A handbook in Visual SLAM

Trym Vegard Haavardsholm

Draft
Updated 12.01.2021

Contents

List of Examples

Chapter 1

Introduction

Consider a robot that navigates the world by the use of its camera sensors. By tracking how the environment appears to move through a sequence of images (Figure 1.1), it is possible to reconstruct both the motion of the robot relative the world, and a 3D map of the environment (Figure 1.2).

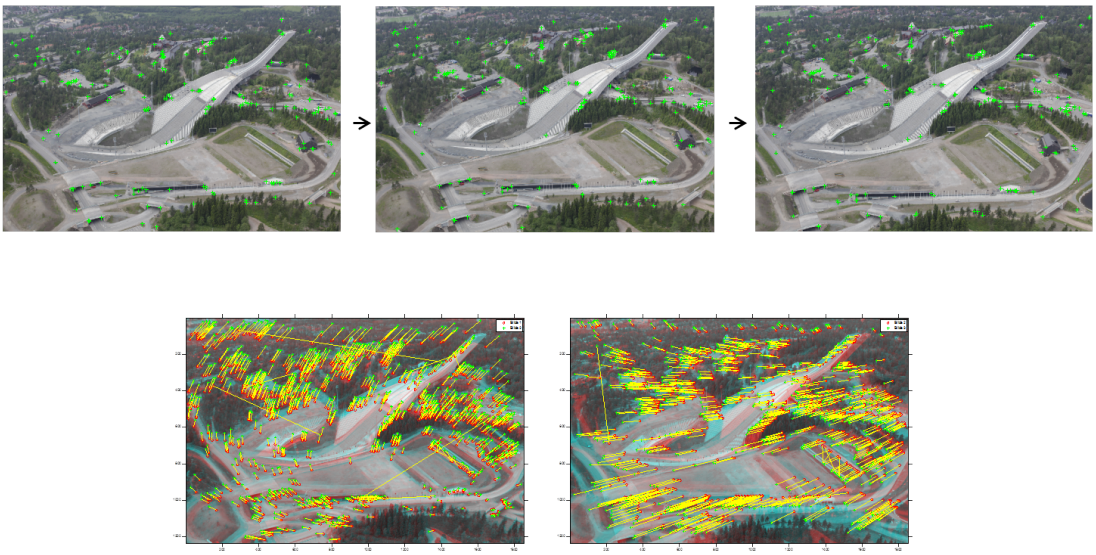


Figure 1.1: Tracking the scene in a sequence of images. In the top row, we see an example of how we can repeatedly detect corresponding points in the scene by extracting keypoint features. By computing local descriptors around these points, we can search for putative point matches between the images, such as shown in the bottom row. Outliers can be removed by only keeping matches that fit with the same type of motion hypothesis.

The map can be used to support tasks such as planning robot motion through rough terrain or avoiding obstacles. It also allows us to limit the error in the motion estimation by serving as a reference for navigation. By recognising revisited areas and correcting for drift through *loop-closure* constraints, the robot can "reset" its localisation error and recover the true topology of the environment, as illustrated

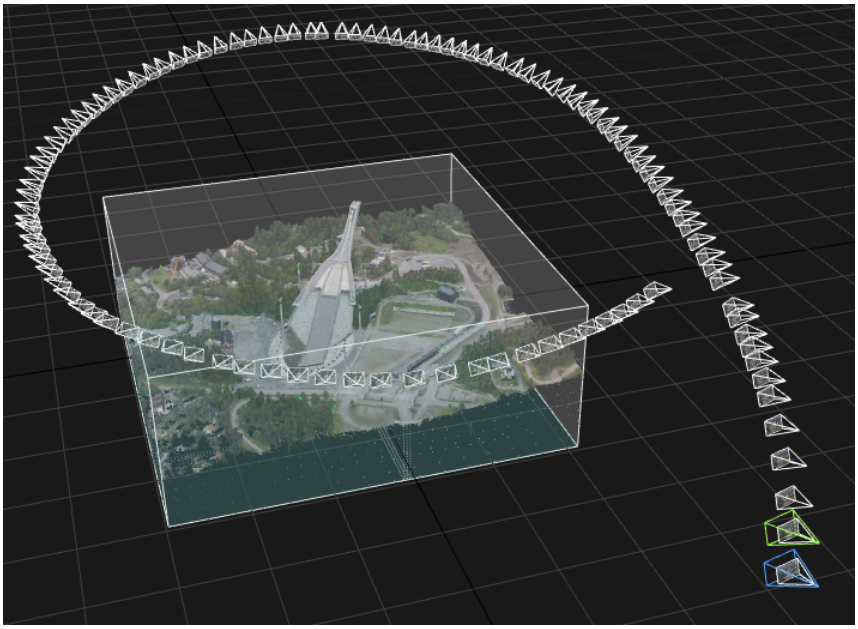


Figure 1.2: An example of estimating motion and a 3D map from a sequence of images.

in Figure 1.3. This approach to navigation is often called *Visual Simultaneous Localisation and Mapping (VSLAM)*. Approaches that reduce computations and complexity by estimating maps locally without loop-closures are often called *Visual Odometry (VO)* systems, and can be considered as reduced VSLAM systems, with the place recognition module disabled. Such approaches reduce drift locally, but are unable to correct for drift in the long-term, and will not recover the global topology of the scene.

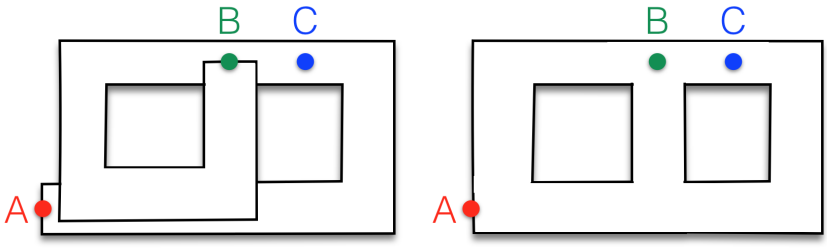


Figure 1.3: Left: Map built from odometry. Points that are close in reality (e.g. B and C) may be arbitrarily far away in the local map. Right: By leveraging loop-closures, VSLAM can recover the true topology of the environment, and discover "shortcuts" in the map. (Image source: [8])

It is customary to separate a SLAM system into two main components: the *front end* and the *back end* [8] (Figure 1.4). The front end extracts relevant data from raw sensor measurements, and performs data associations between the measurements and the map. The back end performs estimation and inference on data from the front end. Notice that the back end may also incorporate measurements from other types of sensors, such as *inertial measurements units (IMUs)*. We will here focus on the back end in VSLAM systems.

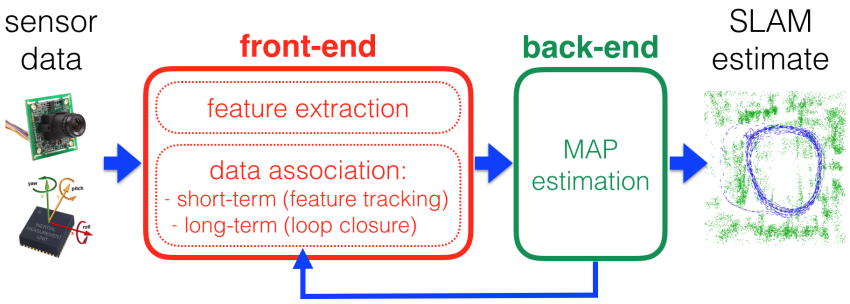


Figure 1.4: Front end and back end in a typical VSLAM system. (Image source: [8])

We may formulate the problem of estimating both motion and structure from noisy sensor measurements by using a Bayesian probabilistic model. Assume that we want to estimate the unknown set of state variables X that includes both the motion of the camera and the structure of the scene. Also assume that we are given a set of sensor measurements Z that depend upon the true state of X. The most often used estimator for X is the *maximum a posteriori* (MAP) estimate, which maximises the posterior density $p(X|Z)$ of the states X given the measurements Z. By applying Bayes' law, we get

$$\begin{aligned} X^{\text{MAP}} &= \arg\max_X p(X|Z) \\ &= \arg\max_X \frac{p(Z|X)p(X)}{p(Z)} \\ &= \arg\max_X l(X;Z)p(X), \end{aligned}$$

where $l(X;Z) \propto p(Z|X)$ is the likelihood of the states X given the measurements Z, defined as any function proportional to $p(Z|X)$. This can be seen as a form of sensor model, that allows us to evaluate how well the measurements fit with our current hypothesis for the states X. The density $p(X)$ is the prior density over the states, which allows us to evaluate how well our current hypothesis fits with any prior knowledge about X.

This handbook is meant to provide a terse, but thorough, introduction to the fundamentals behind how such problems may be expressed and solved. We will start by covering how we can describe states and sensor models using 3D geometry and Lie theory. This includes a framework for expressing derivatives of vectors, orientations and poses, and representing uncertainty in such variables. We will then see how nonlinear least squares can be applied to find the MAP estimate for states, given noisy measurements and corresponding measurement prediction models. Finally we will cover how this can be applied in estimating pose and structure from images.

Hopefully, this handbook will be useful background when studying Visual odometry and Visual SLAM systems.

Chapter 2

3D geometry

We will in this chapter describe how we can relate different objects in 3D by defining local coordinate frames and transformations between them. This will let us express the orientation and pose of objects relative to each other, which we can use to transform points between coordinate frames. We will also introduce notation that makes the direction of these transformations explicit and easy to deduce.

2.1 Points and coordinate frames

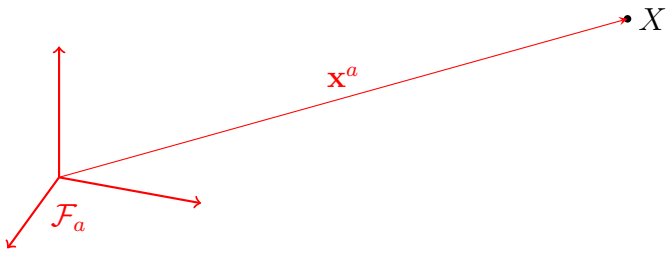


Figure 2.1: A point X represented by a vector $\mathbf{x}^a$ in the coordinate frame $\mathcal{F}_a$.

A point in space may be described by a *coordinate vector*, as shown in Figure 2.1. In this example, the point X is described by the vector $\mathbf{x}^a \in \mathbb{R}^3$, which represents the displacement of the point in Euclidean space with respect to the *coordinate frame* $\mathcal{F}_a$. A *Cartesian coordinate frame* defines a set of orthogonal axes which intersect at a point called the origin.

An alternative to Cartesian coordinates in Euclidean space is to describe points using *homogeneous coordinates* in projective space. A homogeneous coordinate vector in projective space is given by $\tilde{\mathbf{x}} = [\tilde{x}, \tilde{y}, \tilde{z}, \tilde{w}]^\top \in \mathbb{P}^3$, where $\tilde{\mathbf{x}} = \lambda\tilde{\mathbf{x}}$ for all non-zero scalars λ.

Given a Cartesian coordinate vector $\mathbf{x} = [x, y, z]^\top \in \mathbb{R}^3$, we can represent the same point as the homogeneous coordinate vector $\tilde{\mathbf{x}} = \lambda[x, y, z, 1]^\top$. This means that

we can construct a homogeneous vector from a Cartesian vector with the mapping

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \quad \mapsto \quad \tilde{\mathbf{x}} = \breve{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \in \mathbb{P}^3, \tag{2.1}$$

where we use the notation $\breve{\mathbf{x}}$ for the *normalised* homogeneous coordinate $\breve{\mathbf{x}} = \left[\begin{smallmatrix} \mathbf{x} \\ 1 \end{smallmatrix}\right]$, according to *Euclidean normalisation*. We can map a homogeneous vector back into a Cartesian vector with

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{bmatrix} \in \mathbb{P}^3 \quad \mapsto \quad \mathbf{x} = \begin{bmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ \tilde{z}/\tilde{w} \end{bmatrix} \in \mathbb{R}^3. \tag{2.2}$$

Homogeneous vectors with $\tilde{w} = 0$ have no representation in $\mathbb{R}^3$, and correspond to the direction to points infinitely far away from the origin. As we will see later, homogeneous coordinates are a convenient representation when describing transformations and camera projections of points and directions. See Hartley and Zisserman [18] for a detailed description of projective geometry for computer vision.

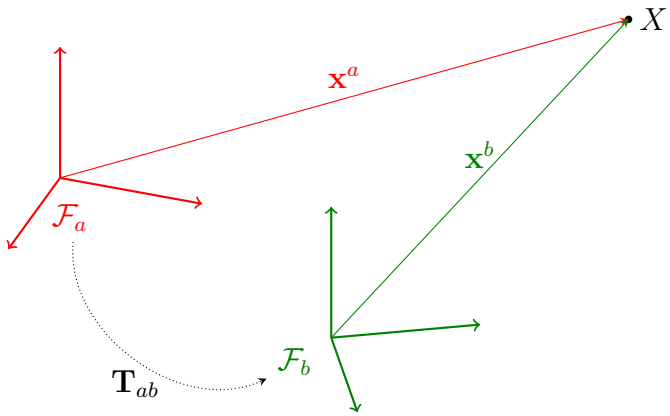


Figure 2.2: A point X represented by a vector $\mathbf{x}^a$ in the coordinate frame $\mathcal{F}_a$, and a corresponding vector $\mathbf{x}^b$ in the coordinate frame $\mathcal{F}_b$. The pose $\mathbf{T}_{ab}$ describes the position and orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$.

It is sometimes necessary to describe the same point relative to several coordinate frames. We might for example want to relate a point given in a world frame to the local coordinate frame of a camera. The relationship between the Cartesian coordinate frames in Figure 2.2 can be described by the *position* and *orientation* of $\mathcal{F}_b$ relative to $\mathcal{F}_a$. This is also called the *pose* of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, and can be expressed as a transformation $\mathbf{T}_{ab}$ that lets us map the point's coordinates $\mathbf{x}^b$ in $\mathcal{F}_b$ to coordinates $\mathbf{x}^a$ in $\mathcal{F}_a$.

We will in the following sections see how we can represent and apply such transformations between coordinate frames.

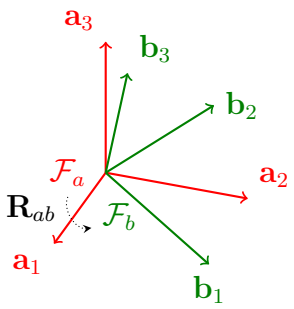


Figure 2.3: Two coordinate frames $\mathcal{F}_a$ and $\mathcal{F}_b$ with the same origin. The orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$ is described by the rotation matrix $\mathbf{R}_{ab}$.

2.2 Representing orientation

We can describe the orientation of coordinate frame $\mathcal{F}_b$ relative to $\mathcal{F}_a$ by how we should rotate $\mathcal{F}_a$ in order to align with $\mathcal{F}_b$. This rotation has 3 degrees of freedom (DOF), and can be represented by the orthogonal rotation matrix

$$\mathbf{R}_{ab} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in SO(3). \tag{2.3}$$

Rotation matrices belong to the $SO(3)$ Lie group (special orthogonal group of dimension 3), which we will cover in more detail in Section 4.7.

Let the coordinate frame $\mathcal{F}_a$ be defined by the orthonormal basis vectors $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$, and the frame $\mathcal{F}_b$ be defined by the orthonormal basis vectors $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$, as illustrated in Figure 2.3. The columns of the rotation matrix representing the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$ correspond to the basis vectors of $\mathcal{F}_b$ expressed in frame $\mathcal{F}_a$

$$\mathbf{R}_{ab} = \begin{bmatrix} \mathbf{b}_1^a & \mathbf{b}_2^a & \mathbf{b}_3^a \end{bmatrix}. \tag{2.4}$$

When the coordinate frames have the same origin, this lets us use the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$ to transform points $\mathbf{x}^b$ given in frame $\mathcal{F}_b$ to points $\mathbf{x}^a$ given in frame $\mathcal{F}_a$ by

$$\mathbf{x}^a = \mathbf{R}_{ab}\mathbf{x}^b. \tag{2.5}$$

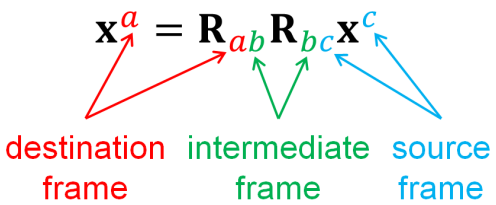


Figure 2.4: The coordinate frame notation helps us combine transformations in the correct order.

We can also compose relative orientations with matrix multiplication

$$\mathbf{R}_{ac} = \mathbf{R}_{ab}\mathbf{R}_{bc}. \tag{2.6}$$

The inverse orientation is given by the inverse rotation matrix, and since rotation matrices are orthogonal, we have

$$\mathbf{R}_{ba} = \mathbf{R}_{ab}^{-1} = \mathbf{R}_{ab}^{\top}. \tag{2.7}$$

Notice in Figure 2.4 how the subscript and superscript notation for coordinate frames aligns, and helps us combine transformations in the correct order.

Example 2.1 Transforming detections between sensors

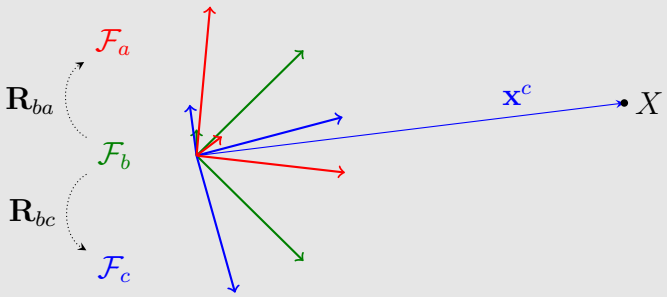


Figure 2.5: Three differently oriented sensors.

Assume that we have three coordinate frames $\mathcal{F}_a$, $\mathcal{F}_b$ and $\mathcal{F}_c$ at the same origin, as shown in Figure 2.5. These correspond to three co-located, but differently oriented sensors. The sensor system has been calibrated with regards to sensor b, so we are given $\mathbf{R}_{ba}$ and $\mathbf{R}_{bc}$. When the sensor c detects a point $\mathbf{x}^c$ represented in frame $\mathcal{F}_c$, what is the corresponding position in the coordinate frame of sensor a?

We can represent the relative orientation between $\mathcal{F}_c$ and $\mathcal{F}_a$ using the calibrated orientations by

$$\mathbf{R}_{ac} = \mathbf{R}_{ab}\mathbf{R}_{bc} = \mathbf{R}_{ba}^{\top}\mathbf{R}_{bc}. \tag{2.8}$$

This lets us compute the point represented in $\mathcal{F}_a$ as

$$\mathbf{x}^a = \mathbf{R}_{ac}\mathbf{x}^c = \mathbf{R}_{ba}^{\top}\mathbf{R}_{bc}\mathbf{x}^c. \tag{2.9}$$

Since $\mathbf{R}_{ba} = \mathbf{R}_{ab}^{\top}$, we see from (2.4) that the rows of $\mathbf{R}_{ba}$ are given by the basis vectors of $\mathcal{F}_a$ expressed in frame $\mathcal{F}_b$

$$\mathbf{R}_{ab} = \begin{bmatrix} \mathbf{a}_1^{b\top} \\ \mathbf{a}_2^{b\top} \\ \mathbf{a}_3^{b\top} \end{bmatrix}. \tag{2.10}$$

This is sometimes useful when assessing how coordinate systems relate to each other.

Example 2.2 Rotation matrix from basis vector perturbations

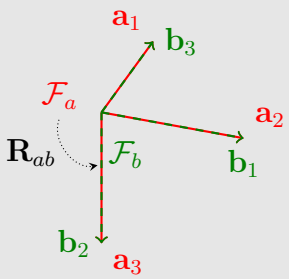


Figure 2.6: Two coordinate frames with their corresponding basis vectors.

As shown in Figure 2.6, we have a coordinate frame $\mathcal{F}_a$, where the x-axis points forwards, the y-axis points right and the z-axis points down. We also have a coordinate frame $\mathcal{F}_b$ where the z-axis points forwards and the x-axis points right. What is the orientation $\mathbf{R}_{ab}$ of $\mathcal{F}_b$ relative to $\mathcal{F}_a$?

From (2.4) we know that the columns of $\mathbf{R}_{ab}$ correspond to the basis vectors of $\mathcal{F}_b$ relative to $\mathcal{F}_a$. Since $\mathbf{b}_1^a = \mathbf{a}_2$, $\mathbf{b}_2^a = \mathbf{a}_3$ and $\mathbf{b}_3^a = \mathbf{a}_1$, we get

$$\mathbf{R}_{ab} = \begin{bmatrix} \mathbf{a}_2 & \mathbf{a}_3 & \mathbf{a}_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.11}$$

With (2.10), we can now also confirm that the rows of (2.11) correspond to the basis vectors of $\mathcal{F}_a$ relative to $\mathcal{F}_b$.

Rotations about the basis vectors are called *principal rotations*. The rotation matrices for a rotation θ about the x-, y- and z-axes are given by

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.12}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.13}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.14}$$

Any orientation can be represented by a sequence of three principal rotations, where no two successive rotations are about the same axis. This lets us specify an orientation as a set of three angles $(\theta_1, \theta_2, \theta_3)$ and a corresponding sequence of principal rotations, which is often referred to as the Euler angle representation.

Example 2.3 Orientation represented as roll, pitch and yaw

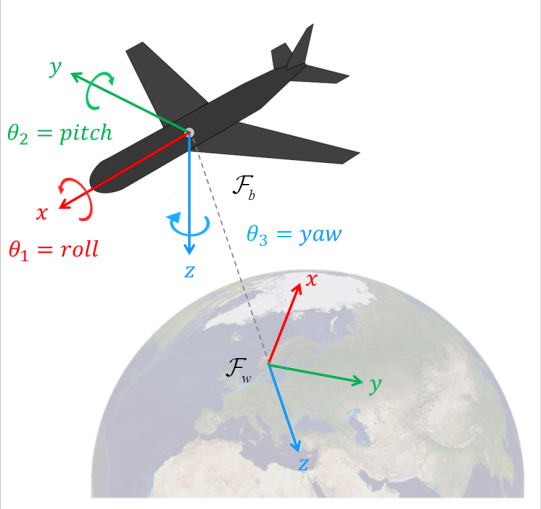


Figure 2.7: The orientation of the body frame $\mathcal{F}_b$ (front, right, down) relative to the world frame $\mathcal{F}_w$ (north, east, down) is represented with roll-, pitch- and yaw-angles about the body axes.

We have an aircraft with body coordinate frame $\mathcal{F}_b$, where the x-axis points forward, the y-axis points right and the z-axis points down. The world coordinate frame $\mathcal{F}_w$ is given as a NED frame, where the x-axis points north, the y-axis points east and the z-axis points down. The roll-, pitch- and yaw-angles around the principal axes of $\mathcal{F}_b$ are given as $(\theta_1, \theta_2, \theta_3)$, as shown in Figure 2.7.

We can compute the rotation matrix corresponding to the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_w$ with the roll-pitch-yaw sequence of principal rotations

$$\mathbf{R}_{wb} = \mathbf{R}_z(\theta_3)\mathbf{R}_y(\theta_2)\mathbf{R}_x(\theta_1). \tag{2.15}$$

It is also possible to describe any orientation as a rotation about a given axis. This is called the *angle-axis* representation, and can be expressed as

$$\boldsymbol{\theta} = \theta\mathbf{u}, \tag{2.16}$$

where θ is the angle of rotation and $\mathbf{u}$ is the unit-length axis of rotation. The corresponding rotation matrix is given by Rodrigues' rotation formula

$$\mathbf{R}(\boldsymbol{\theta}) = \mathbf{I} + \sin\theta[\mathbf{u}]_\times + (1 - \cos\theta)[\mathbf{u}]_\times^2, \tag{2.17}$$

where $[\mathbf{u}]_\times$ is the skew-symmetric matrix[1]

$$[\mathbf{u}]_\times = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}_\times \triangleq \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}. \tag{2.18}$$

When the angle θ is small, the corresponding rotation matrix can be approximated with the matrix

$$\mathbf{R}(\boldsymbol{\theta}) \approx \mathbf{I} + [\boldsymbol{\theta}]_\times. \tag{2.19}$$

This is sometimes called an *infinitesimal rotation*.

Another similar and very popular representation for rotation is the *unit quaternion*, which is covered in detail in e.g. [4].

The orthogonal rotation matrix has 9 elements that parameterise the 3 DOF rotation. The Euler angle and angle-axis representations are both minimal representations, in that they both can be represented by 3 parameters, but they suffer from singularities. In fact, there is no perfect representation that is minimal and also free of singularities. This is because 3D rotations lie on a 3D manifold embedded in a higher dimensional space. We will in Chapters 4, 5 and 6 see how we can describe perturbations, derivatives and probability distributions on this manifold.

2.3 Representing pose

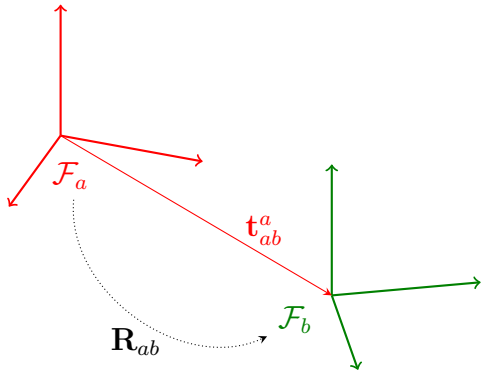


Figure 2.8: Two coordinate frames $\mathcal{F}_a$ and $\mathcal{F}_b$ with different position and orientation. The pose of $\mathcal{F}_b$ relative to $\mathcal{F}_a$ is described by the orientation $\mathbf{R}_{ab}$ and the translation $\mathbf{t}_{ab}^a$ between the frame origins.

When coordinate frames do not share a common origin, we also need to take the relative position between frames into account. The *pose* of $\mathcal{F}_b$ relative to $\mathcal{F}_a$ describes how $\mathcal{F}_a$ should be rotated *and* translated in order to coincide with $\mathcal{F}_b$. This can be represented with the homogeneous transformation matrix

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{t}_{ab}^a \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3), \tag{2.20}$$

where $\mathbf{R}_{ab} \in SO(3)$ represents the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, and $\mathbf{t}_{ab}^a \in \mathbb{R}^3$ is a translation vector given in the coordinate frame of $\mathcal{F}_a$. This vector represents the position of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, as shown in Figure 2.8. The 4×4 pose matrices are known as *3D Euclidean* or *3D rigid body* transformations, and belong to the $SE(3)$ Lie group (special Euclidean group of dimension 3), which we will cover in more detail in Section 4.8.

[1] The cross product between two vectors can also be represented as a matrix multiplication using this skew-symmetric matrix: $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$. This explains the common $[\cdot]_\times$ notation.

The fact that $\mathbf{T}$ is a homogeneous transformation matrix means that we can express the Euclidean transformation linearly using homogeneous vectors. Given a point $\mathbf{x}^b$ represented in $\mathcal{F}_b$ and the pose $\mathbf{T}_{ab}$ of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, we can convert to the homogeneous representation $\tilde{\mathbf{x}}^b$ using (2.1), and transform the point to coordinates given in frame $\mathcal{F}_a$ by

$$\tilde{\mathbf{x}}^a = \mathbf{T}_{ab}\tilde{\mathbf{x}}^b. \tag{2.21}$$

We get the Cartesian coordinate $\mathbf{x}^a$ by normalising $\tilde{\mathbf{x}}^a$ according to (2.2).

The transformation in (2.21) corresponds to rotating $\mathbf{x}^b$ into frame $\mathcal{F}_a$, and translating according to the position of $\mathcal{F}_b$ relative to $\mathcal{F}_a$. We can therefore express the transformation equivalently on Cartesian form as

$$\mathbf{x}^a = \mathbf{R}_{ab}\mathbf{x}^b + \mathbf{t}^a_{ab}. \tag{2.22}$$

We will call this transformation the *action* of the pose $\mathbf{T}$ on the vector $\mathbf{x}$, and denote it as $\mathbf{T} \cdot \mathbf{x}$, where

$$\mathbf{x}^a = \mathbf{T}_{ab} \cdot \mathbf{x}^b = \mathbf{R}_{ab}\mathbf{x}^b + \mathbf{t}^a_{ab}. \tag{2.23}$$

As for rotations, we can compose relative poses with matrix multiplication. This can also be expressed directly as operations on $\mathbf{R}$ and $\mathbf{t}$ by

$$\mathbf{T}_{ac} = \mathbf{T}_{ab}\mathbf{T}_{bc} = \begin{bmatrix} \mathbf{R}_{ab}\mathbf{R}_{bc} & \mathbf{R}_{ab}\mathbf{t}^b_{bc} + \mathbf{t}^a_{ab} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{2.24}$$

The inverse pose is given by

$$\mathbf{T}_{ba} = \mathbf{T}^{-1}_{ab} = \begin{bmatrix} \mathbf{R}^\top_{ab} & -\mathbf{R}^\top_{ab}\mathbf{t}^a_{ab} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{2.25}$$

Since all pose operations can be expressed as operations on the orientation and position directly, it is common to implement poses in practice as a pair $(\mathbf{R}, \mathbf{t})$, rather than the full matrix $\mathbf{T}$. This avoids the use of homogeneous coordinates, and allows alternative representations for orientation, such as quaternions. This dual representation lets us exploit the mathematical properties of $\mathbf{T}$, while simultaneously reaping the benefits of the chosen implementation for orientation and pose operations.

Example 2.4 Seeing a point in the world with a camera

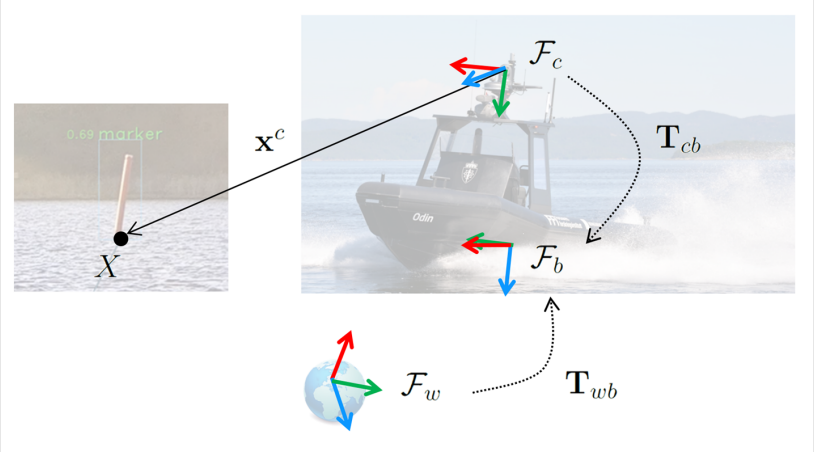


Figure 2.9: A camera observes a sea marker in the camera frame $\mathcal{F}_c$. A navigation system estimates the pose of the boat relative to the world frame $\mathbf{T}_{wb}$, and the pose of the boat relative to the camera $\mathbf{T}_{cb}$ is known.

An autonomous boat observes a sea marker in its camera frame $\mathcal{F}_c$. As illustrated in Figure 2.9, we are given the pose of the boat relative to the world frame $\mathbf{T}_{wb}$, as well as the pose of the boat relative to the camera frame $\mathbf{T}_{cb}$. Where is the sea marker relative to the world frame $\mathcal{F}_w$?

