ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Bachelor Thesis

Motion Planning for an autonomous Drone-Catching Drone based on Riemannian Motion Policies

Spring Term 2020

Supervised by:
Prof. Dr. Roland Siegwart
Michael Pantic
Rik Bähnemann

Author:
Luca Strässle

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Motion Planning for DroGone using Riemannian Motion Policies

is original work which I alone have authored and which is written in my own words.[1]

Author

Luca Strässle

Student supervisors

Michael Pantic
Rik Bähnemann

Supervising lecturer

Prof. Dr. Roland Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette'.[2] The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Place and date

Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

[2] `https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`

Abstract

In this thesis a motion planner is designed for the task of catching an illicit multicopter autonomously with a second multicopter. It extends the focus project "DroGone" where the first prototype was designed and implemented.
The planner is implemented using Riemannian Motion Policies (RMP). With RMPs it is possible to decouple the two planning challenges of keeping the target multicopter in the field of view of the cameras and reducing the distance to the target multicopter. A lot of variability within the planner results from the ability to weigh the importance of these tasks against each other. Thanks to the modular structure of RMPs, adding additional tasks is straight forward. With the help of RMPs, different flight modes are implemented that are responsible for following and catching a target multicopter without crashing into the ground. The planner is independent of a tracker and doesn't need a prediction of the flight path of the target multicopter.
Using a simulation it is shown that the planner can follow and catch target multicopters that are flying a variety of different flight paths.

Preface

This thesis extends the focus project “DroGone” and is supervised by Prof. Dr. Roland Siegwart and the Autonomous Systems Laboratory of ETH Zurich.
I am thankful to everyone whose support and assistance enabled this thesis since the start in March 2020.
Special thanks go to:

- Prof. Dr. Roland Siegwart who gave me the opportunity to work on the focus project “DroGone”.
- Michael Pantic for all the instructive discussions and all the inspiration.
- Rik Bähnemann for his valuable feedback and support.
- Felix Stadler for the valuable discussions and the good times we shared working on our theses.
- The team members of the focus project “DroGone” for the great collaboration and the times we spent together to design and build the drone-catching drone.
- Family, friends and colleagues for their patience, understanding and support.
- Everyone who proofread my thesis.

Symbols

Symbols

A	Riemannian Metric
d	distance
f	acceleration
f_x, f_y	focal lengths
J	Jacobian
R	Rotation matrix
T	Transformation matrix
u	Image u-axis
v	Image v-axis
θ	Roll angle
ϕ	Pitch angle
ψ	Yaw angle
$\mathcal{Q}$	Configuration Manifold
$\mathcal{X}$	Task Manifold

Indices

B	Body frame
C	Camera frame
cm	Carrier multicopter
d	distance
$d2g$	distance to ground
tm	Target multicopter
uv	image
W	World frame
x	x-Axis
y	y-Axis

z	z-Axis

Acronyms and Abbreviations

ASL	Autonomous Systems Lab
ENU	East North Up
ETH	Swiss Federal Institute of Technology
FSM	Finite State Machine
GUI	Graphical User Interface
NMPC	Nonlinear Model Predictive Control
RMP	Riemannian Motion Policy

Contents

Chapter 1

Introduction

This thesis extends the focus project "DroGone" [1] in which a hexacopter was developed that is able to autonomously catch multicopters. The catching mechanism developed in the focus project consists of a net mounted on top of the multicopter and catching the target multicopter means flying into it from below. The sensors in use to make this task autonomous are different cameras mounted on top of the hexacopter facing upwards. Fig. 1.1 shows a picture of the final design of the hexacopter.

Figure 1.1: The Final Prototype of the "DroGone" Focus Project

1.1 Motivation

In the focus project "DroGone", motion planning was done in world frame. With a tracker providing the current position of the target multicopter and the state estimation providing the current position of the carrier multicopter, polynomial trajectories in world frame were generated. To make the planner more variable, Dijsktra's algorithm was used to calculate an optimal interception point within the next two seconds. The trajectories had to be planned in a way such that the target was not lost out of sensor range. This was solved by implementing two modes. In the first mode the carrier multicopter aligns itself vertically with the target multicopter and in the second, the distance is reduced whilst keeping the vertical alignment.
There are two major drawbacks to this approach. First of all, the planner strongly depends on a prediction of the flight path of the target multicopter two seconds into the future. Predicting the flight path of a multicopter can be very challenging and wrong predictions can result in dangerous trajectories. Secondly, by planning in world frame, the planner has no knowledge of the target position on the image. Therefore it cannot react to the target being at the edge of the image.

