ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ROBOTICS &
PERCEPTION
GROUP

Alessio Zanchettin

Autonomous landing on a moving platform

Master Thesis

Robotics and Perception Group
University of Zurich

Supervision

First Supervisor Davide Falanga
Second Supervisor
Prof. Dr. Davide Scaramuzza

September 2016

Contents

Abstract

Compress the introduction in a few key sentences. No more than half a page. The abstract should motivate your work, outline the work that you did, and give some insights into its results.

Nomenclature

Notation

$\mathbf{J}$	Jacobian
$\mathbf{H}$	Hessian
$\mathbf{T}_{WB}$	coordinate transformation from frame B to frame W
$\mathbf{R}_{WB}$	orientation of B with respect to W
${}_W\mathbf{t}_{WB}$	translation of B with respect to W, expressed in coordinate system W

Scalars are written in lower case letters (a), vectors in lower case bold letters ($\mathbf{a}$) and matrices in upper case bold letters ($\mathbf{A}$).

Acronyms and Abbreviations

RPG	Robotics and Perception Group
DoF	Degree of Freedom
IMU	Inertial Measurement Unit
MAV	Micro Aerial Vehicle
ROS	Robot Operating System

Chapter 1

Introduction

Describe the problem and the motivation for this research.

1.1 Related Work

Describe the current state of the art. Provide all necessary citations.

1.2 MBZIRC challenge

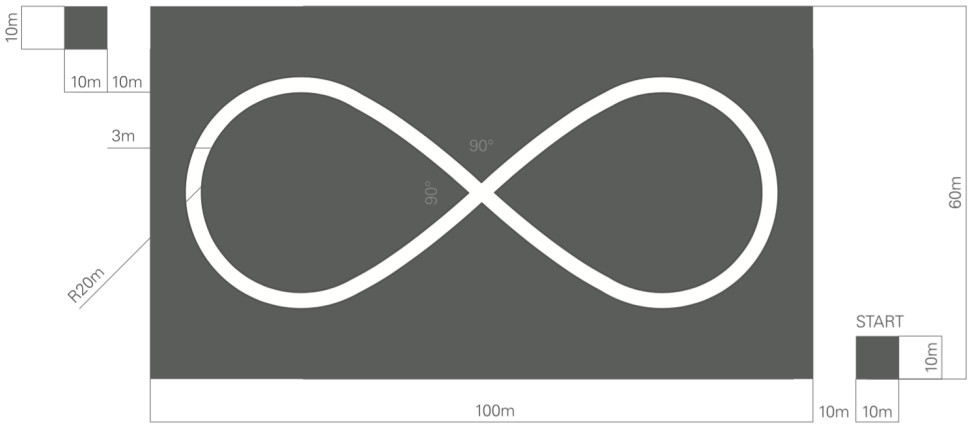


Figure 1.1: Arena of the challenge

Design of the platform

Infinity shape path In the MBZIRC challenge the moving platform will move in an infinity-shape path described in the figure 1.1. We need to describe in a mathematical way this shape in order to use this information in the prediction step of the EKF. From the specification of the challenge:

- the car is moving with constant velocity v along the path
- the radius of the circumferences that forms the trajectory is r_8m
- the path is making a cross in the middle that creates 4 angles of $\frac{\pi}{2}$

The easiest way to describe this path is to define how the angle θ is changing in function of the space.
It easy to see that the shape can be seen as a combination of a cross and two circles. The cross is simply defined as the union between the two line:

$$\begin{aligned} y &= x \\ y &= -x \end{aligned} \tag{1.1}$$

while the two circles

$$\begin{aligned} y^2 + (x - x_0)^2 &= r_8^2 \\ y^2 + (x + x_0)^2 &= r_8^2 \end{aligned} \tag{1.2}$$

It easy to see that if we want the intersections between these two functions to be exactly in the 4 points we have to choose

$$x_0 = \frac{\sqrt{2}}{2} r_8 \tag{1.3}$$

That correspond to the 4 intersections coordinate

$$\Big(\frac{\sqrt{2}}{2}r_8, \frac{\sqrt{2}}{2}r_8\Big); \Big(\frac{\sqrt{2}}{2}r_8, -\frac{\sqrt{2}}{2}r_8\Big); \Big(-\frac{\sqrt{2}}{2}r_8, -\frac{\sqrt{2}}{2}r_8\Big); \Big(-\frac{\sqrt{2}}{2}r_8, \frac{\sqrt{2}}{2}r_8\Big) \tag{1.4}$$

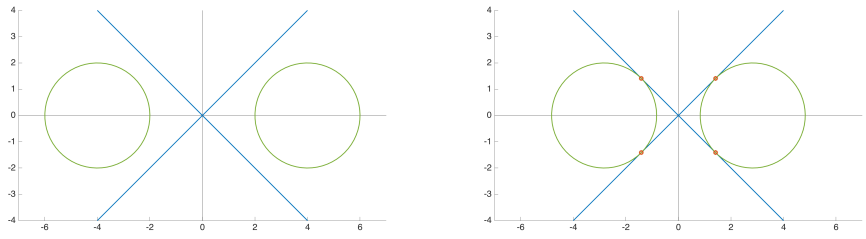


Figure 1.2: How to construct the infinity-shape path

If we travel over the two circumferences the intersections correspond to angles $\theta = \pm\frac{3\pi}{4}$.
Now it is obvious to see that the path is symmetric and it can be divided in 4 parts and describing how the angle is changing in one of this section, the whole

trajectory is defined.
We can observe that:

$$\theta(x) = \begin{cases} -\frac{x}{r_8} & x \in \left[0, \frac{3\pi}{4} r_8\right] \\ -\frac{3\pi}{4} & x \in \left[\frac{3\pi}{4} r_8, \frac{3\pi}{4} r_8 + r_8\right] \end{cases} \tag{1.5}$$

This function define a quarter of the trajectory 1.3 in function of the radius r_8 of the path.
It is now possible to use it to generate the entire trajectory $(x(t), y(t))$1.4: we know that the length of the path is $l = 4(\frac{3\pi}{4} r_8 + r_8)$ and given the constant velocity v we can calculate the time to complete the trajectory $T = \frac{l}{v}$ and it is simple to define $\theta(t)$ just stretching or shrinking $\theta(x)$.
So we can define

$$\begin{aligned} \dot{x} &= v cos(\theta(t)) \\ \dot{y} &= v sin(\theta(t)) \end{aligned} \tag{1.6}$$

And finally we have

$$\begin{aligned} x_k &= x_{k-1} + dt\big(v_{k-1} cos(\theta_{k-1})\big) \\ y_k &= y_{k-1} + dt\big(v_{k-1} sin(\theta_{k-1})\big) \end{aligned} \tag{1.7}$$

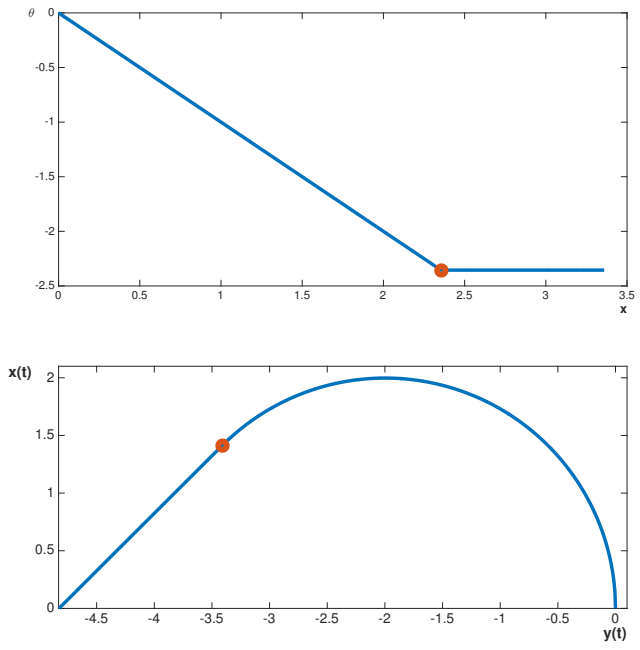


Figure 1.3: The parametrization of a quarter of the path

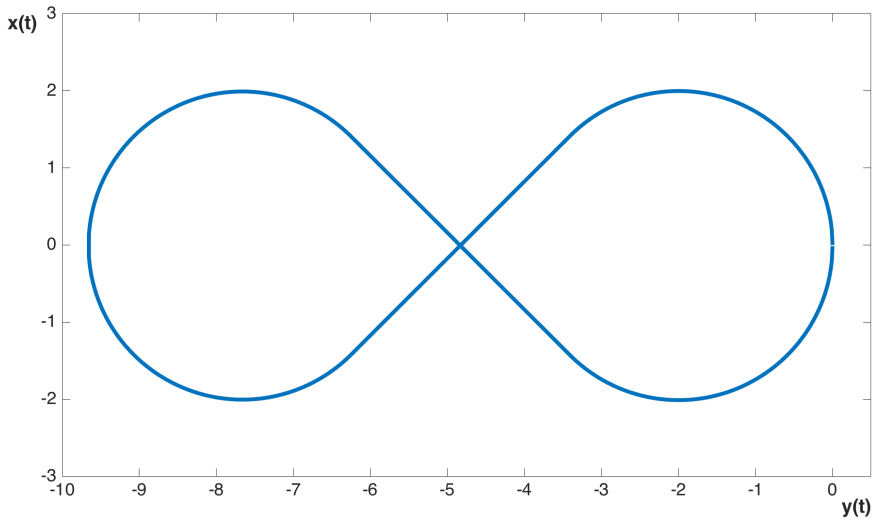


Figure 1.4: Infinity-shape path

Chapter 2

General Framework

2.0.1 SVO & MSF

are calculating the best estimation of the state of quadrotor based on data from a camera and an IMU

SVO front looking camera

explanation about base in fov
Fish eye or not fish eye? pros and cons

2.0.2 Base Detection & Base Tracking

given images taken from a camera on the quadrotor, it is searching the landing platform and estimating its state

2.0.3 Area Exploration & Trajectory Generation

considering the state of the quadrotor and of the landing platform, they are calculating where the quadrotor must go and with which trajectory.