We can compute the pose of the camera relative to the world by

$$\begin{aligned}
\mathbf{T}_{wc} &= \mathbf{T}_{wb}\mathbf{T}_{bc} && \text{(2.26a)}\\
&= \mathbf{T}_{wb}\mathbf{T}_{cb}^{-1} && \text{(2.26b)}\\
&= \begin{bmatrix} \mathbf{R}_{wb}\mathbf{R}_{cb}^{\top} & -\mathbf{R}_{wb}\mathbf{R}_{cb}^{\top}\mathbf{t}_{cb}^{c} + \mathbf{t}_{wb}^{w} \\ \mathbf{0}^{\top} & 1 \end{bmatrix}. && \text{(2.26c)}
\end{aligned}$$

The coordinate of the sea marker in the world frame is then

$$\tilde{\mathbf{x}}^w = \mathbf{T}_{wc}\tilde{\mathbf{x}}^c = \mathbf{T}_{wb}\mathbf{T}_{cb}^{-1}\tilde{\mathbf{x}}^c, \tag{2.27}$$

or equivalently

$$\begin{aligned}
\mathbf{x}^w &= \mathbf{T}_{wc} \cdot \mathbf{x}^c && \text{(2.28a)}\\
&= \mathbf{R}_{wc}\mathbf{x}^c + \mathbf{t}_{wc}^{w} && \text{(2.28b)}\\
&= \mathbf{R}_{wb}\mathbf{R}_{cb}^{\top}\mathbf{x}^c - \mathbf{R}_{wb}\mathbf{R}_{cb}^{\top}\mathbf{t}_{cb}^{c} + \mathbf{t}_{wb}^{w}. && \text{(2.28c)}
\end{aligned}$$

Chapter 3

Camera geometry

This chapter will introduce two important topics in camera geometry: The *geometric camera model*, that describes the camera projection process, and the *epipolar geometry*, that describes the geometry when seeing the same points from two different views. Most of this chapter is based on the wonderful books by Hartley and Zisserman, and Ma et al. [18, 22].

3.1 Geometric camera models

In the general case, a geometric camera model is a *projection* $\pi : \mathbb{R}^3 \to \Omega$, which projects 3D points $\mathbf{x}^c \in \mathbb{R}^3$ in the camera frame $\mathcal{F}_c$ onto pixels $\mathbf{u} \in \Omega$ in the image, so that

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \pi(\mathbf{x}^c). \tag{3.1}$$

Here, $\Omega \subset \mathbb{R}^2$, is the *image domain* of all valid pixels. The inverse of this function is the *backprojection* $\pi^{-1} : \Omega \times \mathbb{R}^+ \to \mathbb{R}^3$, which maps pixels and a corresponding depth z back to 3D points:

$$\mathbf{x}^c = \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = \pi^{-1}(\mathbf{u}, z). \tag{3.2}$$

We will in the following let $\mathbf{u} \in \mathbb{R}^2$, and assume that there is an implicit second step that ensures that $\mathbf{u}$ is a valid pixel $\in \Omega$.

When describing camera projections, it is often convenient to exploit the homogeneous point representation. Equivalent to the homogeneous representation of 3D coordinates in Section 2.1, we can construct a homogeneous 2D vector from a 2D Cartesian vector with the mapping

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} \in \mathbb{R}^2 \quad \mapsto \quad \tilde{\mathbf{u}} = \breve{\mathbf{u}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \in \mathbb{P}^2, \tag{3.3}$$

and we can map a homogeneous vector back into a Cartesian vector with

$$\tilde{\mathbf{u}} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \in \mathbb{P}^2 \quad \mapsto \quad \mathbf{u} = \begin{bmatrix} \tilde{u}/\tilde{w} \\ \tilde{v}/\tilde{w} \end{bmatrix} \in \mathbb{R}^2. \tag{3.4}$$

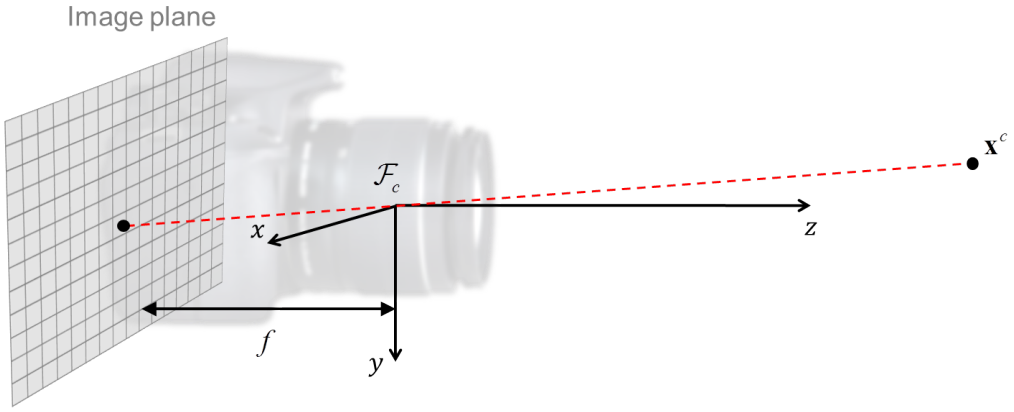


Figure 3.1: The imaging geometry in the perspective camera model. The camera is represented by the camera frame $\mathcal{F}_c$. Points $\mathbf{x}^c$ in the camera frame are projected through the origin and onto the image plane at a distance f behind the projective centre, where f is the focal length.

The most commonly used geometric camera model is the *perspective camera model*[1]. As shown in Figure 3.1, the camera is represented by a 3D coordinate frame $\mathcal{F}_c$, with its origin at the camera's projective centre. The x-axis is pointing right, the y-axis is pointing down, and the z-axis is pointing forwards along the *optical axis* of the camera. The imaging process is modelled as a central projection onto the image plane at $z = -f$, where f is the *focal length*.

Since the projection in Figure 3.1 results in images that are flipped upside down, it is more convenient to put the imaging plane in front of the projection centre using the *frontal projection model*. It is also convenient to define a *normalised image plane* at $z = 1$, which represents an *idealised camera* with focal length 1, as illustrated in Figure 3.2. This lets us describe the imaging process at a fixed imaging plane, independently of the camera-specific focal length parameter f.

We can project $\mathbf{x}^c$ onto the normalised image plane with the homogeneous *perspective projection* matrix $\mathbf{\Pi}_0$, by

$$\tilde{\mathbf{x}}_n = \mathbf{\Pi}_0 \tilde{\mathbf{x}}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{x}}^c = \mathbf{x}^c. \tag{3.5}$$

On normalised form, we have that

$$\breve{\mathbf{x}}_n = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} x^c/z^c \\ y^c/z^c \\ 1 \end{bmatrix} = \frac{1}{z^c}\mathbf{x}^c. \tag{3.6}$$

[1]Sometimes also called the *pinhole model*.

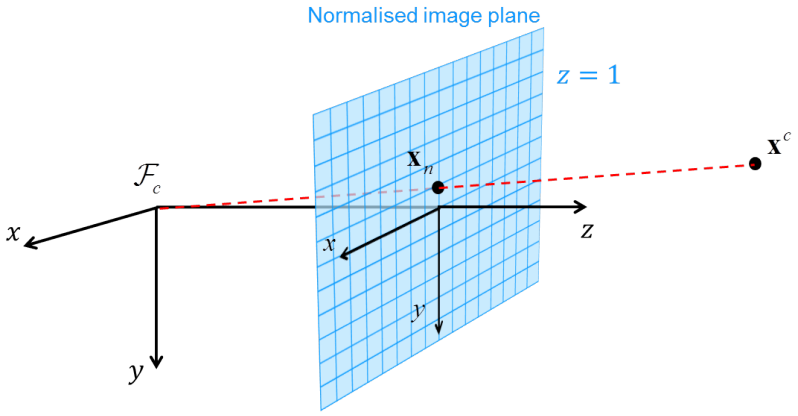


Figure 3.2: The normalised image plane is set at $z = 1$ in front of the projective centre. This represents the perspective camera model for an idealised camera with $f = 1$. The point where the z-axis of $\mathcal{F}_c$ intersects the normalised image plane is called the principal point. The central projection of the 3D point $\mathbf{x}^c$ onto the normalised image plane is the point $\mathbf{x}_n$, which is called the normalised image coordinate.

The projected point $\tilde{\mathbf{x}}_n$ in the normalised image plane can be seen both as a Cartesian representation of the 3D point on the plane when normalised as in (3.6), *as well* as a homogeneous representation of the 2D point in the frame of the normalised image plane. As illustrated in Figure 3.2, this 2D frame is aligned with the x and y axes of the camera frame $\mathcal{F}_c$, and its origin is at the *principal point*, where the z-axis of $\mathcal{F}_c$ intersects the plane. Since $\mathbf{x}_n$ represents the image point for the idealised camera model, it is called the *normalised image coordinate*, and we give it the special subscript n notation.

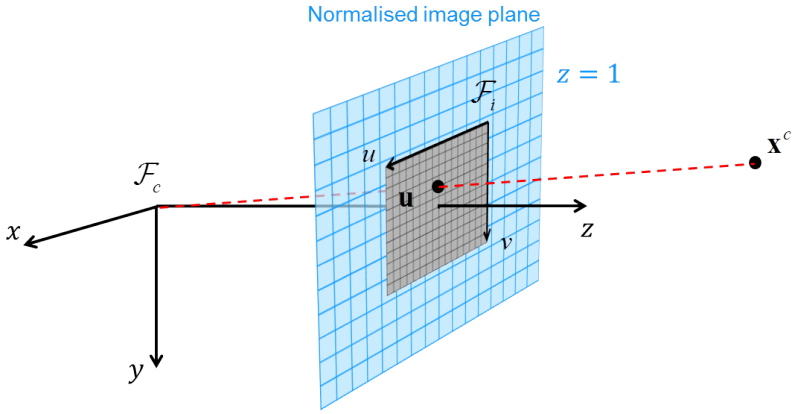


Figure 3.3: The image frame $\mathcal{F}_i$, given in pixels, spans the normalised image plane. Its origin is at the upper-left corner of the image, its u-axis points right along the rows, and its v-axis points down along the columns. The mapping between normalised image coordinates and pixels is given by an affine transformation.

As shown in Figure 3.3, the image is represented by a 2D coordinate frame $\mathcal{F}_i$ that spans the normalised image plane, with the origin at the upper-left corner of the image. We recover the pixel coordinate $\mathbf{u}$ by mapping the normalised image

coordinate $\tilde{\mathbf{x}}_n$ to the image frame $\mathcal{F}_i$ with the affine transformation

$$\tilde{\mathbf{u}} = \mathbf{K}\tilde{\mathbf{x}}_n. \tag{3.7}$$

The upper triangular matrix $\mathbf{K}$ takes us from the idealised case to the camera-specific model. It collects all parameters that are *intrinsic* to a particular camera, and it is therefore known as the *intrinsic matrix* or the *calibration matrix*. The affine transformation matrix is a combined scaling, shear and translation, and is given by

$$\mathbf{K} = \begin{bmatrix} f_u & s_\theta & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.8}$$

where the elements of $\mathbf{K}$ have the following geometric interpretation:

- f_u is the size of unit length in horizontal pixels.
  It can be expressed as $f_u = fs_u$, where f is the focal length in metric units, and s_u is the scaling factor that describes the horizontal pixel density in pixels per metric unit.
- f_v is the size of unit length in vertical pixels.
  It can be expressed as $f_v = fs_v$, where f is the focal length in metric units, and s_v is the scaling factor that describes the vertical pixel density in pixels per metric unit.
- c_u is the u-coordinate of the principal point in $\mathcal{F}_i$.
- c_v is the v-coordinate of the principal point in $\mathcal{F}_i$.
- s_θ is the *skew factor*, proportional to $\cot(\theta)$, where θ is the angle between the u and v axes in $\mathcal{F}_i$.

Because of the structure of $\mathbf{K}$, we preserve the normalised form of the homogeneous coordinates through the transformation:

$$\breve{\mathbf{u}} = \mathbf{K}\breve{\mathbf{x}}_n. \tag{3.9}$$

We can estimate a calibration matrix $\mathbf{K}$ by performing a camera calibration procedure such as [28]. It is common practice to reduce the number of parameters by dropping the skew factor. We will therefore from here on assume that $s_\theta = 0$, which will simplify the following expressions somewhat.

When $\mathbf{K}$ is known, we can obtain the *calibrated* normalised image coordinates from the pixel $\mathbf{u}$ by applying the inverse transformation

$$\tilde{\mathbf{x}}_n = \mathbf{K}^{-1}\tilde{\mathbf{u}}. \tag{3.10}$$

The inverse transformation also preserves the normalised form:

$$\breve{\mathbf{x}}_n = \mathbf{K}^{-1}\breve{\mathbf{u}}. \tag{3.11}$$

If we crop or resample images from a calibrated camera, these transformations can be encoded directly in the calibration matrix. Since cropping corresponds to a possible translation of the principal point, and resampling corresponds to scaling, we can update the calibration matrix by applying the corresponding transformations to $\mathbf{K}$.

Example 3.5 Camera calibration matrix for downsampled images

We calibrate a camera using 1600×1200 pixel images, and we get the following camera calibration matrix:

$$\mathbf{K} = \begin{bmatrix} 100 & 0 & 800 \\ 0 & 100 & 600 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.12}$$

If we downscale our images with a scale factor $s = 0.5$ to 800×600 pixels, how should we modify $\mathbf{K}$ in order for it to be a valid calibration matrix for the downsampled images?

Since the camera calibration matrix is an affine transformation that transforms points on the normalised image plane to pixels in the image, we can apply the scaling to $\mathbf{K}$ with the homogeneous transformation matrix

$$\mathbf{S}_s = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

This results in the camera calibration matrix

$$\mathbf{K}_{0.5} = \mathbf{S}_{0.5}\mathbf{K} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 100 & 0 & 800 \\ 0 & 100 & 600 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 50 & 0 & 400 \\ 0 & 50 & 300 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.14}$$

Putting it all together, the perspective camera model on homogeneous form gives us the projection

$$\tilde{\mathbf{u}} = \mathbf{K}\mathbf{\Pi}_0\tilde{\mathbf{x}}^c = \mathbf{K}\mathbf{x}^c = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}^c, \tag{3.15}$$

which first projects $\mathbf{x}^c$ onto the normalised image plane, and then applies a 2D affine transformation that maps points on the normalised image plane to pixels in the image frame $\mathcal{F}_i$. The equivalent projection function on Euclidean form is given by

$$\mathbf{u} = \pi_p(\mathbf{x}^c; \mathbf{K}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{K} \frac{1}{z^c} \mathbf{x}^c = \begin{bmatrix} f_u \dfrac{x^c}{z^c} + c_u \\ f_v \dfrac{y^c}{z^c} + c_v \end{bmatrix}. \tag{3.16}$$

For normalised image coordinates, we have

$$\mathbf{x}_n = \pi_n(\mathbf{x}^c) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \frac{1}{z^c}\mathbf{x}^c = \begin{bmatrix} x^c/z^c \\ y^c/z^c \end{bmatrix}. \tag{3.17}$$

From (3.6) and (3.11) we get the corresponding backprojection function

$$\mathbf{x}^c = \pi_p^{-1}(\mathbf{u}, z; \mathbf{K}) = z\mathbf{K}^{-1}\breve{\mathbf{u}} = z \begin{bmatrix} \frac{u - c_u}{f_u} \\ \frac{v - c_v}{f_v} \\ 1 \end{bmatrix}. \tag{3.18}$$

For normalised image coordinates, we have

$$\mathbf{x}^c = \pi_n^{-1}(\mathbf{x}_n, z) = z\breve{\mathbf{x}}_n = z \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}. \tag{3.19}$$

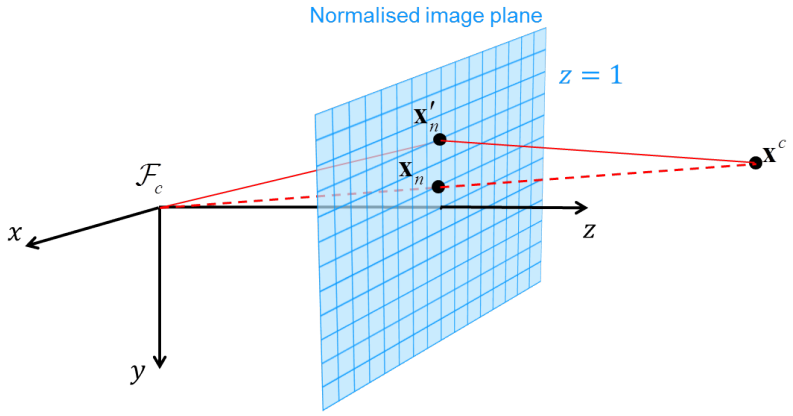


Figure 3.4: Typical cameras deviate from the perspective camera model geometry. A distortion model describes the relationship between undistorted points $\mathbf{x}_n$ and distorted points $\mathbf{x}'_n$ in the normalised image plane. By correcting for these deviations in the normalised image plane, the perspective camera model can still be applied in these situations.

Practical cameras seldom fit perfectly with the perspective camera model, because they typically suffer from some kind of geometric distortion through the optical system. This is illustrated in Figure 3.4. A *distortion model* describes how a camera deviates from the perspective camera geometry, and is typically used to express the relationship between undistorted points $\mathbf{x}_n$ and distorted points $\mathbf{x}'_n$ in the normalised image plane. A typical example is the *radial distortion model*

$$r'^2 = x_n'^2 + y_n'^2 \tag{3.20}$$
$$x_n = x'_n(1 + k_1 r'^2 + k_2 r'^4) \tag{3.21}$$
$$y_n = y'_n(1 + k_1 r'^2 + k_2 r'^4), \tag{3.22}$$

where the parameters k_1 and k_2 can be estimated as part of the camera calibration process.

We will finish this discussion on geometric camera models by mentioning that there are several other models that may be more suitable than the perspective camera model in certain situations. Examples include omnidirectional models [9, 5] and models for underwater cameras [21].

3.2 Epipolar geometry

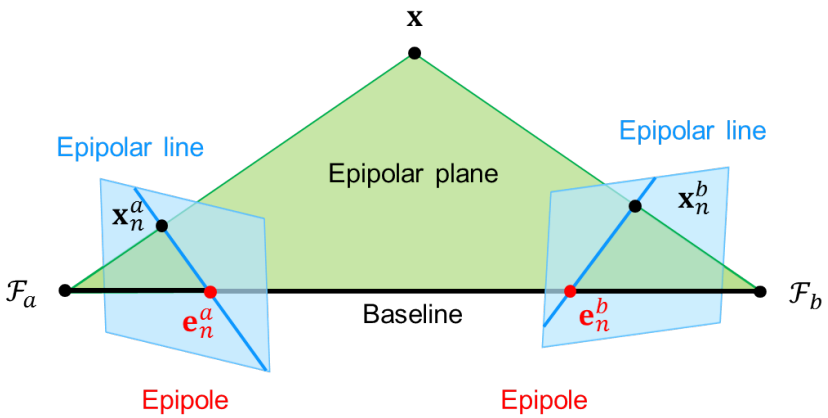


Figure 3.5: An illustration of the epipolar geometry between two perspective cameras, represented by the camera frames $\mathcal{F}_a$ and $\mathcal{F}_b$. The epipolar plane contains the point $\mathbf{x}$ and the two camera centres. The baseline is the line connecting the camera centres. The epipoles $\mathbf{e}_n^a$ and $\mathbf{e}_n^b$ are where the baseline intersects the image planes. The epipolar lines are where the epipolar plane intersects the image planes. Projected image points $\mathbf{x}_n^a$ and $\mathbf{x}_n^b$ corresponding to the same 3D point $\mathbf{x}$ must lie on the corresponding epipolar lines. This is called the epipolar constraint.

Consider two perspective cameras, represented by the camera frames $\mathcal{F}_a$ and $\mathcal{F}_b$, as shown in Figure 3.5. The camera frames are related by the relative pose

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{t}_{ab}^a \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.23}$$

or, equivalently, the inverse pose

$$\mathbf{T}_{ba} = \mathbf{T}_{ab}^{-1} = \begin{bmatrix} \mathbf{R}_{ba} & \mathbf{t}_{ba}^b \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.24}$$

Observing the same point $\mathbf{x}$ with these two cameras puts a strong geometric constraint on the point correspondence in the two images. This constraint is called the *epipolar constraint*.

Figure 3.5 shows the *epipolar plane*, which is the plane containing the point $\mathbf{x}$ and the two camera centres of $\mathcal{F}_a$ and $\mathcal{F}_b$. The line joining the camera centres is the *baseline*, and the *epipoles* are where the baseline intersects the image planes. This

means that the epipoles are the images of the other camera centre. In normalised image coordinates, they are given by

$$\tilde{\mathbf{e}}_n^a = \mathbf{t}_{ab}^a \tag{3.25}$$

$$\tilde{\mathbf{e}}_n^b = \mathbf{t}_{ba}^b. \tag{3.26}$$

Given the camera calibration matrices $\mathbf{K}_a$ and $\mathbf{K}_b$, the corresponding pixel points in the images are given by

$$\tilde{\mathbf{e}}^a = \mathbf{K}_a \mathbf{t}_{ab}^a \tag{3.27}$$

$$\tilde{\mathbf{e}}^b = \mathbf{K}_b \mathbf{t}_{ba}^b. \tag{3.28}$$

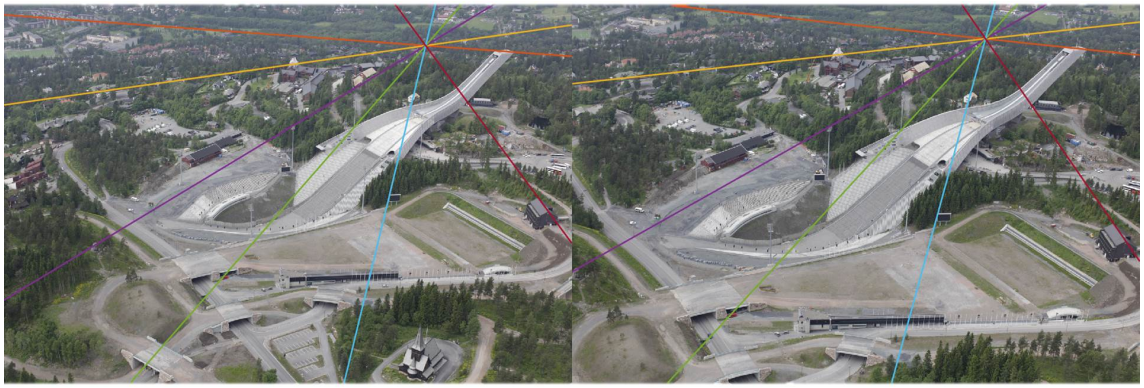


Figure 3.6: An example of epipolar lines between two images that was captured sequentially while moving towards the Holmenkollen ski jump. In the left image, we see the epipolar lines intersect at the epipole, corresponding to the image of the camera centre in the second position (when the right image was captured). In the right image, we see the epipole corresponding to the the first camera position, projected in from behind. Corresponding epipolar lines are given the same colour. Notice that objects in the scene lie on corresponding epipolar lines, in accordance with the epipolar constraint.

The lines where the epipolar plane intersects the image planes is called the *epipolar lines*. Informally, the epipolar constraint says that corresponding points in the two images *must* lie on corresponding epipolar lines, since the cameras and the associated 3D point form a common epipolar plane. Figure 3.6 shows a few examples of corresponding epipolar lines in two different images.

The epipolar constraint can be represented mathematically by the *essential matrix* $\mathbf{E} \in \mathbb{R}^{3\times3}$ in the normalised image plane, and by the *fundamental matrix* $\mathbf{F} \in \mathbb{R}^{3\times3}$ in the image. A correspondence $\mathbf{x}_n^a \leftrightarrow \mathbf{x}_n^b$ in the normalised image planes of the two cameras $\mathcal{F}_a$ and $\mathcal{F}_b$ must satisfy the equation

$$\tilde{\mathbf{x}}_n^{a\top} \mathbf{E}_{ab} \tilde{\mathbf{x}}_n^b = 0. \tag{3.29}$$

The essential matrix $\mathbf{E}_{ab}$ is given by

$$\mathbf{E}_{ab} = [\mathbf{t}_{ab}^a]_\times \mathbf{R}_{ab}, \tag{3.30}$$

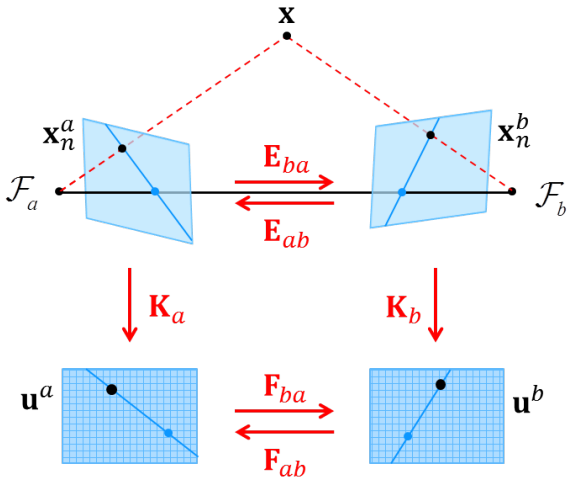


Figure 3.7: The essential matrices $\mathbf{E}_{ab}$ and $\mathbf{E}_{ba}$ represent the epipolar constraint in the normalised image planes. The fundamental matrices $\mathbf{F}_{ab}$ and $\mathbf{F}_{ba}$ represent the same epipolar constraint in the images. They can be computed from the essential matrices via the camera calibration matrices $\mathbf{K}_a$ and $\mathbf{K}_b$.

which is related to the pose $\mathbf{T}_{ab}$ in (3.23). The equation

$$\tilde{\mathbf{x}}_n^{b\top} \mathbf{E}_{ba} \tilde{\mathbf{x}}_n^a = 0, \tag{3.31}$$

with the essential matrix

$$\mathbf{E}_{ba} = \mathbf{E}_{ab}^\top = [\mathbf{t}_{ba}^b]_\times \mathbf{R}_{ba}, \tag{3.32}$$

is an equivalent representation of the constraint in (3.29).

As illustrated in Figure 3.7, the epipolar constraint extends naturally to pixel correspondences $\tilde{\mathbf{u}}^a \leftrightarrow \tilde{\mathbf{u}}^b$ via the camera calibration matrices $\mathbf{K}_a$ and $\mathbf{K}_b$. The pixel correspondences must satisfy the equation

$$\tilde{\mathbf{u}}^{a\top} \mathbf{F}_{ab} \tilde{\mathbf{u}}^b = 0, \tag{3.33}$$

where the fundamental matrix $\mathbf{F}_{ab}$ is given by

$$\mathbf{F}_{ab} = \mathbf{K}_a^{-\top} \mathbf{E}_{ab} \mathbf{K}_b^{-1}. \tag{3.34}$$

The equivalent fundamental matrix $\mathbf{F}_{ba}$ is similarly given by

$$\mathbf{F}_{ba} = \mathbf{F}_{ab}^\top = \mathbf{K}_b^{-\top} \mathbf{E}_{ba} \mathbf{K}_a^{-1}. \tag{3.35}$$

Since

$$\tilde{\mathbf{x}}^\top \tilde{\mathbf{l}} = 0, \qquad \tilde{\mathbf{l}} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \in \mathbb{P}^2 \tag{3.36}$$

is the homogeneous representation of the line

$$ax + by + c = 0, \tag{3.37}$$

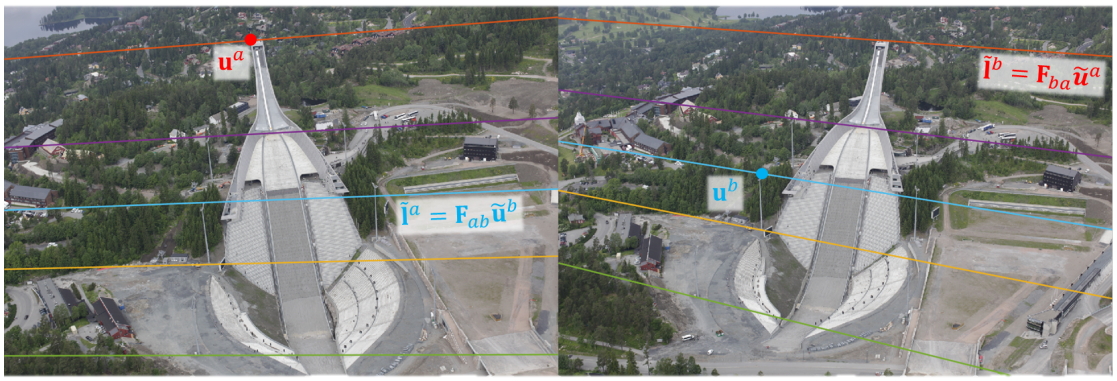


Figure 3.8: An example of computing epipolar lines in one image, given a point in the other image and the fundamental matrix. Corresponding epipolar lines are given the same colour. Notice that objects in the scene lie on corresponding epipolar lines in the images.

we see from (3.29) that

$$\tilde{\mathbf{l}}_n^a = \mathbf{E}_{ab}\tilde{\mathbf{x}}_n^b \tag{3.38}$$

is the epipolar line in the normalised image plane of $\mathcal{F}_a$ corresponding to the point $\tilde{\mathbf{x}}_n^b$ in $\mathcal{F}_b$. Equivalently, we see from (3.33) that

$$\tilde{\mathbf{l}}^a = \mathbf{F}_{ab}\tilde{\mathbf{u}}^b \tag{3.39}$$

is the corresponding epipolar line in the image of $\mathcal{F}_a$. As illustrated in Figure 3.8, we can find the corresponding epipolar lines in $\mathcal{F}_b$ by using $\mathbf{E}_{ba}$ and $\mathbf{F}_{ba}$.

We can restrict the search for correspondences by traversing along the epipolar line. Given a set of putative point correspondences, we can test their validity by computing the distances to their corresponding epipolar lines. The distance $d(\tilde{\mathbf{u}}, \tilde{\mathbf{l}})$ between a homogeneous point $\tilde{\mathbf{u}}$ and a homogeneous line $\tilde{\mathbf{l}}$ is given by

$$d(\tilde{\mathbf{u}}, \tilde{\mathbf{l}}) = \frac{\tilde{\mathbf{u}}^\top \tilde{\mathbf{l}}}{\sqrt{\tilde{l}_1 + \tilde{l}_2}}. \tag{3.40}$$

Although it is a necessary requirement that $d(\tilde{\mathbf{u}}^a, \mathbf{F}_{ab}\tilde{\mathbf{u}}^b)$ is small for $\tilde{\mathbf{u}}^a$ and $\tilde{\mathbf{u}}^b$ to be a valid point correspondence, it important to emphasise that this is not sufficient to guarantee its validity. It only suggests that the two points lie in the same epipolar plane.

We can estimate the fundamental matrix $\mathbf{F}$ from point correspondences using the 7- or 8-point algorithms [18]. The essential matrix $\mathbf{E}$ can be estimated from point correspondences with the 5-point algorithm [25]. We can also compute the essential matrix from the fundamental matrix by

$$\mathbf{E}_{ab} = \mathbf{K}_a^\top \mathbf{F}_{ab}\mathbf{K}_b. \tag{3.41}$$

We will see in Chapter 8 that the structure of the essential matrix (3.30) can be exploited to estimate the relative pose between images up to scale.

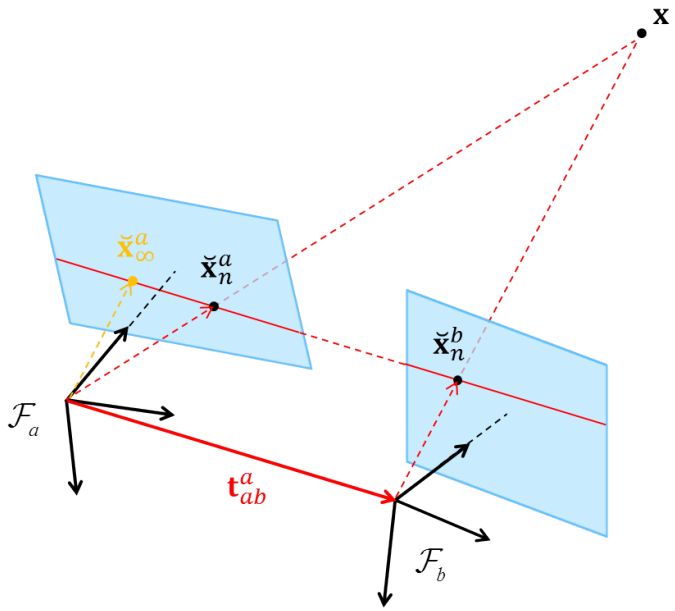


Figure 3.9: When we know the pose of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, we can map the image point $\breve{\mathbf{x}}^b_n$ into $\mathcal{F}_a$ by $\tilde{\mathbf{x}}^a_\infty = \mathbf{R}_{ab}\breve{\mathbf{x}}^b$. This corresponds to the image of $\mathbf{x}$ when $\mathbf{x}$ is infinitely far away. The point $\breve{\mathbf{x}}^a_n$ will in this example lie on the epipolar line to the right of $\tilde{\mathbf{x}}^a_\infty$ (in the direction of $\mathbf{t}^a_{ab}$) for all other depths, restricting the search for valid correspondences further.