1.2 Goals

The goal of this thesis is to implement an alternative motion planner for the "DroGone" project. It should consider the disadvantages of the current planner and propose solutions to improve the performance in the detrimental areas.
A follow mode should be implemented, capable of following a threatening multicopter without losing it out of the range of the sensors responsible for determining its position. To make sure that the target multicopter is not lost, a goal was to treat the challenges of getting closer to the target multicopter and keeping it in the field of view separately. Even though those two challenges are coupled, the possibility should exist to solely consider one task or the other in certain scenarios. For example if the target is about to leave the image, the task of preventing the target from leaving the image should outweigh everything else.
Additionally, the planner should be less dependent on the tracker and its prediction of the future path of the target multicopter. It should be able to follow a target multicopter without having any prediction available.
Lastly, the planner should be robust to image noise and to deviations in the distance, estimated by the detector.

Chapter 2

Related Work

Keeping an object in the field of view of a sensor is a problem that occurs quite frequently. Especially when working with autonomous multicopters like quadro- and hexacopters it is important to consider the visibility of an object in the path planning.

A solution to the task of landing a multicopter on a moving vehicle was presented in [2]. This problem is similar to the one of intercepting another multicopter and was solved by using Dijsktra's algorithm. With a trajectory of the future path of the vehicle to intersect generated by a tracker, the most optimal of all possible intersection points is chosen. This algorithm relies on an accurate prediction of the future path, which in the case of a randomly moving multicopter cannot be guaranteed.

In [3] the authors present a MPC algorithm capable of optimizing both action and perception objectives. The perception objectives are not incorporated as constraints, but rather as components to be optimized, in order not to limit agility. The cost function solved by the MPC, can be weighted with cost matrices for state, perception and input. Since this approach of keeping an object in the field of view is on the controller side, a trajectory in world frame has to be generated.

In [4] an approach is introduced where trajectories are calculated using B-splines, which guarantees that the target stays in sensor range. The perspective projection of one or multiple 3D points must be ensured to be inside the image domain for the whole trajectory. By parametrizing the trajectory as a B-spline polynomial, the kind of motion that can be performed is constrained. There exists an upper bound to agility, because perception is added as a hard constraint. The exact position of the point of interest on the image is not taken into account.

Chapter 3

Method

In this chapter the preliminaries are presented and the method that was implemented is discussed.

3.1 Preliminaries

The method used for motion planning in this thesis is Riemannian Motion Policies [5]. This method makes it possible to decouple different planning challenges and solve them in different spaces. The results can be transformed into a certain space, weighted and summed up.

3.1.1 Riemannian Motion Policies

The Riemannian Motion Policy (RMP) is a mathematical object for motion generation. It consists of an acceleration field f and a Riemannian metric A defined on a manifold. The acceleration field is defined as a motion policy on this manifold. The Riemannian metric is a tool to weigh the axes of the manifold and therefore stretch the space. A RMP on a manifold $\mathcal{M}$ is defined as $(f, A)^{\mathcal{M}}$. For two different manifolds, where one has dimension d and axes q and the other has dimension k and axes x, there exists a task map ϕ, that relates the space of dimension d to the space of dimension k. With the Jacobian of this task map, defined as $J_\phi = \frac{\partial \phi}{\partial q}, \in \mathbb{R}^{k \times d}$ there exists a local mapping between the two manifolds. When working with RMPs for robot motion, one usually works with multiple manifolds. There is one configuration manifold and one or multiple task manifolds. The configuration manifold describes the geometric space in which the output of the motion planner should be in. The task manifolds describe the geometric spaces in which the different planning challenges are computed. An overview of the configuration and the task manifolds is given in Section 3.2. For RMPs, a set of mathematical operations exists. Two of which are especially important for this thesis.