2.0.4 Flight Control & Copilot

given the desired trajectory, they are computing the controls that must be applied to the quad in order to follow such trajectory

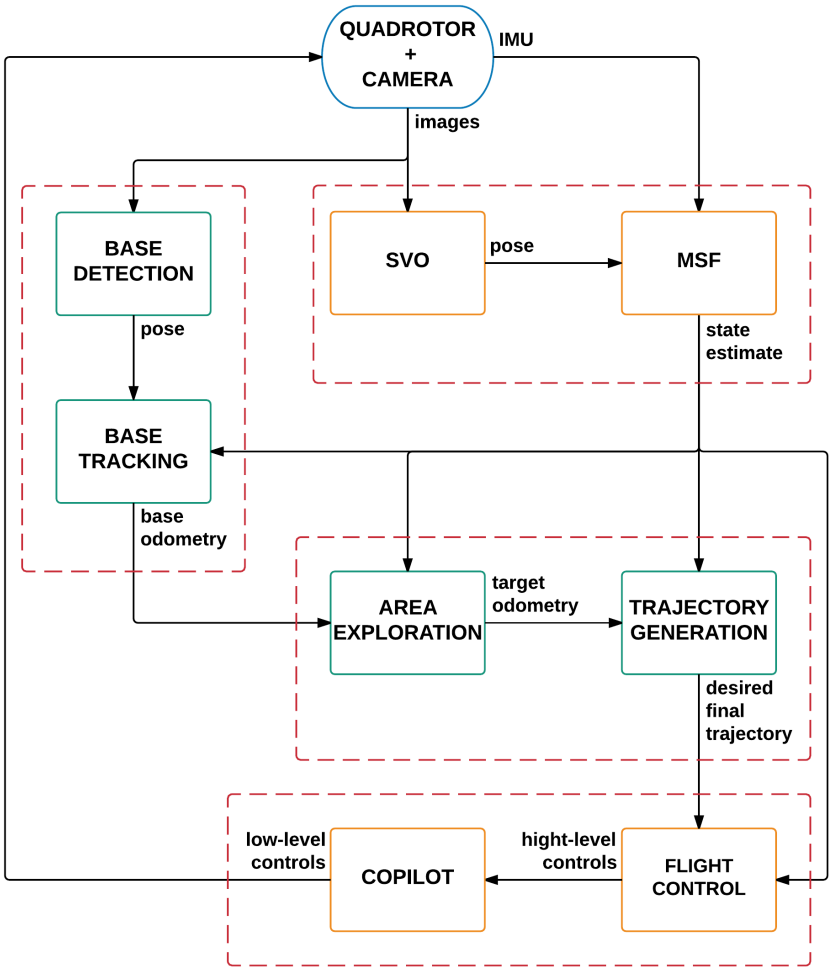


Figure 2.1: Pipeline

Chapter 3

Base Detection and Tracking

One part of the work is focused on the estimation of the state of the moving platform. In the different phases we have to use different methods to detect and track the base:

- to be able to find the platform in the minimum amount of time, at the beginning, we need to inspect the area from a very high altitude. From this height we can see just a few features of the moving car and then the pose estimation are noisy. Furthermore we do not have any assumption on the initial condition of the platform, we just know the magnitude of constant forward velocity $|v_{tan}|$, so we do not know before if at a certain time t the car is moving in a straight line or in a curve, we have to estimate it, and this is possible only tracking the platform for several seconds.
- after knowing the type of movement and a rough pose estimation of the moving car, we can use these information to improve our state estimation: getting close to the platform without loosing the tracking, starting a more precise measure (base on tag detection), and filtering the measurements with a theoretical model of the movement.

3.1 From high altitude

To find the car we assume that the platform is the only white square moving on the arena. Base on this assumption, we analyze the images from the down looking camera to find a moving white square and calculate its optical flow to predict its future position. We perform the following passages:

- threshold the image in order to find the white features.
- find all the close shapes in the image.
- select only the shapes with

- 4 edges
- convex contour
- angles between edges close to $\frac{\pi}{2}$

At this point we have the position of the four corners of all the squares in the image.
Now we try to calculate the optical flow of these points through the sequence of images and we track only the points that are moving with a velocity comparable to the one known v_{tan}.
To perform the optical flow we use the Lucas-Kanade method. It assumes that the displacement of the image contents between two nearby frames is small and approximately constant within a neighborhood of the point p under consideration. Thus the optical flow equation can be assumed to hold for all pixels within a window centered at p. Namely, the local image flow vector (V_x, V_y) must satisfy:

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{3.1}$$

Where $q_1, q_2, \ldots, q_n$ are the pixels inside the window, and $I_x(q_i), I_y(q_i), I_t(q_i)$ are the partial derivatives of the image I I with respect to position x, y and time t, evaluated at the point q_i and at the current time.
These equations can be written in matrix form $Av = b$, where

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{3.2}$$

This system has more equations than unknowns and thus it is usually overdetermined. The Lucas-Kanade method obtains a compromise solution by the least squares principle. Namely, it solves the 2×2 system:

$$\begin{aligned} A^T Av &= A^T b \\ \mathrm{v} &= (A^T A)^{-1} A^T b \end{aligned} \tag{3.3}$$

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_y(q_i) I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix} \tag{3.4}$$

With this method we can track the interesting points from frame to frame, calculate direction and velocity of the car and so predict where it will be after

a time t.

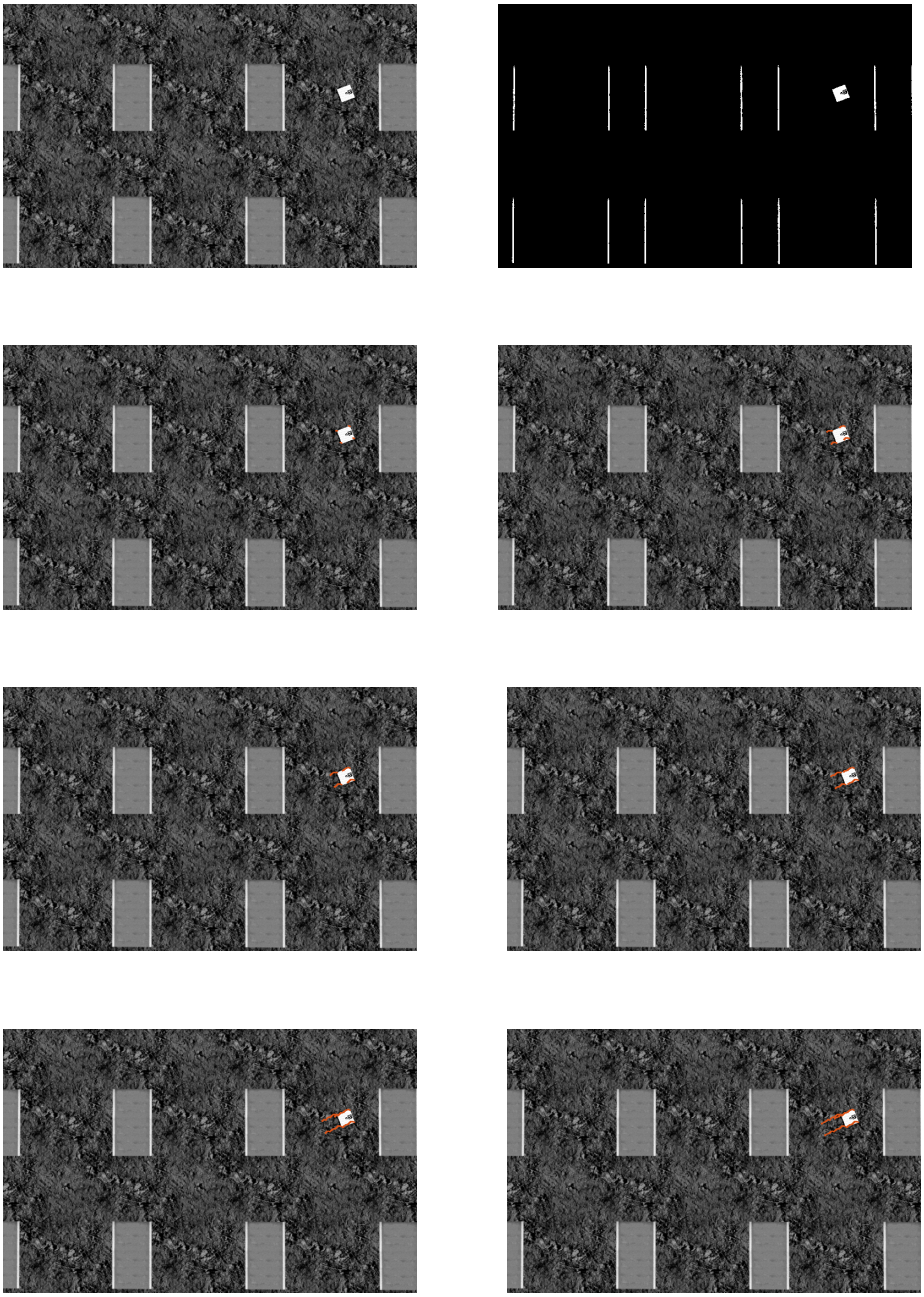


Figure 3.1: A sequence of images where the moving car is detected and tracked. First image is the original image. Then the one after thresholding. Then all the subsequent images where the corners of the platform are tracked.

From images to real world

After tracking the platform in the images, we have to find its position in the 3D real world. This position is calculate using the pinhole model of the camera:

$$wm = A[R|t]M \tag{3.5}$$

In expanded form:

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.6}$$

Where:

- m homogeneous coordinate of the projection point in pixel.
- M homogeneous coordinate of a 3D point in the world coordinate frame.
- A is the camera matrix or the matrix of intrinsic parameters. It is Composed by f_x, f_y the focal lengths and c_x, c_y the principal point.
- $[R|t]$ is the joint rotation-translation matrix or matrix of extrinsic parameters. It express the camera motion around the static scene. This matrix denote the coordinate system transformations from 3D world coordinates to 3D camera coordinates. The position C of the camera expressed in world coordinates is $C = -R^{-1}t = -R^T t$.

We can calculate the depth of the platform using the known dimension of the base: given the length l_w of the square in the real world and the average dimension of the edges in the image l_i, we can calculate the depth with respect to the camera frame

$$z = \frac{l_w f}{l_i} \tag{3.7}$$

To calculate the dimension l_i we need at least 3 corner of the base and we calculate all the pairwise distances between the corners 3.2:

- if we have 4 corners there are 6 different distances: 4 of which equal to l_i and 2 $\sqrt{2}l_i$
- if we have 3 corners there are 3 different distances: 2 of which equal to l_i and 1 $\sqrt{2}l_i$

This approximation is not really precise when we see the platform with a camera not perpendicular to the base, but we need just a rough approximation of the height in this first phase.