When the relative pose between the cameras are known, we can also express the epipolar line as a function of depth from the other camera. This can for $\mathcal{F}_a$ be expressed as as

$$\tilde{\mathbf{x}}^a_n = \mathbf{R}_{ab}\breve{\mathbf{x}}^b_n + \frac{1}{z^b}\mathbf{t}^a_{ab}, \tag{3.42}$$

where $\breve{\mathbf{x}}^b_n$ is the normalised point correspondence in the normalised image plane of $\mathcal{F}_b$ and z^b is the depth from $\mathcal{F}_b$. Since we can think of $\breve{\mathbf{x}}^b_n$ as a 3D vector on the normalised image plane of $\mathcal{F}_b$, the first term in (3.42) rotates $\breve{\mathbf{x}}^b_n$ into the frame $\mathcal{F}_a$ of the other camera (Figure 3.9). These lines of sight are parallel, and correspond to the projections of a 3D point without observable depth. This is because the motion between the cameras were without translation, or because the 3D point is "infinitely" far away. We therefore write this projection as

$$\tilde{\mathbf{x}}^a_\infty = \mathbf{R}_{ab}\breve{\mathbf{x}}^b_n. \tag{3.43}$$

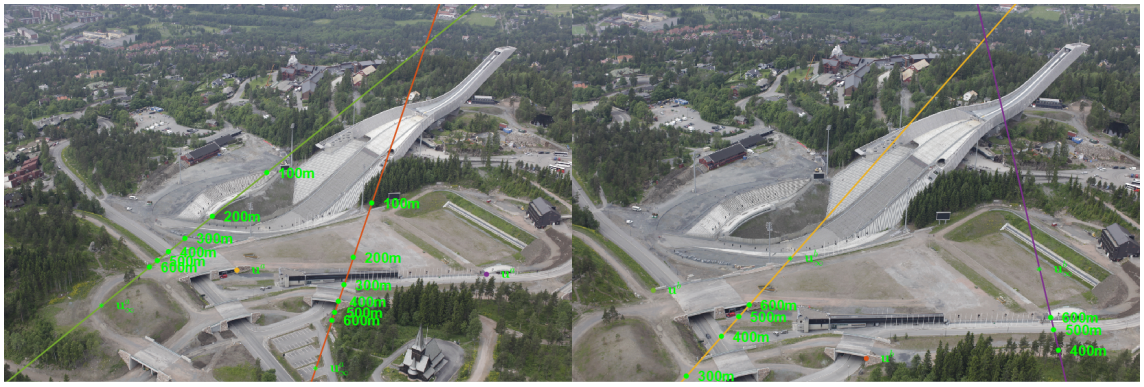
The second term in (3.42) moves $\tilde{\mathbf{x}}^a_n$ along the epipolar line defined by $\tilde{\mathbf{x}}^a_\infty$ and the epipole $\mathbf{t}^a_{ab}$. By increasing the *inverse depth* $1/z^b$, we move away from $\tilde{\mathbf{x}}^a_\infty$, faster and faster as the depth decreases.

The corresponding representation of the epipolar line in the image is given by

$$\tilde{\mathbf{u}}^a = \tilde{\mathbf{u}}^a_\infty + \frac{1}{z^b}\mathbf{K}_a\mathbf{t}^a_{ab}, \tag{3.44}$$

with

$$\tilde{\mathbf{u}}^a_\infty = \mathbf{K}_a\mathbf{R}_{ab}\mathbf{K}_b^{-1}\breve{\mathbf{u}}^b \tag{3.45}$$



img_a img_b

Figure 3.10: Epipolar lines with depth. This example shows two chosen points in each image, with the corresponding epipolar lines in the other image given the same colour. The corresponding depth for the chosen points along the epipolar lines according to (3.44) are shown, together with the limit at $\mathbf{u}_\infty$.

The distance between the pixels

$$d = \|\mathbf{u}^a - \mathbf{u}^a_\infty\| \tag{3.46}$$

is called the *disparity*, and is a measure of the observed parallax between the cameras.

Example 3.6 Stereo geometry

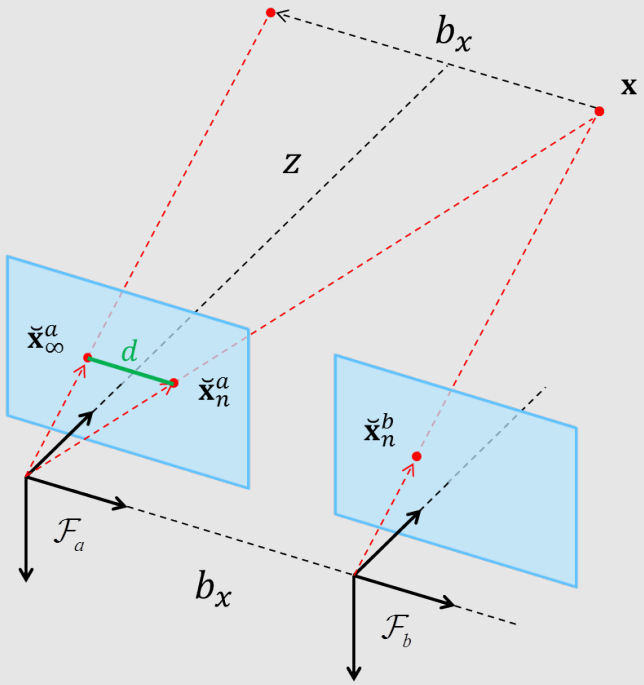


Figure 3.11: The ideal stereo geometry. The cameras have identical orientation, and baseline along the x-axis with distance b_x.

Stereo imaging exploits a special case of the epipolar geometry presented in

this section. In stereo geometry

$$\mathbf{K}_a = \mathbf{K}_b = \mathbf{K}, \tag{3.47}$$

$$\mathbf{R}_{ab} = \mathbf{R}_{ba} = \mathbf{I}, \tag{3.48}$$

and the translation between the cameras are typically along the x-axis

$$\mathbf{t}_{ab}^{a} = \begin{bmatrix} b_x \\ 0 \\ 0 \end{bmatrix}, \tag{3.49}$$

where b_x is the length of the baseline. This means that the epipoles are at the infinity,

$$\breve{\mathbf{u}}_{\infty}^{a} = \breve{\mathbf{u}}^{b}, \tag{3.50}$$

and the epipolar lines are horizontal along the rows in the images:

$$\breve{\mathbf{u}}^{a} = \breve{\mathbf{u}}^{b} + \begin{bmatrix} f_u \dfrac{b_x}{z} \\ 0 \\ 0 \end{bmatrix}. \tag{3.51}$$

The disparity is therefore given as

$$d = \|\mathbf{u}^a - \mathbf{u}_{\infty}^{a}\| = u^a - u^b = f_u \frac{b_x}{z}. \tag{3.52}$$

This results in a simple expression for the depth given the disparity $d = u^a - u^b$:

$$z = f_u \frac{b_x}{d}. \tag{3.53}$$

We can use this representation of the epipolar lines to restrict the search for correspondences between images even further. Clearly, the relevant part of the epipolar line can always be restricted on one end by $\tilde{\mathbf{u}}_{\infty}^{a}$ (or $\tilde{\mathbf{x}}_{\infty}^{a}$ on the normalised image plane), if it is visible in the image. With information about the relevant interval of the depth z^b, we can use (3.42) and (3.44) to define the corresponding relevant segment on the epipolar line. We will see in Chapter 8 that this representation can also be used to estimate the depth to a point, given the relative pose and point correspondence.

Chapter 4

A brief introduction to Lie theory

In Chapter 2 we saw that orientations and poses lie on manifolds in higher-dimensional spaces. This makes it complicated to add increments, represent uncertainty and perform differentiation for such variables, since they do not have the same properties as vectors in vector spaces.

As an example, consider perturbing a rotation matrix $\mathbf{R} \in SO(3)$ by adding a matrix $\delta\mathbf{R} \in \mathbb{R}^{3\times3}$. This is problematic since for some $\delta\mathbf{R}$ we have $\mathbf{R} + \delta\mathbf{R} \notin SO(3)$, which means that the perturbation in general will not produce a new rotation matrix. This also means that such perturbations is not a proper way to express the derivative of functions of rotation matrices.

We will in this chapter give a very brief introduction to a selection of extremely useful principles of *Lie theory*. The resulting mathematical framework will enable us to formulate estimation problems properly and with ease, by giving us the tools needed to work with rotations and poses *on the manifold*.

Most of this chapter is heavily based on [26], as well as [12, 4]. It is meant as a practical introduction to the subject, and is not a proper introduction to the underlying theory as such. Please see to the references above for more information.

4.1 The Lie group

Informally, a Lie group is both a smooth differentiable manifold and a group satisfying the group axioms. A group $(\mathcal{G}, \circ)$ is a set $\mathcal{G}$ with a composition operation $\circ$ that satisfies the axioms

$$\text{Closure under } \circ : \quad \mathcal{X} \circ \mathcal{Y} \in \mathcal{G} \tag{4.1}$$

$$\text{Identity } \mathcal{E} : \quad \mathcal{E} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{E} = \mathcal{X} \tag{4.2}$$

$$\text{Inverse } \mathcal{X}^{-1} : \quad \mathcal{X}^{-1} \circ \mathcal{X} = \mathcal{X} \circ \mathcal{X}^{-1} = \mathcal{E} \tag{4.3}$$

$$\text{Associativity} : \quad (\mathcal{X} \circ \mathcal{Y}) \circ \mathcal{Z} = \mathcal{X} \circ (\mathcal{Y} \circ \mathcal{Z}) \tag{4.4}$$

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

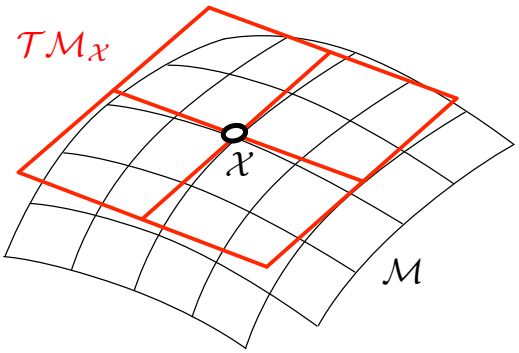


Figure 4.1: A manifold $\mathcal{M}$ with the tangent vector space $\mathcal{T}\mathcal{M}_\mathcal{X}$ at the point $\mathcal{X}$. In Lie groups, the manifold looks the same at every point, and all tangent spaces at any point are therefore alike.
(Image source: [26]; Cropped; licensed under CC BY-NC-SA 4.0)

Lie groups can also transform elements of other sets. Given a Lie group $\mathcal{M}$ and a set $\mathcal{V}$, we denote by $\mathcal{X} \cdot v$ the *action* of $\mathcal{X} \in \mathcal{M}$ on $v \in \mathcal{V}$. A group action must satisfy the axioms

$$\text{Identity} \ : \quad \mathcal{E} \cdot v = v \tag{4.5}$$

$$\text{Compatibility} \ : \quad (\mathcal{X} \circ \mathcal{Y}) \cdot v = \mathcal{X} \cdot (\mathcal{Y} \cdot v). \tag{4.6}$$

Since Lie groups are smooth manifolds, the group operations and actions are smooth, and look like operations on a vector space locally. In fact, there is a unique tangent space at each point on the manifold that captures the local structure of the group (Figure 4.1). This tangent space is a vector space that allows linear algebra and calculus.

In Lie groups, the local properties of smooth manifolds, which lets us do calculus, are combined with the global properties of groups, which lets us perform nonlinear composition and transformations on different objects. We will in this chapter see how we can exploit these properties to perform perturbations and handle uncertainty on any element of the Lie group manifolds. In Chapter 5 we will build upon this to define and compute derivatives on the manifolds.

The orientation group $SO(3)$ and pose group $SE(3)$ introduced in chapter 2 are both examples of *matrix Lie groups*. These are Lie groups where the elements of the group are matrices, the composition operation is matrix multiplication, and the inversion operation is matrix inversion. The action on vectors is $\mathbf{R} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x}$ for rotations $\mathbf{R} \in SO(3)$ and $\mathbf{T} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x} + \mathbf{t}$ for poses $\mathbf{T} \in SE(3)$. We will continue with a general introduction to Lie theory, before the properties of these groups are summarised in Sections 4.7 and 4.8.

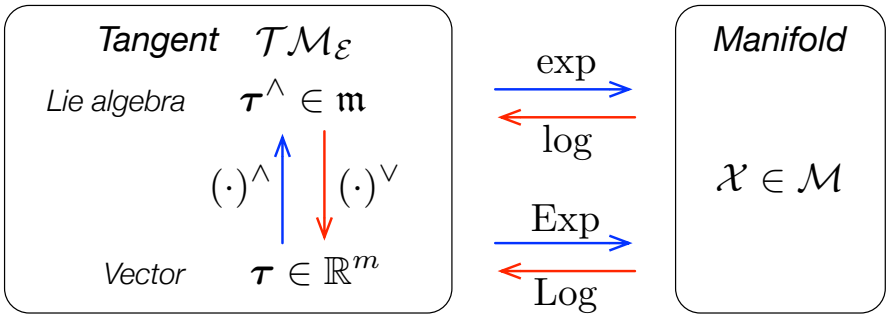


Figure 4.2: Mappings between the manifold $\mathcal{M}$ and the representations of its tangent space at the identity $\mathcal{T}\mathcal{M}_\mathcal{E}$. The hat $(\cdot)^\wedge$ and vee $(\cdot)^\vee$ operators map to/from the Lie algebra $\mathfrak{m}$ and tangent vector space $\mathbb{R}^m$. $\exp(\cdot)$ and $\log(\cdot)$ map the Lie algebra to/from the manifold, and $\mathrm{Exp}(\cdot)$ and $\mathrm{Log}(\cdot)$ are shortcuts to map the tangent vector space directly to/from $\mathcal{M}$.
(Image source: [26]; licensed under CC BY-NC-SA 4.0)

4.2 The Lie algebra

The structure of the tangent spaces on a Lie group manifold $\mathcal{M}$ is the same everywhere. We will denote the tangent space to $\mathcal{M}$ at $\mathcal{X}$ as $\mathcal{T}\mathcal{M}_\mathcal{X}$, as illustrated in Figure 4.1. The tangent space at the identity $\mathcal{T}\mathcal{M}_\mathcal{E}$ is called the *Lie algebra* of $\mathcal{M}$, and is denoted $\mathfrak{m}$:

$$\text{Lie algebra} \ : \quad \mathfrak{m} \triangleq \mathcal{T}\mathcal{M}_\mathcal{E}. \tag{4.7}$$

The Lie algebra $\mathfrak{m}$ is a vector space, and although its elements $\boldsymbol{\tau}^\wedge \in \mathfrak{m}$ can have non-trivial structures, they can be *identified* with vectors $\boldsymbol{\tau} \in \mathbb{R}^m$, where m is the dimensionality of $\mathcal{M}$. This is done by expressing $\boldsymbol{\tau}^\wedge$ as a linear combination of some base elements $\mathbf{E}_i$, where $\mathbf{E}_i$ are called the *generators* of $\mathfrak{m}$. We may in this way map between $\mathbb{R}^m$ to $\mathfrak{m}$ and vice versa with two mutually inverse linear maps called *hat* and *vee*:

$$\text{Hat: } (\cdot)^\wedge : \mathbb{R}^m \to \mathfrak{m}; \qquad \boldsymbol{\tau}^\wedge = \sum_{i=1}^{m} \tau_i \mathbf{E}_i \tag{4.8}$$

$$\text{Vee: } (\cdot)^\vee : \mathfrak{m} \to \mathbb{R}^m; \qquad \boldsymbol{\tau} = (\boldsymbol{\tau}^\wedge)^\vee = \sum_{i=1}^{m} \tau_i \mathbf{e}_i, \tag{4.9}$$

where $\mathbf{e}_i$ are the basis vectors of $\mathbb{R}^m$ so that $\mathbf{e}_i^\wedge = \mathbf{E}_i$ (Figure 4.2). This lets us represent elements in the Lie algebra using vectors in a Cartesian vector space, which may be manipulated with linear algebra using matrix operations.

4.3 The exponential map

The *exponential map* transfers elements of the Lie algebra $\boldsymbol{\tau}^\wedge \in \mathfrak{m}$ to elements of the group $\mathcal{X} \in \mathcal{M}$:

$$\exp : \mathfrak{m} \to \mathcal{M}; \qquad \mathcal{X} = \exp(\boldsymbol{\tau}^\wedge). \tag{4.10}$$

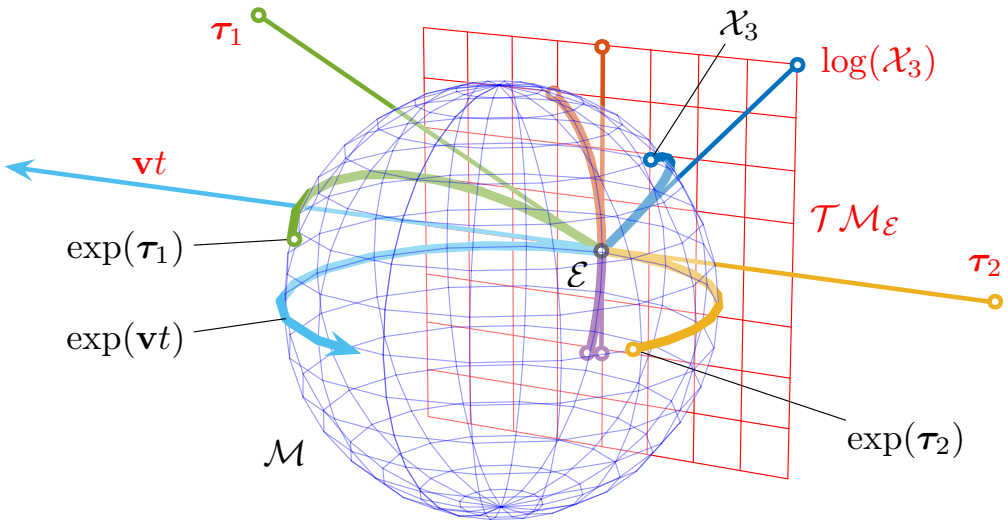


Figure 4.3: An illustration of the relation between the Lie group and the Lie algebra. The Lie algebra $\mathcal{T}\mathcal{M}_\mathcal{E}$ is the tangent space to the Lie group's manifold $\mathcal{M}$ at the identity $\mathcal{E}$. The exponential map wraps straight paths through the origin on the Lie algebra over the manifold along the corresponding geodesic. The logarithmic map unwraps curved and nonlinear paths on the manifold onto the linear Lie algebra. All points on the manifold has an exact equivalent on the tangent space.
(Image source: [26]; licensed under CC BY-NC-SA 4.0)

Intuitively, it wraps the tangent element over the manifold following the *geodesic*, which in some sense is the shortest route along the manifold (Figure 4.3). The exponential map is *exact* for all tangent elements, but does not always have a closed form expression.

The inverse transformation is the *logarithmic map*,

$$\log : \mathcal{M} \to \mathfrak{m}; \qquad \boldsymbol{\tau}^\wedge = \log(\mathcal{X}), \tag{4.11}$$

which unwraps elements on the manifold onto the tangent space (Figure 4.3).

The *capitalised exponential and logarithmic maps* are convenient compositions that lets us map vector elements $\boldsymbol{\tau} \in \mathbb{R}^m$ directly to group elements $\mathcal{X} \in \mathcal{M}$, and vice versa. These are given by

$$\mathrm{Exp} : \mathbb{R}^m \to \mathcal{M}; \qquad \mathcal{X} = \mathrm{Exp}(\boldsymbol{\tau}) \triangleq \exp(\boldsymbol{\tau}^\wedge) \tag{4.12}$$

$$\mathrm{Log} : \mathcal{M} \to \mathbb{R}^m; \qquad \boldsymbol{\tau} = \mathrm{Log}(\mathcal{X}) \triangleq \log(\mathcal{X})^\vee. \tag{4.13}$$

Figure 4.2 gives an overview of the different operators we can use to map between the manifold $\mathcal{M}$ and the representations of its tangent space at the identity $\mathcal{T}\mathcal{M}_\mathcal{E}$. Additional properties of the exponential map is given in [26, 4].

4.4 Right and left perturbations

The exponential map allows us to perform perturbations on elements $\mathcal{X}$ on the manifold expressed as tangent space vectors $\boldsymbol{\tau}$ by combining one Exp/Log operation with one composition. Since the composition is non-commutative, the order of the operands is important for how the perturbations are performed.

With the combination

$$\mathcal{X} \circ \text{Exp}({}^{\mathcal{X}}\boldsymbol{\tau}), \tag{4.14}$$

the perturbation is on the right side of the composition. The tangent space vector ${}^{\mathcal{X}}\boldsymbol{\tau}$ therefore belongs to the tangent space at $\mathcal{X}$, as denoted by the left superscript. Following the convention we have used in Chapter 2, we say that ${}^{\mathcal{X}}\boldsymbol{\tau}$ is expressed in the *local* frame at $\mathcal{X}$.

When we perform the perturbation on the left

$$\text{Exp}({}^{\mathcal{E}}\boldsymbol{\tau}) \circ \mathcal{X}, \tag{4.15}$$

we have ${}^{\mathcal{E}}\boldsymbol{\tau} \in \mathcal{TM}_{\mathcal{E}}$, and we say that ${}^{\mathcal{E}}\boldsymbol{\tau}$ is expressed in the *global* frame.

Example 4.7 Local and global perturbations on poses

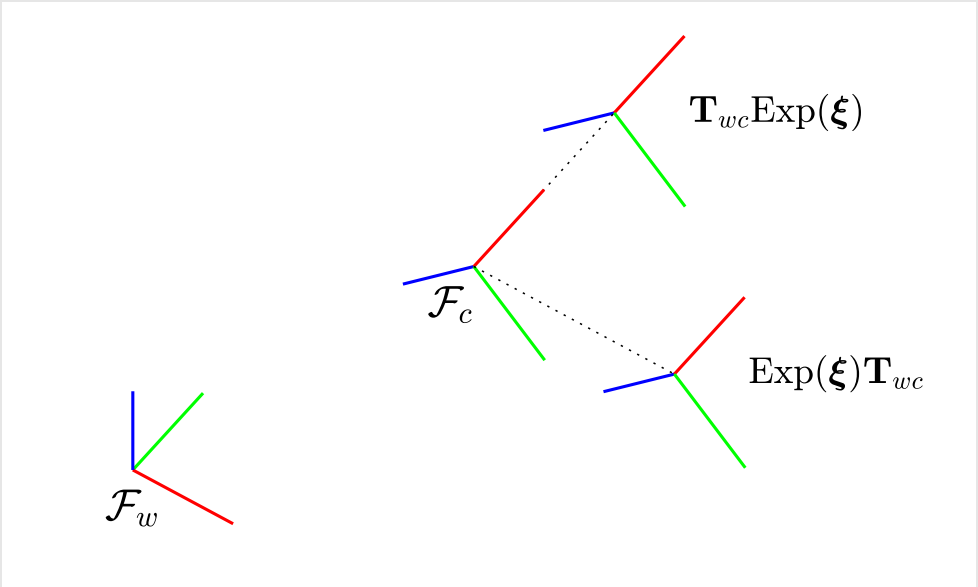


Figure 4.4: A world frame $\mathcal{F}_w$ and a camera frame $\mathcal{F}_c$. A perturbation along the x-axis is performed on the right- and left-hand sides of the pose $\mathbf{T}_{wc}$. This corresponds to perturbations locally in the camera frame, and globally in the world frame, respectively.

The pose of a camera relative to the world frame is expressed as $\mathbf{T}_{wc}$. Perturbations on the right side of $\mathbf{T}_{wc}$ are performed in the local camera coordinate frame, *before* the transformation between the coordinate frames. Perturbations on the left side are performed in the global world coordinate frame, *after* the transformation.

The tangent vector $\boldsymbol{\xi} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$ corresponds to an increment of 2 units in the x-direction (see Section 4.8). Figure 4.4 shows the result of perturbing $\mathbf{T}_{wc}$ with $\boldsymbol{\xi}$ locally along the x-axis of $\mathcal{F}_c$ with $\mathbf{T}_{wc}\operatorname{Exp}(\boldsymbol{\xi})$, and globally along the x-axis of $\mathcal{F}_w$ with $\operatorname{Exp}(\boldsymbol{\xi})\mathbf{T}_{wc}$.

We will in the following mostly consider local right side perturbations, although the left side convention is also common. See [26, 4] for corresponding discussions covering left perturbations.

4.5 Plus and minus operators

It is convenient to express right perturbations using the right plus and minus operators

$$\mathcal{Y} = \mathcal{X} \oplus {}^{\mathcal{X}}\boldsymbol{\tau} \triangleq \mathcal{X} \circ \operatorname{Exp}({}^{\mathcal{X}}\boldsymbol{\tau}) \in \mathcal{M} \tag{4.16}$$

$${}^{\mathcal{X}}\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X} \triangleq \operatorname{Log}(\mathcal{X}^{-1} \circ \mathcal{Y}) \in \mathcal{TM}_{\mathcal{X}}. \tag{4.17}$$

The plus operator lets us increment an element $\mathcal{X}$ with a tangent space vector ${}^{\mathcal{X}}\boldsymbol{\tau}$, while the minus operator lets us compute the corresponding tangent space vector between the two elements $\mathcal{X}$ and $\mathcal{Y}$. We will later use these operators to define derivatives on the manifold (Section 5.2).

It is also possible to define left plus and minus operators

$$\mathcal{Y} = {}^{\mathcal{E}}\boldsymbol{\tau} \oplus \mathcal{X} \triangleq \operatorname{Exp}({}^{\mathcal{E}}\boldsymbol{\tau}) \circ \mathcal{X} \in \mathcal{M} \tag{4.18}$$

$${}^{\mathcal{E}}\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X} \triangleq \operatorname{Log}(\mathcal{Y} \circ \mathcal{X}^{-1}) \in \mathcal{TM}_{\mathcal{E}}. \tag{4.19}$$

Notice that while left and right $\oplus$ are distinguished by the operands, $\ominus$ is ambiguous. Since we mostly consider right perturbations, we will also use the right definitions of $\oplus$ and $\ominus$ as default.

Example 4.8 Interpolation on the manifold

We can interpolate between two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ with the linear interpolation scheme

$$\mathbf{x} = (1-\alpha)\mathbf{x}_1 + \alpha\mathbf{x}_2, \quad \alpha \in [0,1]. \tag{4.20}$$

This scheme will not work in general for Lie groups, since not all Lie groups are closed under addition. For poses $\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$ for example, we will get

$$(1-\alpha)\mathbf{T}_1 + \alpha\mathbf{T}_2 \notin SE(3) \tag{4.21}$$

for some values of $\alpha \in [0,1]$.

We can instead define a proper interpolation scheme on the manifold using Lie algebra. The tangent vector between $\mathcal{X}_1$ and $\mathcal{X}_2$ in $\mathcal{TM}_{\mathcal{X}_1}$ is given by $\mathcal{X}_2 \ominus \mathcal{X}_1$.

The interpolation scheme

$$\mathcal{X} = \mathcal{X}_1 \oplus \alpha(\mathcal{X}_2 \ominus \mathcal{X}_1) \tag{4.22a}$$
$$= \mathcal{X}_1 \circ \operatorname{Exp}(\alpha \operatorname{Log}(\mathcal{X}_1^{-1} \circ \mathcal{X}_2)) \tag{4.22b}$$

will therefore move steadily along the geodesic from $\mathcal{X}_1$ to $\mathcal{X}_2$.

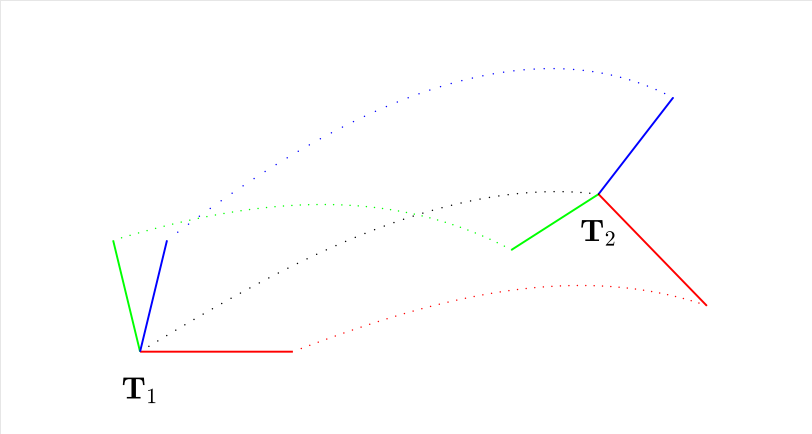


Figure 4.5: Interpolation along the manifold between two poses $\mathbf{T}_1$ and $\mathbf{T}_2$.

For poses $\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$, we get

$$\mathbf{T} = \mathbf{T}_1 \circ \operatorname{Exp}(\alpha \operatorname{Log}(\mathbf{T}_1^{-1} \circ \mathbf{T}_2)) \tag{4.23a}$$
$$= \mathbf{T}_1(\mathbf{T}_1^{-1}\mathbf{T}_2)^{\alpha}. \tag{4.23b}$$

An example of the interpolated motion along the manifold between two poses is shown in Figure 4.5. We see that it follows a screw motion, corresponding to constant angular velocity, and constant translational velocity locally, so that the direction is constantly changed in the global frame.

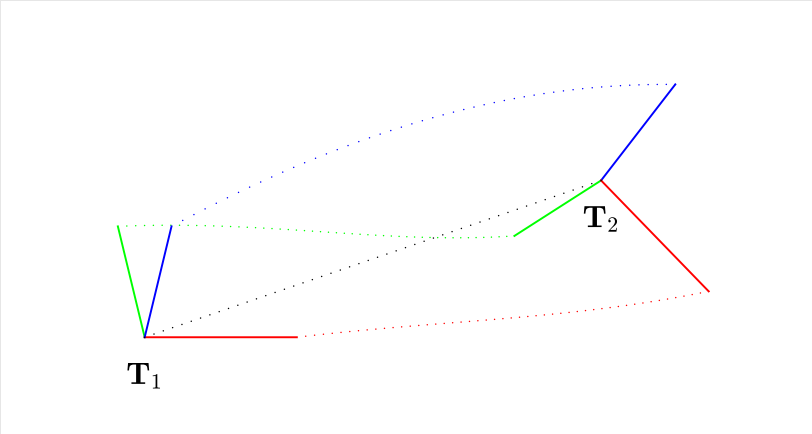


Figure 4.6: Separate interpolation of rotation and position between two poses $\mathbf{T}_1$ and $\mathbf{T}_2$.

An alternative is to interpolate orientation and position separately (see the hybrid representation in Section 4.8.4). For the poses $\mathbf{T}_1 = \left[\begin{smallmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{smallmatrix}\right]$ and $\mathbf{T}_2 =$

$\left[\begin{smallmatrix}\mathbf{R}_2 & \mathbf{t}_2 \\ \mathbf{0}^\top & 1\end{smallmatrix}\right]$, we get the separated interpolation scheme

$$\mathbf{R} = \mathbf{R}_1 \oplus \alpha(\mathbf{R}_2 \ominus \mathbf{R}_1) \tag{4.24}$$
$$\mathbf{t} = \mathbf{t}_1 + \alpha(\mathbf{t}_2 - \mathbf{t}_1), \tag{4.25}$$

where the interpolated pose is given by $\mathbf{T} = \left[\begin{smallmatrix}\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1\end{smallmatrix}\right]$, and we have rearranged (4.20) a bit to show how similar the vector space and manifold schemes are. Similarly to (4.23b), (4.24) can also be expressed as $\mathbf{R} = \mathbf{R}_1(\mathbf{R}_1^\top \mathbf{R}_2)^\alpha$. This separated scheme results in the linear translation motion shown in Figure 4.6.

