertainty. Green points are the locations after the update step. Green numbers are the tracking counts (1 for newly initialized features). In the top left a virtual horizon is depicted.

B. Experiment with Slow Motions

An experiment with slow to medium fast hand-held motions of about 1 min was carried out to evaluate the performance of the framework with different numbers of total features (from 10 to 50 in steps of 10). The performance was assessed by computing the relative position error w.r.t. the traveled distance [9]. Furthermore we compared the obtained results to a batch optimization framework along the lines of [16]. Figure 3 depicts the extracted relative error values. The achieved performance tends to be similar to the one of the batch optimization framework and often achieves slightly higher accuracy. While these results depend on the specific dataset and parameter tuning, we also have to mention that the relatively high rotational motion (average of around 1.5 rad/s) favors approaches which can handle arbitrarily short feature tracks. Given the *undelayed* initialization of feature within our approach, the resulting filter is able to extract visual information from a feature's second observation onwards.

Surprisingly, the performance was relatively independent of the total amount of tracked features. A significant drop in accuracy could only be observed with feature counts below 20. This observation can have different reasons. One could be the type of sensor motions with relatively high rotational rates, which can lead to more bad features or outliers. Another point is also that our approach considers $256 = 4 \times 8 \times 8$ intensity errors per tracked features and thus we cannot directly compare to standard feature tracking based visual odometry frameworks, which typically require much higher feature counts. More in-depth evaluation of this effect will be part of future work. The timings of the proposed framework are listed in table I for a single core

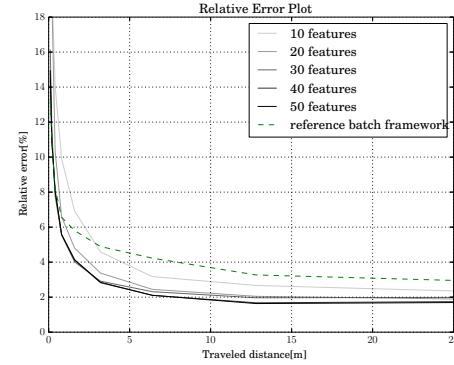


Fig. 3. Gray lines are the relative errors of the presented approach, where the darkest lines corresponds to 50 features and the brightest line to 10 features respectively. The dashed green line represents the performance of the reference batch optimization framework.

TABLE I
TIMINGS OF PRESENTED APPROACH PER PROCESSED IMAGE

Tot. Features	10	20	30	40	50
Timing [ms]	6.65	10.50	14.87	21.48	29.72

of an Intel i7-2760QM. The setup with 50 features uses an average processing time of 29.72 ms per processed image and can thus easily be run at 20 Hz .

C. Experiment with Fast Motions

Here, we evaluate the robustness of the proposed approach w.r.t. very fast motions. We recorded a hand-held dataset with mean rotational rate of around 3.5 rad/s and with peaks of up to 8 rad/s . The motion capture system exhibited a relative high number of bad tracking, whereby we filtered them out as good as possible. We investigate the tracking performance of the attitude and of the robocentric velocities, where the corresponding estimates with 3σ -bounds are plotted in figs. 4 and 5 respectively. It can clearly be seen that the estimates nicely fit the ground truth data from the motion capture. As known from previous work the inclination angles and the robocentric velocities of visual-inertial setups are fully observable [17], and we can nicely observe the initial decrease of the corresponding covariance (especially when the system gets excited). On the other hand the yaw angle is unobservable and drifts slowly with time.

Figures 6 and 7 depict the estimation of the calibration parameters. Again, the estimates together with their 3σ -bounds are plotted. Depending on the excitation of the system the estimated values converge relatively quickly. It can be observed, that the translational term of the IMU-camera calibration requires a lot of rotational motion in order to converge appropriately. For the presented experiment, the accelerometer bias exhibits the worse convergence rate but is still within a reasonable range.

Furthermore, we also observed a divergence mode for the presented approach. It can occur when the velocity estimate diverge, e.g., due to missing motion or too many outliers. The problem is then, that the filter attempts to minimize the effect of the erroneous velocity on the bearing vectors by setting the distance of the features to infinity. This again lowers any corrective effect on the diverging velocity resulting in further divergence. All in all this was very rarely observed for regular usage, especially if the system was properly excited at the start.