Pullback

With the pullback operation it is possible to pull a RMP from the manifold it is defined on to another manifold. It is important to be able to get the RMPs from their task manifolds into the configuration manifold. Pulling from the task space to configuration space is possible with the Jacobian matrix of the task map ϕ that relates the configuration manifold $\mathcal{Q}$ to the task manifold $\mathcal{X}$.

$$pull_{\phi}\left((f, A)^{\mathcal{X}}\right) = \left((J^T A J)^{+} J^T A f,\ J^T A J\right)^{\mathcal{Q}} \tag{3.1}$$

The result of this equation, is the RMP that was defined on the task manifold, in the configuration manifold.

Addition

If two or more RMPs are on the same manifold, they can be added up with this equation:

$$\sum_i (f_i, A_i)^{\mathcal{M}} = \left(\left(\sum_i A_i\right)^{+} \sum_i A_i f_i,\ \sum_i A_i\right)^{\mathcal{M}} \tag{3.2}$$

This summation is a metric-weighted average, where the Riemannian metric is considered. With the metric an importance can be given to the axes of the task manifolds. It is then considered when the RMPs are added up in configuration space.

3.2 Manifolds

As mentioned previously, there is a configuration manifold and task manifolds. All the manifolds relevant for this thesis and the necessary local mappings are presented in this section.

3.2.1 Configuration Manifold

"DroGone" uses a Nonlinear Model-Predictive Controller (NMPC) [6]. The controller's input are trajectories in the $\left(x^W, y^W, z^W\right)$ world frame which last for at least two seconds. These trajectories are generated by the motion planner. As the configuration manifold is of the geometry of the output of the planner, the axes of the configuration manifold have to be the $\left(x^W, y^W, z^W\right)$ world frame coordinates. The coordinates of the world frame are aligned with the ENU (East, North, Up) local tangent plane coordinate frame.

3.2.2 Task Manifolds

Three planning policies in three different manifolds were implemented in this thesis. In a later section Section 3.3 the policies are explained further. In this section an overview of the task manifolds is given.

Distance to Target Manifold

The first manifold is the one-dimensional distance to target manifold. The distance to target is defined as the absolute value of the vector connecting the target and the carrier multicopter. As the carrier multicopters position is only used in world frame, the superscript indicating world frame is dropped here and in the following. The carrier multicopters position in world frame is referred to as (x, y, z). With $(x_{tm}^W, y_{tm}^W, z_{tm}^W)$ being the target multicopters position in world frame, the distance in world frame writes as

$$d = \sqrt{\left(x_{tm}^W - x\right)^2 + \left(y_{tm}^W - y\right)^2 + \left(z_{tm}^W - z\right)^2} \tag{3.3}$$

With this definition the Jacobian relating the configuration manifold to the distance to target manifold can be calculated.

$$J_d = \begin{bmatrix} \frac{\partial d}{\partial x} & \frac{\partial d}{\partial y} & \frac{\partial d}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{x - x_{tm}^W}{d} & \frac{y - y_{tm}^W}{d} & \frac{z - z_{tm}^W}{d} \end{bmatrix} \tag{3.4}$$

Image Manifold

The image manifold is two-dimensional. It's two axes are the (u, v) pixel coordinates, the coodinate frame of the image plane. To build a Jacobian, the coordinates have to be expressed depending on (x, y, z), the position of the carrier multicopter in world frame. In homogeneous coordinates u and v are defined as

$$\begin{bmatrix} u' \\ v' \\ \lambda \end{bmatrix} = K \cdot T_{C \leftarrow B} \cdot T_{B \leftarrow W} \cdot \boldsymbol{p}_{tm}^W \tag{3.5}$$

For this equation one needs the camera matrix K, the transformation matrices from world to body $T_{B \leftarrow W}$ and from body to camera $T_{C \leftarrow B}$ and the target multicopters position in world frame $\boldsymbol{p}_{tm}^W$. From here on, the target multicopter will just be referred to as the target. The camera matrix that defines the properties of the pinhole camera model of the camera used looks like this.