4.6 The adjoint

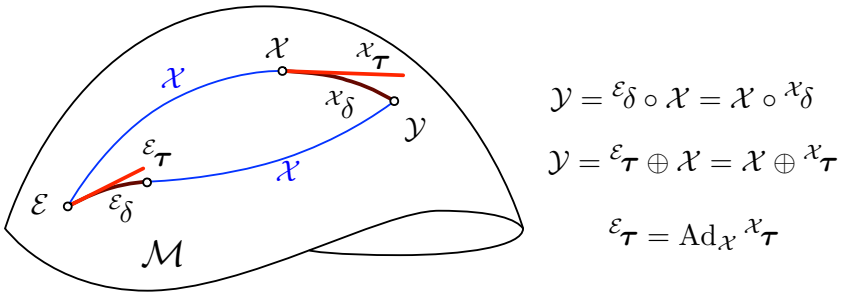


Figure 4.7: Two compositions $\mathcal{X} \circ {}^{\mathcal{X}}\delta$ and ${}^{\mathcal{E}}\delta \circ \mathcal{X}$ join the identity $\mathcal{E}$ with the element $\mathcal{Y}$. Due to non-commutativity, the elements ${}^{\mathcal{X}}\delta$ and ${}^{\mathcal{E}}\delta$ are not equal. The corresponding tangent space vectors ${}^{\mathcal{X}}\boldsymbol{\tau} = \mathrm{Log}({}^{\mathcal{X}}\delta)$ and ${}^{\mathcal{E}}\boldsymbol{\tau} = \mathrm{Log}({}^{\mathcal{E}}\delta)$ are related by ${}^{\mathcal{E}}\boldsymbol{\tau} = \mathbf{Ad}_{\mathcal{X}}\, {}^{\mathcal{X}}\boldsymbol{\tau}$, where $\mathbf{Ad}_{\mathcal{X}}$ is the adjoint matrix of $\mathcal{M}$ at $\mathcal{X}$.
(Image source: [26]; licensed under CC BY-NC-SA 4.0)

In Section 4.4 we saw that there are two natural ways of perturbing an element $\mathcal{X}$ on the manifold by a tangent vector. We can either right-perturb by a vector of the local tangent space, $\mathcal{X} \circ \mathrm{Exp}({}^{\mathcal{X}}\boldsymbol{\tau})$, or we can left-perturb by a vector of the tangent space at the identity, $\mathrm{Exp}({}^{\mathcal{E}}\boldsymbol{\tau}) \circ \mathcal{X}$. One can show that this duality induces a relationship between the two tangent spaces called the *adjoint of* $\mathcal{M}$ *at* $\mathcal{X}$, denoted by $\mathrm{Ad}_{\mathcal{X}}$.

The *adjoint action* of a group on its own Lie algebra is defined as

$$\mathrm{Ad}_{\mathcal{X}} : \mathfrak{m} \to \mathfrak{m}; \qquad \mathrm{Ad}_{\mathcal{X}}(\boldsymbol{\tau}^\wedge) \triangleq \mathcal{X}\boldsymbol{\tau}^\wedge\mathcal{X}^{-1}. \tag{4.26}$$

As illustrated in Figure 4.7, the adjoint determines a relation between the local and global tangent elements

$${}^{\mathcal{E}}\boldsymbol{\tau}^\wedge = \mathrm{Ad}_{\mathcal{X}}({}^{\mathcal{X}}\boldsymbol{\tau}^\wedge), \tag{4.27}$$

so that

$${}^{\mathcal{E}}\boldsymbol{\tau}^\wedge \oplus \mathcal{X} = \mathcal{X} \oplus {}^{\mathcal{X}}\boldsymbol{\tau}^\wedge. \tag{4.28}$$

$\mathrm{Ad}_{\mathcal{X}}()$ is a linear transformation, so we can define an equivalent matrix operator that maps the tangent vectors directly:

$$\mathbf{Ad}_{\mathcal{X}} : \mathbb{R}^m \to \mathbb{R}^m; \qquad {}^{\mathcal{E}}\boldsymbol{\tau} = \mathbf{Ad}_{\mathcal{X}}\, {}^{\mathcal{X}}\boldsymbol{\tau}. \tag{4.29}$$

The matrix $\mathbf{Ad}_{\mathcal{X}}$ is called the *adjoint matrix*, and it lets us linearly transform vectors of the tangent space at $\mathcal{X}$ onto vectors of the tangent space at the identity $\mathcal{E}$, corresponding to the same perturbation of $\mathcal{X}$. The adjoint matrix can be computed by applying the vee operator to (4.26), so that

$$\mathbf{Ad}_{\mathcal{X}} \triangleq (\mathcal{X}\boldsymbol{\tau}^{\wedge}\mathcal{X}^{-1})^{\vee}. \tag{4.30}$$

Additional properties of the adjoint matrix is given in [26, 4].

Example 4.9 Computing global from local and vice versa

In Example 4.7 we experienced the difference between applying a perturbation $\boldsymbol{\xi} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\top}$ to the pose $\mathbf{T}_{wc}$ locally on the right, and globally on the left. The results are shown in Figure 4.4. We can use the adjoint matrix for poses $\mathbf{Ad}_{\mathbf{T}}$ (see Section 4.8) to compute the corresponding perturbations in the other frame.

When $\boldsymbol{\xi}$ is applied locally on the right side, the corresponding global perturbation is given by

$$\boldsymbol{\xi}^{w} = \mathbf{Ad}_{\mathbf{T}_{wc}}\boldsymbol{\xi} = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}^{\top}, \tag{4.31}$$

which is an increment of 2 units along the y-axis of $\mathcal{F}_w$.

When $\boldsymbol{\xi}$ is applied globally on the left side, we can use the inverse pose to compute the corresponding local perturbation by

$$\boldsymbol{\xi}^{c} = \mathbf{Ad}_{\mathbf{T}_{wc}^{-1}}\boldsymbol{\xi} = \begin{bmatrix} 0 & \sqrt{2} & -\sqrt{2} & 0 & 0 & 0 \end{bmatrix}^{\top}, \tag{4.32}$$

which is an increment of $\sqrt{2}$ units along the y-axis of $\mathcal{F}_c$ and $-\sqrt{2}$ units along the z-axis of $\mathcal{F}_c$.

We can use the adjoint matrix to transform tangent space vectors between coordinate frames, in the same way that we use poses to transform points.

4.7 The 3D rotation group $SO(3)$

The *special orthogonal group* in 3D is the set of valid rotation matrices

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} \;\middle|\; \mathbf{R}\mathbf{R}^{\top} = \mathbf{I}, \det \mathbf{R} = 1 \right\}, \tag{4.33}$$

and is closed under matrix multiplication with identity $\mathbf{I}$. Inversion is achieved with transposition

$$\mathbf{R}^{-1} = \mathbf{R}^{\top}, \tag{4.34}$$

and composition with the product

$$\mathbf{R}_a \circ \mathbf{R}_b = \mathbf{R}_a\mathbf{R}_b. \tag{4.35}$$

The group action on vectors is given by the product

$$\mathbf{R} \cdot \mathbf{x} = \mathbf{R}\mathbf{x}. \tag{4.36}$$

4.7.1 Lie algebra

The Lie algebra of $SO(3)$ is given by

$$\mathfrak{so}(3) = \left\{ \boldsymbol{\theta}^\wedge = [\boldsymbol{\theta}]_\times \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{\theta} \in \mathbb{R}^3 \right\}, \tag{4.37}$$

where the tangent space vector $\boldsymbol{\theta} \triangleq \theta\mathbf{u}$ corresponds to the rotation on angle-axis form, with angle θ and unit rotation axis $\mathbf{u}$ (see Section 2.2). Since $\boldsymbol{\theta} \in \mathbb{R}^3$, the dimension of $SO(3)$ is $m = 3$.

The Lie algebra is a vector space that can be decomposed into

$$\boldsymbol{\theta}^\wedge = [\boldsymbol{\theta}]_\times = \theta_1\mathbf{E}_1 + \theta_2\mathbf{E}_2 + \theta_3\mathbf{E}_3, \tag{4.38}$$

where

$$\mathbf{E}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, \; \mathbf{E}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \; \mathbf{E}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.39}$$

are the generators of $\mathfrak{so}(3)$. The hat and vee operators are given by

$$\begin{aligned} \text{Hat} \;:\quad & \boldsymbol{\theta}^\wedge = [\boldsymbol{\theta}]_\times \\ \text{Vee} \;:\quad & \boldsymbol{\theta} \;\; = [\boldsymbol{\theta}]_\times^\vee. \end{aligned} \tag{4.40}$$

4.7.2 Exp and Log maps

Since the tangent space vector $\boldsymbol{\theta} = \theta\mathbf{u}$ corresponds to the axis-angle representation, the Exp map is simply the Rodrigues' rotation formula

$$\mathbf{R} = \mathrm{Exp}(\boldsymbol{\theta}) \triangleq \mathbf{I} + \sin\theta[\mathbf{u}]_\times + (1 - \cos\theta)[\mathbf{u}]_\times^2. \tag{4.41}$$

The Log map is given by

$$\boldsymbol{\theta} = \mathrm{Log}(\mathbf{R}) \triangleq \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^\top)^\vee, \tag{4.42}$$

where

$$\theta = \arccos\left(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}\right). \tag{4.43}$$

Special care must be taken when θ is small. Practical implementations can in these cases use a Taylor expansion of the coefficient $\theta/(2\sin\theta)$ [12]. Also, when θ is small, the following approximation holds (see (2.19)):

$$\mathbf{R} = \mathrm{Exp}(\boldsymbol{\theta}) \approx \mathbf{I} + \boldsymbol{\theta}^\wedge. \tag{4.44}$$

4.7.3 The adjoint

The adjoint matrix for $SO(3)$ at $\mathbf{R}$ is given by

$$\mathbf{Ad}_{\mathbf{R}} = \mathbf{R} \in \mathbb{R}^{3\times 3}. \tag{4.45}$$

4.8 The 3D rigid motion group $SE(3)$

The *special Euclidean group* in 3D is the set of valid Euclidean transformation matrices

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \;\middle|\; \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\}, \tag{4.46}$$

and is closed under matrix multiplication with identity $\mathbf{I}$. Inversion is given by

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{4.47}$$

and composition with the product

$$\mathbf{T}_a \circ \mathbf{T}_b = \mathbf{T}_a \mathbf{T}_b = \begin{bmatrix} \mathbf{R}_a \mathbf{R}_b & \mathbf{R}_a \mathbf{t}_b + \mathbf{t}_a \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{4.48}$$

The group action on vectors is given by

$$\mathbf{T} \cdot \mathbf{x} = \mathbf{T}\tilde{\mathbf{x}} = \mathbf{R}\mathbf{x} + \mathbf{t}. \tag{4.49}$$

4.8.1 Lie algebra

The Lie algebra of $SE(3)$ is given by

$$\mathfrak{se}(3) = \left\{ \boldsymbol{\xi}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} \;\middle|\; \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix} \in \mathbb{R}^6 \right\}, \tag{4.50}$$

Since $\boldsymbol{\xi} \in \mathbb{R}^6$, the dimension of $SE(3)$ is $m = 6$. The vectors $\boldsymbol{\rho}, \boldsymbol{\theta} \in \mathbb{R}^3$ correspond the translational and rotational parts, respectively.

The Lie algebra is a vector space that can be decomposed into

$$\boldsymbol{\xi}^\wedge = \xi_1 \mathbf{E}_1 + \xi_2 \mathbf{E}_2 + \xi_3 \mathbf{E}_3 + \xi_4 \mathbf{E}_4 + \xi_5 \mathbf{E}_5 + \xi_6 \mathbf{E}_6, \tag{4.51}$$

where

$$\begin{aligned} \mathbf{E}_1 &= \begin{bmatrix} 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}, \quad \mathbf{E}_2 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}, \quad \mathbf{E}_3 = \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&0&0 \end{bmatrix}, \\ \mathbf{E}_4 &= \begin{bmatrix} 0&0&0&0 \\ 0&0&-1&0 \\ 0&1&0&0 \\ 0&0&0&0 \end{bmatrix}, \mathbf{E}_5 = \begin{bmatrix} 0&0&1&0 \\ 0&0&0&0 \\ -1&0&0&0 \\ 0&0&0&0 \end{bmatrix}, \mathbf{E}_6 = \begin{bmatrix} 0&-1&0&0 \\ 1&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}, \end{aligned} \tag{4.52}$$

are the generators of $\mathfrak{se}(3)$. The hat and vee operators are given by

$$\begin{aligned} \text{Hat} &: \quad \boldsymbol{\xi}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix} \\ \text{Vee} &: \quad \boldsymbol{\xi} \;\; = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix}^\vee. \end{aligned} \tag{4.53}$$

4.8.2 Exp and Log maps

The Exp map for $SE(3)$ is given by

$$\mathbf{T} = \text{Exp}(\boldsymbol{\xi}) \triangleq \begin{bmatrix} \text{Exp}(\boldsymbol{\theta}) & \mathbf{V}(\boldsymbol{\theta})\boldsymbol{\rho} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{4.54}$$

where $\text{Exp}(\boldsymbol{\theta})$ is given by (4.41) and

$$\mathbf{V}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1-\cos\theta}{\theta}[\mathbf{u}]_\times + \frac{\theta - \sin\theta}{\theta}[\mathbf{u}]_\times^2. \tag{4.55}$$

The Log map is given by

$$\boldsymbol{\xi} = \text{Log}(\mathbf{T}) \triangleq \begin{bmatrix} \mathbf{V}^{-1}(\boldsymbol{\theta})\mathbf{t} \\ \boldsymbol{\theta} \end{bmatrix}, \tag{4.56}$$

where $\boldsymbol{\theta} = \text{Log}(\mathbf{R})$ from (4.42) and

$$\mathbf{V}^{-1}(\boldsymbol{\theta}) = \mathbf{I} - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\theta\sin\theta}{2(1-\cos\theta)}\right)[\mathbf{u}]_\times^2. \tag{4.57}$$

As for $SO(3)$ in Section 4.7.2, special care must be taken when θ is small [12]. Also, when θ is small, the following approximation holds:

$$\mathbf{T} = \text{Exp}(\boldsymbol{\xi}) \approx \mathbf{I} + \boldsymbol{\xi}^\wedge. \tag{4.58}$$

4.8.3 The adjoint

The adjoint matrix for $SE(3)$ at $\mathbf{T}$ is given by

$$\mathbf{Ad}_\mathbf{T} = \begin{bmatrix} \mathbf{R} & [\mathbf{t}]_\times\mathbf{R} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{4.59}$$

4.8.4 Hybrid representation

An alternative to representing poses on the $SE(3)$ manifold, is to represent the orientation $\mathbf{R} \in SO(3)$ and position $\mathbf{t} \in \mathbb{R}^3$ as two separate states without interaction. Since $\mathbf{t}$ is a vector, we will in this case only need to compute the Exp and Log maps for the orientation on the $SO(3)$ manifold.

We can define maps the corresponding to Exp and Log for the hybrid representation as the pseudo-Exp and pseudo-Log maps [6]:

$$\mathbf{T} = \text{pseudo-Exp}(\boldsymbol{\xi}) \triangleq \begin{bmatrix} \text{Exp}(\boldsymbol{\theta}) & \boldsymbol{\rho} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{4.60}$$

$$\boldsymbol{\xi} = \text{pseudo-Log}(\mathbf{T}) \triangleq \begin{bmatrix} \mathbf{t} \\ \text{Log}(\mathbf{R}) \end{bmatrix}. \tag{4.61}$$

A *retraction* is a more general way to define a mapping between the tangent space $\mathcal{TM}$ and the manifold $\mathcal{M}$ [17, 11, 1]. For $SE(3)$, the exponential map (4.54) is one example of retraction, while the pseudo-Exp (4.60) is another, and sometimes more attractive choice. This is because the pseudo-versions saves computations both for the Exp and Log maps, as well for the Jacobians, as we shall see in the next chapter.

Chapter 5

Jacobians

Problems in computer vision often involve nonlinear functions, such as camera projections and 3D transformations. It is also common that these functions operate on and produce variables that are defined on manifolds, such as rotations on $SO(3)$ and poses on $SE(3)$. As we will see in the following chapters, propagating uncertainty and estimating states from observations will typically involve linearising such functions with Taylor expansions. Multivariate differentiation on vector spaces and Lie groups is therefore a key mathematical tool in vision-based state estimation.

This chapter will introduce useful notation and a procedure for computing derivatives on vector spaces and Lie groups based on [26]. We will finish this section with listing useful Jacobians for common operations on $SO(3)$ and $SE(3)$.

5.1 Derivatives in vector spaces

Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a multivariate function that takes vectors $\mathbf{x} \in \mathbb{R}^m$ as input and produces vectors $f(\mathbf{x}) \in \mathbb{R}^n$ as output. The Jacobian matrix of f is defined as the $n \times m$ matrix that stacks all the partial derivatives

$$\mathbf{J} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \in \mathbb{R}^{n \times m}. \tag{5.1}$$

Each column vector $\mathbf{j}_i = [\frac{\partial f_1}{\partial x_i} \dots \frac{\partial f_n}{\partial x_i}]^\top$ of the Jacobian (5.1) corresponds to

$$\mathbf{j}_i = \frac{\partial f(\mathbf{x})}{\partial x_i} \triangleq \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} \in \mathbb{R}^n, \tag{5.2}$$

where $\mathbf{e}_i$ is the i-th natural basis vector of $\mathbb{R}^m$. This represents the instantaneous variation of $f(\mathbf{x})$ when $\mathbf{x}$ is perturbed in the direction of $\mathbf{e}_i$. We will in the following adopt the convenient notation from [26], where the Jacobian is given on the compact form

$$\mathbf{J} = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \triangleq \lim_{\mathbf{h} \to 0} \frac{f(\mathbf{x} + \mathbf{h}) - f(\mathbf{x})}{\mathbf{h}} \in \mathbb{R}^{n \times m}, \tag{5.3}$$

with $\mathbf{h} \in \mathbb{R}^m$, which combines all the columns (5.2) to form the definition of (5.1). Like the Jacobian matrix in (5.1), the form used in (5.3) is just a notational convenience, since division by the vector $\mathbf{h}$ is undefined, and proper computation requires (5.2). However, by using this notation we can follow a straight-forward procedure to calculate Jacobians, by developing the numerator into a form linear in $\mathbf{h}$ and identifying the left hand side as the Jacobian:

$$\lim_{\mathbf{h}\to 0} \frac{f(\mathbf{x}+\mathbf{h}) - f(\mathbf{x})}{\mathbf{h}} = \ldots = \lim_{\mathbf{h}\to 0} \frac{\mathbf{J}\mathbf{h}}{\mathbf{h}} \triangleq \frac{\partial(\mathbf{J}\mathbf{h})}{\partial \mathbf{h}} = \mathbf{J}. \tag{5.4}$$

To distinguish and identify different Jacobians, we will use the notation $\mathbf{J}_{\mathbf{x}}^{f(\mathbf{x})} \triangleq \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ and $\mathbf{J}_{\mathbf{x}}^{\mathbf{y}} \triangleq \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$.

Example 5.10 Computing the Jacobian $\mathbf{J}_{\mathbf{x}}^{\mathbf{A}\mathbf{x}}$

For a vector $\mathbf{x} \in \mathbb{R}^m$ and a constant conformable matrix $\mathbf{A}$, we want to compute the derivative of $f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ with respect to $\mathbf{x}$. Using (5.3) we get

$$\begin{aligned}
\mathbf{J}_{\mathbf{x}}^{\mathbf{A}\mathbf{x}} = \frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} &= \lim_{\mathbf{h}\to 0} \frac{\mathbf{A}(\mathbf{x}+\mathbf{h}) - \mathbf{A}\mathbf{x}}{\mathbf{h}} && \text{(5.5a)}\\
&= \lim_{\mathbf{h}\to 0} \frac{\mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{h} - \mathbf{A}\mathbf{x}}{\mathbf{h}} && \text{(5.5b)}\\
&= \lim_{\mathbf{h}\to 0} \frac{\mathbf{A}\mathbf{h}}{\mathbf{h}} && \text{(5.5c)}\\
&= \mathbf{A}. && \text{(5.5d)}
\end{aligned}$$

Example 5.11 Computing the Jacobian $\mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{A}\mathbf{x}}$

For a vector $\mathbf{x} \in \mathbb{R}^m$ and a constant matrix $\mathbf{A} \in \mathbb{R}^{m\times m}$, we now want to compute the derivative of the scalar-valued quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x}$ with respect to $\mathbf{x}$. By applying (5.3) we get

$$\begin{aligned}
\mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{A}\mathbf{x}} &= \lim_{\mathbf{h}\to 0} \frac{(\mathbf{x}+\mathbf{h})^\top \mathbf{A}(\mathbf{x}+\mathbf{h}) - \mathbf{x}^\top \mathbf{A}\mathbf{x}}{\mathbf{h}} && \text{(5.6a)}\\
&= \lim_{\mathbf{h}\to 0} \frac{\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{h}^\top \mathbf{A}\mathbf{x} + \mathbf{x}^\top \mathbf{A}\mathbf{h} + \mathbf{h}^\top \mathbf{A}\mathbf{h} - \mathbf{x}^\top \mathbf{A}\mathbf{x}}{\mathbf{h}}. && \text{(5.6b)}
\end{aligned}$$

By dropping the higher order term and using that for scalars $a^\top = a$, we get

$$\mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{A}\mathbf{x}} = \lim_{\mathbf{h}\to 0} \frac{\mathbf{x}^\top \mathbf{A}\mathbf{h} + (\mathbf{h}^\top \mathbf{A}\mathbf{x})^\top}{\mathbf{h}} \tag{5.6c}$$

$$= \lim_{\mathbf{h}\to 0} \frac{\mathbf{x}^\top \mathbf{A}\mathbf{h} + \mathbf{x}^\top \mathbf{A}^\top \mathbf{h}}{\mathbf{h}} \tag{5.6d}$$

$$= \lim_{\mathbf{h}\to 0} \frac{\mathbf{x}^\top (\mathbf{A} + \mathbf{A}^\top)\mathbf{h}}{\mathbf{h}} \tag{5.6e}$$

$$= \mathbf{x}^\top (\mathbf{A} + \mathbf{A}^\top). \tag{5.6f}$$

For $\mathbf{y} = f(\mathbf{x})$ and $\mathbf{z} = g(\mathbf{y})$ we have $\mathbf{z} = g(f(\mathbf{x}))$. The chain rule also applies in multivariate calculus, and lets us compute the derivative of $\mathbf{z}$ with respect to $\mathbf{x}$ as

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}. \tag{5.7}$$

We can therefore express the Jacobian $\mathbf{J}_{\mathbf{x}}^{\mathbf{z}}$ as a composition of several Jacobian blocks

$$\mathbf{J}_{\mathbf{x}}^{\mathbf{z}} = \mathbf{J}_{\mathbf{y}}^{\mathbf{z}} \mathbf{J}_{\mathbf{x}}^{\mathbf{y}}. \tag{5.8}$$

Example 5.12 Computing the Jacobian $\mathbf{J}_{\mathbf{x}}^{\|\mathbf{x}-\mathbf{a}\|}$

For a vector $\mathbf{x} \in \mathbb{R}^m$ and the constant vector $\mathbf{a} \in \mathbb{R}^m$, we want to compute the derivative of the scalar-valued L2 norm of the difference between these vectors $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{a}\|$ with respect to $\mathbf{x}$. Using the chain rule, we can represent the Jacobian as

$$\mathbf{J}_{\mathbf{x}}^{\|\mathbf{x}-\mathbf{a}\|} = \mathbf{J}_{\mathbf{x}-\mathbf{a}}^{\|\mathbf{x}-\mathbf{a}\|} \mathbf{J}_{\mathbf{x}}^{\mathbf{x}-\mathbf{a}}. \tag{5.9}$$

We can compute the first Jacobian by using the chain rule again:

$$\mathbf{J}_{\mathbf{x}}^{\|\mathbf{x}\|} = \mathbf{J}_{\mathbf{x}^\top \mathbf{x}}^{\sqrt{\mathbf{x}^\top \mathbf{x}}} \mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{x}}. \tag{5.10}$$

We can here recognise that $\mathbf{J}_{\mathbf{x}^\top \mathbf{x}}^{\sqrt{\mathbf{x}^\top \mathbf{x}}}$ is simply $\frac{d\sqrt{u}}{du}$ with $u = \mathbf{x}^\top \mathbf{x}$, and $\mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{x}}$ is a special case of $\mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{A}\mathbf{x}}$ from (5.6) with $\mathbf{A} = \mathbf{I}$. This gives us

$$\mathbf{J}_{\mathbf{x}}^{\|\mathbf{x}\|} = \mathbf{J}_{\mathbf{x}^\top \mathbf{x}}^{\sqrt{\mathbf{x}^\top \mathbf{x}}} \mathbf{J}_{\mathbf{x}}^{\mathbf{x}^\top \mathbf{x}} \tag{5.11a}$$

$$= \frac{1}{2\sqrt{\mathbf{x}^\top \mathbf{x}}} \mathbf{x}^\top (\mathbf{I} + \mathbf{I}^\top) \tag{5.11b}$$

$$= \frac{2\mathbf{x}^\top}{2\,\|\mathbf{x}\|} \tag{5.11c}$$

$$= \frac{\mathbf{x}^\top}{\|\mathbf{x}\|}. \tag{5.11d}$$

By applying (5.3) on the second Jacobian in (5.9), we see that

$$\mathbf{J}_{\mathbf{x}}^{\mathbf{x}-\mathbf{a}} = \mathbf{I}. \tag{5.12}$$

We can now finally compose the Jacobian blocks (5.11) and (5.12) in (5.9) to compute the desired Jacobian

$$\mathbf{J}_{\mathbf{x}}^{\|\mathbf{x}-\mathbf{a}\|} = \mathbf{J}_{\mathbf{x}-\mathbf{a}}^{\|\mathbf{x}-\mathbf{a}\|}\mathbf{J}_{\mathbf{x}}^{\mathbf{x}-\mathbf{a}} \tag{5.13a}$$

$$= \frac{(\mathbf{x}-\mathbf{a})^\top}{\|\mathbf{x}-\mathbf{a}\|}\,\mathbf{I} \tag{5.13b}$$

$$= \frac{(\mathbf{x}-\mathbf{a})^\top}{\|\mathbf{x}-\mathbf{a}\|}. \tag{5.13c}$$

For small values of $\mathbf{h}$, we have the linear first-order Taylor approximation

$$f(\mathbf{x}+\mathbf{h}) \xrightarrow[\mathbf{h}\to 0]{} f(\mathbf{x}) + \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\mathbf{h}. \tag{5.14}$$

This concludes our brief introduction to derivatives in vector spaces. Those seeking further details should consult an appropriate text on multivariate calculus. We will finish this section by emphasising that the procedure in (5.3) is not applicable for 3D rotations and poses, which are defined on manifolds rather than on vector spaces. The next section will describe how we apply the Lie theory from Chapter 4 to differentiate functions on these manifolds.

5.2 Derivatives on Lie groups

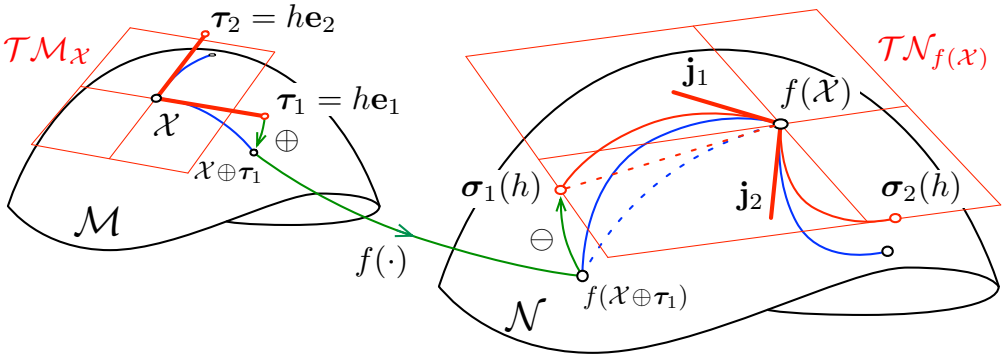


Figure 5.1: The right derivative of a function $f : \mathcal{M} \to \mathcal{N}$. The perturbation vectors in the canonical directions, $\boldsymbol{\tau}_i = h\mathbf{e}_i \in \mathcal{T}\mathcal{M}_\mathcal{X}$ are propagated to perturbation vectors $\boldsymbol{\sigma}_i \in \mathcal{T}\mathcal{N}_{f(\mathcal{X})}$ through $\boldsymbol{\sigma}_i(h) = f(\mathcal{X} \oplus h\mathbf{e}_i) \ominus f(\mathcal{X})$. The column vectors $\mathbf{j}_i$ of $\mathbf{J}$ are the derivatives $\mathbf{j}_i = \lim_{h\to 0} \boldsymbol{\sigma}_i(h)/h$, and the Jacobian matrix $\mathbf{J} = [\mathbf{j}_1 \cdots \mathbf{j}_m]$ linearly maps tangent space vectors from $\mathcal{T}\mathcal{M}_\mathcal{X}$ to $\mathcal{T}\mathcal{N}_{f(\mathcal{X})}$.
(Image source: [26]; licensed under CC BY-NC-SA 4.0)

For functions acting on Lie groups, we can express the derivative similarly to that of (5.3) by using the $\oplus$ and $\ominus$ operators from Section 4.5 with infinitesimal perturbations in the tangent space. For a function $f : \mathcal{M} \to \mathcal{N}$, the right (local)

derivative is given by

$$\frac{{}^{\mathcal{X}}\partial f(\mathcal{X})}{\partial \mathcal{X}} \triangleq \lim_{\boldsymbol{\tau}\to 0} \frac{f(\mathcal{X}\oplus\boldsymbol{\tau})\ominus f(\mathcal{X})}{\boldsymbol{\tau}} \in \mathbb{R}^{n\times m} \tag{5.15}$$

$$= \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Log}\left(f(\mathcal{X})^{-1}\circ f(\mathcal{X}\circ \mathrm{Exp}(\boldsymbol{\tau}))\right)}{\boldsymbol{\tau}}, \tag{5.16}$$

where $\boldsymbol{\tau}\in\mathcal{TM}_{\mathcal{X}}$.

Variations in $\mathcal{X}$ and $f(\mathcal{X})$ are here expressed in terms of vectors in the local tangent spaces at $\mathcal{X}\in\mathcal{M}$ and $f(\mathcal{X})\in\mathcal{N}$ respectively. The tangent space vector

$$\boldsymbol{\sigma}_i(h) = f(\mathcal{X}\oplus h\mathbf{e}_i)\ominus f(\mathcal{X}) \quad \in \mathcal{TN}_{f(\mathcal{X})} \tag{5.17}$$

is the variation of $f(\mathcal{X})$ when $\mathcal{X}$ is perturbed in the direction of $\mathbf{e}_i$. The column vectors $\mathbf{j}_i$ of the Jacobian are the derivatives $\mathbf{j}_i = \lim_{h\to 0}\boldsymbol{\sigma}_i(h)/h$, and $\mathbf{J} = [\mathbf{j}_1\cdots\mathbf{j}_m]$ is a proper Jacobian matrix in $\mathbb{R}^{n\times m}$, linearly mapping the local tangent spaces $\mathcal{TM}_{\mathcal{X}}\to\mathcal{TN}_{f(\mathcal{X})}$ (Figure 5.1).

As in the previous section, we can use (5.15) to compute the Jacobians by applying the same procedure as in (5.4).

Example 5.13 Computing the Jacobian $\mathbf{J}_{\mathcal{Y}}^{\mathcal{X}\circ\mathcal{Y}}$ on the manifold

We can compute the Jacobian of the composition $\mathcal{X}\circ\mathcal{Y}$ between two elements $\mathcal{X},\mathcal{Y}\in\mathcal{M}$ with respect to $\mathcal{Y}$, by using (5.15) with the procedure in (5.4).

$$\mathbf{J}_{\mathcal{Y}}^{\mathcal{X}\circ\mathcal{Y}} = \lim_{\boldsymbol{\tau}\to 0} \frac{(\mathcal{X}\circ(\mathcal{Y}\oplus\boldsymbol{\tau}))\ominus(\mathcal{X}\circ\mathcal{Y})}{\boldsymbol{\tau}} \tag{5.18a}$$

$$= \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Log}\left((\mathcal{X}\circ\mathcal{Y})^{-1}\circ(\mathcal{X}\circ(\mathcal{Y}\circ\mathrm{Exp}(\boldsymbol{\tau})))\right)}{\boldsymbol{\tau}} \tag{5.18b}$$

$$= \lim_{\boldsymbol{\tau}\to 0} \frac{\mathrm{Log}\left((\mathcal{X}\circ\mathcal{Y})^{-1}(\mathcal{X}\circ\mathcal{Y})\circ\mathrm{Exp}(\boldsymbol{\tau})\right)}{\boldsymbol{\tau}} \tag{5.18c}$$

$$= \lim_{\boldsymbol{\tau}\to 0} \frac{\boldsymbol{\tau}}{\boldsymbol{\tau}} \tag{5.18d}$$

$$= \mathbf{I}. \tag{5.18e}$$

For small values of $\boldsymbol{\tau}$, the following first order-Taylor approximation holds,

$$f(\mathcal{X}\oplus{}^{\mathcal{X}}\boldsymbol{\tau}) \xrightarrow[{}^{\mathcal{X}}\boldsymbol{\tau}\to 0]{} f(\mathcal{X}) \oplus \frac{{}^{\mathcal{X}}\partial f(\mathcal{X})}{\partial\mathcal{X}}\,{}^{\mathcal{X}}\boldsymbol{\tau} \quad \in\mathcal{N}, \tag{5.19}$$

and can be used to linearise the function $f(\mathcal{X})$ on the manifold, similarly to (5.14).