If this depth $z! = 0$ we can solve the system of equation 3.5 finding an unique

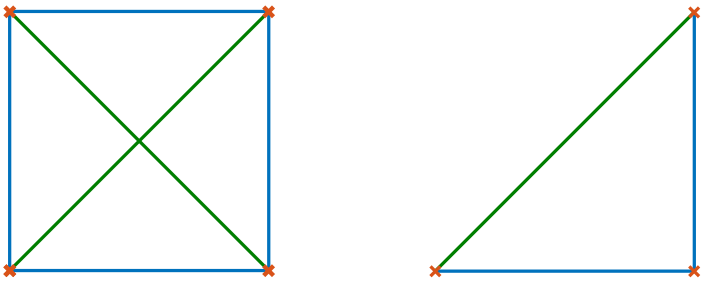


Figure 3.2: Model of the square platform detected on the image. Red crosses corner detected. Blue lines edges with length l_i. Green lines edges with length $\sqrt{2}l_i$

solution using the following equivalent equations:

$$\begin{aligned} x &= z\frac{u - c_x}{f_x} \\ y &= z\frac{u - c_y}{f_y} \\ \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \end{aligned} \tag{3.8}$$

A better method to find the position of the platform, without the approximation of the depth z is to resolve a Perspective-n-Point problem that estimates the pose of a camera given a set of n 3D points in the world and their corresponding 2D projections in the image. The only issue is that to solve this problem without ambiguity the minimum number of points is 4, and sometimes we can track only 3 corners of the base, but when all the 4 points are available we solve the correspondent PnP problem to find a better estimation of the base position.

With this method every time we detect the car we can estimate its position and velocity vector in world coordinate frame, so we can predict where the platform will be in t seconds and where the quadrotor should go to following the base.

3.2 From low altitude

3.2.1 Extended Kalman Filter

An Extended Kalman Filter is design in order to have the most reliable value of the state of the platform.

Kalman filtering is an algorithm that uses a series of noisy measurements observed over time and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each time frame.
The algorithm works in a two-step process:

- In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties, based on a model of the system:

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) + \boldsymbol{w}_k \tag{3.9}$$

- Once the outcome of the next measurement is observed:

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{3.10}$$

  these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty.

In the extended Kalman filter, the state transition and observation models don't need to be linear functions of the state but may instead be differentiable functions.
($\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the process and observation noises which are both assumed to be zero mean multivariate Gaussian noises with covariance $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ respectively. $\boldsymbol{u}_k$ is the control vector).

The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.
The Kalman filter does not require any assumption that the errors are Gaussian. However, the filter yields the exact conditional probability estimate in the special case that all errors are Gaussian-distributed.
Initialization

$$\begin{aligned} \boldsymbol{x}_{0|0} &= x_0 \\ \boldsymbol{P}_{0|0} &= P_0 \end{aligned} \tag{3.11}$$

In this case the prediction equations are continuous in time, so for the prediction step of the EKF we have to solve:

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}}(t) &= f(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) \\ \dot{\boldsymbol{P}}(t) &= \boldsymbol{F}(t)\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{F}(t)^\top + \boldsymbol{Q}(t) \end{aligned} \tag{3.12}$$

for $t \in (t_{k-1}, t_k)$ where

$$\begin{aligned} \hat{\boldsymbol{x}}(t_{k-1}) &= \hat{x}_{k-1|k-1} \\ \boldsymbol{P}(t_{k-1}) &= P_{k-1|k-1} \\ \boldsymbol{F}(t) &= \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)} \\ \hat{\boldsymbol{x}}_{k|k-1} &= \hat{\boldsymbol{x}}(t_k) \\ \boldsymbol{P}_{k|k-1} &= \boldsymbol{P}(t_k) \end{aligned} \tag{3.13}$$

In order to save some computation we can discretize the dynamicin order to have shorter computation during the prediction step of the EKF:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k-1} &= f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k) \\ \boldsymbol{P}_{k|k-1} &= \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^{\top} + \boldsymbol{Q}_k \end{aligned} \tag{3.14}$$

where the state transition matrix is defined to be the following Jacobians:

$$\boldsymbol{F}_{k-1} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k} \tag{3.15}$$

While the update equations are discrete in time and they yield to the following update step:

$$\begin{aligned} \boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\top}(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\top} + \boldsymbol{R}_k)^{-1} \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})) \\ \boldsymbol{P}_{k|k} &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1} \end{aligned} \tag{3.16}$$

where the observation matrix is defined to be the following Jacobian:

$$\boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}} \tag{3.17}$$

Prediction update: non-holonomic model

The platform is considered as a car and simulated with a non-holonomic model. In this model the state is defined as $\boldsymbol{x} = (x, y, z, \theta, v, \phi)$: It corresponds to the 3 position in a space (x, y, z) and the yaw angle of the platform (θ) w.r.t. the world frame, the forward velocity (v) and the angle of the front wheels (ϕ). The system depends on a parameter L that corresponds to the distance between the front and the back wheels.
In this model the control input are the change in velocity v and in the angle of curvature ϕ.
The equation of motion in continuous time are:

$$\begin{aligned} \dot{\boldsymbol{x}} &= f(\boldsymbol{x}, \boldsymbol{u}) \\ \\ \dot{x} &= vcos(\theta) \\ \dot{y} &= vsin(\theta) \\ \dot{z} &= 0 \\ \dot{\theta} &= \frac{v}{L}tan(\phi) \\ \dot{v} &= u_1 \\ \dot{\phi} &= u_2 \end{aligned} \tag{3.18}$$

It is possible to discretize these dynamics in $t \in (t_{k-1}, t_k)$ with a first order finite difference:

$$\dot{\boldsymbol{x}} \approx \frac{\boldsymbol{x}_k - \boldsymbol{x}_{k-1}}{dt} \approx f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k)$$

with $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$, $\boldsymbol{x}_{k-1} = \boldsymbol{x}(t_{k-1})$, $dt = t_k - t_{k-1}$

$$\begin{aligned} x_k &= x_{k-1} + dt\big(v_{k-1} cos(\theta_{k-1})\big) \\ y_k &= y_{k-1} + dt\big(v_{k-1} sin(\theta_{k-1})\big) \\ z_k &= z_{k-1} \\ \theta_k &= \theta_{k-1} + dt\Big(\frac{v_{k-1}}{L} tan(\phi_{k-1})\Big) \\ v_k &= v_{k-1} + dt\big(u_{1k}\big) \\ \phi_k &= \phi_{k-1} + dt\big(u_{2k}\big) \end{aligned} \tag{3.19}$$

In order to solve the former system, we have anyway to find a numerical solution. For this purpose we use a RUNGE-KUTTA scheme

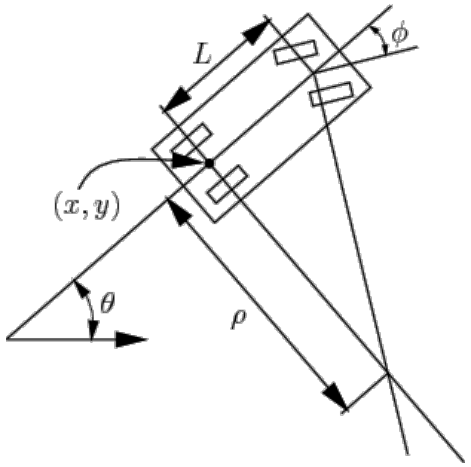


Figure 3.3: Non-holonomic model

Straight and circular path For now we assume the input u_1 and u_2 are equal to zero, so the platform can be static ($v_f(0) = 0$) can move in a straight line ($v_f(0) \neq 0$ and $\phi(0) = 0$) or in a circle ($v_f(0) \neq 0$ and $\phi(0) \neq 0$)

3.2.2 Measurement update

Tag Detector

April Tag vs Ar Sys precision comparison vs frequency nodlet

Cross Detector

pnp problem

Covariance Estimation

3.3 Results

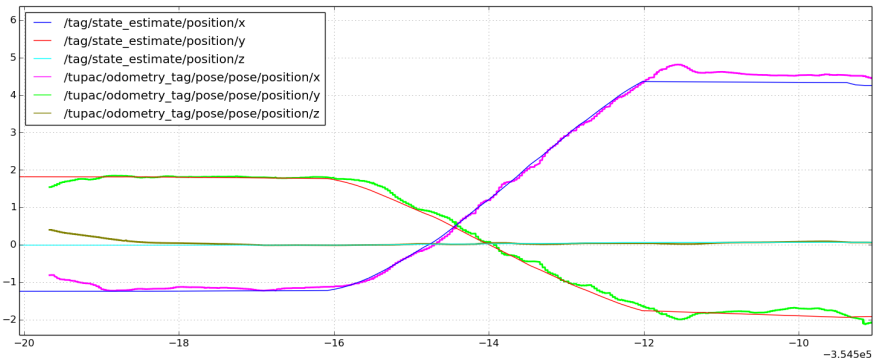


Figure 3.4: EKF 1

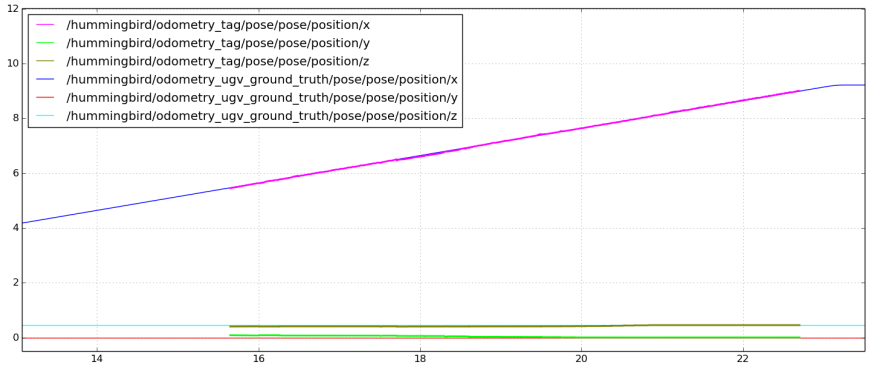


Figure 3.5: EKF 2

Chapter 4

Area exploration

different time exploration imu to switch off time following

jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg
jksdbkvbdsjg

4.1 Phases

4.1.1 First phase - Searching for the base

In this phase the quadrotor starts from a given position and has to find the moving car. Given the rectangle in which the platform can move the UAV follows a list of way-points in order to span the whole area at high altitude. In this way the downward camera can collect information from a large section of the space and the searching task can be performed faster.