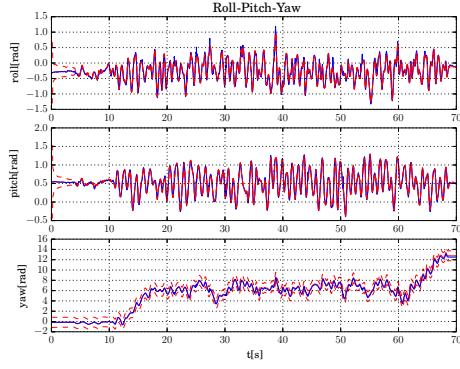


Fig. 4. Euler angle estimates. Red: estimate, blue: motion capture, red dashed: 3σ -bound. Only the yaw angle is not observable and exhibits a growing covariance. The inclination angles (roll and pitch) exhibit a high quality tracking accuracy.

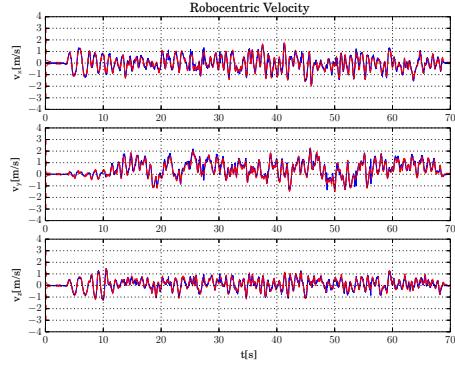


Fig. 5. Velocity estimates. Red: estimate, blue: motion capture, red dashed: 3σ -bound. The robo-centric velocity is fully observable and thus exhibits a bounded uncertainty. It very nicely tracks the reference from the motion capture system (and probably also exhibits a higher precision).

D. Flying Experiments

Implementing the framework on-board a UAV with a forward oriented visual-inertial sensor, we also performed

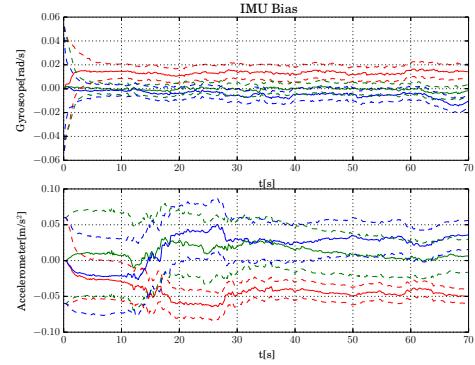


Fig. 6. Estimated IMU biases. Top: gyroscope bias (red: x, blue: y, green: z), bottom: accelerometer bias (red: x, blue: y, green: z). The gyroscope biases exhibit a better convergence than the accelerometer biases, probably due to the more direct link of rotational rates to visual errors.

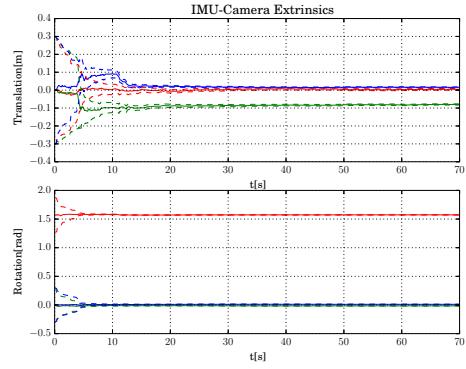


Fig. 7. Estimated IMU-camera extrinsics. Top: translation (red: x, blue: y, green: z), bottom: orientation (red: yaw, blue: pitch, green: roll). Especially when sufficiently excited, the estimates converge quickly. The reached values correspond approximately to the ones obtained from an offline calibration.

preliminary experiments on a real robot. The special aspect here is that the visual-inertial odometry framework was initialized on the ground without any previous calibration motions, i.e. the calibration parameters had to converge during take-off. The output of the filter was directly used for feedback control of the UAV. Figure 8 depicts the estimated position output of the framework during take-off, flying and landing. If compared to the motion capture system the filter exhibits a certain offset which can be mainly attributed to the online calibration of the filter.