We can apply left perturbations to compute the corresponding left (global) derivative [26]. The left derivative is also related to the right (local) derivative by the adjoints of $\mathcal{M}$ and $\mathcal{N}$

$$\frac{{}^{\mathcal{E}}\partial f(\mathcal{X})}{\partial\mathcal{X}}\mathbf{Ad}_{\mathcal{X}} = \mathbf{Ad}_{f(\mathcal{X})}\frac{{}^{\mathcal{X}}\partial f(\mathcal{X})}{\partial\mathcal{X}} \tag{5.20}$$

so that

$$\frac{^{\mathcal{E}}\partial f(\mathcal{X})}{\partial \mathcal{X}} = \mathbf{Ad}_{f(\mathcal{X})} \frac{^{\mathcal{X}}\partial f(\mathcal{X})}{\partial \mathcal{X}} \mathbf{Ad}_{\mathcal{X}}{}^{-1}. \tag{5.21}$$

It can be shown that the chain rule also applies to derivatives on the manifold (as long as we do not mix right and left derivatives) [26]. This means that for $\mathcal{Y} = f(\mathcal{X})$ and $\mathcal{Z} = g(\mathcal{Y})$, we have $\mathcal{Z} = g(f(\mathcal{X}))$ and

$$\frac{\partial \mathcal{Z}}{\partial \mathcal{X}} = \frac{\partial \mathcal{Z}}{\partial \mathcal{Y}} \frac{\partial \mathcal{Y}}{\partial \mathcal{X}} \qquad \text{or} \qquad \mathbf{J}^{\mathcal{Z}}_{\mathcal{X}} = \mathbf{J}^{\mathcal{Z}}_{\mathcal{Y}} \mathbf{J}^{\mathcal{Y}}_{\mathcal{X}}. \tag{5.22}$$

5.3 Elementary Lie group Jacobian blocks

With the derivative properly defined, we are able to compute the Jacobians for the common operations on Lie groups that are listed here. We will use these elementary Jacobians to define the corresponding Jacobians for $SO(3)$ and $SE(3)$ in Section 5.4 and Section 5.5. With the chain rule, these Jacobians are the elementary building blocks for expressing the derivative of functions operating on orientations and poses.

5.3.1 Jacobian of the inverse operation

The Jacobian of the inverse operation is given by

$$\mathbf{J}^{\mathcal{X}^{-1}}_{\mathcal{X}} = -\mathbf{Ad}_{\mathcal{X}}. \tag{5.23}$$

5.3.2 Jacobians of the composition operation

For $\mathcal{X}, \mathcal{Y} \in \mathcal{M}$, the Jacobians of the composition operation is given by

$$\mathbf{J}^{\mathcal{X}\circ\mathcal{Y}}_{\mathcal{X}} = \mathbf{Ad}_{\mathcal{Y}}{}^{-1} \tag{5.24}$$

$$\mathbf{J}^{\mathcal{X}\circ\mathcal{Y}}_{\mathcal{Y}} = \mathbf{I}. \tag{5.25}$$

5.3.3 Jacobians of the group action

For $\mathcal{X} \in \mathcal{M}$ and $v \in \mathcal{V}$, the Jacobians of the group action $\mathcal{X} \cdot v$ is defined as

$$\mathbf{J}^{\mathcal{X}\cdot v}_{\mathcal{X}} \triangleq \frac{^{\mathcal{X}}\partial(\mathcal{X} \cdot v)}{\partial \mathcal{X}} \tag{5.26}$$

$$\mathbf{J}^{\mathcal{X}\cdot v}_{v} \triangleq \frac{^{v}\partial(\mathcal{X} \cdot v)}{\partial v}. \tag{5.27}$$

Since the group action depends on $\mathcal{V}$, these expressions cannot be generalised. The following examples show how to compute these Jacobians for $SO(3)$ and $SE(3)$.

Example 5.14 Computing the Jacobians for $\mathbf{R} \cdot \mathbf{x}$

For a rotation matrix $\mathbf{R} \in SO(3)$ and a vector $\mathbf{x} \in \mathbb{R}^3$, the group action of $\mathbf{R}$ on $\mathbf{x}$ is defined in (4.36) as $\mathbf{R} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x}$.

The Jacobian of $\mathbf{R} \cdot \mathbf{x}$ with respect to $\mathbf{R}$ is given by

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{R}\cdot\mathbf{x}} = \lim_{\boldsymbol{\theta}\to 0} \frac{(\mathbf{R} \oplus \boldsymbol{\theta}) \cdot \mathbf{x} - (\mathbf{R} \cdot \mathbf{x})}{\boldsymbol{\theta}} \tag{5.28a}$$

$$= \lim_{\boldsymbol{\theta}\to 0} \frac{(\mathbf{R} \circ \operatorname{Exp}(\boldsymbol{\theta})) \cdot \mathbf{x} - (\mathbf{R} \cdot \mathbf{x})}{\boldsymbol{\theta}}. \tag{5.28b}$$

Since we are interested in the limit where $\boldsymbol{\theta} \to 0$, we can use the approximation $\operatorname{Exp}(\boldsymbol{\theta}) \approx \mathbf{I} + \boldsymbol{\theta}^\wedge$ from (4.44). By also switching to matrix operations, we get

$$= \lim_{\boldsymbol{\theta}\to 0} \frac{\mathbf{R}(\mathbf{I} + [\boldsymbol{\theta}]_\times)\mathbf{x} - \mathbf{R}\mathbf{x}}{\boldsymbol{\theta}} \tag{5.28c}$$

$$= \lim_{\boldsymbol{\theta}\to 0} \frac{\mathbf{R}\mathbf{x} + \mathbf{R}[\boldsymbol{\theta}]_\times\mathbf{x} - \mathbf{R}\mathbf{x}}{\boldsymbol{\theta}} \tag{5.28d}$$

$$= \lim_{\boldsymbol{\theta}\to 0} \frac{\mathbf{R}[\boldsymbol{\theta}]_\times\mathbf{x}}{\boldsymbol{\theta}}. \tag{5.28e}$$

With the property $[\mathbf{a}]_\times\mathbf{b} = -[\mathbf{b}]_\times\mathbf{a}$, we get the desired form

$$= \lim_{\boldsymbol{\theta}\to 0} \frac{-\mathbf{R}[\mathbf{x}]_\times\boldsymbol{\theta}}{\boldsymbol{\theta}}, \tag{5.28f}$$

which results in the Jacobian

$$= -\mathbf{R}[\mathbf{x}]_\times. \tag{5.28g}$$

The Jacobian of $\mathbf{R} \cdot \mathbf{x}$ with respect to $\mathbf{x}$ is given by

$$\mathbf{J}_{\mathbf{x}}^{\mathbf{R}\cdot\mathbf{x}} = \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{R} \cdot (\mathbf{x} + \partial\mathbf{x}) - (\mathbf{R} \cdot \mathbf{x})}{\partial\mathbf{x}} \tag{5.29a}$$

$$= \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{R}\mathbf{x} + \mathbf{R}\partial\mathbf{x} - \mathbf{R}\mathbf{x}}{\partial\mathbf{x}} \tag{5.29b}$$

$$= \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{R}\partial\mathbf{x}}{\partial\mathbf{x}} \tag{5.29c}$$

$$= \mathbf{R}. \tag{5.29d}$$

Example 5.15 Computing the Jacobians for $\mathbf{T} \cdot \mathbf{x}$

For a pose $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{smallmatrix}\right] \in SE(3)$ and a vector $\mathbf{x} \in \mathbb{R}^3$, the group action of $\mathbf{T}$ on $\mathbf{x}$ is defined in (4.49) as $\mathbf{T} \cdot \mathbf{x} \triangleq \mathbf{R}\mathbf{x} + \mathbf{t}$.

The Jacobian of $\mathbf{T} \cdot \mathbf{x}$ with respect to $\mathbf{T}$ is given by

$$\mathbf{J}_{\mathbf{T}}^{\mathbf{T}\cdot\mathbf{x}} = \lim_{\boldsymbol{\xi}\to 0} \frac{(\mathbf{T} \oplus \boldsymbol{\xi}) \cdot \mathbf{x} - (\mathbf{T} \cdot \mathbf{x})}{\boldsymbol{\xi}} \tag{5.30a}$$

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{(\mathbf{T} \circ \operatorname{Exp}(\boldsymbol{\xi})) \cdot \mathbf{x} - (\mathbf{T} \cdot \mathbf{x})}{\boldsymbol{\xi}}. \tag{5.30b}$$

Since we are interested in the limit where $\boldsymbol{\xi} \to 0$, we can use the approximation $\operatorname{Exp}(\boldsymbol{\xi}) \approx \mathbf{I} + \boldsymbol{\xi}^\wedge$ from (4.58).

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\mathbf{T} \circ (\mathbf{I} + \boldsymbol{\xi}^\wedge) \cdot \mathbf{x} - (\mathbf{T} \cdot \mathbf{x})}{\boldsymbol{\xi}} \tag{5.30c}$$

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\mathbf{T} \circ \begin{bmatrix} \mathbf{I} + [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 1 \end{bmatrix} \cdot \mathbf{x} - (\mathbf{T} \cdot \mathbf{x})}{\boldsymbol{\xi}}. \tag{5.30d}$$

By performing the composition as matrix multiplication,

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\begin{bmatrix} \mathbf{R}(\mathbf{I} + [\boldsymbol{\theta}]_\times) & \mathbf{R}\boldsymbol{\rho} + \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \cdot \mathbf{x} - (\mathbf{T} \cdot \mathbf{x})}{\boldsymbol{\xi}}, \tag{5.30e}$$

and performing the group action according to (4.49),

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\mathbf{R}(\mathbf{I} + [\boldsymbol{\theta}]_\times)\mathbf{x} + \mathbf{R}\boldsymbol{\rho} + \mathbf{t} - (\mathbf{R}\mathbf{x} + \mathbf{t})}{\boldsymbol{\xi}}, \tag{5.30f}$$

we get

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\mathbf{R}\mathbf{x} + \mathbf{R}[\boldsymbol{\theta}]_\times\mathbf{x} + \mathbf{R}\boldsymbol{\rho} + \mathbf{t} - \mathbf{R}\mathbf{x} - \mathbf{t}}{\boldsymbol{\xi}} \tag{5.30g}$$

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\mathbf{R}\boldsymbol{\rho} + \mathbf{R}[\boldsymbol{\theta}]_\times\mathbf{x}}{\boldsymbol{\xi}}. \tag{5.30h}$$

With the property $[\mathbf{a}]_\times\mathbf{b} = -[\mathbf{b}]_\times\mathbf{a}$, we get the desired form

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\mathbf{R}\boldsymbol{\rho} - \mathbf{R}[\mathbf{x}]_\times\boldsymbol{\theta}}{\boldsymbol{\xi}} \tag{5.30i}$$

$$= \lim_{\boldsymbol{\xi}\to 0} \frac{\begin{bmatrix} \mathbf{R} & -\mathbf{R}[\mathbf{x}]_\times \end{bmatrix} \boldsymbol{\xi}}{\boldsymbol{\xi}}, \tag{5.30j}$$

which results in the Jacobian

$$= \begin{bmatrix} \mathbf{R} & -\mathbf{R}[\mathbf{x}]_\times \end{bmatrix}. \tag{5.30k}$$

The Jacobian of $\mathbf{T} \cdot \mathbf{x}$ with respect to $\mathbf{x}$ is given by

$$\begin{aligned}
\mathbf{J}_{\mathbf{x}}^{\mathbf{T}\cdot\mathbf{x}} &= \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{T}\cdot(\mathbf{x}+\partial\mathbf{x}) - (\mathbf{T}\cdot\mathbf{x})}{\partial\mathbf{x}} && \text{(5.31a)}\\
&= \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{R}(\mathbf{x}+\partial\mathbf{x}) + \mathbf{t} - (\mathbf{R}\mathbf{x}+\mathbf{t})}{\partial\mathbf{x}} && \text{(5.31b)}\\
&= \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{R}\mathbf{x} + \mathbf{R}\partial\mathbf{x} + \mathbf{t} - \mathbf{R}\mathbf{x} - \mathbf{t}}{\partial\mathbf{x}} && \text{(5.31c)}\\
&= \lim_{\partial\mathbf{x}\to 0} \frac{\mathbf{R}\partial\mathbf{x}}{\partial\mathbf{x}} && \text{(5.31d)}\\
&= \mathbf{R}. && \text{(5.31e)}
\end{aligned}$$

5.3.4 Jacobians of $\mathcal{M}$

The right Jacobian of $\mathcal{M}$ is defined as the right derivative of $\mathcal{X} = \operatorname{Exp}(\boldsymbol{\tau})$

$$\mathbf{J}_r(\boldsymbol{\tau}) \triangleq \frac{{}^{\mathcal{X}}\partial \operatorname{Exp}(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}} \in \mathbb{R}^{m\times m}, \tag{5.32}$$

where $\boldsymbol{\tau} \in \mathbb{R}^m$.

The right Jacobian maps variations of the vector $\boldsymbol{\tau}$ into variations in the *local* tangent space at $\mathcal{X} = \operatorname{Exp}(\boldsymbol{\tau})$. For small $\delta\boldsymbol{\tau}$, the following approximations hold

$$\operatorname{Exp}(\boldsymbol{\tau}+\delta\boldsymbol{\tau}) \approx \operatorname{Exp}(\boldsymbol{\tau})\operatorname{Exp}(\mathbf{J}_r(\boldsymbol{\tau})\delta\boldsymbol{\tau}) \tag{5.33}$$

$$\operatorname{Exp}(\boldsymbol{\tau})\operatorname{Exp}(\delta\boldsymbol{\tau}) \approx \operatorname{Exp}(\boldsymbol{\tau} + \mathbf{J}_r{}^{-1}(\boldsymbol{\tau})\delta\boldsymbol{\tau}) \tag{5.34}$$

$$\operatorname{Log}(\operatorname{Exp}(\boldsymbol{\tau})\operatorname{Exp}(\delta\boldsymbol{\tau})) \approx \boldsymbol{\tau} + \mathbf{J}_r{}^{-1}(\boldsymbol{\tau})\delta\boldsymbol{\tau}. \tag{5.35}$$

For $\boldsymbol{\tau} = \operatorname{Log}(\mathcal{X})$, we have from (5.35) that the Jacobian of the Log operation is given by

$$\mathbf{J}_{\mathcal{X}}^{\operatorname{Log}(\mathcal{X})} \triangleq \frac{{}^{\mathcal{X}}\partial \operatorname{Log}(\mathcal{X})}{\partial \mathcal{X}} = \mathbf{J}_r{}^{-1}(\boldsymbol{\tau}). \tag{5.36}$$

The corresponding left Jacobian

$$\mathbf{J}_l(\boldsymbol{\tau}) \triangleq \frac{{}^{\mathcal{E}}\partial \operatorname{Exp}(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}} \in \mathbb{R}^{m\times m}, \tag{5.37}$$

maps variation of $\boldsymbol{\tau}$ into variations in the *global* tangent space. We can relate the left and right Jacobians with the adjoint matrix:

$$\mathbf{Ad}_{\operatorname{Exp}(\boldsymbol{\tau})} = \mathbf{J}_l(\boldsymbol{\tau})\mathbf{J}_r{}^{-1}(\boldsymbol{\tau}). \tag{5.38}$$

Also, as shown in Example 5.16 below, applying the chain rule to $\mathbf{J}_r(-\boldsymbol{\tau})$ gives us

$$\mathbf{J}_r(-\boldsymbol{\tau}) = \mathbf{J}_l(\boldsymbol{\tau}). \tag{5.39}$$

Example 5.16 Applying the chain rule to $\mathbf{J}_r(-\boldsymbol{\tau})$

The definition of the right Jacobian (5.32) gives us

$$\mathbf{J}_r(-\boldsymbol{\tau}) = \mathbf{J}_{-\boldsymbol{\tau}}^{\mathrm{Exp}(-\boldsymbol{\tau})}. \tag{5.40a}$$

Since $\mathrm{Exp}(-\boldsymbol{\tau}) = \mathrm{Exp}(\boldsymbol{\tau})^{-1}$, we can express this as

$$= \mathbf{J}_{-\boldsymbol{\tau}}^{\mathrm{Exp}(\boldsymbol{\tau})^{-1}}. \tag{5.40b}$$

By applying the chain rule and using (5.38) and (5.23), we get

$$= \mathbf{J}_{\mathrm{Exp}(\boldsymbol{\tau})}^{\mathrm{Exp}(\boldsymbol{\tau})^{-1}} \mathbf{J}_{\boldsymbol{\tau}}^{\mathrm{Exp}(\boldsymbol{\tau})} \mathbf{J}_{-\boldsymbol{\tau}}^{\boldsymbol{\tau}} \tag{5.40c}$$

$$= -\mathbf{Ad}_{\mathrm{Exp}(\boldsymbol{\tau})} \mathbf{J}_r(\boldsymbol{\tau})(-\mathbf{I}) \tag{5.40d}$$

$$= \mathbf{J}_l(\boldsymbol{\tau}). \tag{5.40e}$$

5.3.5 Jacobians of the plus and minus operators

The Jacobians of the plus and minus operators can be derived by applying the chain rule to the definitions in (4.16) and (4.17). We have

$$\mathbf{J}_{\mathcal{X}}^{\mathcal{X}\oplus\boldsymbol{\tau}} = \mathbf{Ad}_{\mathrm{Exp}(\boldsymbol{\tau})}{}^{-1} \tag{5.41}$$

$$\mathbf{J}_{\boldsymbol{\tau}}^{\mathcal{X}\oplus\boldsymbol{\tau}} = \mathbf{J}_r(\boldsymbol{\tau}), \tag{5.42}$$

and with $\boldsymbol{\tau} = \mathcal{Y} \ominus \mathcal{X}$,

$$\mathbf{J}_{\mathcal{X}}^{\mathcal{Y}\ominus\mathcal{X}} = -\mathbf{J}_l{}^{-1}(\boldsymbol{\tau}) \tag{5.43}$$

$$\mathbf{J}_{\mathcal{Y}}^{\mathcal{Y}\ominus\mathcal{X}} = \mathbf{J}_r{}^{-1}(\boldsymbol{\tau}). \tag{5.44}$$

5.4 Jacobian blocks for $SO(3)$

The right and left Jacobians, and their inverses, have the following closed form expressions:

$$\mathbf{J}_r(\boldsymbol{\theta}) = \mathbf{I} - \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2 \tag{5.45}$$

$$\mathbf{J}_r{}^{-1}(\boldsymbol{\theta}) = \mathbf{I} + \frac{1}{2}[\boldsymbol{\theta}]_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)[\boldsymbol{\theta}]_\times^2 \tag{5.46}$$

$$\mathbf{J}_l(\boldsymbol{\theta}) = \mathbf{I} + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\theta}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\theta}]_\times^2 \tag{5.47}$$

$$\mathbf{J}_l{}^{-1}(\boldsymbol{\theta}) = \mathbf{I} - \frac{1}{2}[\boldsymbol{\theta}]_\times + \left(\frac{1}{\theta^2} - \frac{1+\cos\theta}{2\theta\sin\theta}\right)[\boldsymbol{\theta}]_\times^2. \tag{5.48}$$

Notice that

$$\mathbf{J}_l(\boldsymbol{\theta}) = \mathbf{J}_r(\boldsymbol{\theta})^\top, \qquad \mathbf{J}_l(\boldsymbol{\theta})^{-1} = \mathbf{J}_r(\boldsymbol{\theta})^{-\top}. \tag{5.49}$$

In the remainder of this section, we list the Jacobian blocks in Section 5.3 developed for $SO(3)$.

5.4.1 Jacobian of the inverse operation

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{R}^{-1}} = -\mathbf{Ad}_{\mathbf{R}} = -\mathbf{R}. \tag{5.50}$$

5.4.2 Jacobians of the composition operation

$$\mathbf{J}_{\mathbf{R}_a}^{\mathbf{R}_a\mathbf{R}_b} = \mathbf{Ad}_{\mathbf{R}_b}{}^{-1} = \mathbf{R}_b^{\top} \tag{5.51}$$
$$\mathbf{J}_{\mathbf{R}_b}^{\mathbf{R}_a\mathbf{R}_b} = \mathbf{I}. \tag{5.52}$$

5.4.3 Jacobians of the group action

From Example 5.14 we have

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{R}\cdot\mathbf{x}} = -\mathbf{R}[\mathbf{x}]_{\times} \tag{5.53}$$
$$\mathbf{J}_{\mathbf{x}}^{\mathbf{R}\cdot\mathbf{x}} = \mathbf{R}. \tag{5.54}$$

5.4.4 Jacobians of the plus and minus operators

$$\mathbf{J}_{\mathbf{R}}^{\mathbf{R}\oplus\boldsymbol{\theta}} = \mathbf{Ad}_{\mathrm{Exp}(\boldsymbol{\theta})}{}^{-1} = \mathbf{R}(\boldsymbol{\theta})^{\top} \tag{5.55}$$
$$\mathbf{J}_{\boldsymbol{\theta}}^{\mathbf{R}\oplus\boldsymbol{\theta}} = \mathbf{J}_r(\boldsymbol{\theta}). \tag{5.56}$$

For $\boldsymbol{\theta} = \mathbf{R}_b \ominus \mathbf{R}_a$, we have

$$\mathbf{J}_{\mathbf{R}_a}^{\mathbf{R}_b\ominus\mathbf{R}_a} = -\mathbf{J}_l{}^{-1}(\boldsymbol{\theta}) \tag{5.57}$$
$$\mathbf{J}_{\mathbf{R}_b}^{\mathbf{R}_b\ominus\mathbf{R}_a} = \mathbf{J}_r{}^{-1}(\boldsymbol{\theta}). \tag{5.58}$$

5.5 Jacobian blocks for $SE(3)$

The left Jacobian and its inverse have the following closed form expressions:

$$\mathbf{J}_l(\boldsymbol{\xi}) = \begin{bmatrix} \mathbf{J}_l(\boldsymbol{\theta}) & \mathbf{Q}(\boldsymbol{\xi}) \\ \mathbf{0} & \mathbf{J}_l(\boldsymbol{\theta}) \end{bmatrix} \tag{5.59}$$
$$\mathbf{J}_l{}^{-1}(\boldsymbol{\xi}) = \begin{bmatrix} \mathbf{J}_l{}^{-1}(\boldsymbol{\theta}) & -\mathbf{J}_l{}^{-1}(\boldsymbol{\theta})\mathbf{Q}(\boldsymbol{\xi})\mathbf{J}_l{}^{-1}(\boldsymbol{\theta}) \\ \mathbf{0} & \mathbf{J}_l{}^{-1}(\boldsymbol{\theta}) \end{bmatrix}. \tag{5.60}$$

Here $\mathbf{J}_l(\boldsymbol{\theta})$ is the left Jacobian of $SO(3)$ in (5.47) and $\mathbf{Q}(\boldsymbol{\xi})$ is given by

$$\begin{aligned}\mathbf{Q}(\boldsymbol{\xi}) =& \frac{1}{2}[\boldsymbol{\rho}]_\times + \frac{\theta - \sin\theta}{\theta^3}([\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times + [\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times + [\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times) \\ & - \frac{1 - \frac{\theta^2}{2} - \cos\theta}{\theta^4}([\boldsymbol{\theta}]_\times^2[\boldsymbol{\rho}]_\times + [\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times^2 - 3[\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times) \\ & - \frac{1}{2}\left(\frac{1 - \frac{\theta^2}{2} - \cos\theta}{\theta^4} - 3\frac{\theta - \sin\theta - \frac{\theta^3}{6}}{\theta^5}\right)([\boldsymbol{\theta}]_\times[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times^2 + [\boldsymbol{\theta}]_\times^2[\boldsymbol{\rho}]_\times[\boldsymbol{\theta}]_\times).\end{aligned} \tag{5.61}$$

The right Jacobian and its inverse can be obtained using (5.39).

In the remainder of this section, we list the Jacobian blocks in Section 5.3 developed for $SE(3)$.

5.5.1 Jacobian of the inverse operation

$$\mathbf{J}_{\mathbf{T}}^{\mathbf{T}^{-1}} = -\mathbf{Ad}_{\mathbf{T}} = -\begin{bmatrix}\mathbf{R} & [\mathbf{t}]_\times\mathbf{R} \\ \mathbf{0} & \mathbf{R}\end{bmatrix}. \tag{5.62}$$

5.5.2 Jacobians of the composition operation

$$\mathbf{J}_{\mathbf{T}_a}^{\mathbf{T}_a\mathbf{T}_b} = \mathbf{Ad}_{\mathbf{T}_b}{}^{-1} = \begin{bmatrix}\mathbf{R}_b^\top & -\mathbf{R}_b^\top[\mathbf{t}_b]_\times \\ \mathbf{0} & \mathbf{R}_b^\top\end{bmatrix} \tag{5.63}$$

$$\mathbf{J}_{\mathbf{T}_b}^{\mathbf{T}_a\mathbf{T}_b} = \mathbf{I}. \tag{5.64}$$

5.5.3 Jacobians of the group action

From Example 5.15 we have

$$\mathbf{J}_{\mathbf{T}}^{\mathbf{T}\cdot\mathbf{x}} = \begin{bmatrix}\mathbf{R} & -\mathbf{R}[\mathbf{x}]_\times\end{bmatrix} \tag{5.65}$$

$$\mathbf{J}_{\mathbf{x}}^{\mathbf{T}\cdot\mathbf{x}} = \mathbf{R}. \tag{5.66}$$

5.5.4 Jacobians of the plus and minus operators

$$\mathbf{J}_{\mathbf{T}}^{\mathbf{T}\oplus\boldsymbol{\xi}} = \mathbf{Ad}_{\mathrm{Exp}(\boldsymbol{\xi})}{}^{-1} \tag{5.67}$$

$$\mathbf{J}_{\boldsymbol{\xi}}^{\mathbf{T}\oplus\boldsymbol{\xi}} = \mathbf{J}_r(\boldsymbol{\xi}). \tag{5.68}$$

For $\boldsymbol{\xi} = \mathbf{T}_b \ominus \mathbf{T}_a$, we have

$$\mathbf{J}_{\mathbf{T}_a}^{\mathbf{T}_b\ominus\mathbf{T}_a} = -\mathbf{J}_l{}^{-1}(\boldsymbol{\xi}) \tag{5.69}$$

$$\mathbf{J}_{\mathbf{T}_b}^{\mathbf{T}_b\ominus\mathbf{T}_a} = \mathbf{J}_r{}^{-1}(\boldsymbol{\xi}). \tag{5.70}$$

Chapter 6

Representing uncertainty

This chapter will cover how we can represent uncertainty in vectors and Lie group elements using Gaussian *probability distribution functions (PDFs)*. We will also see how we can propagate uncertainty through functions of random variables. This is a useful tool when we want to express uncertainty in different coordinate frames, or we want to propagate uncertainty through sensor models. Most of this chapter is based on [4, 26].

6.1 The multivariate normal distribution

A *multivariate normal* or *Gaussian* PDF over the *random variable* $\mathbf{x} \in \mathbb{R}^m$ can be expressed as

$$p(\mathbf{x}; \bar{\mathbf{x}}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^m \det \boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right), \tag{6.1}$$

where the *mean* $\bar{\mathbf{x}}$ is the expected value of $\mathbf{x}$,

$$\bar{\mathbf{x}} = \mathbb{E}[\mathbf{x}] \in \mathbb{R}^m, \tag{6.2}$$

and the *covariance matrix* $\boldsymbol{\Sigma}$ is defined as the expected product of deviations

$$\boldsymbol{\Sigma} = \mathbb{E}\left[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^\top\right] \in \mathbb{R}^{m \times m}, \tag{6.3}$$

where $\mathbb{E}[\cdot]$ is the *expectation operator*. We say that $\mathbf{x}$ is *normally distributed* by using the notation

$$\mathbf{x} \sim \mathcal{N}(\bar{\mathbf{x}}, \boldsymbol{\Sigma}), \tag{6.4}$$

where $\mathcal{N}(\bar{\mathbf{x}}, \boldsymbol{\Sigma})$ is shorthand for the PDF in (6.1).

The *joint Gaussian distribution* over a pair of random variables, $(\mathbf{x}, \mathbf{y})$, is given by

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\bar{\mathbf{x}} \\ \bar{\mathbf{y}}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\mathbf{xx}} & \boldsymbol{\Sigma}_{\mathbf{xy}} \\ \boldsymbol{\Sigma}_{\mathbf{yx}} & \boldsymbol{\Sigma}_{\mathbf{yy}}\end{bmatrix}\right), \tag{6.5}$$

where $\boldsymbol{\Sigma}_{\mathbf{xy}} = \mathbb{E}[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{y}-\bar{\mathbf{y}})^\top]$ and $\boldsymbol{\Sigma}_{\mathbf{yx}} = \boldsymbol{\Sigma}_{\mathbf{xy}}^\top$. The convenient block structure of the joint covariance matrix lets us easily *marginalise* the joint distribution to recover the two *marginal Gaussian distributions*

$$p(\mathbf{x}) = \mathcal{N}(\bar{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{xx}}) \tag{6.6}$$
$$p(\mathbf{y}) = \mathcal{N}(\bar{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{yy}}). \tag{6.7}$$

It is also possible to break the joint distribution into the product of the two factors

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y}), \tag{6.8}$$

where the *conditional distribution* of $\mathbf{x}$ given $\mathbf{y}$ is

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\bar{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}(\mathbf{y} - \bar{\mathbf{y}}),\ \boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}}), \tag{6.9}$$

and marginal distribution $p(\mathbf{y})$ is given in (6.7). Given a measurement $\mathbf{y}$, we can use (6.9) to update the distribution of $\mathbf{x}$ given the measurement by adjusting $\bar{\mathbf{x}}$ and reducing the uncertainty in $\boldsymbol{\Sigma}_{\mathbf{xx}}$.

6.2 Uncertainty for Lie groups

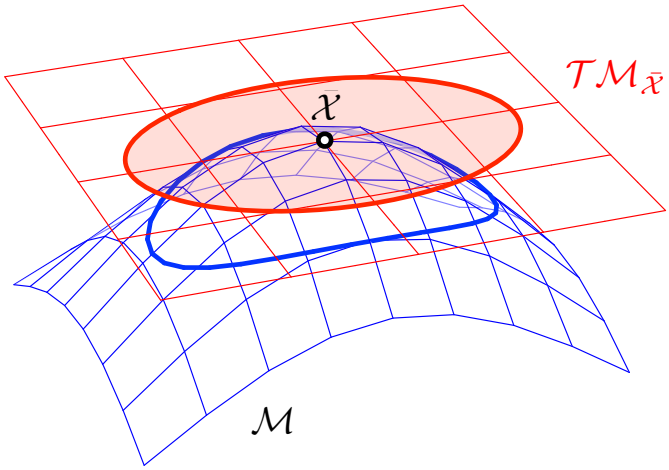


Figure 6.1: Uncertainty around an element $\bar{\mathcal{X}}$ on the manifold $\mathcal{M}$ can be expressed as a covariance on the tangent vector space at the element (red). This lets us define a random variable on the tangent space that through perturbation with $\bar{\mathcal{X}}$ induces the corresponding probability distribution on the manifold (blue).
(Image source: [26]; licensed under CC BY-NC-SA 4.0)

We can represent a random variable on the manifold as a perturbation

$$\mathcal{X} = \bar{\mathcal{X}} \oplus \boldsymbol{\tau}, \quad \boldsymbol{\tau} = \mathcal{X} \ominus \bar{\mathcal{X}}, \tag{6.10}$$

where $\bar{\mathcal{X}} \in \mathcal{M}$ is a nominal (noise-free) element, and $\boldsymbol{\tau} \in \mathbb{R}^m$ is a random variable in the tangent space $\mathcal{T}\mathcal{M}_{\bar{\mathcal{X}}}$. We can in this way define a PDF over $\boldsymbol{\tau}$ to induce a PDF on $\mathcal{X}$.