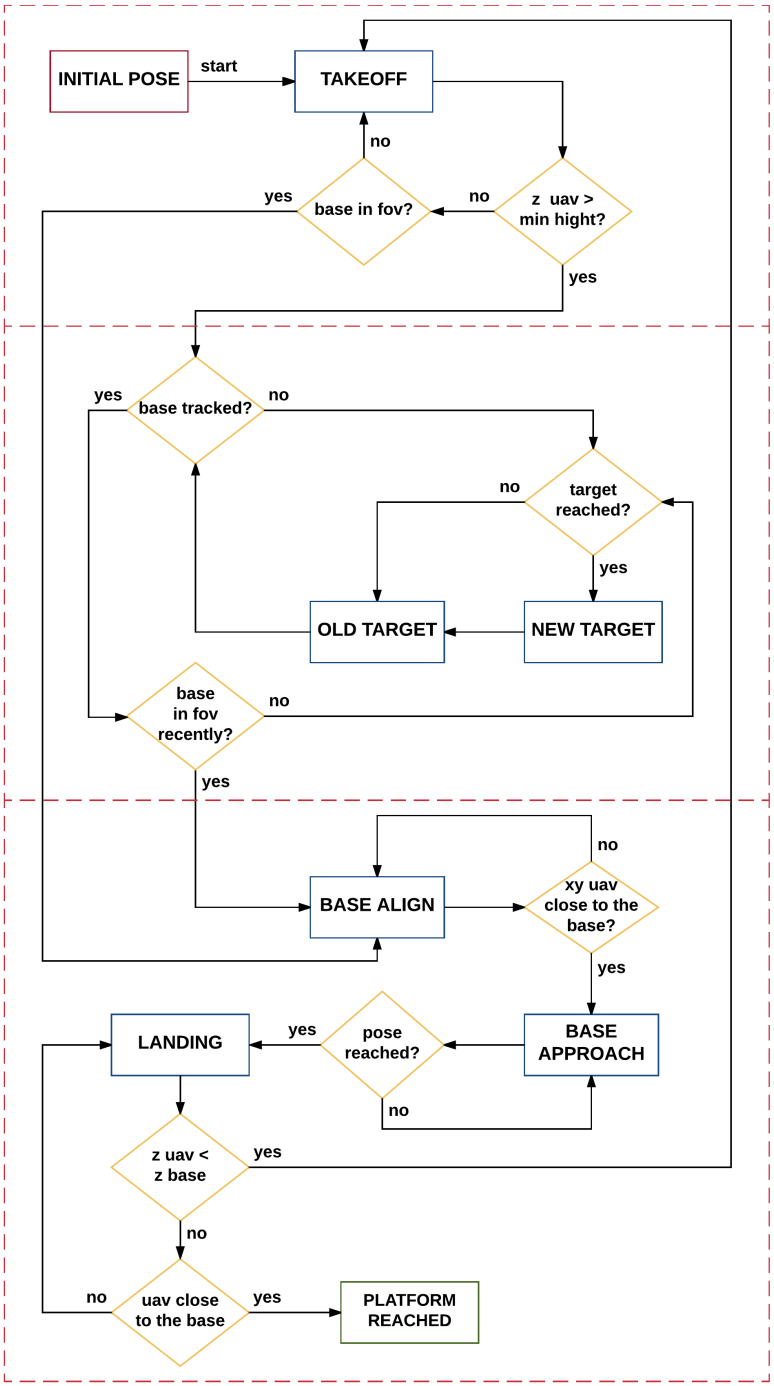


Figure 4.1: Area Exploration

Understand type of movement

From the challenge description 1.1 we know that the car is moving in a shape composed by straight lines and circumference sectors. We need to understand in which part is the platform at a given time: this information is important in order to calculate properly where the platform will be in t seconds and because we want to proceed with the following phases, and finally perform the landing maneuver, when the platform is going straight.
To understand the trajectory of the moving base we collect all the estimated positions of the base, and we perform a linear regression on the last n estimations: the platform is moving in a straight line if the linear regression is a good approximation of the data trends otherwise it driving in a curve.

We have a series of n points, each of these is consider as a pair of coordinates (x_i, y_i), and we are searching for he best-fit line that can describe the data as a linear function:

$$y = mx + q$$

(we perform the following analysis considering before the coordinates y_i as dependent variables, then x_i, and we are peaking the best of this two fits).
We want find the best best parameters m and q, and to do so we need to have some measure of quality to optimize. Unless all our n points are already in a perfect line (trivial solution), there will be an error between the value predicted by the line, and the observed dependent variable:

$$e_i = y_i - (mx_i + q)$$

These differences are called residuals and what we want is to find a line that minimizes:

$$\sum_{i=1}^{n} e_i^2$$

The model we find is the Least Squares Fit of the data.
We define also the cumulative residual as:

$$e_{tot} = \sqrt{\sum_{i=1}^{n} e_i^2}$$

The parameters m and q of the model, are found where e_{tot}^2 is minimized:

$$\begin{aligned} \frac{e_{tot}^2}{\partial m} &= 0 \\ \frac{e_{tot}^2}{\partial q} &= 0 \end{aligned} \tag{4.1}$$

It is easy to demonstrate that the solution of 4.1 is:

$$\begin{aligned} m &= \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \\ q &= \frac{\sum_{i=1}^{n} y_i}{n} - m\frac{\sum_{i=1}^{n} x_i}{n} \end{aligned} \tag{4.2}$$

The platform is moving in the straight line if the cumulative residual e_{tot} is below a threshold th_{line}, while if the error is above th_{curve} the base is following the circumference.

To have a good interpretation of the data it is important to decide the three parameters n, t_{line}, t_{curve} correctly.

- The first parameter n is the number of samples to consider when we perform the linear regression. We chose it in order to consider poses that are along a curve with length

$$l_{curve} = \frac{r_8 \pi}{4} \tag{4.3}$$

We know the forward constant velocity of the car v_{tan}, so we can calculate the time in which the platform is performing the curve

$$t_{curve} = \frac{l_{curve}}{v_{tan}} \tag{4.4}$$

When we receive a pose at time t_i we store it and we perform the linear regression with all the data stored from $[t_i - t_{curve}, t_i]$.

- The threshold parameters are calculating considering that each measure is corrupted by an additive Gaussian noise with 0 mean and σ_e^2 variance:

$$\tilde{y_i} = \mathcal{N}(y_i, \sigma_e^2)$$

When we perform the linear regression on the measured data, the average residual square is

$$\begin{aligned} < \tilde{e_i}^2 > = \\ < (\tilde{y_i} - (mx_i + q))^2 > = \\ < \tilde{y_i}^2 - 2\tilde{y_i}(mx_i + q) + (mx_i + q)^2 > = \\ < \tilde{y_i}^2 > -2 < \tilde{y_i} > (mx_i + q) + (mx_i + q)^2 = \\ \sigma_e^2 + y_i^2 - 2y_i(mx_i + q) + (mx_i + q)^2 = \\ \sigma_e^2 + e_i^2 \end{aligned} \tag{4.5}$$

  - when we perform the linear regression on linear data the theoretical residual is

$$e_i = 0$$

And the average residual squares on the measured data

$$< \tilde{e_i}^2 >= \sigma_e^2$$

.

The parameter th_{line} is then:

$$th_{line} = \sqrt{\sum_{i=1}^{n} \tilde{e_i}^2} = \sqrt{\sum_{i=1}^{n} \sigma_e^2} = \sigma_e \sqrt{n} \tag{4.6}$$

– when we perform the linear regression on data along a circumference arch with radius ρ and angles $\theta_i \in [\theta_1, \theta_2]$ the theoretical data are distributed as:

$$(\rho\cos\theta_i, \rho\sin\theta_i)$$

so the theoretical residual is:

$$e_i = \rho\sin\theta_i - (m\rho\cos\theta_i + q)$$

To find m and q we use equation 4.2, but we want a general approximation of these values. To do so, we have to consider all the sums in the equations as integrals, using the relation 4.7

$$\begin{aligned} \lim_{n\to\infty} \frac{b-a}{n}\sum_{i=0}^{n} f(x_i) &= \int_a^b f(x)\mathrm{d}x \\ \sum_{i=0}^{n} f(x_i) &\simeq \frac{n}{b-a}\int_a^b f(x)\mathrm{d}x \end{aligned} \tag{4.7}$$

So now if we calculate this approximation for our values we have:

$$\begin{aligned} \sum_{i=1}^{n} x_i y_i &= \sum_{i=1}^{n} \rho^2\cos\theta_i\sin\theta_i \\ &\simeq \frac{n}{\theta_2-\theta_1}\rho^2\int_{\theta_1}^{\theta_2}\cos x\sin x\mathrm{d}x = \frac{n}{\theta_2-\theta_1}\frac{\rho^2}{2}\Big[-\cos^2 x\Big]_{\theta_2}^{\theta_2} \\ \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} \rho\cos\theta_i \\ &\simeq \frac{n}{\theta_2-\theta_1}\rho\int_{\theta_1}^{\theta_2}\cos x\mathrm{d}x = \frac{n}{\theta_2-\theta_1}\rho\Big[\sin x\Big]_{\theta_2}^{\theta_2} \\ \sum_{i=1}^{n} y_i &= \sum_{i=1}^{n} \rho\sin\theta_i \\ &\simeq \frac{n}{\theta_2-\theta_1}\rho\int_{\theta_1}^{\theta_2}\sin x\mathrm{d}x = \frac{n}{\theta_2-\theta_1}\rho\Big[-\cos x\Big]_{\theta_2}^{\theta_2} \\ \sum_{i=1}^{n} x_i^2 &= \sum_{i=1}^{n} \rho^2\cos^2\theta_i \\ &\simeq \frac{n}{\theta_2-\theta_1}\rho^2\int_{\theta_1}^{\theta_2}\cos^2 x\mathrm{d}x = \frac{n}{\theta_2-\theta_1}\frac{\rho^2}{2}\Big[x+\cos x\sin x\Big]_{\theta_2}^{\theta_2} \end{aligned} \tag{4.8}$$

In our case we consider pieces of curve with length l_{curve} 4.3, that corresponds to a circumference arch with:

$$\rho = r_8 \qquad\qquad \theta_i \in \left[0, \frac{\pi}{4}\right] \tag{4.9}$$

We can now calculate the approximate values of m and q using 4.2 4.8 4.9:

$$m = \frac{nr_8^2\frac{n}{\pi} - r_8\frac{n2\sqrt{2}}{\pi}r_8\frac{n2(2-\sqrt{2})}{\pi}}{n\frac{nr_8^2(2+\pi)}{2\pi} - (r_8\frac{n2\sqrt{2}}{\pi})^2} = \frac{2\pi - 16\sqrt{2} + 16}{\pi^2 + 2\pi - 16}$$
$$q = \frac{r_8\frac{n2(2-\sqrt{2})}{\pi}}{n} - m\frac{r_8\frac{n2\sqrt{2}}{\pi}}{n} = r_8\frac{4 - 2\sqrt{2}(m+1)}{\pi} = r_8\bar{q} \tag{4.10}$$

Now we can calculate the theoretical average residual square, using again the approximations 4.8:

$$\begin{aligned} < e_i^2 > &= \frac{\sum_{i=1}^{n}\Big(r_8\sin\theta_i - (mr_8\cos\theta_i + q)\Big)^2}{n} \\ &= \frac{\sum_{i=1}^{n}\Big(r_8^2\sin^2\theta_i - 2r_8\sin\theta_i(mr_8\cos\theta_i + q) + (mr_8\cos\theta_i + q)^2\Big)}{n} \\ &= \frac{\sum_{i=1}^{n} r_8^2\xi}{n} = r_8^2\xi \\ \xi &= \frac{\pi - 2 + m^2(\pi+2) + 2\pi\bar{q}^2 - 4m - 8\bar{q}(2-\sqrt{2}) + 8m\bar{q}\sqrt{2}}{2\pi} \end{aligned}$$

Finally we calculate

$$< \tilde{e_i}^2 >=< e_i^2 > +\sigma_e^2 = \rho^2\xi + \sigma_e^2$$

.
The parameter th_{curve} is:

$$th_{curve} = \sqrt{\sum_{i=1}^{n}\tilde{e_i}^2} = \sqrt{\sum_{i=1}^{n}\rho^2\xi + \sigma_e^2} = \sqrt{n}(\sigma_e + \rho\sqrt{\xi}) \tag{4.11}$$

The figure 4.2 shows the typical evolution of the total residual during this first phase: the different phases of linear and circular movement can be detect in the graph. Furthermore the point of regime change can be seen both in figure 4.2 and in the map 4.3 in which also all the estimate positions of the base are plotted.