$$K = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

f_x and f_y stand for the focal lengths, where u_0 and v_0 define the principal point on the image. The transformation matrix from body to camera is defined as

$$T_{C \leftarrow B} = \begin{bmatrix} R_{C \leftarrow B} & -R_{C \leftarrow B} \cdot \boldsymbol{t}_{C \leftarrow B} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.7}$$

where $R_{C \leftarrow B}$ is the rotation matrix determined from the orientation $(\theta_C^B, \phi_C^B, \psi_C^B)$ of the camera mounting and $t_{C \leftarrow B} = \left(x_C^B, y_C^B, z_C^B\right)^T$ is the translation determined from the position of the camera mounting with respect to the carrier

multicopter.
The Transformation matrix from world to body is defined as

$$T_{B\leftarrow W} = \begin{bmatrix} R_{B\leftarrow W} & -R_{B\leftarrow W} \cdot \boldsymbol{t}_{B\leftarrow W} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.8}$$

where $R_{B\leftarrow W}$ is the rotation matrix determined from the orientation $(\theta_B^W, \phi_B^W, \psi_B^W)$ of the carrier multicopter with respect to the world frame. However here the roll and pitch angles are assumed to be zero $\theta_B^W = \phi_B^W = 0$. This is necessary because acceleration and roll as well as pitch depend on each other. The Jacobian that is being derived here, should relate change in world frame to change in image frame. An acceleration of the target in the image frame can be transformed into an acceleration of the carrier multicopter in world frame with the inverse of this Jacobian. This acceleration is calculated for a specific orientation of the carrier multicopter. As the orientation of a hexacopter and its acceleration depend on each other, the acceleration of the carrier multicopter is defined by its orientation. If now an acceleration is calculated for a particular orientation, the acceleration is overdefined. For the assumption that roll and pitch are zero, the acceleration defined by the orientation is zero. Therefore the acceleration calculated with the Jacobian can be used. $t_{B\leftarrow W} = (x, y, z)^T$ is the position of the carrier multicopter in world frame.

On the "DroGone" hardware the cameras are mounted facing upwards and close to the centre of the multicopter. With this mounting, the camera frame matches the body frame almost exactly and for this reason, zero rotation and zero translation was assumed for this thesis. With this assumption, the equation above and normalisation of the homogeneous coordinates u and v in dependency of (x, y, z) write

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x}{z_{tm}^W - z} \left(\left(x_{tm}^W - x \right) \cos\psi + \left(y_{tm}^W - y \right) \sin\psi \right) + u_0 \\ \frac{f_y}{z_{tm}^W - z} \left(- \left(x_{tm}^W - x \right) \sin\psi + \left(y_{tm}^W - y \right) \cos\psi \right) + v_0 \end{bmatrix} \tag{3.9}$$

where the sub- and superscript were dropped for the yaw angle of the carrier multicopter $\psi = \psi_B^W$.
With this definition the Jacobian can be calculated

$$\begin{aligned} J_{uv} &= \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \end{bmatrix} \\ &= \begin{bmatrix} \frac{f_x \cos\psi}{z - z_{tm}^W} & \frac{f_x \sin\psi}{z - z_{tm}^W} & -\frac{f_x}{\left(z - z_{tm}^W \right)^2} \left(\left(x - x_{tm}^W \right) \cos\psi + \left(y - y_{tm}^W \right) \sin\psi \right) \\ -\frac{f_y \sin\psi}{z - z_{tm}^W} & \frac{f_y \cos\psi}{z - z_{tm}^W} & -\frac{f_y}{\left(z - z_{tm}^W \right)^2} \left(\left(x - x_{tm}^W \right) \sin\psi - \left(y - y_{tm}^W \right) \sin\psi \right) \end{bmatrix} \end{aligned} \tag{3.10}$$

Distance to Ground Manifold

The distance to ground manifold is one-dimensional and has the distance to ground as variable. The distance to ground is defined as the difference between

the z coordinates of the carrier multicopter and the ground in the world frame.

$$d_{gnd} = z - z_{gnd} \tag{3.11}$$

This results in the following Jacobian matrix.

$$J_{d2g} = \begin{bmatrix} \frac{\partial d_{gnd}}{\partial x} & \frac{\partial d_{gnd}}{\partial y} & \frac{\partial d_{gnd}}{\partial z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

3.3 Policies

On each of the task manifolds there is a policy that generates an acceleration based on the current state of the manifold variables. The policies are explained in the following.

3.3.1 Distance to Target Policy

Motion Policy

The motion policy on the distance to target manifold tries to drive the distance to a goal value.

$$f_d = s\left(d_{goal} - d\right) \cdot f_{max} - \beta_d \cdot \dot{d} \tag{3.13}$$

The first term of the equation considers the current error. This error is normalized by a soft normalization function $s(x)$ (see Section 3.3.4) and then multiplied by the maximum acceleration that the carrier multicopter is able to achieve. The second term is a damping term. It depends on the current velocity of the distance to the target, or how the distance to the target is changing.