Covariance matrices can be properly defined on $\mathcal{TM}_{\bar{\mathcal{X}}}$ with the expectation operator $\mathbb{E}[\cdot]$ as

$$\boldsymbol{\Sigma}_{\mathcal{X}} = \mathbb{E}[\boldsymbol{\tau}\boldsymbol{\tau}^\top] = \mathbb{E}[(\mathcal{X} \ominus \bar{\mathcal{X}})(\mathcal{X} \ominus \bar{\mathcal{X}})^\top] \in \mathbb{R}^{m \times m}. \tag{6.11}$$

This lets us define the Gaussian PDF for the tangent space vector

$$\boldsymbol{\tau} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathcal{X}}), \tag{6.12}$$

and the induced Gaussian PDF on the manifold (Figure 6.1)

$$\mathcal{X} \sim \mathcal{N}(\bar{\mathcal{X}}, \boldsymbol{\Sigma}_{\mathcal{X}}). \tag{6.13}$$

Example 6.17 Drawing random poses from a Gaussian distribution

Using the framework presented above, we can draw random poses from a distribution $\mathbf{T} \sim \mathcal{N}(\bar{\mathbf{T}}, \boldsymbol{\Sigma}_{\mathbf{T}})$ by perturbing the mean pose $\bar{\mathbf{T}}$ with tangent vectors drawn from the distribution $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\mathbf{T}})$, so that $\mathbf{T} = \bar{\mathbf{T}} \oplus \boldsymbol{\xi}$.

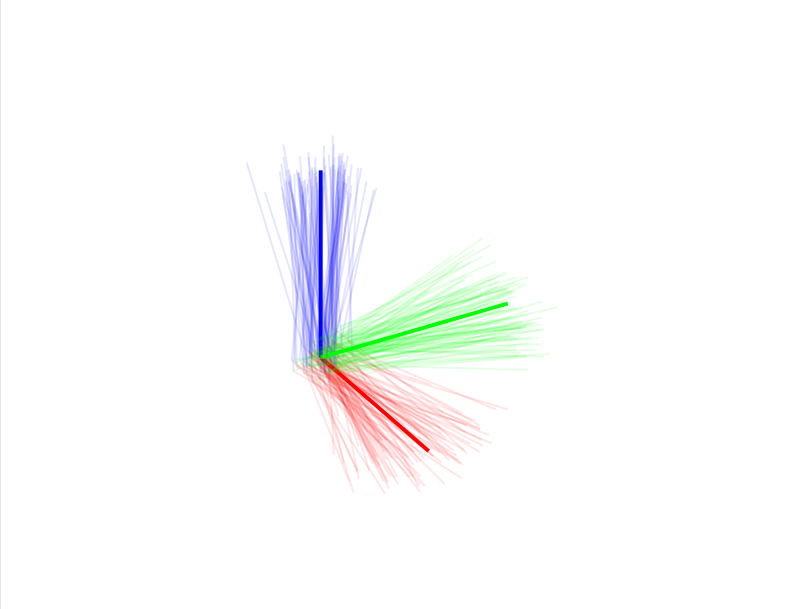


Figure 6.2: The mean pose $\bar{\mathbf{T}}$ is shown in thick axes, while 50 poses drawn randomly from $\mathcal{N}(\bar{\mathbf{T}}, \boldsymbol{\Sigma}_{\mathbf{T}})$ are shown in thin transparent axes.

Figure 6.2 shows the result of drawing 50 poses with $\bar{\mathbf{T}} = \mathbf{I}$ and

$$\boldsymbol{\Sigma}_{\mathbf{T}} = \begin{bmatrix} 0.05 & & & & & & \\ & 0.05 & & & & & \\ & & 0.05 & & & & \\ & & & 0.1 & & & \\ & & & & 0.1 & & \\ & & & & & 0.2 & \end{bmatrix}.$$

6.3 Propagation of uncertainty

Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a function that takes the random variable $\mathbf{x} \in \mathbb{R}^m$ as input and produces the random variable $\mathbf{y} = f(\mathbf{x}) \in \mathbb{R}^n$ as output. We will assume that $\mathbf{x} \sim \mathcal{N}(\bar{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})$.

Computing the uncertainty of the output $\mathbf{y}$ based on the effect of the uncertainties in the input $\mathbf{x}$ is sometimes called *propagation of uncertainty*. This is useful when we need to know the uncertainty in transformed variables, such as the uncertainty of poses in different coordinate frames, and the uncertainty in pixel position based on the uncertainty of a projected 3D point.

When $f(\mathbf{x})$ is a linear function $\mathbf{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, we can compute the probability distribution of $\mathbf{y}$ exactly. Since the expectation operator is linear, we can compute the mean of $\mathbf{y}$ from the definition in (6.2) as

$$\bar{\mathbf{y}} = \mathbb{E}[\mathbf{y}] = \mathbb{E}[\mathbf{A}\mathbf{x} + \mathbf{b}] = \mathbf{A}\mathbb{E}[\mathbf{x}] + \mathbf{b} = \mathbf{A}\bar{\mathbf{x}} + \mathbf{b} = f(\bar{\mathbf{x}}). \tag{6.14}$$

The covariance matrix of $\mathbf{y}$ is developed similarly from the definition in (6.3) by

$$\begin{aligned}
\boldsymbol{\Sigma}_{\mathbf{y}} &= \mathbb{E}[(\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^\top] && \text{(6.15a)}\\
&= \mathbb{E}\left[((\mathbf{A}\mathbf{x} + \mathbf{b}) - (\mathbf{A}\bar{\mathbf{x}} + \mathbf{b}))\left((\mathbf{A}\mathbf{x} + \mathbf{b}) - (\mathbf{A}\bar{\mathbf{x}} + \mathbf{b})\right)^\top\right] && \text{(6.15b)}\\
&= \mathbb{E}\left[(\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}))(\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}}))^\top\right] && \text{(6.15c)}\\
&= \mathbb{E}\left[\mathbf{A}(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^\top\mathbf{A}^\top\right] && \text{(6.15d)}\\
&= \mathbf{A}\mathbb{E}\left[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^\top\right]\mathbf{A}^\top && \text{(6.15e)}\\
&= \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{A}^\top. && \text{(6.15f)}
\end{aligned}$$

The resulting probability distribution for $\mathbf{y}$ when $\mathbf{y} = f(\mathbf{x})$ is linear is therefore given by

$$\mathbf{y} \sim \mathcal{N}(f(\bar{\mathbf{x}}), \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{A}^\top). \tag{6.16}$$

Example 6.18 Transforming random points with deterministic poses

We are given the random 3D point $\mathbf{x}^c \sim \mathcal{N}(\bar{\mathbf{x}}^c, \boldsymbol{\Sigma}_{\mathbf{x}^c})$ in the camera frame $\mathcal{F}_c$, and the *deterministic* pose $\mathbf{T}_{wc}$ of the camera frame relative to the world frame $\mathcal{F}_w$.

Since $\mathbf{x}^w = \mathbf{T}_{wc} \cdot \mathbf{x}^c = \mathbf{R}_{wc}\mathbf{x}^c + \mathbf{t}^w_{wc}$ is a linear function in $\mathbf{x}^c$, we can apply (6.16) to compute the exact distribution of the point in the world frame as

$$\mathbf{x}^w \sim \mathcal{N}(\mathbf{R}_{wc}\bar{\mathbf{x}}^c + \mathbf{t}^w_{wc},\ \mathbf{R}_{wc}\boldsymbol{\Sigma}_{\mathbf{x}^c}\mathbf{R}^\top_{wc}). \tag{6.17}$$

For a random Lie group variable, $\mathcal{X} \sim \mathcal{N}(\bar{\mathcal{X}}, \boldsymbol{\Sigma}_{\mathcal{X}})$, since global and local perturbations are related linearly by the adjoint matrix (4.30), we can transform the covariance matrices exactly between frames according to

$${}^{\mathcal{E}}\boldsymbol{\Sigma}_{\mathcal{X}} = \mathbf{Ad}_{\mathcal{X}}\,{}^{\mathcal{X}}\boldsymbol{\Sigma}_{\mathcal{X}}\mathbf{Ad}_{\mathcal{X}}{}^\top. \tag{6.18}$$

This allows us to express the local distribution of $\mathcal{X}$ exactly in the global frame.

For a *nonlinear* function $g : \mathbb{R}^m \to \mathbb{R}^n$, where the random variable $\mathbf{y}$ is now given by $\mathbf{y} = g(\mathbf{x})$, we will in general not get an exact Gaussian distribution over $\mathbf{y}$ when propagating the uncertainty of $\mathbf{x}$. We can instead assume that $g(\mathbf{x})$ is approximately linear around the mean of $\mathbf{x}$, so that we can use the propagation procedure above with the linear approximation.

We can linearise $g(\mathbf{x})$ with a first order Taylor expansion (5.14) at the mean $\bar{\mathbf{x}}$ to get the linear form

$$g(\mathbf{x}) = g(\bar{\mathbf{x}} + (\mathbf{x} - \bar{\mathbf{x}})) \approx g(\bar{\mathbf{x}}) + \mathbf{J}^{\mathbf{y}}_{\bar{\mathbf{x}}}(\mathbf{x} - \bar{\mathbf{x}}), \tag{6.19}$$

where the Jacobian is defined as

$$\mathbf{J}^{\mathbf{y}}_{\bar{\mathbf{x}}} \triangleq \frac{\partial g(\mathbf{x})}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\bar{\mathbf{x}}}. \tag{6.20}$$

Since $\mathbb{E}[\mathbf{x} - \bar{\mathbf{x}}] = \mathbf{0}$, developing the mean of (6.19) according to (6.14) gives us

$$\bar{\mathbf{y}} \approx g(\bar{\mathbf{x}}). \tag{6.21}$$

Developing the covariance matrix of (6.19) according to (6.15) results in

$$\mathbf{\Sigma}_{\mathbf{y}} \approx \mathbf{J}^{\mathbf{y}}_{\bar{\mathbf{x}}} \mathbf{\Sigma}_{\mathbf{x}} \mathbf{J}^{\mathbf{y}\top}_{\bar{\mathbf{x}}}. \tag{6.22}$$

We can therefore propagate the uncertainty through the nonlinear function $g(\mathbf{x})$ by approximating it locally as a linear function $g(\bar{\mathbf{x}}) + \mathbf{J}^{\mathbf{y}}_{\bar{\mathbf{x}}}(\mathbf{x} - \bar{\mathbf{x}})$ to get the approximate PDF for $\mathbf{y} = g(\mathbf{x})$ as

$$\mathbf{y} \sim \mathcal{N}(g(\bar{\mathbf{x}}), \mathbf{J}^{\mathbf{y}}_{\bar{\mathbf{x}}} \mathbf{\Sigma}_{\mathbf{x}} \mathbf{J}^{\mathbf{y}\top}_{\bar{\mathbf{x}}}). \tag{6.23}$$

It is also possible to perform uncertainty propagation using higher order approximations [4].

Example 6.19 Uncertainty in backprojection

Given a calibrated camera with intrinsic matrix

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.24}$$

we can use the inverse perspective camera model,

$$\mathbf{x}^c = \pi^{-1}(\mathbf{u}, z) = z\mathbf{K}^{-1}\breve{\mathbf{u}} = z \begin{bmatrix} \frac{u - c_u}{f_u} \\ \frac{v - c_v}{f_v} \\ 1 \end{bmatrix}, \tag{6.25}$$

to compute the 3D point $\mathbf{x}^c$ in the camera frame that corresponds to a pixel $\mathbf{u}$ with depth z. Given the pose $\mathbf{T}_{wc}$ of the camera relative to the world frame, we

get the corresponding world point from $\mathbf{x}^w = \mathbf{T}_{wc} \cdot \mathbf{x}^c$. This gives us the model

$$\mathbf{x}^w = f(\mathbf{T}_{wc}, \mathbf{u}, z) = \mathbf{T}_{wc} \cdot \pi^{-1}(\mathbf{u}, z). \tag{6.26}$$

We are given the parameters for the model as random variables

$$\mathbf{T}_{wc} \sim \mathcal{N}(\bar{\mathbf{T}}_{wc}, \mathbf{\Sigma}_{\mathbf{T}_{wc}}) \tag{6.27}$$
$$\mathbf{u} \sim \mathcal{N}(\bar{\mathbf{u}}, \mathbf{\Sigma}_{\mathbf{u}}) \tag{6.28}$$
$$z \sim \mathcal{N}(\bar{z}, \sigma_z^2). \tag{6.29}$$

We can approximate the distribution for the backprojected point as

$$\bar{\mathbf{x}}^w = f(\bar{\mathbf{T}}_{wc}, \bar{\mathbf{u}}, \bar{z}) \tag{6.30}$$
$$\mathbf{\Sigma}_{\mathbf{x}^w} = \mathbf{J}^f_{\bar{\mathbf{T}}_{wc}} \mathbf{\Sigma}_{\mathbf{T}_{wc}} \mathbf{J}^{f\top}_{\bar{\mathbf{T}}_{wc}} + \mathbf{J}^f_{\bar{\mathbf{u}}} \mathbf{\Sigma}_{\mathbf{u}} \mathbf{J}^{f\top}_{\bar{\mathbf{u}}} + \mathbf{J}^f_{\bar{z}} \sigma_z^2 \mathbf{J}^{f\top}_{\bar{z}}. \tag{6.31}$$

Using the Jacobian blocks from section 5.5, we get the Jacobians

$$\mathbf{J}^f_{\bar{\mathbf{T}}_{wc}} = \mathbf{J}^{\bar{\mathbf{T}}_{wc} \cdot \pi^{-1}}_{\bar{\mathbf{T}}_{wc}} = \mathbf{J}^{\bar{\mathbf{T}}_{wc} \cdot \mathbf{x}^c}_{\bar{\mathbf{T}}_{wc}} = \begin{bmatrix} \bar{\mathbf{R}}_{wc} & -\bar{\mathbf{R}}_{wc} [\mathbf{x}^c]_\times \end{bmatrix} \tag{6.32}$$

$$\mathbf{J}^f_{\bar{\mathbf{u}}} = \mathbf{J}^{\bar{\mathbf{T}}_{wc} \cdot \pi^{-1}}_{\pi^{-1}} \mathbf{J}^{\pi^{-1}}_{\bar{\mathbf{u}}} = \bar{\mathbf{R}}_{wc} \begin{bmatrix} \frac{\bar{z}}{f_u} & 0 \\ 0 & \frac{\bar{z}}{f_v} \\ 0 & 0 \end{bmatrix} \tag{6.33}$$

$$\mathbf{J}^f_{\bar{z}} = \mathbf{J}^{\bar{\mathbf{T}}_{wc} \cdot \pi^{-1}}_{\pi^{-1}} \mathbf{J}^{\pi^{-1}}_{\bar{z}} = \bar{\mathbf{R}}_{wc} \, \mathbf{K}^{-1} \breve{\mathbf{u}}, \tag{6.34}$$

which allows us to compute the approximation $\mathbf{x}^w \sim \mathcal{N}(\bar{\mathbf{x}}^w, \mathbf{\Sigma}_{\mathbf{x}^w})$.

Chapter 7

Nonlinear state estimation

This chapter will introduce a powerful framework for nonlinear state estimation based on nonlinear least squares. This includes a simple notation for expressing such problems, and a procedure for obtaining the MAP estimate and its uncertainty, given Gaussian measurement noise. Most of this chapter is based on [11, 4].

7.1 Linear least squares

Consider a set of n *linear* equations in m unknowns $\mathbf{x} = [x_1, \ldots, x_m]^\top$

$$e_i(\mathbf{x}) = 0, \quad i = 1, \ldots, n. \tag{7.1}$$

We can combine these equations on vector form as

$$e(\mathbf{x}) = \begin{bmatrix} e_1(\mathbf{x}) \\ \vdots \\ e_n(\mathbf{x}) \end{bmatrix} = \mathbf{0}, \tag{7.2}$$

where $e : \mathbb{R}^m \to \mathbb{R}^n$.

When $n > m$, it is typically not possible to find an exact solution to (7.2). We can instead seek a solution that minimises the sum of squares of the *residuals* $e(\mathbf{x})$,

$$f(\mathbf{x}) = e(\mathbf{x})^\top e(\mathbf{x}) = \|e(\mathbf{x})\|^2, \tag{7.3}$$

where $f(\mathbf{x})$ is often called the *objective function*. Since (7.2) is linear, we can obtain an objective function on the form

$$f(\mathbf{x}) = \|e(\mathbf{x})\|^2 = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2, \tag{7.4}$$

which means that we want to find the vector $\mathbf{x}$ so that

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2. \tag{7.5}$$

This is called a *linear least squares* problem.

A solution to (7.5) is required to have zero gradient, so that

$$\frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x}^*} = 2\mathbf{A}^\top(\mathbf{A}\mathbf{x}^* - \mathbf{b}) = \mathbf{0}. \tag{7.6}$$

This results in *the normal equations*

$$\mathbf{A}^\top\mathbf{A}\mathbf{x}^* = \mathbf{A}^\top\mathbf{b} \tag{7.7}$$

$$\mathbf{x}^* = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b}, \tag{7.8}$$

which can be solved with Cholesky- or QR factorisation[1].

7.2 Nonlinear least squares

When the equations in $e(\mathbf{x})$ are nonlinear, we have a *nonlinear least squares* problem. This results in a nonlinear objective function, which cannot be minimised directly using the normal equations in (7.7). We can instead assume that $f(\mathbf{x})$ is approximately linear locally around the current estimate. This lets us define a procedure starting from a suitable initial estimate, where we iteratively linearise the problem, solve the linearised problem using the normal equations, and update the estimate until convergence (Figure 7.1).

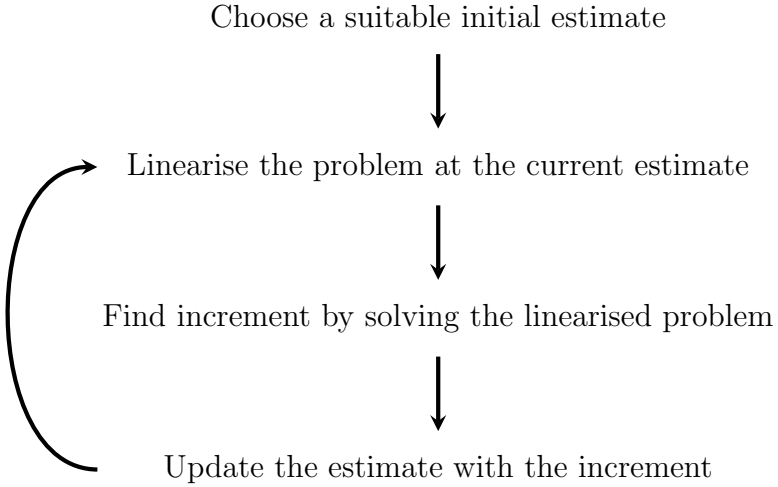


Figure 7.1: An iterative procedure for solving nonlinear least squares problems.

The vector $\mathbf{x}$ is sometimes called a *state variable*, which is typically used to the describe the physical state of an object or a system. The pose of a camera in the world is an example of such a state variable. Notice that we can concatenate several

[1] Equivalent to `x = A\b` in Matlab.

different state variables into the vector $\mathbf{x}$,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_p \end{bmatrix}, \tag{7.9}$$

and that the equations $e_i(\mathbf{x})$ can be defined to operate on one or more these p different state variables. This lets us use the least squares framework to estimate several state variables at once, such as the poses of a whole set of cameras, instead of only one.

To conveniently represent state variables over both vector spaces and manifolds, we will broaden the definition of the plus and minus operators given in Section 4.5. First, the vectors $\mathbf{x} \in \mathbb{R}^m$ is a Lie group under addition, where the exponential map is the identity [26]:

$$\text{Exp} : \mathbb{R}^m \to \mathbb{R}^m; \qquad \mathbf{x} = \text{Exp}(\mathbf{x}). \tag{7.10}$$

This means that $\mathbf{x}$ is its own tangent vector, and that $\oplus$ and $\ominus$ are in fact valid representations of vector addition and subtraction:

$$\mathbf{x}_a \oplus \mathbf{x}_b = \mathbf{x}_a + \mathbf{x}_b \tag{7.11}$$

$$\mathbf{x}_b \ominus \mathbf{x}_a = \mathbf{x}_b - \mathbf{x}_a, \tag{7.12}$$

where $\mathbf{x}_a, \mathbf{x}_b \in \mathbb{R}^m$.

We can now define $\underline{\mathcal{X}}$ as the concatenation of state variables

$$\underline{\mathcal{X}} \triangleq \begin{Bmatrix} \mathcal{X}_1 \\ \vdots \\ \mathcal{X}_p \end{Bmatrix} \in \mathcal{M}, \tag{7.13}$$

where $\mathcal{X}_i \in \mathcal{M}_i$ and $\mathcal{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_p\}$ is the *composite manifold*, which can cover both vector spaces, orientations, poses and other manifolds. We define $\underline{\boldsymbol{\tau}}$ to be the corresponding concatenation of tangent vectors

$$\underline{\boldsymbol{\tau}} \triangleq \begin{bmatrix} \boldsymbol{\tau}_1 \\ \vdots \\ \boldsymbol{\tau}_p \end{bmatrix} \in \mathbb{R}^m, \tag{7.14}$$

where $\boldsymbol{\tau}_i$ is in the tangent space of $\mathcal{M}_i$. This lets us define the plus and minus operators for the concatenated state variable $\underline{\mathcal{X}}$ as

$$\underline{\mathcal{X}} \oplus \underline{\boldsymbol{\tau}} \triangleq \begin{Bmatrix} \mathcal{X}_1 \oplus \boldsymbol{\tau}_1 \\ \vdots \\ \mathcal{X}_p \oplus \boldsymbol{\tau}_p \end{Bmatrix} \in \mathcal{M} \tag{7.15}$$

$$\underline{\mathcal{Y}} \ominus \underline{\mathcal{X}} \triangleq \begin{bmatrix} \mathcal{Y}_1 \ominus \mathcal{X}_1 \\ \vdots \\ \mathcal{Y}_p \ominus \mathcal{X}_p \end{bmatrix} \in \mathbb{R}^m. \tag{7.16}$$

If we define $\underline{\mathcal{X}}_i$ to be the concatenated set of state variables taken as input by the i-th nonlinear equation $e_i(\underline{\mathcal{X}}_i)$, we get the nonlinear objective function

$$f(\underline{\mathcal{X}}) = \|e(\underline{\mathcal{X}})\|^2 = \sum_{i=1}^{n} \|e_i(\underline{\mathcal{X}}_i)\|^2 . \tag{7.17}$$

We can linearise $e_i(\underline{\mathcal{X}}_i)$ around the current estimate $\hat{\underline{\mathcal{X}}}_i$ using a first order Taylor expansion (5.19) as

$$e_i(\underline{\mathcal{X}}_i) = e_i(\hat{\underline{\mathcal{X}}}_i \oplus \underline{\boldsymbol{\tau}}_i) \approx e_i(\hat{\underline{\mathcal{X}}}_i) + \mathbf{J}^{e_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i, \tag{7.18}$$

where $\underline{\boldsymbol{\tau}}_i = \underline{\mathcal{X}}_i \ominus \hat{\underline{\mathcal{X}}}_i$.

Applying the linearised residual functions in (7.17) allows us linearise the objective function as

$$\begin{aligned} f(\underline{\mathcal{X}}) = f(\hat{\underline{\mathcal{X}}} \oplus \underline{\boldsymbol{\tau}}) &\approx \sum_{i=1}^{n} \left\| e_i(\hat{\underline{\mathcal{X}}}_i) + \mathbf{J}^{e_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i \right\|^2 \\ &= \sum_{i=1}^{n} \left\| \mathbf{J}^{e_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i - (-e_i(\hat{\underline{\mathcal{X}}}_i)) \right\|^2 \\ &= \sum_{i=1}^{n} \|\mathbf{A}_i \underline{\boldsymbol{\tau}}_i - \mathbf{b}_i\|^2 \\ &= \|\mathbf{A}\underline{\boldsymbol{\tau}} - \mathbf{b}\|^2 , \end{aligned} \tag{7.19}$$

where $\underline{\boldsymbol{\tau}} = \underline{\mathcal{X}} \ominus \hat{\underline{\mathcal{X}}}$ is the *state update vector*, the Jacobians $\mathbf{A}_i = \mathbf{J}^{e_i}_{\hat{\underline{\mathcal{X}}}_i}$ are submatrices of the Jacobian $\mathbf{A} = \mathbf{J}^{f}_{\hat{\underline{\mathcal{X}}}}$, and the errors $\mathbf{b}_i = -e_i(\hat{\underline{\mathcal{X}}}_i)$ are the subvectors of the errors $\mathbf{b} = -e(\hat{\underline{\mathcal{X}}})$.

Since the linearised objective function in (7.19) now describes a linear least squares problem, we can compute the state update vector $\underline{\boldsymbol{\tau}}^*$ that minimises this function by solving the normal equations in (7.7). This leads to the well known Gauss-Newton algorithm, which is given i Algorithm 1.

Algorithm 1: The Gauss-Newton algorithm

Data: An objective function $f(\underline{\mathcal{X}})$ and a good initial state estimate $\hat{\underline{\mathcal{X}}}^0$
Result: An estimate for the states $\hat{\underline{\mathcal{X}}}$

for $t = 0, 1, \ldots, t^{max}$ **do**
  $\mathbf{A}, \mathbf{b} \leftarrow$ Linearise $f(\underline{\mathcal{X}})$ at $\hat{\underline{\mathcal{X}}}^t$
  $\underline{\boldsymbol{\tau}} \leftarrow$ Solve the linearised problem $\mathbf{A}^\top \mathbf{A} \underline{\boldsymbol{\tau}} = \mathbf{A}^\top \mathbf{b}$
  $\hat{\underline{\mathcal{X}}}^{t+1} \leftarrow \hat{\underline{\mathcal{X}}}^t \oplus \underline{\boldsymbol{\tau}}$

  if $f(\hat{\underline{\mathcal{X}}}^{t+1})$ *is very small or* $\hat{\underline{\mathcal{X}}}^{t+1} \approx \hat{\underline{\mathcal{X}}}^t$ **then**
    $\hat{\underline{\mathcal{X}}} \leftarrow \hat{\underline{\mathcal{X}}}^{t+1}$
    return
  end
end

Example 7.20 Estimating the mean of a set of poses

Given a set of poses $\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}$, we define an estimate for the mean pose $\bar{\mathbf{T}}$ as the pose that minimises

$$f(\bar{\mathbf{T}}) = \sum_{i=1}^{n} \left\| \mathbf{T}_i \ominus \bar{\mathbf{T}} \right\|^2. \tag{7.20}$$

Here, the state variable we wish to estimate is $\bar{\mathbf{T}}$, and the equations we wish to minimise are

$$e_i(\bar{\mathbf{T}}) = \mathbf{T}_i \ominus \bar{\mathbf{T}}, \tag{7.21}$$

which correspond to the tangent vectors at $\bar{\mathbf{T}}$, between $\bar{\mathbf{T}}$ and each of the given poses $\mathbf{T}_i$. We need to solve this iteratively, since we want to minimise $f(\bar{\mathbf{T}})$ at the correct tangent space.

We can linearise the residual functions using a first order Taylor expansion by

$$e_i(\bar{\mathbf{T}} \oplus \boldsymbol{\xi}) \approx e_i(\bar{\mathbf{T}}) + \mathbf{J}_{\bar{\mathbf{T}}}^{e_i} \boldsymbol{\xi}. \tag{7.22}$$

From (5.69) we see that the Jacobian is given by

$$\mathbf{J}_{\bar{\mathbf{T}}}^{e_i} = \mathbf{J}_{\bar{\mathbf{T}}}^{\mathbf{T}_i \ominus \bar{\mathbf{T}}} = -\mathbf{J}_l{}^{-1}(\mathbf{T}_i \ominus \bar{\mathbf{T}}), \tag{7.23}$$

where the inverse left Jacobian $\mathbf{J}_l{}^{-1}$ for $SE(3)$ is given in (5.60).

The linearised least squares problem is then given by

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}} \sum_{i=1}^{n} \left\| \mathbf{A}_i \boldsymbol{\xi} - \mathbf{b}_i \right\|^2 \tag{7.24}$$

$$= \arg\min_{\boldsymbol{\xi}} \left\| \mathbf{A}\boldsymbol{\xi} - \mathbf{b} \right\|^2, \tag{7.25}$$

where

$$\mathbf{A}_i = -\mathbf{J}_l{}^{-1}(\mathbf{T}_i \ominus \bar{\mathbf{T}}) \tag{7.26}$$

$$\mathbf{b}_i = -\mathbf{T}_i \ominus \bar{\mathbf{T}}, \tag{7.27}$$

and

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{bmatrix}. \tag{7.28}$$

This lets us solve the problem with Gauss-Newton, as shown in Algorithm 2.

Algorithm 2: Estimating the mean pose with Gauss-Newton

Data: A set of poses $\{\mathbf{T}_1, \ldots, \mathbf{T}_n\}$
Result: An estimate of the mean pose $\bar{\mathbf{T}}$

Initialise for example with $\bar{\mathbf{T}}^0 \leftarrow \mathbf{T}_1$

for $t = 0, 1, \ldots, t^{max}$ **do**
  $\mathbf{A}, \mathbf{b} \leftarrow$ Linearise at $\bar{\mathbf{T}}^t$ according to (7.26) and (7.27)
  $\boldsymbol{\xi} \leftarrow$ Solve the linearised problem $\mathbf{A}^\top \mathbf{A}\boldsymbol{\xi} = \mathbf{A}^\top \mathbf{b}$
  $\bar{\mathbf{T}}^{t+1} \leftarrow \bar{\mathbf{T}}^t \oplus \boldsymbol{\xi}$

  if $||\bar{\mathbf{T}}^t \ominus \bar{\mathbf{T}}^{t+1}||^2 < \epsilon$ **then**
    $\bar{\mathbf{T}} \leftarrow \bar{\mathbf{T}}^{t+1}$
    return
  end
end

When $\bar{\mathbf{T}}$ is close to $\mathbf{T}_i$, $-\mathbf{A}_i$ is close to the identity $\mathbf{I}$. With the approximation $-\mathbf{A}_i \approx \mathbf{I}$, the least-squares problem in (7.24) corresponds to finding the mean tangent vector $\mathbf{T}_i \ominus \bar{\mathbf{T}}$ in the tangent space of $\bar{\mathbf{T}}$ as

$$\boldsymbol{\xi}^* = \frac{1}{n} \sum_{i=1}^{n} \mathbf{T}_i \ominus \bar{\mathbf{T}}. \tag{7.29}$$

By updating the tangent space iteratively with the current hypothesis for $\bar{\mathbf{T}}$, we

get the simpler iterative procedure in Algorithm 3.