V. CONCLUSION

In this paper we presented a visual-inertial filtering framework which uses direct intensity errors as visual measurements within the extended Kalman filter update. By choosing

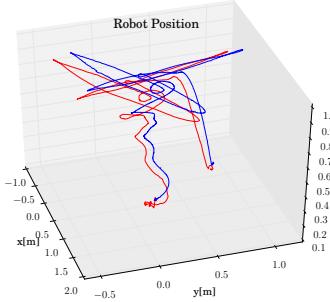


Fig. 8. Estimated trajectory (red) on-board a UAV compared to groundtruth (blue) from the motion capture system. During take-off, flying, and landing the output of the filter is used to stabilize and control the UAV. Calibration is performed online.

a fully robocentric representation of the filter state together with a numerically minimal bearing/distance representation of features, we avoid major consistency problems while exhibiting accurate tracking performance and high robustness. Especially in difficult situations with very fast motions or outliers the presented approach manages to keep track of the state with only minor drift of the yaw and position estimates. The framework can be run on-board a UAV with a feature count of 50 at a framerate of 20 Hz and was used to stabilize the flight of a UAV from take-off to landing.

Future work will include more extensive evaluation of the multilevel patch features in context of intensity error based visual-inertial odometry frameworks. Furthermore we would also like to try to extend the online calibration in order to include the camera intrinsics. Also, the framework could be relatively easily adapted in order to handle multiple cameras. This could improve the filter performance, especially for cases with lack of translational motion. Another option to avoid divergence would be to use some heuristics based methods in order to detect such modes and to add zero-velocity pseudo-measurements in order to stabilize the filter. A detailed observability analysis could also be performed, where the dependency of unobservable modes w.r.t. sensor motions would be of high interest.

REFERENCES

- [1] C. Audras, A. I. Comport, M. Meillard, and P. Rives, "Real-time dense appearance-based SLAM for RGB-D sensors," in *Australasian Conf. on Robotics and Automation*, 2011.
- [2] M. Bloesch, S. Omari, and A. Jaeger, "ROVIO," 2015. [Online]. Available: <https://github.com/ethz-asl/rovio>
- [3] J. A. Castellanos, J. Neira, and J. D. Tardos, "Limits to the consistency of EKF-based SLAM," in *5th IFAC Symp. on Intelligent Autonomous Vehicles*, 2004.
- [4] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point RANSAC for EKF-based Structure from Motion," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009.
- [5] A. J. Davison, "Real-Time Simultaneous Localisation and Mapping with a Single Camera," in *IEEE Int. Conference on Computer Vision*, 2003.
- [6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conf. on Computer Vision*, 2014.
- [7] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO : Fast Semi-Direct Monocular Visual Odometry," in *IEEE Int. Conf. on Robotics and Automation*, 2014.
- [8] S. Gehrig, F. Eberli, and T. Meyer, "A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching," *Computer Vision Systems*, vol. 5815, pp. 134–143, 2009.
- [9] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. October, pp. 1231–1237, 2013.
- [10] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2011.
- [11] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and improvement of the consistency of extended Kalman filter based SLAM," in *IEEE Int. Conf. on Robotics and Automation*, May 2008.
- [12] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.
- [13] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *IEEE Int. Conf. on Robotics and Automation*, May 2001.
- [14] J. Kelly and G. S. Sukhatme, "Visual-Inertial Sensor Fusion: Localization, Mapping, and Sensor-to-Sensor Self-calibration," *Int. Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, Nov. 2011.
- [15] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *IEEE Int. Conf. on Robotics and Automation*, IEEE, May 2013.
- [16] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.
- [17] M. Li and a. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.
- [18] J. Montiel, J. Civera, and A. Davison, "Unified Inverse Depth Parametrization for Monocular SLAM," in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, Aug. 2006.
- [19] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *IEEE Int. Conf. on Robotics and Automation*, 2007.
- [20] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM," in *IEEE Int. Conf. on Robotics and Automation*, 2014.
- [21] S. Shen, M. N. Mulgaonkar, Y., and V. Kumar, "Initialization-free monocular visual-inertial estimation with application to autonomous MAVs," in *International Symposium on Experimental Robotics*, 2014.
- [22] J. Solà, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *International Journal of Computer Vision*, vol. 97, pp. 339–368, 2012.
- [23] A. Stelzer, H. Hirschmüller, and M. Gorner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 381–402, Feb. 2012.
- [24] S. Weiss, M. Achterlik, S. Lynen, L. Kneip, M. Chli, and R. Siegwart, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.
- [25] D. Wooden, M. Malchano, K. Blankepoor, A. Howard, A. A. Rizzi, and M. Raibert, "Autonomous navigation for BigDog," in *IEEE Int. Conf. on Robotics and Automation*, 2010.