Metric

The metric for the distance to target RMP is one-dimensional.

$$A_d = \begin{bmatrix} a_d \end{bmatrix} \tag{3.14}$$

3.3.2 Image Policy

Motion Policy

The motion policy on the image manifold has the same structure as the one on the distance to target manifold.

$$f_{uv} = s\begin{pmatrix} u_{goal} - u \\ v_{goal} - v \end{pmatrix} \cdot f_{max} - \beta_{uv} \cdot \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} \tag{3.15}$$

The error is scaled by a soft normalization function $s(x)$ (see Section 3.3.4). The maximum acceleration is also transformed from world frame into image frame. Additionally there's a damping term proportional to the velocity of the target on the image. The position and velocity of the target on the image used in this

policy are calculated for a carrier multicopter with zero roll and pitch. This is because the policy should be fulfilled for steady state of the multicopter. The steady state of any not omnidirectional multicopter is at roll and pitch equal to zero.

Metric

The metric for the image RMP is two-dimensional. For this thesis a diagonal matrix was chosen, where u and v could be weighted separately.

$$A_{uv} = \begin{bmatrix} a_u & 0 \\ 0 & a_v \end{bmatrix} \tag{3.16}$$

a_u and a_v are set in the image manifold. Since the axes of the image manifold have a different scaling than the axes of the other two task manifolds, the weight has to be scaled as well to be able to set the metric intuitively.

$$\begin{aligned} a_u &= \left(\frac{z - z_{tm}}{f_x} \right)^2 \cdot a_{u,set} \\ a_v &= \left(\frac{z - z_{tm}}{f_y} \right)^2 \cdot a_{v,set} \end{aligned} \tag{3.17}$$

3.3.3 Distance to Ground Policy

Motion Policy

The motion policy on the distance to ground manifold drives the distance to ground away from zero.

$$f_{d2g} = \frac{\alpha_{d2g}}{d_{gnd}} - \beta_{d2g} \cdot \dot{d}_{gnd} \tag{3.18}$$

The first part of the equation gets bigger as the distance to ground gets smaller. In this policy there's also a damping part. With $f_{d2g} = min\left(f_{d2g}, f_{max}\right)$ it is made sure, that the acceleration does not get bigger than the maximum acceleration.

Metric

The metric for the distance to ground RMP is one-dimensional.

$$A_{d2g} = \begin{bmatrix} a_{d2g} \end{bmatrix} \tag{3.19}$$

3.3.4 Soft Normalization Function

For the two policies that try to drive the variables to a certain value a soft normalization function is used. This function makes sure that the value of the error is smoothly normalized when it approaches some maximum value. This

maximum value is different for each policy. With σ one can define at what value the function should flatten. With the parameter c the steepness of the curve of the soft normalization function can be varied.

$$\begin{aligned} s\left(\boldsymbol{x}\right) &= \frac{\boldsymbol{x}}{h(|\boldsymbol{x}|)} \\ h(z) &= z + \sigma c \log\left(1 + \exp\left(-2\frac{cz}{\sigma}\right)\right) \end{aligned} \tag{3.20}$$

See Fig. 3.1 for the soft normalization function of the distance to target and the image policy, which have a different σ.