Calculate future position

Now knowing the platform regime of movement at a specific time we can estimate correctly where it will be after t_s seconds and proceed with the following phases when it starts a straight portion of the trajectory.
Thanks to the algorithm described before we can estimate that at the moment t_0 the car is at position (x_0, y_0) with a direction angle of θ_0 and forward velocity of v_{tan}, so at time $t_1 = t_0 + t_s$ seconds the car will be at position (x_1, y_1) with an angle θ_1, and it has traveled $v_{tan}t_s$.

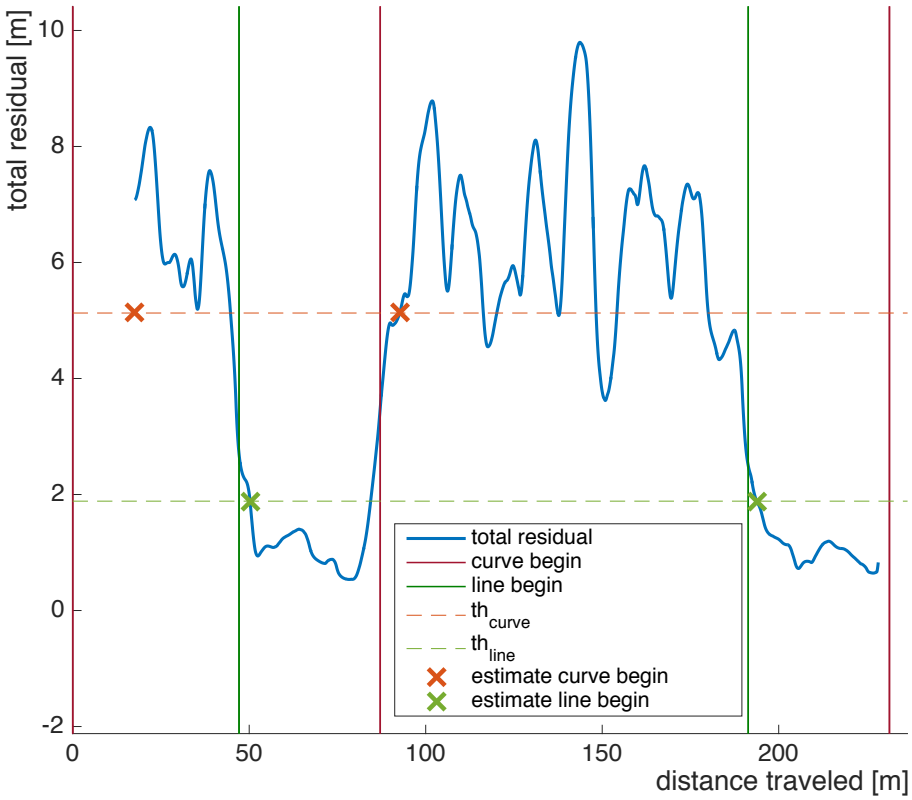


Figure 4.2: Evolution of the total residual during this first phase (in blue). The vertical lines are the real moments in which the car changes movement types: green a linear phase starts, red a circular phase begins. The horizontal lines are the thresholds for the detection of the two different regimes. The cross are the moment in which the algorithm understands the change.

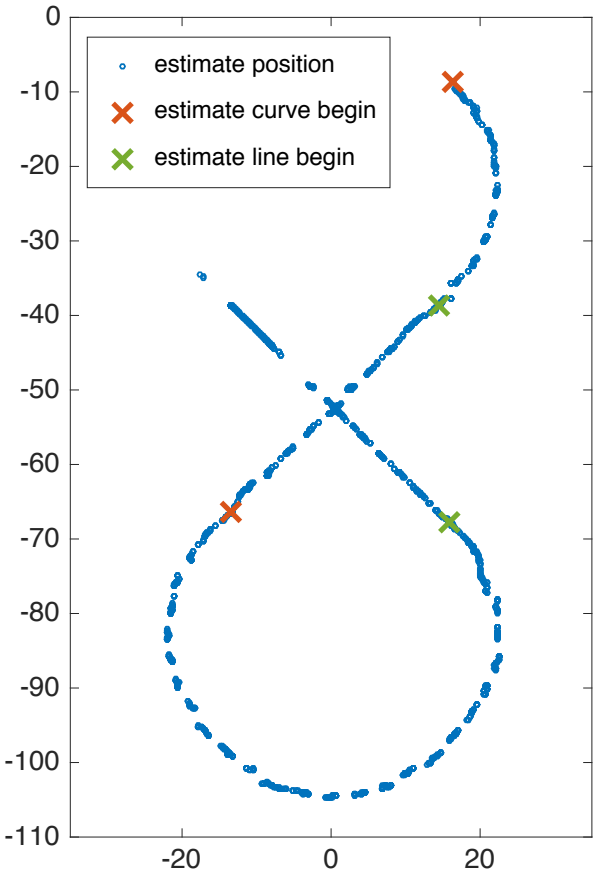


Figure 4.3: Map of the estimated position of the car in blue. The cross are the moments in which the algorithm understands the change. Red crosses from line to curve. Green crosses from curve to line.

- When no regime is found (at the begging) or when a line movement is detect, the predicted state is simply

$$\begin{cases} x_1 &= x_0 + v_{tan} t_s \cos\theta_0 \\ y_1 &= y_0 + v_{tan} t_s \sin\theta_0 \\ \theta_1 &= \theta_0 \end{cases} \tag{4.12}$$

- When a movement in the circumference is detected we have to perform some calculations in order to find the final state of the platform.
  First of all we use the relation

$$l_{curve} = \rho|\beta_s|$$

  To find the angle β_s that the platform will span in t_s seconds. In our case:

$$\begin{aligned} v_{tan} t_s &= r_8|\beta_s| \\ |\beta_s| &= \frac{v_{tan} t_s}{r_8} \end{aligned} \tag{4.13}$$

  The final angle will be:

$$\theta_1 = \theta_0 + \beta_s \tag{4.14}$$

  The segment connecting (x_0, y_0) and (x_1, y_1) has 4.4:

  – direction θ_{chord} found as bisection between θ_0 and θ_1

$$\theta_{chord} = \theta_0 + \frac{\theta_0 + \theta_1}{2} \tag{4.15}$$

  – length l_{chord}, found with the chord theorem:

$$l_{chord} = 2r_8 \sin\frac{|\beta_s|}{2} \tag{4.16}$$

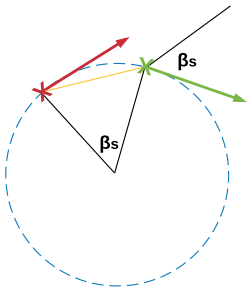


Figure 4.4: In red position of initial state at time t_0. In green final estimate state at time t_1. In yellow the chord between the two states with length l_{chord} and orientation θ_{chord}.

In order to properly find the final point (x_1, y_1) we have to resolve an other last problem: both β_s and $-\beta_s$ span a curve of length $v_{tan}t$, and due to the symmetry of our trajectory is impossible to know before which angle is the right one.
What we can do is calculate both the two possible final states using 4.13 4.15 4.16:

$$\begin{cases} x_1^a &= x_0 + l_{chord} \cos\left(\theta_0 + \frac{|\beta_s|}{2}\right) \\ y_1^a &= y_0 + l_{chord} \sin\left(\theta_0 + \frac{|\beta_s|}{2}\right) \\ \theta_1^a &= \theta_0 + |\beta_s| \end{cases} \tag{4.17}$$

$$\begin{cases} x_1^b &= x_0 + l_{chord} \cos\left(\theta_0 - \frac{|\beta_s|}{2}\right) \\ y_1^b &= y_0 + l_{chord} \sin\left(\theta_0 - \frac{|\beta_s|}{2}\right) \\ \theta_1^b &= \theta_0 - |\beta_s| \end{cases} \tag{4.18}$$

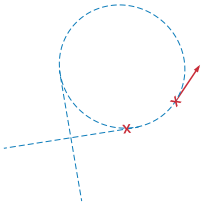
In order to understand which is the correct state we can calculate the distance between the two possible final points and a point of the trajectory estimated at time $t_{-\alpha} < t_0$. For sure the state with smaller distance will be the right final state, because the wrong one leads to a position further away.
The images 4.5 show all the passages we perform to find the right final state.

The image 4.6 shows where the algorithm calculates the way points for the quadrotor in order to following the moving car.

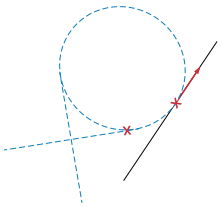
Following the base and proceed with following phase

At this point we can use the predict position of the platform to control the quadrotor in order to following the base and at the right moment proceed with the other phases.
The right moment to perform the landing is at the start of a line segment:

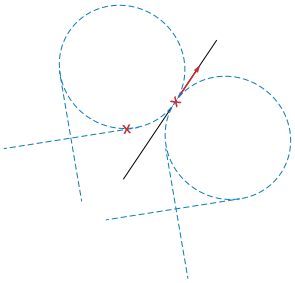
- if we first detect the base and we understand that it is moving in a circumference we cannot land, we can follow the base and waiting when we will detect a change in the regime from curve to line. At this point we can proceed with the landing.
- if we first detect the base and we understand that it is moving in a straight line we should not land, because we do not know when it actually started the line, so it can be almost at the end of it, and we do not have time to perform the entire landing maneuver. What we do is following the car and waiting when it changes from the line movement to the curve. At this point we can calculate where the next change point will be:
  - we know the orientation of the straight line portion just finished and it is θ_{line}



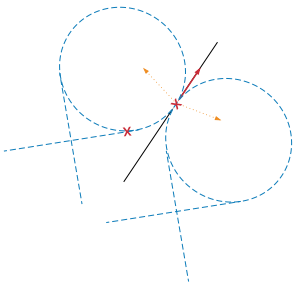
(a) Red cross with arrow: state a t_0. Red arrow: current velocity vector. Red cross: previous position a $t_{-\alpha}$.