Algorithm 3: Iterative estimation of mean pose

Data: A set of poses $\mathbf{T}_1, \ldots, \mathbf{T}_n$
Result: An estimate of the mean pose $\bar{\mathbf{T}}$

Initialise for example with $\bar{\mathbf{T}}^0 \leftarrow \mathbf{T}_1$

for $t = 0, 1, \ldots, t^{max}$ **do**
  Compute the mean tangent vector in the tangent space at $\bar{\mathbf{T}}^t$
  $\bar{\boldsymbol{\xi}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{T}_i \ominus \bar{\mathbf{T}}^t$

  Update the hypothesis
  $\bar{\mathbf{T}}^{t+1} \leftarrow \bar{\mathbf{T}}^t \oplus \bar{\boldsymbol{\xi}}$

  if $||\bar{\mathbf{T}}^t \ominus \bar{\mathbf{T}}^{t+1}||^2 < \epsilon$ **then**
    $\bar{\mathbf{T}} \leftarrow \bar{\mathbf{T}}^{t+1}$
    return
  end
end

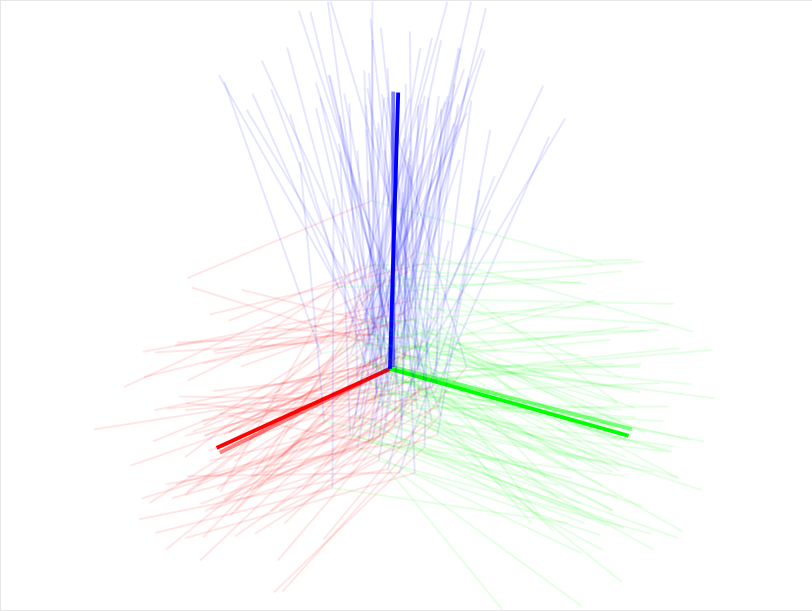


Figure 7.2: 100 randomly drawn poses are shown in thin transparent axes, while the true mean is shown in thick transparent axes. The mean estimated using Algorithm 3 is shown in thick solid axes.

The procedure in Algorithm 3 is shown to converge at least linearly if the initial estimate is close enough to the solution [2]. We can also apply the hybrid representation in Section 4.8.4 to simplify the procedure even further. Figure 7.2 shows an example of estimating the mean of a set of randomly drawn poses.

The Gauss-Newton algorithm can approach quadratic convergence rate in some situations. This is because it computes step lengths by approximating the Hessian of the objective function $f(\underline{\mathcal{X}})$ at $\underline{\hat{\mathcal{X}}}$ as

$$\begin{aligned}\frac{\partial^2 f(\underline{\hat{\mathcal{X}}})}{\partial \underline{\hat{\mathcal{X}}} \partial \underline{\hat{\mathcal{X}}}^\top} &= \left(\frac{\partial e(\underline{\hat{\mathcal{X}}})}{\partial \underline{\hat{\mathcal{X}}}}\right)^\top \left(\frac{\partial e(\underline{\hat{\mathcal{X}}})}{\partial \underline{\hat{\mathcal{X}}}}\right) + \sum_{i=1}^{n} e_i(\underline{\hat{\mathcal{X}}}_i) \frac{\partial^2 e_i(\underline{\hat{\mathcal{X}}}_i)}{\partial \underline{\hat{\mathcal{X}}}_i \partial \underline{\hat{\mathcal{X}}}_i^\top} \\ &= \mathbf{A}^\top \mathbf{A} + \mathbf{Q} \approx \mathbf{A}^\top \mathbf{A}.\end{aligned} \tag{7.30}$$

This approximation is good if we are near the solution, and the objective function is nearly quadratic. If the quadratic fit is poor, Gauss-Newton might still estimate decent update directions, but the step lengths may be bad. This may lead to new estimates that are further away from the minimum, and subsequent divergence.

Since the Gauss-Newton algorithm is not guaranteed to converge because of the Hessian approximation, it is sometimes beneficial to modify the algorithm to be more conservative towards robustness, rather than speed. One solution is to update the estimate with only a fraction α of the update vector

$$\underline{\hat{\mathcal{X}}}^{t+1} \leftarrow \underline{\hat{\mathcal{X}}}^{t} \oplus \alpha \underline{\boldsymbol{\tau}}, \tag{7.31}$$

We can find good values for α by performing a *line search* [4].

Another solution is to apply a *trust region method*, which controls in which region one is willing to trust the quadratic Hessian approximation made by Gauss-Newton. The *Levenberg-Marquardt* algorithm is an example of such a method, where the normal equations (7.7) are modified by adding a non-negative constant to the diagonal

$$\left(\mathbf{A}^\top \mathbf{A} + \lambda \operatorname{diag}(\mathbf{A}^\top \mathbf{A})\right) \underline{\boldsymbol{\tau}} = \mathbf{A}^\top \mathbf{b}. \tag{7.32}$$

When $\lambda = 0$ we obtain Gauss-Newton, while larger λ causes larger steps in the steepest descent direction when the gradient is small, and more cautious steps when the gradient is large. The complete Levenberg-Marquardt method is given in Algo-

rithm 4.

Algorithm 4: The Levenberg-Marquardt algorithm

Data: An objective function $f(\underline{\mathcal{X}})$ and a good initial state estimate $\underline{\hat{\mathcal{X}}}^0$
Result: An estimate for the states $\underline{\hat{\mathcal{X}}}$

$\lambda \leftarrow 10^{-4}$

for $t = 0, 1, \ldots, t^{max}$ **do**
  $\mathbf{A}, \mathbf{b} \leftarrow$ Linearise $f(\underline{\mathcal{X}})$ at $\underline{\hat{\mathcal{X}}}^t$
  $\underline{\boldsymbol{\tau}} \leftarrow$ Solve the linearised problem $(\mathbf{A}^\top \mathbf{A} + \lambda \operatorname{diag}(\mathbf{A}^\top \mathbf{A}))\underline{\boldsymbol{\tau}} = \mathbf{A}^\top \mathbf{b}$

  if $f(\underline{\hat{\mathcal{X}}}^t \oplus \underline{\boldsymbol{\tau}}) < f(\underline{\hat{\mathcal{X}}}^t)$ **then**
    Accept update, increase trust region
    $\underline{\hat{\mathcal{X}}}^{t+1} \leftarrow \underline{\hat{\mathcal{X}}}^t \oplus \underline{\boldsymbol{\tau}}$
    $\lambda \leftarrow \lambda / 10$
  else
    Reject update, reduce trust region
    $\underline{\hat{\mathcal{X}}}^{t+1} \leftarrow \underline{\hat{\mathcal{X}}}^t$
    $\lambda \leftarrow \lambda * 10$
  end

  if $f(\underline{\hat{\mathcal{X}}}^{t+1})$ *is very small or* $\underline{\hat{\mathcal{X}}}^{t+1} \approx \underline{\hat{\mathcal{X}}}^t$ **then**
    $\underline{\hat{\mathcal{X}}} \leftarrow \underline{\hat{\mathcal{X}}}^{t+1}$
    return
  end
end

Example 7.21 Range-based localisation

This example is inspired by an example in [7].

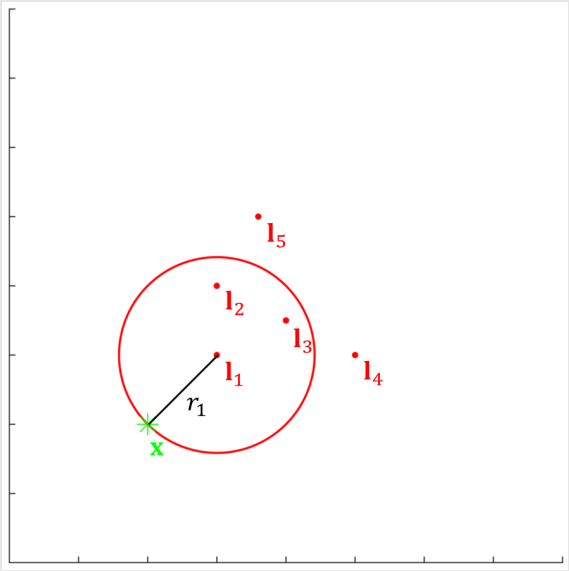


Figure 7.3: Range-based localisation example. Our true position $\mathbf{x}$ is shown in green, while the known landmarks $\{\mathbf{l}_1, \ldots, \mathbf{l}_5\}$ are shown in red The range r_1 from $\mathbf{l}_1$ to our position is visualised as a red circle around the landmark.

We are given a set of ranges $\{r_1, \ldots, r_n\}$ to landmarks with known 2D positions $\{\mathbf{l}_1, \ldots, \mathbf{l}_n\}$, and we want to estimate our location $\mathbf{x}$ that minimises the errors between the predicted range $\|\mathbf{x} - \mathbf{l}_i\|$ and the range measurement r_i,

$$e_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{l}_i\| - r_i. \tag{7.33}$$

This leads to the objective function

$$f(\mathbf{x}) = \sum_{i=1}^{n} \| \|\mathbf{x} - \mathbf{l}_i\| - r_i \|^2 \tag{7.34}$$

$$= \sum_{i=1}^{n} \left(\|\mathbf{x} - \mathbf{l}_i\| - r_i \right)^2, \tag{7.35}$$

which we can solve as the nonlinear least squares problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{n} \left(\|\mathbf{x} - \mathbf{l}_i\| - r_i \right)^2. \tag{7.36}$$

We can linearise the error function at the current estimate $\hat{\mathbf{x}}$ as

$$e_i(\hat{\mathbf{x}} + \delta\mathbf{x}) \approx e_i(\hat{\mathbf{x}}) + \mathbf{J}_{\hat{\mathbf{x}}}^{e_i} \delta\mathbf{x} \tag{7.37}$$

The Jacobian is given by (see Example 5.12)

$$\mathbf{J}_{\hat{\mathbf{x}}}^{e_i} = \mathbf{J}_{\hat{\mathbf{x}}}^{\|\hat{\mathbf{x}} - \mathbf{l}_i\|} = \frac{(\hat{\mathbf{x}} - \mathbf{l}_i)^\top}{\|\hat{\mathbf{x}} - \mathbf{l}_i\|}. \tag{7.38}$$

Using (7.19), we get the linearised least squares problem

$$\delta\mathbf{x}^* = \arg\min_{\delta\mathbf{x}} \sum_{i=1}^{n} \left(\mathbf{A}_i \delta\mathbf{x} - b_i \right)^2 \tag{7.39}$$

$$= \arg\min_{\delta\mathbf{x}} \|\mathbf{A}\delta\mathbf{x} - \mathbf{b}\|^2, \tag{7.40}$$

where

$$\mathbf{A}_i = \mathbf{J}_{\hat{\mathbf{x}}}^{e_i} = \frac{(\hat{\mathbf{x}} - \mathbf{l}_i)^\top}{\|\hat{\mathbf{x}} - \mathbf{l}_i\|} \tag{7.41}$$

$$b_i = r_i - e_i(\hat{\mathbf{x}}) = r_i - \|\hat{\mathbf{x}} - \mathbf{l}_i\| \tag{7.42}$$

and

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{bmatrix}. \tag{7.43}$$

This lets us estimate $\hat{\mathbf{x}}$ using Gauss-Newton (Algorithm 1) or Levenberg-Marquardt (Algorithm 4).

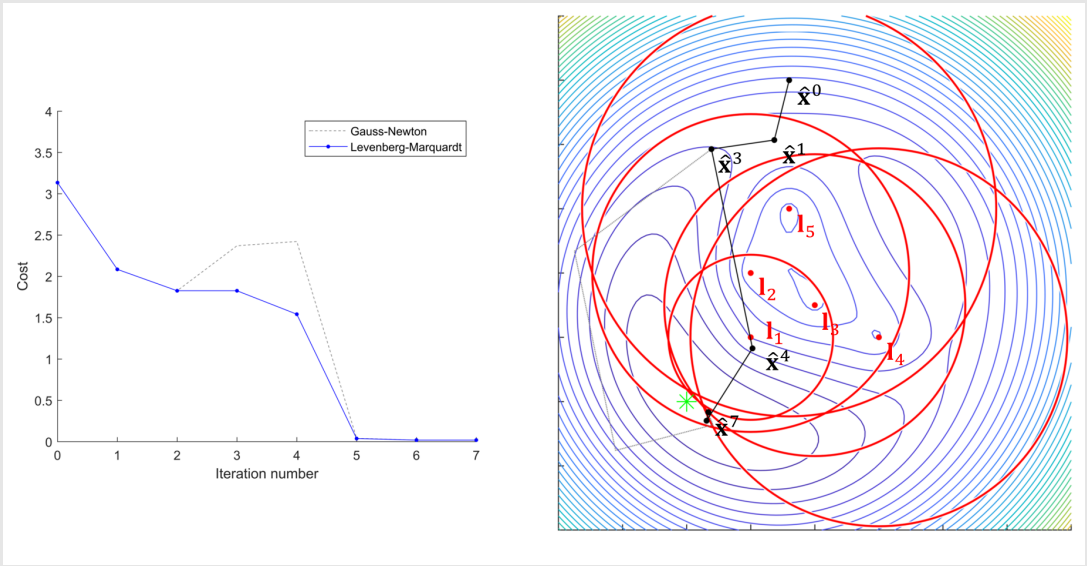


Figure 7.4: Example of applying Gauss-Newton and Levenberg-Marquardt to the range-based localisation problem. The range measurements are shown as red circles around their corresponding landmarks, and the coloured contours visualises the values of objective function at different locations. The iteration steps of the Levenberg-Marquardt algorithm are shown in black, while the steps of the Gauss-Newton algorithm are shown in grey. To the left we see how the error evolves with iterations for the two algorithms. We see that Gauss-Newton performs two steps that increases the error, while Levenberg-Marquardt always reduces the error in each step.

Figure 7.4 shows the difference between applying Gauss-Newton and Levenberg-Marquardt. We see that both methods converge to a solution that minimises the objective function, but Gauss-Newton performs two steps that increases the error. Figure 7.5 shows that perturbing the initial estimate slightly results in convergence to an erroneous local minimum.

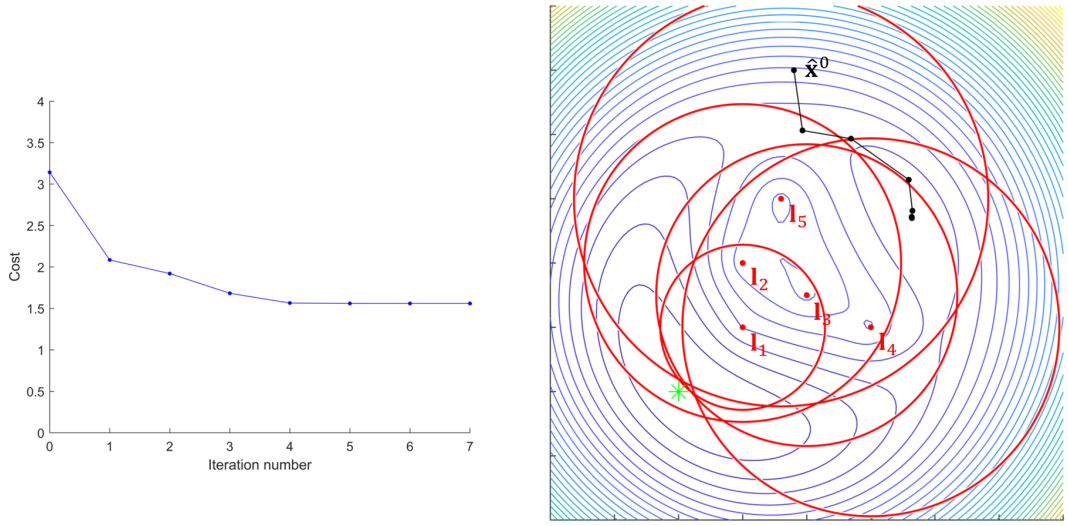


Figure 7.5: The result when perturbing the initial estimate slightly to the right. In this situation, Levenberg-Marquardt converges to a local minima. Even though Levenberg-Marquardt is guaranteed to converge, it is not guaranteed to converge to the correct solution.

7.3 Nonlinear MAP inference

We will now turn to the problem of solving *state estimation problems* based on noisy *measurements*. Assume that we want to estimate the unknown set of state variables X, and that we are given a set of sensor measurements Z that depends upon the true state of X. The most often used estimator for X is the *maximum a posteriori* (MAP) estimate, which maximises the *posterior density* $p(X|Z)$ of the states X given the measurements Z,

$$X^{\text{MAP}} = \arg\max_X p(X|Z). \tag{7.44}$$

Let the states X be represented by the concatenation $\underline{\mathcal{X}}$ from (7.13), and Z be the set of measurement vectors $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$. Let $\underline{\mathcal{X}}_i$ be the states involved in measurement $\mathbf{z}_i$. Assuming that the measurements are corrupted by zero-mean Gaussian noise, we define the *measurement model* as

$$\mathbf{z}_i = h_i(\underline{\mathcal{X}}_i) + \eta_i, \qquad \eta_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i), \tag{7.45}$$

where $h_i(\underline{\mathcal{X}}_i)$ is the *measurement prediction function*. We define the *measurement error function* as the difference between the predicted measurement, given the states $\underline{\mathcal{X}}_i$, and the measurement $\mathbf{z}_i$,

$$e_i(\underline{\mathcal{X}}_i) = h_i(\underline{\mathcal{X}}_i) - \mathbf{z}_i. \tag{7.46}$$

It can be shown that the MAP estimate for the states $\underline{\mathcal{X}}$ given the measurements Z

is the solution to the nonlinear least squares problem,

$$\hat{\underline{\mathcal{X}}}^{\text{MAP}} = \arg\min_{\underline{\mathcal{X}}} \sum_{i=1}^{n} \|h_i(\underline{\mathcal{X}}_i) - \mathbf{z}_i\|^2_{\mathbf{\Sigma}_i}, \tag{7.47}$$

where

$$\|\mathbf{e}\|^2_{\mathbf{\Sigma}} \triangleq \mathbf{e}^\top \mathbf{\Sigma}^{-1} \mathbf{e} = \left(\mathbf{\Sigma}^{-1/2}\mathbf{e}\right)^\top \left(\mathbf{\Sigma}^{-1/2}\mathbf{e}\right) = \left\|\mathbf{\Sigma}^{-1/2}\mathbf{e}\right\|^2, \tag{7.48}$$

is the squared Mahalanobis distance. We see from (7.48) that the measurement errors are weighted according to the measurement uncertainties, so that good measurements will contribute more to the total error than uncertain measurements. This is called *covariance weighting*, and (7.47) is therefore sometimes called a *weighted nonlinear least squares* problem.

We can solve (7.47) by first linearising the measurement prediction functions around the current estimate $\hat{\underline{\mathcal{X}}}$,

$$h_i(\underline{\mathcal{X}}_i) = h_i(\hat{\underline{\mathcal{X}}}_i \oplus \underline{\boldsymbol{\tau}}_i) \approx h_i(\hat{\underline{\mathcal{X}}}_i) + \mathbf{J}^{h_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i. \tag{7.49}$$

This leads to the linearised measurement error function

$$e_i(\underline{\mathcal{X}}_i) = e_i(\hat{\underline{\mathcal{X}}}_i \oplus \underline{\boldsymbol{\tau}}_i) \approx h_i(\hat{\underline{\mathcal{X}}}_i) + \mathbf{J}^{h_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i - \mathbf{z}_i, \tag{7.50}$$

and the linearised objective function

$$\begin{aligned}
f(\underline{\mathcal{X}}) = f(\hat{\underline{\mathcal{X}}} \oplus \underline{\boldsymbol{\tau}}) &= \sum_{i=1}^{n} \left\| e_i(\hat{\underline{\mathcal{X}}}_i \oplus \underline{\boldsymbol{\tau}}_i) \right\|^2_{\mathbf{\Sigma}_i} \\
&\approx \sum_{i=1}^{n} \left\| h_i(\hat{\underline{\mathcal{X}}}_i) + \mathbf{J}^{h_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i - \mathbf{z}_i \right\|^2_{\mathbf{\Sigma}_i} \\
&= \sum_{i=1}^{n} \left\| \mathbf{J}^{h_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i - (\mathbf{z}_i - h_i(\hat{\underline{\mathcal{X}}}_i)) \right\|^2_{\mathbf{\Sigma}_i} \\
&= \sum_{i=1}^{n} \left\| \mathbf{\Sigma}_i^{-1/2} \mathbf{J}^{h_i}_{\hat{\underline{\mathcal{X}}}_i} \underline{\boldsymbol{\tau}}_i - \mathbf{\Sigma}_i^{-1/2} (\mathbf{z}_i - h_i(\hat{\underline{\mathcal{X}}}_i)) \right\|^2 \\
&= \sum_{i=1}^{n} \|\mathbf{A}_i \underline{\boldsymbol{\tau}}_i - \mathbf{b}_i\|^2 \\
&= \|\mathbf{A}\underline{\boldsymbol{\tau}} - \mathbf{b}\|^2 .
\end{aligned} \tag{7.51}$$

Notice that the weighted Jacobians

$$\mathbf{A}_i = \mathbf{\Sigma}^{-1/2} \mathbf{J}^{h_i}_{\hat{\underline{\mathcal{X}}}_i}, \tag{7.52}$$

and the weighted *measurement prediction errors*

$$\mathbf{b}_i = \mathbf{\Sigma}^{-1/2} (\mathbf{z}_i - h_i(\hat{\underline{\mathcal{X}}}_i)), \tag{7.53}$$

have now undergone a form of *whitening*, that eliminates the units of the measurements.

With the problem on linearised form, we can apply Gauss-Newton (Algorithm 1) or Levenberg-Marquardt (Algorithm 4) to compute the MAP estimate $\hat{\underline{\mathcal{X}}}$, given a suitable initial estimate $\hat{\underline{\mathcal{X}}}^0$. We can estimate the uncertainty in the MAP estimate by recognising that the Hessian of $f(\underline{\mathcal{X}})$ at the solution corresponds to the *information matrix* of the estimate. Using the Gauss-Newton approximation to the Hessian, we obtain a first order approximation of the true covariance, given by

$$\boldsymbol{\Sigma}_{\hat{\underline{\mathcal{X}}}} = \boldsymbol{\Lambda}_{\hat{\underline{\mathcal{X}}}}^{-1} \approx (\mathbf{A}_{\hat{\underline{\mathcal{X}}}}^{\top} \mathbf{A}_{\hat{\underline{\mathcal{X}}}})^{-1}. \tag{7.54}$$

Example 7.22 Range-based localisation with measurement noise

We will revisit the range-based localisation example in Example 7.21, and see how we can apply the framework given in this section to naturally describe the state estimation problem, and solve it in the presence of measurement noise.

We are given a set of noisy range measurements $\{r_1, \ldots, r_n\}$ to landmarks with known 2D positions $\{\mathbf{l}_1, \ldots, \mathbf{l}_n\}$. The measurement model is given by

$$r_i = h_i(\mathbf{x}) + \eta_i, \qquad \eta \sim \mathcal{N}(0, \sigma_i^2), \tag{7.55}$$

where the measurement prediction function is

$$h_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{l}_i\|. \tag{7.56}$$

The measurement error is given by

$$e_i(\mathbf{x}) = h_i(\mathbf{x}) - r_i = \|\mathbf{x} - \mathbf{l}_i\| - r_i, \tag{7.57}$$

leading to the covariance weighted nonlinear least squares problem

$$\hat{\mathbf{x}}^{\text{MAP}} = \arg\min_{\mathbf{x}} \sum_{i=1}^{n} \frac{1}{\sigma_i} \left(\|\mathbf{x} - \mathbf{l}_i\| - r_i\right)^2 \tag{7.58}$$

By linearising the problem according to (7.37) and (7.51), we get the linearised least squares problem

$$\delta\mathbf{x}^* = \arg\min_{\delta\mathbf{x}} \sum_{i=1}^{n} \left(\mathbf{A}_i \delta\mathbf{x} - b_i\right)^2 \tag{7.59}$$

$$= \arg\min_{\delta\mathbf{x}} \|\mathbf{A}\delta\mathbf{x} - \mathbf{b}\|^2, \tag{7.60}$$

where

$$\mathbf{A}_i = \frac{1}{\sigma_i} \mathbf{J}_{\hat{\mathbf{x}}}^{e_i} = \frac{1}{\sigma_i} \frac{(\hat{\mathbf{x}} - \mathbf{l}_i)^{\top}}{\|\hat{\mathbf{x}} - \mathbf{l}_i\|} \tag{7.61}$$

$$b_i = \frac{1}{\sigma_i}(r_i - e_i(\hat{\mathbf{x}})) = \frac{1}{\sigma_i}(r_i - \|\hat{\mathbf{x}} - \mathbf{l}_i\|). \tag{7.62}$$

Figure 7.6 illustrates the difference between plain least squares and covariance weighted least squares. In Figure 7.7, we see an illustration of the covariance estimated from the approximation to the Hessian, in the idealised case when all measurements are equal to the expected values.

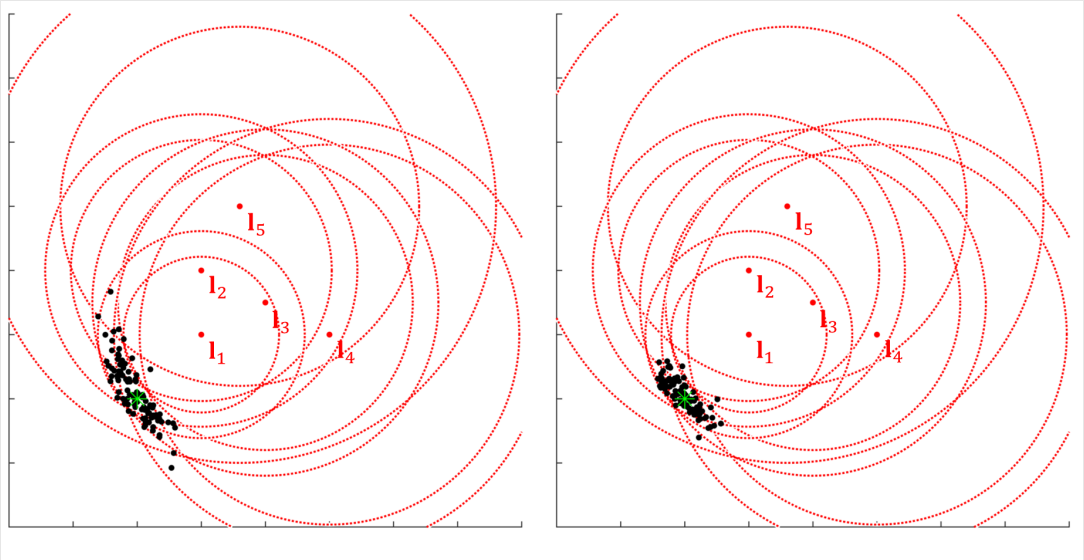


Figure 7.6: The difference between not taking measurement noise into account, and performing covariance weighting. The noise on the range measurements is visualised as $\pm\sigma$ dotted circles around the landmarks. In this example, landmark $\mathbf{l}_5$ has triple the measurement noise compared to the other landmarks. Based on simulating 100 sets of measurements, we see on the left the result of estimating $\mathbf{x}$ without taking the noise into account. The uncertainty in the results are dominated by the uncertain measurements. On right, the same measurements have been used in the covariance weighted scheme in (7.58). We see that the influence of the noisy measurements have been significantly reduced.

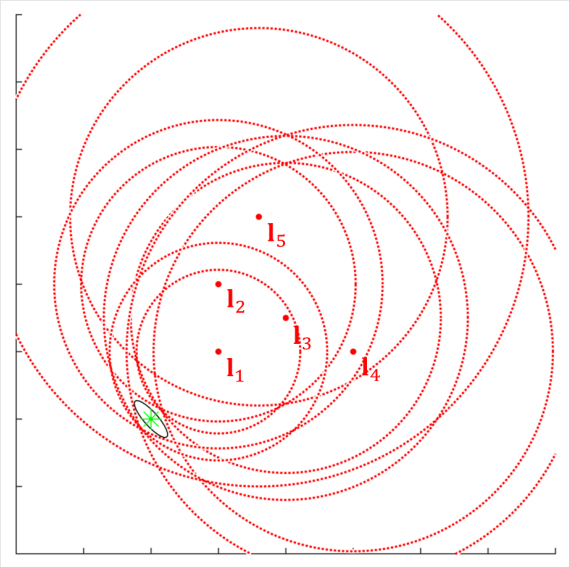


Figure 7.7: The covariance estimated from the approximation of the Hessian at the solution. Here, all measurements are set to the mean, to illustrate the connection between the uncertainty in the measurements, visualised as $\pm\sigma$ dotted circles around the landmarks, and the covariance of the estimate, shown as a black ellipse corresponding to the Mahalanobis distance at 1 standard deviation.

The framework for MAP estimation described in this section makes it easy to express estimation problems where we want to find the best estimates for the states $\mathcal{X}$, given a set of measurements Z. By defining measurement prediction functions $h_i(\mathcal{X}_i)$ for each measurement $\mathbf{z}_i$, and assuming Gaussian measurement noise, we can now obtain the MAP estimate and an estimate for its uncertainty. This is done by linearising the measurement prediction function, using the tools from Chapter 5, and applying the Gauss-Newton method from Algorithm 1 or the Levenberg-Marquardt method in Algorithm 4.

Notice that we have not assumed that all the measurements have the same type of measurement prediction function. This means that we also can use this framework to put several different types of constraints on the state variables, such as priors on states, as well as measurements from different types of sensors. We will in the next section see how we can apply this framework to estimate pose and structure from images.

Example 7.23 Range-based localisation with prior

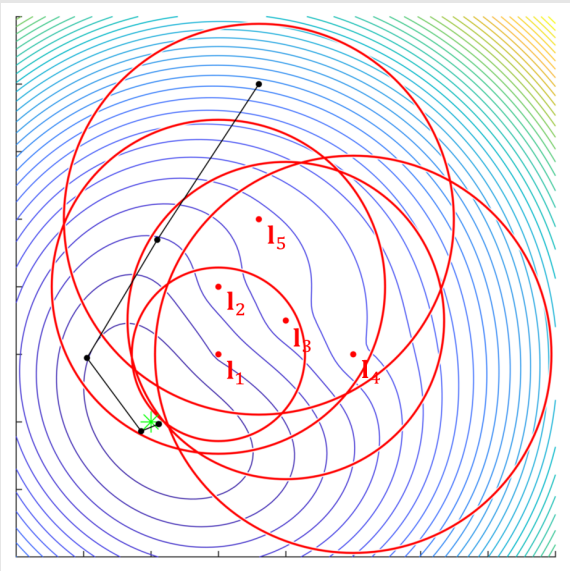


Figure 7.8: The result of adding a prior to the problem in (7.59). The expected value of the prior is set to the true state, and the standard deviation in both directions correspond to the measurement noise for the noisiest landmark $\mathbf{l}_5$. Compared to Figure 7.4, we see that the cost function is smoother, with a much better defined global minimium. The Gauss-Newton iterations shown in black traverses more directly towards the solution.

We can add a prior distribution $\mathcal{N}(\bar{\mathbf{x}}_p, \boldsymbol{\Sigma}_p)$ on our position in the problem in Example 7.22, by adding another constraint on the state $\mathbf{x}$.

We can define the "measurement model" for the prior distribution as

$$\mathbf{0} = h_p(\mathbf{x}) + \eta_p, \qquad \eta \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p), \tag{7.63}$$

where the measurement prediction function simply is

$$h_p(\mathbf{x}) = \mathbf{x} - \bar{\mathbf{x}}_p. \tag{7.64}$$

Since $\mathbf{J}_{\hat{\mathbf{x}}}^{h_p} = \mathbf{I}$, we can add the prior constraint by adding the following Jacobian blocks and prediction errors to $\mathbf{A}$ and $\mathbf{b}$ in (7.59):

$$\mathbf{A}_p = \boldsymbol{\Sigma}_p^{-1/2}\mathbf{I} = \boldsymbol{\Sigma}_p^{-1/2} \tag{7.65}$$

$$\mathbf{b}_p = \boldsymbol{\Sigma}_p^{-1/2}(\mathbf{0} - (\mathbf{x} - \bar{\mathbf{x}}_p)) = \boldsymbol{\Sigma}_p^{-1/2}(\bar{\mathbf{x}}_p - \mathbf{x}). \tag{7.66}$$

Figure 7.8 shows an example of adding a strict prior on $\mathbf{x}$.