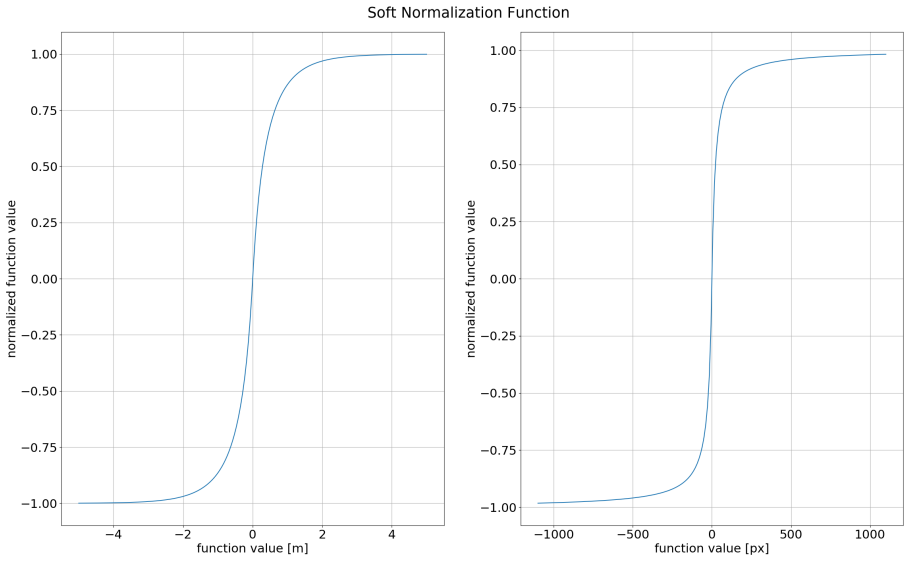

Figure 3.1: Soft Normalization Functions for Distance to Target (left) and Image Policy (right)

From here on the motion policy is referred to as policy. If the combination of motion policy and metric is meant, it is referred to as RMP.

3.3.5 Policy Tuning

In the three policies presented, there are four parameters to tune, besides the parameters of the soft normalization function. For the distance to target and image policies there are the damping terms β_d and β_{uv}. They must be tuned such that the target reaches it's goal position fast without overshooting. When tuning the parameters one has to consider that the two policies act together and have to be balanced.
When the damping of the image policy is too big the target cannot be followed and if the damping is to small, the target is overshot. Both are undesirable and can cause a loss of the target. To prevent that, the damping for this policy has to

be tuned differently for different target velocities. For a high target velocity the damping term should be small, whereas for a small target velocity the damping term should be high. To balance the effect of the image policy, the damping term of the distance to target policy also has to be adjusted with the target speed.
The distance to ground policy has two parameters to tune. α_{d2g} to weigh the contribution to the acceleration of the inverse of the error and β_{d2g} to weigh the damping. Those two parameters have to be tuned such that the carrier multicopter switches it's flight direction fast when flying towards the ground. After switching direction the acceleration shouldn't be too high, so that if possible the target is not leaving the field of view. See Table 3.1 for the tuning values for different target velocities.

	$0\frac{m}{s}$	$2\frac{m}{s}$	$4\frac{m}{s}$
β_{uv}	5.0	2.9	1.8
β_d	5.0	5.0	2.7
β_{d2g}	0.5	0.5	0.5
α_{d2g}	3.0	3.0	3.0

Table 3.1: Value of Tuning Parameters for different Flight Speeds

3.4 Policy Summation

Recall that the addition proposed by RMP (3.2) is a metric-weighted average summation. The metric-weighted average sums up the policies weighted with the metrics. This sum is then normalized with the inverse of the sum of metrics. Perpendicular axes cannot be weighted against each other using this normalization. For "DroGone" the camera, mounted on top of the carrier multicopter, is facing upwards. This results in an acceleration of the image policy that is always exactly perpendicular to the acceleration of the distance policy, given that both accelerations are in world frame. For this thesis it is important to be able to weigh the polices against each other with the metrics. As the policies are summed in world frame, two perpendicular directions are always of the same unit. Therefore a weighing of perpendicular axes is possible. The metric-weighted sum from (3.2) can be used to calculate the direction of the total acceleration as the axes are weighted against each other before the normalization.

$$f = |f| \cdot \frac{\sum_i A_i f_i}{|\sum_i A_i f_i|} \tag{3.21}$$

An alternative normalization was necessary to calculate the magnitude $|f|$ to the direction of the total acceleration. The total magnitude is also calculated using a metric-weighted average sum. First the total weight of each RMP is

calculated by taking the average of the weights of each metric.

$$w_i = \sum_{j=0}^{n} \frac{a_{i,j}}{n} \quad with \quad A_i = \begin{bmatrix} a_{i,1} & 0 & \cdots & 0 \\ 0 & a_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & a_{i,n} \end{bmatrix} \tag{3.22}$$

Those weights can then be used to calculate the weighted average sum.

$$|f| = \frac{\sum_i w_i \cdot |f_i|}{\sum_i w_i} \tag{3.23}$$

This yields the magnitude and together with the direction the complete desired acceleration can be obtained.