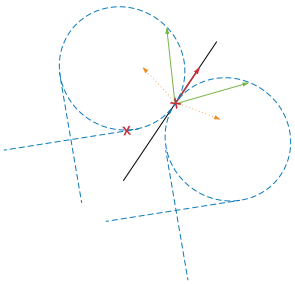
(b) Black line: direction of the velocity vector, with angle θ_0. We have symmetry with respect to this axes.



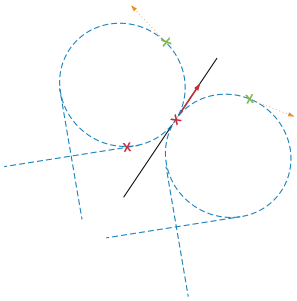
(c) Blue lines: real and symmetric path. We do not know which of the two trajectories is correct.



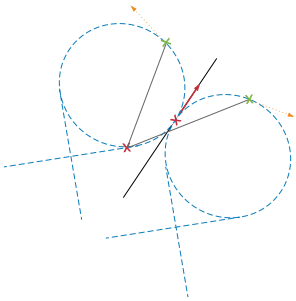
(d) Yellow arrows: estimated future angles θ_1^a and θ_1^b that the base will assume at time t_1.



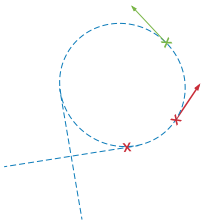
(e) Green arrows: segment from state at t_0 and possible states at t_1: with angles θ^a_{chord} and θ^b_{chord} and length l_{chord}



(f) Green crosses: future possible states of the platform at positions (x_1^a, y_1^a) and (x_1^b, y_1^b) .



(g) Dark grey lines: distances from previous position at $t_{-\alpha}$ and possible future positions at t_1.



(h) Green cross with arrow: future state selected taking the position with minimum distance.

Figure 4.5: The sequence of passages computed in order to select the future position when the platform is moving on the circumference.

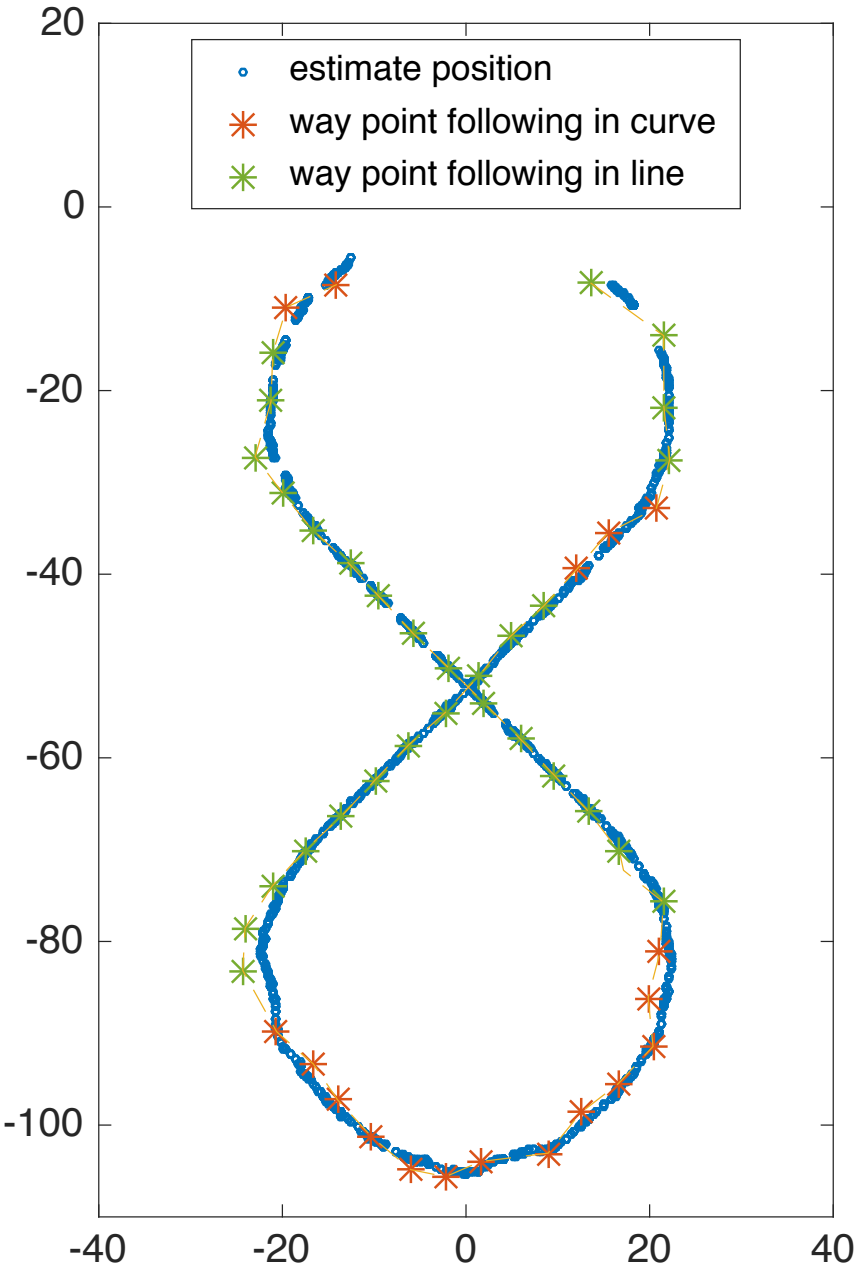


Figure 4.6: Data from the first phase of the area exploration. Blue circles position of the base estimate. Stars the position in which the quadrotor should go to following the base until a proper moment to proceed with the following phases is detected.

– given the point of change between line and curve, the future point will be in the circumference after an angle of $\left|\frac{3\pi}{2}\right|$.

– from equations 4.15 4.16 we know that the segment connecting the change point and the future intersection point has length $\sqrt{2}r_8$ and angle $\theta_{line} \pm \frac{3\pi}{4}$

– we can apply the same method described before to find the two possible intersection points:

$$\begin{cases} x^a_{intersection} &= x_{changing} + \sqrt{2}r_8 \cos\left(\theta_{line} + \frac{3\pi}{4}\right) \\ y^a_{intersection} &= y_{changing} + \sqrt{2}r_8 \sin\left(\theta_{line} + \frac{3\pi}{4}\right) \\ \theta^a_{intersection} &= \theta_{line} + \frac{3\pi}{2} \end{cases} \tag{4.19}$$

$$\begin{cases} x^b_{intersection} &= x_{changing} + \sqrt{2}r_8 \cos\left(\theta_{line} - \frac{3\pi}{4}\right) \\ y^b_{intersection} &= y_{changing} + \sqrt{2}r_8 \sin\left(\theta_{line} - \frac{3\pi}{4}\right) \\ \theta^b_{intersection} &= \theta_{line} - \frac{3\pi}{2} \end{cases} \tag{4.20}$$

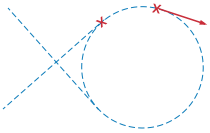
and select the right one with minimum distance with the current estimate position of the platform. The images 4.7 show all the passages we perform to find the right intersection point.

Now that we know where the next change of regime will be, we can follow the base and continue with the following phases when the base is close to the intersection point.

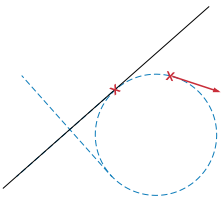
4.1.2 Second phase - Approaching the base

4.1.3 Second phase - Following the base

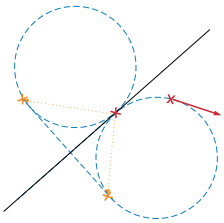
4.1.4 Second phase - Landing on the base



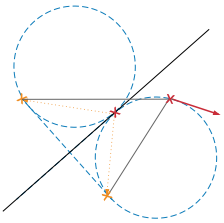
(a) Red cross with arrow: state a t_0. Red arrow: current velocity vector. Red cross: changing point from line to curve.



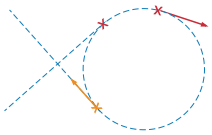
(b) Black line: direction of the line sector just finished. The direction is taken as the slope of the best linear fit found in the previous regime.



(c) Blue lines: real and symmetric path. We do not know which of the two trajectories is correct. Yellow crosses: in both the path we can calculate the future intersection point.



(d) Dark grey lines: distances from current position and the two possible future intersections. Both are eligible becouse of the symmetry of the trajectory.



(e) Yellow cross with arrow: future intersection point selected taking the position with minimum distance from the current state.

Figure 4.7: The sequence of passages computed in order to select the future intersection point where the platform will start the movement in line.