Chapter 8

Estimating pose and structure

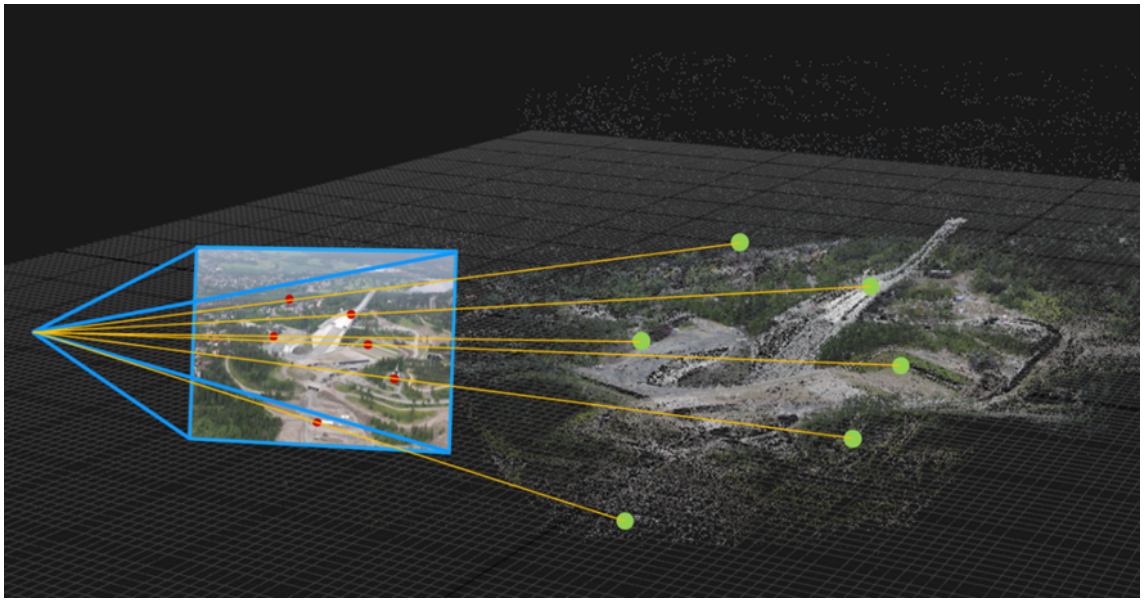


Figure 8.1: An example of estimating pose from correspondences between pixel points and 3D points in a map.

This chapter gives a brief introduction to how we can estimate pose and structure from images, based on the principles we have covered in the previous chapters.

Pose estimation using cameras is also sometimes called *motion estimation* or *localisation*, and is usually estimated relative to known structure, as illustrated in Figure 8.1. The *structure* is here a representation of the 3D structure in the scene, which is also sometimes called a *map*. When we know the relative poses between two or more cameras, we can estimate the structure by triangulating correspondences between the cameras.

When neither pose nor structure is known a priori, we typically need to *initialise* a map based on relative pose estimates between camera images. We can then estimate new poses relative to this map, and continue extending the map while exploring the scene. This procedure is often called *simultaneous localisation and mapping (SLAM)*, and performing SLAM with images is often called *Visual SLAM*

(VSLAM).

A major problem with monocular vision is that scale in the world cannot be observed without prior knowledge. Monocular VSLAM will therefore drift in scale over time, and this results in a major contribution to the overall error. But as pointed out in [14], an interesting paradox is that its inherent scale-ambiguity is also a major benefit, since monocular vision can seamlessly operate in vastly differently scaled environments, such as switching from a small-scale door handle to the large-scale environment just outside.

Adding more sensors, such as another camera with known baseline, active range sensors or an inertial navigation system, can enable us to recover metric scale, as well as the horizontal plane, and even the geographical pose.

8.1 Problem formulations

It is common to formulate pose estimation problems as either *direct* or *indirect*, and structure estimation problems as either *dense* or *sparse* [13].

Indirect methods first extract an intermediate geometrical representation of the scene, such as keypoint correspondences or optical flow vectors, and use these to optimise a *geometric error*. Because these methods are based on some kind of feature extraction step, they are also often called *feature-based methods*. Indirect methods provide robustness to photometric and geometric distortions, but come at the price of higher computational cost and are dependent on the feature extraction step to work.

Direct methods skip the feature extraction step, and use the image intensity values directly to optimise a *photometric error*. Direct methods do not require that geometrical primitives are recognisable by themselves, but can sample across all parts of the image. This makes direct methods more robust to effects such as motion blur and better suited in environments that are sparsely textured, but they are vulnerable to photometric and geometric distortions.

Sparse methods reconstruct and use only a selected set of independent points in the scene, while *dense methods* attempt to reconstruct and use a 3D map for all the pixels in the images. An important difference between the sparse and dense methods is that, while the sparse methods map independent points with no notion of neighbourhoods, the dense methods typically add a geometric prior, assuming that the structure is locally smooth. Because of the resulting correlated structure in the dense maps, it is difficult to optimise pose and dense structure jointly in real-time.

The most widely used formulation is the sparse indirect formulation, which is used in methods such as PTAM [19] and ORB-SLAM [23]. The sparse direct formulation is used in DSO [13], while the dense direct formulation is used in DTAM [24] and LSD-SLAM [14]. SVO [16] uses a hybrid formulation, where the direct formulation is used for initial alignment and to obtain correspondences, while the

indirect formulation is used for joint model optimisation.

We will in the following sections take a closer look at how we can express and solve some of these problems.

8.2 Indirect methods

We will here consider sparse, indirect methods based on point correspondence measurements between images. Given correspondences $\mathbf{u} \leftrightarrow \mathbf{x}^w$ between pixels and points in the world frame (defined by the map), these methods minimise the *geometric error*, which for each correspondence can be expressed as

$$e_g(\mathbf{T}_{wc}, \mathbf{x}^w) = \pi_p(\mathbf{T}_{wc}^{-1} \cdot \mathbf{x}^w) - \mathbf{u}. \tag{8.1}$$

Here, π_p is the projection function for the perspective camera given in (3.16) and $\mathbf{T}_{wc}$ is the pose of the camera in the world frame. The geometric error measures the difference between the predicted pixel position, given the current hypotheses for the pose $\mathbf{T}_{wc}$ and the point $\mathbf{x}^w$, and the measured pixel correspondence. It is therefore also often called the *reprojection error*. Estimating poses and structure based on the geometric error is often called *bundle adjustment* [27].

The following sections will discuss how we can estimate pose, structure and both, based on minimising the geometric error.

8.2.1 Estimating pose with known structure

Assume that the world points $\mathbf{x}_j^w$ are known, and we are given the correspondences $\mathbf{u}_j \leftrightarrow \mathbf{x}_j^w$, with measurement noise $\mathbf{\Sigma}_j$.

A suitable initial estimate for the pose $\mathbf{T}_{wc}$ can be found by applying a fast *perspective-n-point (PnP)* algorithm, such as P3P [20]. Estimating the camera pose from known 3D points by minimising the geometric error is sometimes called *motion-only bundle adjustment*. Assuming that the camera is calibrated, we can pre-calibrate the measurements to normalised image coordinates with (3.11). This gives us the measurement prediction function

$$h_j(\mathbf{T}_{wc}) = \pi_n(\mathbf{T}_{wc}^{-1} \cdot \mathbf{x}_j^w), \tag{8.2}$$

where π_n is the projection function to normalised image coordinates in (3.17). The measurement error function is the geometric error

$$e_j(\mathbf{T}_{wc}) = \pi_n(\mathbf{T}_{wc}^{-1} \cdot \mathbf{x}_j^w) - \mathbf{x}_{n\,j}. \tag{8.3}$$

The Jacobian of the measurement prediction function is given by (omitting the

index j for brevity)

$$\mathbf{J}^{h}_{\mathbf{T}_{wc}} = \mathbf{J}^{\pi_n(\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w)}_{\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w}\mathbf{J}^{\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w}_{\mathbf{T}_{wc}^{-1}}\mathbf{J}^{\mathbf{T}_{wc}^{-1}}_{\mathbf{T}_{wc}} \tag{8.4a}$$

$$= \mathbf{J}^{\pi_n(\mathbf{x}^c)}_{\mathbf{x}^c}\mathbf{J}^{\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w}_{\mathbf{T}_{wc}^{-1}}\mathbf{J}^{\mathbf{T}_{wc}^{-1}}_{\mathbf{T}_{wc}} \tag{8.4b}$$

$$= \frac{1}{z^c}\begin{bmatrix}1 & 0 & -x^c/z^c\\ 0 & 1 & -y^c/z^c\end{bmatrix}\begin{bmatrix}\mathbf{R}_{wc}^\top & -\mathbf{R}_{wc}^\top[\mathbf{x}^w]_\times\end{bmatrix}\cdot -\begin{bmatrix}\mathbf{R}_{wc} & [\mathbf{t}^w_{wc}]_\times\mathbf{R}_{wc}\\ \mathbf{0} & \mathbf{R}_{wc}\end{bmatrix} \tag{8.4c}$$

$$= d\begin{bmatrix}1 & 0 & -x_n\\ 0 & 1 & -y_n\end{bmatrix}\begin{bmatrix}-\mathbf{I} & [\mathbf{x}^c]_\times\end{bmatrix} \tag{8.4d}$$

$$= \begin{bmatrix}-d & 0 & dx_n & x_ny_n & -1-x_n^2 & y_n\\ 0 & -d & dy_n & 1+y_n^2 & -x_ny_n & -x_n\end{bmatrix}, \tag{8.4e}$$

where $\mathbf{x}^c = \mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w$ is the point in camera frame, and $d = 1/z^c$ is the inverse depth.

By propagating the noise to the normalised image plane, we get the linearised weighted least squares problem

$$\boldsymbol{\xi}^* = \arg\min_{\boldsymbol{\xi}}\sum_{j=1}^{n}\|\mathbf{A}_j\boldsymbol{\xi}-\mathbf{b}_j\|^2 \tag{8.5}$$

$$= \arg\min_{\boldsymbol{\xi}}\|\mathbf{A}\boldsymbol{\xi}-\mathbf{b}\|^2, \tag{8.6}$$

where

$$\mathbf{A}_j = \boldsymbol{\Sigma}_{n\,j}^{-1/2}\mathbf{J}^{h_j}_{\mathbf{T}_{wc}} \tag{8.7}$$

$$\mathbf{b}_j = \boldsymbol{\Sigma}_{n\,j}^{-1/2}(\mathbf{x}_{n\,j} - h_j(\mathbf{T}_{wc})), \tag{8.8}$$

and

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_1\\ \vdots\\ \mathbf{A}_n\end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix}\mathbf{b}_1\\ \vdots\\ \mathbf{b}_n\end{bmatrix}. \tag{8.9}$$

This lets us compute the MAP estimate $\hat{\mathbf{T}}_{wc}$ with Gauss-Newton or Levenberg-Marquardt.

8.2.2 Estimating structure with known poses

Assume that we know the poses of the two or more cameras $\{\mathbf{T}_{wc_i}\}$, and we are given the correspondences $\mathbf{u}^i_j \leftrightarrow \mathbf{x}^w_j$, with measurement noise $\boldsymbol{\Sigma}_{ij}$.

A suitable initial estimate for the structure $\{\mathbf{x}^w_j\}$ can be found by for example triangulating the points linearly by minimising algebraic error [18]. Estimating the structure from known camera poses by minimising the geometric error is sometimes called *structure-only bundle adjustment*. Assuming that the cameras are calibrated, we can pre-calibrate the measurements to normalised image coordinates with (3.11). This gives us the measurement prediction function

$$h_{ij}(\mathbf{x}^w_j) = \pi_n(\mathbf{T}^{-1}_{wc_i}\cdot\mathbf{x}^w_j), \tag{8.10}$$

where π_n is the projection function to normalised image coordinates in (3.17). The measurement error function is the geometric error

$$e_{ij}(\mathbf{x}_j^w) = \pi_n(\mathbf{T}_{wc_i}^{-1} \cdot \mathbf{x}_j^w) - \mathbf{x}_{n\,j}^i. \tag{8.11}$$

The Jacobian of the measurement prediction function is given by (omitting the indexes i and j for brevity)

$$\begin{aligned}
\mathbf{J}_{\mathbf{x}^w}^h &= \mathbf{J}_{\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w}^{\pi_n(\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w)} \mathbf{J}_{\mathbf{x}^w}^{\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w} && \text{(8.12a)}\\
&= \mathbf{J}_{\mathbf{x}^c}^{\pi_n(\mathbf{x}^c)} \mathbf{J}_{\mathbf{x}^w}^{\mathbf{T}_{wc}^{-1}\cdot\mathbf{x}^w} && \text{(8.12b)}\\
&= \frac{1}{z^c} \begin{bmatrix} 1 & 0 & -x^c/z^c \\ 0 & 1 & -y^c/z^c \end{bmatrix} \mathbf{R}_{wc}^\top && \text{(8.12c)}\\
&= d \begin{bmatrix} 1 & 0 & -x_n \\ 0 & 1 & -y_n \end{bmatrix} \mathbf{R}_{wc}^\top, && \text{(8.12d)}
\end{aligned}$$

where $\mathbf{x}^c = \mathbf{T}_{wc}^{-1} \cdot \mathbf{x}^w$ is the point in camera frame, and $d = 1/z^c$ is the inverse depth.

By propagating the noise to the normalised image plane, we get the linearised weighted least squares problem

$$\begin{aligned}
\delta\mathbf{x}^* &= \arg\min_{\delta\mathbf{x}} \sum_{i=1}^{k} \sum_{j=1}^{n} \left\| \mathbf{A}_{ij}\delta\mathbf{x}_j - \mathbf{b}_{ij} \right\|^2 && \text{(8.13)}\\
&= \arg\min_{\delta\mathbf{x}} \left\| \mathbf{A}\delta\mathbf{x} - \mathbf{b} \right\|^2, && \text{(8.14)}
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{A}_{ij} &= \mathbf{\Sigma}_{n\,ij}^{-1/2} \mathbf{J}_{\mathbf{x}_j^w}^{h_{ij}} && \text{(8.15)}\\
\mathbf{b}_{ij} &= \mathbf{\Sigma}_{n\,ij}^{-1/2} (\mathbf{x}_{n\,j}^i - h_{ij}(\mathbf{x}_j^w)), && \text{(8.16)}
\end{aligned}$$

and

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & & \\ & \ddots & \\ & & \mathbf{A}_{1n} \\ & \vdots & \\ \mathbf{A}_{k1} & & \\ & \ddots & \\ & & \mathbf{A}_{kn} \end{bmatrix} \qquad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{x}_1 \\ \vdots \\ \delta\mathbf{x}_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{11} \\ \vdots \\ \mathbf{b}_{1n} \\ \vdots \\ \mathbf{b}_{k1} \\ \vdots \\ \mathbf{b}_{kn} \end{bmatrix}. \tag{8.17}$$

This lets us compute the MAP estimates $\{\hat{\mathbf{x}}_j^w\}$ with Gauss-Newton or Levenberg-Marquardt.

8.2.3 Estimating pose and structure

In this situation we do not know either the camera poses $\{\mathbf{T}_{wc_i}\}$ nor the world points $\{\mathbf{x}_j^w\}$. We are just given the correspondences $\mathbf{u}_j^i \leftrightarrow \mathbf{x}_j^w$, with measurement noise $\mathbf{\Sigma}_{ij}$.

A suitable initial estimate can be found by first applying pairwise two-view relative pose estimation, based on estimating the essential matrix $\mathbf{E}$ using the 5-point algorithm [25], and decomposing it to obtain the relative pose [18]. Estimating the both pose and structure by minimising the geometric error is sometimes called *full bundle adjustment*. Assuming that the cameras are calibrated, we can pre-calibrate the measurements to normalised image coordinates with (3.11). This gives us the measurement prediction function

$$h_{ij}(\mathbf{T}_{wc_i}, \mathbf{x}_j^w) = \pi_n(\mathbf{T}_{wc_i}^{-1} \cdot \mathbf{x}_j^w), \tag{8.18}$$

where π_n is the projection function to normalised image coordinates in (3.17). The measurement error function is the geometric error

$$e_{ij}(\mathbf{T}_{wc_i}, \mathbf{x}_j^w) = \pi_n(\mathbf{T}_{wc_i}^{-1} \cdot \mathbf{x}_j^w) - \mathbf{x}_{n\,j}^i. \tag{8.19}$$

Since the measurement function is a function of two variables, we linearise it at the current state estimates as

$$\begin{aligned} h_{ij}(\mathbf{T}_{wc_i}, \mathbf{x}_j^w) &= h_{ij}(\hat{\mathbf{T}}_{wc_i} \oplus \boldsymbol{\xi}_i, \hat{\mathbf{x}}_j^w + \delta\mathbf{x}_j) && \text{(8.20a)} \\ &\approx h_{ij}(\hat{\mathbf{T}}_{wc_i}, \hat{\mathbf{x}}_j^w) + \mathbf{J}^{h_{ij}}_{\hat{\mathbf{T}}_{wc_i}} \boldsymbol{\xi}_i + \mathbf{J}^{h_{ij}}_{\hat{\mathbf{x}}_j^w} \delta\mathbf{x}_j, && \text{(8.20b)} \end{aligned}$$

The Jacobians are given in (8.4) and (8.12).

By propagating the noise to the normalised image plane, we get the linearised weighted least squares problem

$$\begin{aligned} \underline{\boldsymbol{\tau}}^* &= \arg\min_{\underline{\boldsymbol{\tau}}} \sum_{i=1}^{k} \sum_{j=1}^{n} \left\| \mathbf{P}_{ij}\boldsymbol{\xi}_i + \mathbf{S}_{ij}\delta\mathbf{x}_j - \mathbf{b}_{ij} \right\|^2 && \text{(8.21)} \\ &= \arg\min_{\underline{\boldsymbol{\tau}}} \left\| \mathbf{A}\underline{\boldsymbol{\tau}} - \mathbf{b} \right\|^2, && \text{(8.22)} \end{aligned}$$

where

$$\begin{aligned} \mathbf{P}_{ij} &= \mathbf{\Sigma}_{n\,ij}^{-1/2} \mathbf{J}^{h_{ij}}_{\mathbf{T}_{wc_i}} && \text{(8.23)} \\ \mathbf{S}_{ij} &= \mathbf{\Sigma}_{n\,ij}^{-1/2} \mathbf{J}^{h_{ij}}_{\mathbf{x}_j^w} && \text{(8.24)} \\ \mathbf{b}_{ij} &= \mathbf{\Sigma}_{n\,ij}^{-1/2} (\mathbf{x}_{n\,j}^i - h_{ij}(\mathbf{T}_{wc_i}, \mathbf{x}_j^w)), && \text{(8.25)} \end{aligned}$$

and

$$\mathbf{A} = \begin{bmatrix} \mathbf{P}_{11} & & & \mathbf{S}_{11} & & \\ \vdots & & & & \ddots & \\ \mathbf{P}_{1n} & & & & & \mathbf{S}_{1n} \\ & \ddots & & & \vdots & \\ & & \mathbf{P}_{k1} & \mathbf{S}_{k1} & & \\ & & \vdots & & \ddots & \\ & & \mathbf{P}_{kn} & & & \mathbf{S}_{kn} \end{bmatrix} \quad \underline{\boldsymbol{\tau}} = \begin{bmatrix} \boldsymbol{\xi}_1 \\ \vdots \\ \boldsymbol{\xi}_k \\ \delta\mathbf{x}_1 \\ \vdots \\ \delta\mathbf{x}_n \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{11} \\ \vdots \\ \mathbf{b}_{1n} \\ \vdots \\ \mathbf{b}_{k1} \\ \vdots \\ \mathbf{b}_{kn} \end{bmatrix}. \tag{8.26}$$

The linear least squares problem in (8.26) is not uniquely determined and has a singular approximation to the Hessian. This is because the global frame and scale is not defined, so we can apply any similarity transform to the camera poses without affecting the objective function. This is sometimes called *gauge freedom*. We fix this by adding priors on at least one pose, defining the global frame, and at least one point, defining the global scale.

This lets us compute the MAP estimates $\{\hat{\mathbf{T}}_{wc_i}\}$ and $\{\hat{\mathbf{x}}_j^w\}$ with Gauss-Newton or Levenberg-Marquardt.

8.3 Direct methods

We will here consider direct methods that optimise structure separately from pose.

Consider two images $I_a(\mathbf{u})$ and $I_b(\mathbf{u})$. Given the relative pose $\mathbf{T}_{ab}$ between the two cameras a and b, we can define the *warp function* [3]

$$\mathbf{u}^a = w(\mathbf{u}^b, z^b, \mathbf{T}_{ab}) = \pi_p(\mathbf{T}_{ab} \cdot \pi_p^{-1}(\mathbf{u}^b, z^b)), \tag{8.27}$$

that maps a pixel $\mathbf{u}^b$ in image I_b to a pixel $\mathbf{u}_a$ in image I_a. This lets us define the *photometric error* between these two images at a given pixel as

$$e_p(\mathbf{u}^b, z^b, \mathbf{T}_{ab}) = I_a(w(\mathbf{u}^b, z^b, \mathbf{T}_{ab})) - I_b(\mathbf{u}^b). \tag{8.28}$$

Direct methods typically represent structure as a sparse or dense depth map for pixels rather than 3D points. The depth is typically represented as the *inverse depth* $d = z^{-1}$, because the inverse depth parameterisation is better suited when we want to model the uncertainty in depth using a Gaussian noise model [10]. Depth can be estimated by collecting small-baseline measurements over time, letting the uncertainty in depth fall below a threshold before using it for pose estimation [15, 14, 16].

If we know the pose $\mathbf{T}_{ab}$ between the two images, we can exploit the epipolar geometry to search for good matches that minimises (8.28) over a relevant interval of depths, guided by the current depth uncertainty (Section 3.2). Since

$$z^a \mathbf{K}_a^{-1} \breve{\mathbf{u}}^a = z^b \mathbf{R}_{ab} \mathbf{K}_b^{-1} \breve{\mathbf{u}}^b + \mathbf{t}_{ab}^a, \tag{8.29}$$

we can compute both depths by solving the linear least squares problem

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \|\mathbf{A}\mathbf{z} - \mathbf{b}\|^2, \tag{8.30}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{K}_a^{-1}\breve{\mathbf{u}}^a & -\mathbf{R}_{ab}\mathbf{K}_b^{-1}\breve{\mathbf{u}}^b \end{bmatrix} \qquad \mathbf{z} = \begin{bmatrix} z^a \\ z^b \end{bmatrix} \qquad \mathbf{b} = \mathbf{t}_{ab}^a. \tag{8.31}$$

When interested in z^j only, the law of sines applied on the epipolar plane gives us

$$z^b = \frac{\left\|\mathbf{K}_a^{-1}\breve{\mathbf{u}}^a \times \mathbf{t}_{ab}^a\right\|}{\left\|\mathbf{K}_a^{-1}\breve{\mathbf{u}}^a \times \mathbf{R}_{ab}\mathbf{K}_b^{-1}\breve{\mathbf{u}}^b\right\|}. \tag{8.32}$$

We can avoid the computations of the norms by

$$z^b = \frac{\mathbf{a}^\top(\mathbf{K}_a^{-1}\breve{\mathbf{u}}^a \times \mathbf{t}_{ab}^a)}{\mathbf{a}^\top(\mathbf{K}_a^{-1}\breve{\mathbf{u}}^a \times \mathbf{R}_{ab}\mathbf{K}_b^{-1}\breve{\mathbf{u}}^b)}, \tag{8.33}$$

where $\mathbf{a} \in \mathbb{R}^3$ is a suitable vector that is not perpendicular to the epipolar plane, or parallel to any of the vectors in the cross products.

Assuming that the cameras are calibrated, we can pre-calibrate the measurements to normalised image coordinates with (3.11). This results in the warp function

$$w_n(\mathbf{x}_n^b, z^b, \mathbf{T}_{ab})) = \pi_n(\mathbf{T}_{ab} \cdot \pi_n^{-1}(\mathbf{x}_n^b, z^b)). \tag{8.34}$$

Given the depths $\{z_i^b\}$ for normalise image coordinates $\{\mathbf{x}_{n\,i}^b\}$ and the images I_a and I_b, we can estimate the pose $\mathbf{T}_{ab}$ by defining the measurement prediction function for the pixel intensity in image I_b as

$$h_i(\mathbf{T}_{ab}) = I_a(w_n(\mathbf{x}_{n\,i}^b, z_i^b, \mathbf{T}_{ab})), \tag{8.35}$$

and the measurement error function as

$$e_i(\mathbf{T}_{ab}) = I_a(w_n(\mathbf{x}_{n\,i}^b, z_i^b, \mathbf{T}_{ab})) - I_b(\mathbf{x}_{n\,i}^b). \tag{8.36}$$

The Jacobian of the measurement prediction function is given by (omitting the indexes i for brevity)

$$\begin{aligned}
\mathbf{J}_{\mathbf{T}_{ab}}^h &= \mathbf{J}_{w_n(\mathbf{x}_n^b, z^b, \mathbf{T}_{ab})}^{I_a} \mathbf{J}_{\mathbf{T}_{ab}}^{w_n(\mathbf{x}_n^b, z^b, \mathbf{T}_{ab})} && \text{(8.37a)} \\
&= \mathbf{J}_{\pi_n(\mathbf{x}^a)}^{I_a} \mathbf{J}_{\mathbf{T}_{ab}\cdot\mathbf{x}^b}^{\pi_n(\mathbf{T}_{ab}\cdot\mathbf{x}^b)} \mathbf{J}_{\mathbf{T}_{ab}}^{\mathbf{T}_{ab}\cdot\mathbf{x}^b} && \text{(8.37b)} \\
&= \mathbf{J}_{\mathbf{x}_n^a}^{I_a} \mathbf{J}_{\mathbf{x}^a}^{\pi_n(\mathbf{x}^a)} \mathbf{J}_{\mathbf{T}_{ab}}^{\mathbf{T}_{ab}\cdot\mathbf{x}^b} && \text{(8.37c)} \\
&= \nabla I_a(\mathbf{x}_n^a)\; d \begin{bmatrix} 1 & 0 & -x_n^a \\ 0 & 1 & -y_n^a \end{bmatrix} \begin{bmatrix} \mathbf{R}_{ab} & -\mathbf{R}_{ab}[\mathbf{x}^b]_\times \end{bmatrix}, && \text{(8.37d)}
\end{aligned}$$

where $\nabla I_a(\mathbf{x}_n^a)$ is the image gradient at $\breve{\mathbf{u}}^a = \mathbf{K}_a\breve{\mathbf{x}}_n^a$.

This lets us compute the MAP estimate $\hat{\mathbf{T}}_{ab}$ with Gauss-Newton or Levenberg-Marquardt.

Bibliography

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton: Princeton University Press, Dec. 2008. ISBN: 9781400830244. DOI: 10.1515/9781400830244. URL: https://www.degruyter.com/view/books/9781400830244/9781400830244/9781400830244.xml%20http://www.degruyter.com/view/books/9781400830244/9781400830244/9781400830244.xml.

[2] Vincent Arsigny, Xavier Pennec, and Nicholas Ayache. "Bi-invariant Means in Lie Groups. Application to Left-invariant Polyaffine Transformations ." In: *Notes* (2006), p. 52.

[3] Simon Baker and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *International Journal of Computer Vision* 56.3 (Feb. 2004), pp. 221–255. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000011205.11775.fd. URL: http://link.springer.com/10.1023/B:VISI.0000011205.11775.fd.

[4] Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge: Cambridge University Press, 2017. ISBN: 9781316671528. DOI: 10.1017/9781316671528. URL: http://ebooks.cambridge.org/ref/id/CBO9781316671528.

[5] J. P. Barreto. "Unifying Image Plane Liftings for Central Catadioptric and Dioptric Cameras". In: *Imaging Beyond the Pinhole Camera*. Springer Netherlands, 2006, pp. 21–38. ISBN: 1402048939. DOI: 10.1007/978-1-4020-4894-4{_}2. URL: http://link.springer.com/10.1007/978-1-4020-4894-4_2.

[6] Jose-Luis Blanco. *A tutorial on SE(3) transformation parameterizations and on-manifold optimization*. Tech. rep. URL: http://www.ual.es/personal/jlblanco/%20http://mapir.isa.uma.es/.

[7] S Boyd and L Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. 2017, p. 476. ISBN: 978-1-316-51896-0. DOI: 10.1017/9781108583664. URL: https://web.stanford.edu/~boyd/vmls/.

[8] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (Dec. 2016), pp. 1309–1332. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2624754. URL: http://ieeexplore.ieee.org/document/7747236/.

[9] David Caruso, Jakob Engel, and Daniel Cremers. "Large-scale direct SLAM for omnidirectional cameras". In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2015-Decem. 2015, pp. 141–148. ISBN: 9781479999941. DOI: 10.1109/IROS.2015.7353366.

[10] Javier Civera, Andrew J. Davison, and José María Martínez Montiel. "Inverse Depth Parametrization". In: *Springer Tracts in Advanced Robotics*. Vol. 75. 5. 2012, pp. 33–63. ISBN: 1552-3098 VO - 24. DOI: 10.1007/978-3-642-24834-4{_}3. URL: http://link.springer.com/10.1007/978-3-642-24834-4_3.

[11] Frank Dellaert and Michael Kaess. "Factor Graphs for Robot Perception". In: *Foundations and Trends in Robotics* 6.1-2 (2017), pp. 1–139. ISSN: 1935-8253. DOI: 10.1561/2300000043. URL: http://www.nowpublishers.com/article/Details/ROB-043.

[12] Ethan Eade. "Lie Groups for 2D and 3D Transformations". In: (2013), pp. 1–24.

[13] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP.99 (2017), pp. 1–1. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2658577. URL: http://ieeexplore.ieee.org/document/7898369/.

[14] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. Vol. 8690. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10604-5. DOI: 10.1007/978-3-319-10605-2. URL: http://link.springer.com/10.1007/978-3-319-10605-2.

[15] Jakob Engel, Jurgen Sturm, and Daniel Cremers. "Semi-dense Visual Odometry for a Monocular Camera". In: *2013 IEEE International Conference on Computer Vision*. IEEE, Dec. 2013, pp. 1449–1456. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.183. URL: http://ieeexplore.ieee.org/document/6751290/.

[16] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014, pp. 15–22. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6906584. URL: http://ieeexplore.ieee.org/document/6906584/.

[17] Christian Forster et al. "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry". In: *IEEE Transactions on Robotics* 33.1 (Feb. 2017), pp. 1–21. ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597321. URL: http://ieeexplore.ieee.org/document/7557075/.

[18] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd Ed. Cambridge University Press, 2004. ISBN: 9780521540513.

[19] Georg Klein and David Murray. "Parallel tracking and mapping for small AR workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*. 2007. ISBN: 9781424417506. DOI: 10.1109/ISMAR.2007.4538852.

[20] Laurent Kneip, Davide Scaramuzza, and Roland Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *CVPR 2011*. IEEE, June 2011, pp. 2969–2976. ISBN: 978-1-4577-0394-2. DOI: 10.1109/CVPR.2011.5995464. URL: http://ieeexplore.ieee.org/document/5995464/.

[21] Tomasz Łuczyński, Max Pfingsthorn, and Andreas Birk. "The Pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings". In: *Ocean Engineering* 133 (Mar. 2017), pp. 9–22. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2017.01.029. URL: https://linkinghub.elsevier.com/retrieve/pii/S0029801817300434.

[22] Y Ma et al. *An invitation to 3D vision: from images to geometric models*. 2004. URL: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:An+Invitation+to+3D+Vision#1.

[23] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2463671. URL: http://ieeexplore.ieee.org/document/7219438/.

[24] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. "DTAM: Dense tracking and mapping in real-time". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011, pp. 2320–2327. ISBN: 978-1-4577-1102-2. DOI: 10.1109/ICCV.2011.6126513. URL: http://ieeexplore.ieee.org/document/6126513/.

[25] D. Nister. "An efficient solution to the five-point relative pose problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (June 2004), pp. 756–770. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.17. URL: http://ieeexplore.ieee.org/document/1288525/.

[26] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics*. Tech. rep. URL: https://github.com/artivis/manif..

[27] Bill Triggs et al. "Bundle Adjustment — A Modern Synthesis". In: *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*. 2000, pp. 298–372. ISBN: 3-540-67973-1. DOI: 10.1007/3-540-44480-7{_}21. URL: http://link.springer.com/10.1007/3-540-44480-7_21.

[28] Zhengyou Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. ISSN: 01628828. DOI: 10.1109/34.888718. URL: http://ieeexplore.ieee.org/document/888718/.