$$f = \frac{\sum_i w_i \cdot |f_i|}{\sum_i w_i} \cdot \frac{\sum_i A_i f_i}{|\sum_i A_i f_i|} \tag{3.24}$$

3.5 Flight Modes

With these three different policies and the metrics to weigh them, different flight modes can be implemented. There are two main modes, the follow and the catch mode. Additionally there are three safety modes, which the main modes temporarily switch to in critical situations. In the different modes not only the metrics and policy parameters are changed, but also the Jacobian of the image policy. The image policy generates an acceleration into the negative z^W-direction as a downwards flying carrier multicopter causes the target to get closer to the middle of the image. Given that the target is not moving in the negative z^W-direction, this increases the distance to the target and when the target is too far away it cannot be detected anymore. Therefore the carrier multicopter flying downwards can be desirable or not desirable, depending on the mode of the planner. With the Jacobian, which relates change in one manifold to change in another manifold, it is possible to reduce or increase the effect of the image policy on the z^W-direction. The third column of J_{uv} (3.10) relates the image coordinates with the z^W-coordinate. If the entries of this column are scaled down, the influence of the image policy on the acceleration in z^W-direction is reduced and if they are scaled up, the influence is increased.
In almost all the modes that are presented in the following, the distance to ground policy is deactivated by setting its metric to zero. Only in “prevent ground crash mode”, the policy is active. The weightings in the metrics for the different modes can be seen in Table 3.2

3.5.1 Follow Mode

In follow mode the goal is to follow the moving target multicopter without losing it out of the field of view of the sensors. For this purpose, the image and the distance to target policies are active. The image policy tries to get the target

into the middle of the image ($u_{goal} = v_{goal} = 0$). For following the target without losing it, the influence of the image policy on the z^W-direction is too high. As a solution, the entries in the third column of J_{uv} are halved.
The distance to target policy aims for a distance of five meters ($d_{goal} = 5$). For this mode, the two policies are equally weighted.

3.5.2 Catch Mode

When in catch mode, the carrier multicopter tries to catch the target multicopter. The image and the distance to target policies are active. In catch mode a downwards effect on the z^W-direction is undesirable. Therefore the effect of the image policy on the z^W-axis is deactivated, by setting the third column of the Jacobian to zero. The goal positions in the image are zero ($u_{goal} = v_{goal} = 0$). The goal position of the distance to target is set to zero ($d_{goal} = 0$). In this mode the weight of the distance to target policy is four times higher than the weight on the image policy. Once the distance between carrier and target multicopter is very small, the dampers of the image and the distance to target policies are deactivated. The damper of the distance to target motion policy is deactivated, to ensure that the carrier multicopter does not stop at the interception point and therefore catches the target. Disabling the damper of the image policy causes a small overshoot in the direction of the target. This overshoot is necessary to catch moving targets, since the planner works without prediction of future position of the target multicopter.

3.5.3 Safety Modes

Recover Mode

The recover mode comes into play when the target is at the edge of the image. In this mode the goal is to get the target closer to the middle of the image as fast as possible. The goal positions in the image and the distance to target policies are equal to the ones in follow mode. The fact that the target gets closer into the middle of the image when the carrier multicopter flies downwards can be useful in this mode. For this reason, the entries of the third column of J_{uv} are doubled. The damper of the image policy is deactivated completely, in order to make the mode even more aggressive. The distance policy is not of importance in this mode and therefore the image policy is weighted five times higher.

Prevent Ground Crash Mode

The prevent ground crash mode is the safety mode that prevents crashing into the ground. In this mode the distance to ground policy is the only policy that matters and the other two are deactivated.

Recover Distance Mode

The recover distance mode comes into play when the target multicopter is far away from the carrier multicopter. It tries to reduce the distance to the target. As the downwards effect of the image policy on the z^W-coordinate is a drawback for that purpose, the third column of J_{uv} is set to zero to deactivate that effect. Additionally the damping term of the distance to target policy is reduced to get to the target even faster.
This mode is not necessary if the full autonomy pipeline is active. For a certain horizontal distance between the carrier and the target multicopter, the pixel error on the image is smaller if the complete distance is bigger. So for a large distance, where the recover distance mode would come into play, the planner will switch to catch mode anyway. The catch mode is designed to reduce the distance fast and can act as a recover distance mode. However, if the follow mode wants to stand on it's own this mode is necessary, especially for the case where the target is flying upwards on a steep curve. This mode was implemented during the evaluation of the follow mode as a reaction to the drawbacks discussed in Chapter 5.