14

Vision Based SLAM

Do relative-state SLAM. James, Tim, Dan.

Bibliography

Timothy Barfoot. *State Estimation For Robotics*. Cambridge University Press, 2019.

Randal W Beard and Timothy W McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.

Ian D Cowling, Oleg A Yakimenko, James F Whidborne, and Alastair K Cooke. A prototype of an autonomous controller for a quadrotor UAV. In *Proceedings of the European Control Conference*, Kos, Greece, jul 2007.

Tom Drummond. Lie groups, lie algebras, projective geometry and optimization for 3d geometry, engineering and computer vision, 2014.

Available at <https://www.dropbox.com/s/5y3tvypzps59s29/3DGeometry.pdf?dl=0>.

Ethan Eade. Lie groups for 2d and 3d transformations, 2017. Available at <http://ethaneade.com/lie.pdf>.

Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, February 2017.

Fei Gao, William Wu, Yi Lin, and Shaojie Shen. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 344–351, Brisbane, Australia, May 2018.

Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Springer-Verlag New York, Inc, 2003. ISBN 0-387-40122-9.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.

Dinesh Atchuthan Joan Sola, Jeremie Deray. A micro lie theory for state estimation in robotics. 2019. Available at <https://arxiv.org/pdf/1812.01537.pdf>.

M K Kaiser, N R Gans, and W E Dixon. Vision-based estimation for guidance, navigation, and control of an aerial vehicle. *IEEE Transactions on Aerospace and Electronic Systems*, 46(3):1064–1077, jul 2010.

Hassan K Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2002.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the Imaging Understanding Workshop*, pages 121–130, 1981.

Max Lutz and Thomas Meurer. Efficient formulation of collision avoidance constraints in optimization based trajectory planning and control. In *Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*, pages 228–233, San Diego, CA, August 2021.

Yi Ma, Stefano Soatto, Jan Kosecka, and Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2003.

R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of a quadrotor. *IEEE Robotics & Automation Magazine*, pages 20–32, 2012a.

Robert Mahony, Tarek Hamel, Pascal Morin, and Ezio Malis. Non-linear complementary filters on special linear group. *International Journal of Control*, 85(10):1557–1573, oct 2012b.

Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Technical Report RR-6303, INRIA, <https://hal.inria.fr/inria-00174036v3>, sep 2007.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525, Shanghai, China, May 2011.

Les Piegl and Wayne Tiller. *The NURBS Book*. Springer, second edition edition, 1995.

Romulo T. Rodrigues, Nikolaos Tsiovas, A. Pedro Aguiar, and Antonio Pascoal. B-spline surfaces for range-based environment mapping.

In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, pages 10774–10779, Las Vegas, Nevada, October 2020.

Soheil Sarabandi and Federico Thomas. Accurate computation of quaternions from rotation matrices. In J. Lenarcic and V. Parenti-Castelli, editors, *Advances in Robot Kinematics*, pages 39–46. Springer, 2019.

Jakob Schwichtenberg. *Physics from symmetry*. Springer International Publishing, 2015.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, pages 593–600. IEEE, 1994.

Joan Solà, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics, 2018. URL <http://arxiv.org/abs/1812.01537>.

Brian L Stevens and Frank L Lewis. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2003.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2006.

Nikolas Trawny and Stergios I. Roumeliotis. Indirect kalman filter for 3d attitude estimation a tutorial for quaternion algebra. Technical Report 2005-002, Rev. 57, University of Minnesota, March 2005.

Index

Accelerometers, [355](#)

Rate Gyros, [358](#)