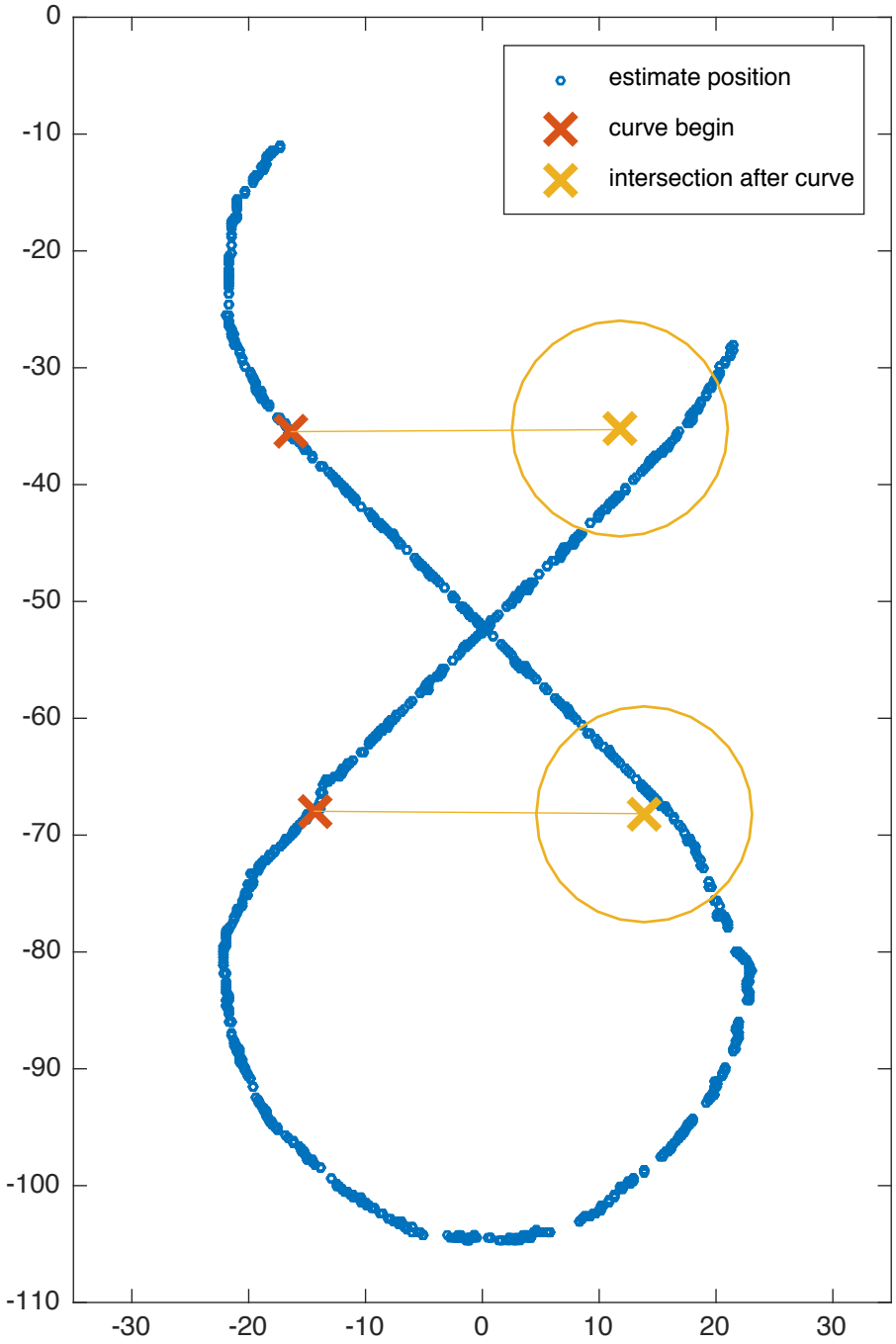


Figure 4.8: Data from the first phase of the area exploration. Blue circles position of the base estimate. Red cross the points in which the algorithm detect a passage from a line phase to a curve. Green cross from a curve to a line. Yellow cross the position where the quadrotor should go in order to intersect the platform when is about to start a line phase.

Chapter 5

Trajectory Generator

5.1 Base trajectory prediction

5.2 Rapid Trajectory

how to compute the acceleration... comparision between diff imu and thrust

Compute the acceleration

The Rapid trajectory generator needs an initial and a final state. The initial state is always selected as the current position velocity and acceleration of the quadrotor. From the state estimate of MSF we have the first two information, while we have to compute the acceleration.
There are several ways to make this calculation:

- IMU measurements:
- finite difference of velocity:
- total thrust:

5.3 Results

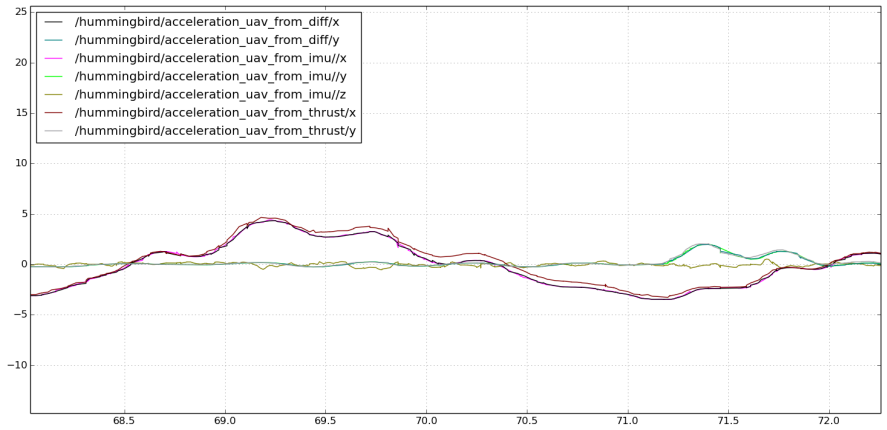


Figure 5.1: Comparison Acceleration

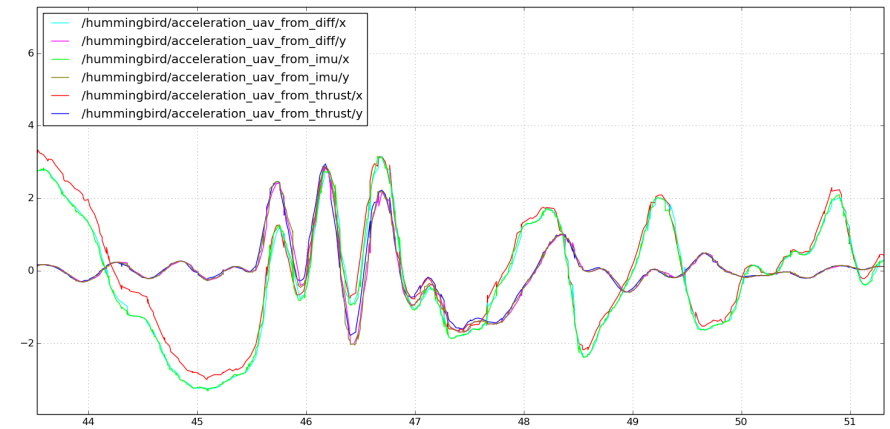


Figure 5.2: Comparison Acceleration Different Mass

Chapter 6

Nonlinear Model Predictive Control

6.1 Model

6.2 Cost Function

6.3 Acado Library

6.4 Learning

6.5 Results

Chapter 7

Experiments

Provide numerical results, plots, and timings. Interpret the data.

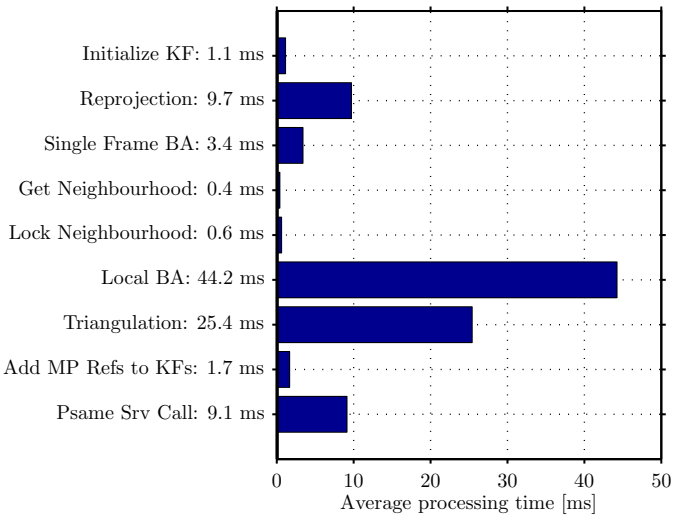


Figure 7.1: Example of a figure.

Chapter 8

Scientific Writing

This chapter gives you some tips on how to write scientifically. It should prevent you from making the most common mistakes people do and help you with creating a well written report.

8.1 General Style

- A report/paper is not a short-story. There is no build-up to a climax. The climax should be in the abstract. Even better, in the title.
- Hierarchical exposition, not linear: this goes in hand with the previous point. A hierarchical exposition means that you start with the core of your work (The main thing your project was about) and then go into details in following sections. Do not build up to the core of your work with too much background/preliminaries as it would be the case in a linear exposition.
- At the beginning of every chapter/major section, you should summarize what the content of the section will be. A person should get a good sense of the report by reading the first paragraph of each section.
- Express your thoughts succinctly. Avoid unnecessary words or phrases and be precise and specific.
- Definitions are useful if they are used often. Do not define something if it is only used once.
- Be generous with your references. Do not compare your results with others by pointing the deficiencies of their work; rather, state how your results are adding to the body of knowledge others have created.
- Notation is extremely important. Good notation facilitates understanding. You do not want the reader to mentally perform translations every time they see a symbol.

8.2 Important Stuff

- Use active verb tense whenever possible: instead of *An analysis of the signal noise is performed using a discrete Fourier transform.* write **We perform an analysis of the signal noise using a discrete Fourier transform.**
- Make short sentences with one statement. Long sentences with multiple statements are complicated and hard to understand. Write to be understood, not to impress!
- Be concrete/specific: instead of *We use a model to predict the state* write **We use a linear model of the attitude dynamics to predict the quadrotor's state at time** $t + \Delta t$.
- Be precise: instead of *We assume the model to be linear*, say **We design a linear model of the system dynamics**. (You assume the *system dynamics* to be linear and hence you create a linear model.)
- Be consistent: this basically applies to every level. Denote the same thing always with the same word, create figures with a similar style, etc.
- Do not make unsubstantiated statements. Do not use *It is common knowledge* or *Several researchers have shown.* Instead use constructs like **Recently, several researchers [?, ?] have shown**.

8.3 Small Things

- Do not use *don't*, *aren't*, etc., use **do not** and **are not**.
- Do not use words like *simply*, *highly*, *just*, *very*, etc.
- Use **because** instead of *due to the fact that*, **to** instead of *in order to*, etc.
- When referencing to figures, sections, etc., use capital letters: see **Figure** 1, see **Section** 2.
- Every figure must be referenced in the text.
- Use $\sim$ to make spaces which LaTeX must not separate: Figure$\sim$`\ref{fig:bla}`. This avoids having the word and the number on different lines.
- Put punctuation marks after each formula as if they were text. Separate multiple consecutive formulas by commas and put a dot if you start a new sentence after the formula. For more details, see Section 9.4.
- Use `\left(` and `\right)` when you have mathematical expressions that are higher than normal brackets, e.g., $\left(\frac{pV}{RT}\right)$ instead of $(\frac{pV}{RT})$.
- Avoid brackets. If something is important enough to be mentioned it does not need brackets; if not, it does not need to be mentioned at all.
- In English, after a colon (:) you continue with small letters.
- Use *we* to refer to yourself, i.e. **We** developed an algorithm to ...

- Do not use *ours*.
- Number all equations.
- Put details in an appendix.
- Avoid single-sentence paragraphs.

Chapter 9

LaTeX Tips and Tricks

In this chapter, we show some useful tips and tricks when working with LaTeX.

9.1 Using Git

We recommend you to use *Git* also for your LaTeX files such as this report. If you do so, we suggest to write every sentence in your TeX file on a new line. This will make it easier to keep track of changes since *Git* tracks them line by line. So if you change one sentence, *Git* will tell you that only that sentence has changed instead of the entire paragraph otherwise. Furthermore, if you are using the PDF viewer of *texmaker*, you can jump from the PDF directly to the sentence in the TeX file by clicking on it (instead of just jumping to the corresponding paragraph).

9.2 Headings

Your report can be structured using several different types of headings. Use the commands `\chapter{.}`, `\section{.}`, `\subsection{.}`, and `\subsubsection{.}`. Use the asterisk symbol `*` to suppress numbering of a certain heading if necessary, for example, `\section*{.}`.

9.3 References

References to literature are included using the command `\cite{·}`. For example [**?**, **?**]. Your references must be entered in the file `bibliography.bib`. Making changes or adding new references in the bibliography file can be done manually or by using specialized software such as *JabRef* which is free of charge.

Cross-referencing within the text is easily done using `\label{·}` and `\ref{·}`. For example, this paragraph is part of Chapter 9; more specifically on page 38.

9.4 Writing Equations

The most common way to include equations is using the `equation` environment. Use `\eqref{·}` to reference an equation, e.g., (9.1).

Embed equations in the text. Thus you must use proper punctuation. You must introduce all symbols that you use. You should define these before you use them. However, they must be introduced in the same sentence at the latest.

Example 1

For n detections and m LEDs on the object, we will obtain N pose candidates,

$$N = 4\alpha \binom{n}{3} \frac{m!}{(m-3)!}, \tag{9.1}$$

where $\alpha \in \{1, 2\}$ is a magic factor.

Example 2

The transformation matrix in homogeneous coordinates, $\mathtt{T}$, is composed of the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{p}$,

$$\mathtt{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix}, \qquad \text{with} \quad \mathbf{R} \in SO(3), \quad \mathbf{p} \in \mathbb{R}^3. \tag{9.2}$$

9.5 Including Graphics

The easiest way to include figures in your document is to use PDF figures if you use `pdflatex` to compile. Figure 9.1 was created with the use of the open-source program `ipe`.

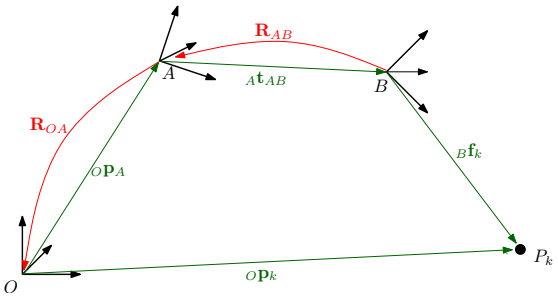


Figure 9.1: Example of a figure.

9.6 Including Matlab Figures

When including figures into your report you want them as a vector graphic such that you can zoom into the figure without getting blurry. Furthermore it is nice when the text in the figure gets substituted by LaTeX such that you have the same font and the same font size. Figure 9.2 shows an example of such an imported matlab figure. An easy way of achieving this is by using the `matlab2tikz` script. You can find a short example on how to use this script in the `matlab_figures` folder. The `create_figures.m` script creates a plot and then the tikz file which you can include in your document. For using tikz, you need to make use of the `pgfplots` package in your TeX document. More information on using `matlab2tikz` can be found on Matlab Central where you can also download the necessary files (`matlab2tikz.m`, `matlab2tikzInputParser.m`, `updater.m`).

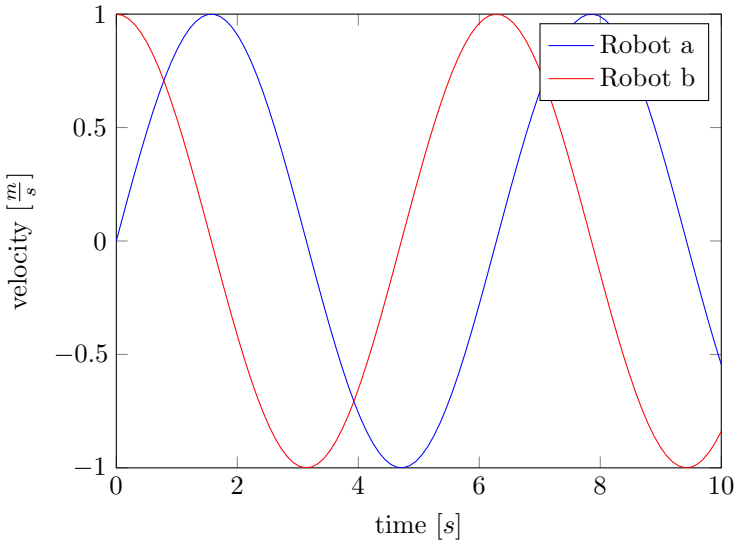


Figure 9.2: Example figure created with `matlab2tikz`.

An alternative which you might want to consider is `matlabfrag` and `mlf2pdf`. Especially when there are many data points in your figure you might run into problems when using tikz. Again, you can find a short example on how to use `mlf2pdf` in the `create_figures.m` scriptin in the `matlab_figures` folder. This script makes use of the two functions `matlabfrag.m` and `mlf2pdf.m` to create a PDF which you can then include into matlab. These two files can be downloaded here and here.

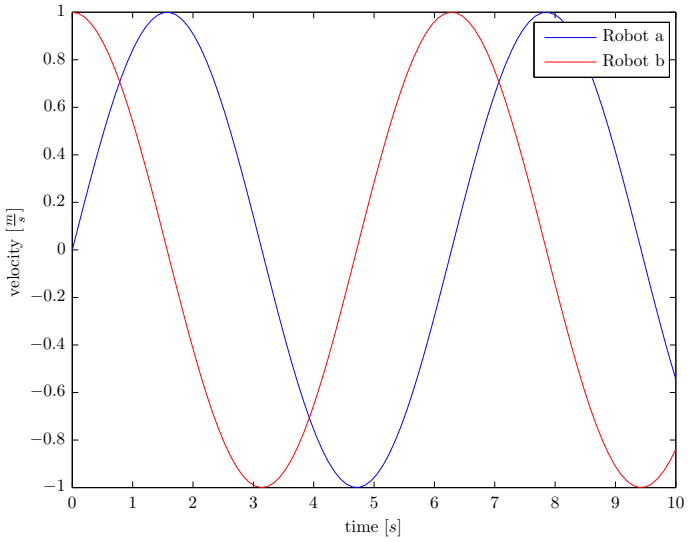


Figure 9.3: Example figure created with `mlf2pdf`.

9.7 Including Code in your Document

You may include samples from your Matlab code using the `lstlistings` environment, for example:

Listing 9.1: Matlab Example

```
% Evaluate y = 2x
for i = 1:length(x)

  y(i) = 2*x(i);

end
```

Listing 9.2: C++ Example

```
// sum all elements in a list
int sum=0;
for(list<int>::iterator it=mylist.begin(); it!=mylist.end(); ++it)
  sum += *it;
```

Chapter 10

RPG Notation Style

This chapter presents some conventions on notation that we use at the Robotics and Perception Group. Try to stick to those conventions since a unique style makes it easier to review the report.

10.1 Variable styles in LaTeX

Use lowercase and bold letters for vectors, e.g. $\mathbf{x}$, uppercase and bold letters for matrices, e.g. $\mathbf{R}$, and lowercase letters with normal weight for scalars, e.g. s.

10.2 Coordinate Systems and Rotations

We use the notation introduced by Prof. Glocker in the course "Mechanik 3" at ETHZ to express coordinate frames, rotations and vectors. Refer to Chapter 5 "Kinematik" in the lecture script for more details [1]. Figure 10.1 gives an overview of how coordinate transformations and vectors are specified. Observe that the coordinate system in which a vector is expressed is always written as index before the variable, e.g. ${}_B\mathbf{t}_{AB}$ is the vector from A to B expressed the coordinate system B. For the ease of reading, the index for the origin coordinate frame can be omitted: ${}_O\mathbf{t}_k := \mathbf{t}_k$.

[1] http://mitschriften.amiv.ethz.ch/main.php?page=3&scrid=1&pid=87&eid=1

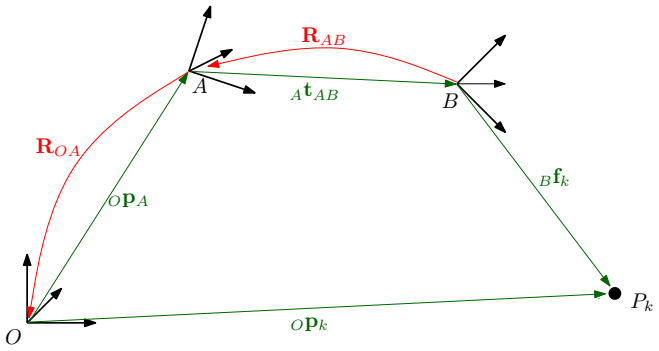


Figure 10.1: Notation overview.

A and B are two adjacent coordinate frames and O is the frame of origin. $\mathbf{R}_{AB}$ describes the coordinate transformation from frame B to frame A, thus it holds that

$$
\begin{aligned}
{}_O\mathbf{t}_k &= \mathbf{R}_{OB}\ {}_B\mathbf{f}_k, \\
\mathbf{R}_{OB} &= \mathbf{R}_{OA}\ \mathbf{R}_{AB}.
\end{aligned}
$$

10.3 Measured, estimated and target values

For controllers and estimators please specify the variables as follows in the report:

true value:	$\mathbf{x}$
estimated value:	$\hat{\mathbf{x}}$
measured value:	$\tilde{\mathbf{x}}$
desired value:	$\mathbf{x}_{\text{des}}$
error value:	$\mathbf{x}_{\text{e}}$
equilibrium value:	$\mathbf{x}^*$

Chapter 11

Discussion

Explain both the advantages and limitations of your approach.

11.1 Conclusion

Summarize your work and what came out of it.

11.2 Future Work

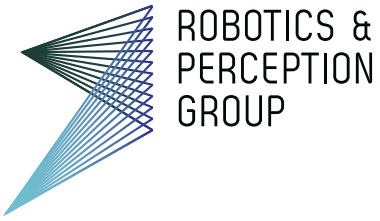
How would you extend the work? Can you propose another approach?

Appendix A

Something

In the appendix, you can provide some more data, a tutorial on how to run your code, a detailed proof, etc.

It is, however, not a requirement to have an appendix.



Title of work:

Autonomous landing on a moving platform

Thesis type and date:

Master Thesis, September 2016

Supervision:

First Supervisor Davide Falanga
Second Supervisor
Prof. Dr. Davide Scaramuzza

Student:

Name:	Alessio Zanchettin
E-mail:	zalessio@student.ethz.ch
Legi-Nr.:	97-906-739

Statement regarding plagiarism:

By signing this statement, I affirm that I have read the information notice on plagiarism, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Information notice on plagiarism:

`http://www.lehre.uzh.ch/plagiate/20110314_LK_Plagiarism.pdf`

Zurich, 25. 8. 2016: _____