3.5.4 Mode Switches

Image triggered switches

The switches between the follow, recover and catch mode occur based on the position of the target on the image assuming roll and pitch zero. In Fig. 3.2 one can see the image for the three modes. In the images white stands for follow mode, red for recover mode and green for catch mode. When the planner is in follow mode and the target gets to the outer fifth of the image, the planner switches to recover mode. If the target gets to the inner fifth of the image a switch to catch mode occurs. At the beginning of the recover mode, the target is in the outer fifth of the image. To have enough time to recover, the switch back to follow occurs when the target is in the inner three fifths of the image again. Once the planner is in catch mode it only aborts the catch mode when the target is in the outer fifth of the image. In this case the target is almost lost and the planner switches directly from catch into recover mode. The reason for not having a follow mode in between is that a catch attempt should only be aborted when the target is close to being lost. When the carrier multicopter is very close to the target multicopter, the switch to recover is completely deactivated. This is because when the target is close, it is in the outer fifth of the image very fast which can cause an unnecessary switch to recover mode.

Distance triggered Switches

The prevent ground crash mode is triggered once the altitude is smaller than two meters. This mode has highest priority and a transition into this mode occurs no matter where the target is in the image. Once the carrier multicopter has an altitude of more than two meters and twenty centimetres again, the planner

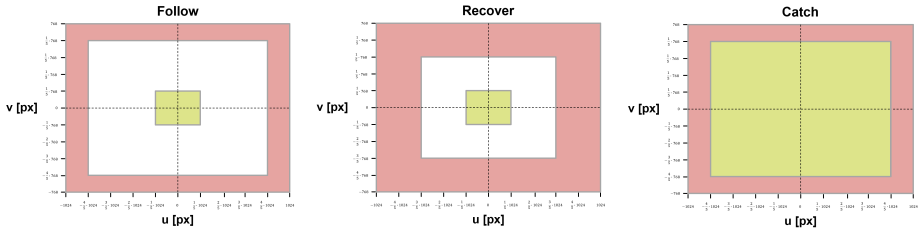

Figure 3.2: Image-triggered Mode Switches in Follow, Recover and Catch mode

switches back to follow, recover or catch mode, depending on the target position on the image.
A switch to the recover distance mode occurs if the planner is in follow mode and the distance is bigger than seventeen meters. In this mode, switches to catch or recover mode can occur. The switches occur based on the target position on the image with the rules of the follow mode (see left image in Fig. 3.2). A switch back to follow mode only happens when the distance to the target is smaller than twelve meters again.

	Follow	Catch	Recover	Prevent Ground Crash	Recover Distance
a_u	1.0	1.0	5.0	0.0	1.0
a_v	1.0	1.0	5.0	0.0	1.0
a_d	1.0	4.0	1.0	0.0	2.0
a_{d2g}	0.0	0.0	0.0	1.0	0.0

Table 3.2: Values of Metrics for different Flight Modes

3.6 Implementation

At the ASL an implementation of RMPs in C++ is being developed. As a basis for this thesis, the current version of this implementation was used. The implementation of RMPs can be found in the git repository [7], where special access was granted for this thesis. This implementation contains the RMP operations and templates for policies and geometries. With the help of the templates, the geometries and policies on the three task manifolds (see Section 3.2.2) were implemented. The alternative summation described in Section 3.4 was added to the RMP operations. With the policies and the summation, an acceleration in world frame can be calculated for a certain state of the carrier and target multicopters. For the planner to be compatible with the NMPC used by “DroGone”, the output had to be a two second long trajectory consisting of an array of two hundred positions, velocities and accelerations. Point by point, the acceleration

is calculated and forward integrated to velocity and position with a trapezoidal integrator. With the new position and velocity, the next acceleration can be calculated with the policies. The target multicopter is assumed to be stationary for the whole trajectory, as no prediction is used in this thesis. See Algorithm 1 for the structure of the algorithm. The graphical user interface (GUI), the finite state machine (FSM) and the planner for the flight path of the target multicopter were taken from "DroGone". The new implementations can be found in the git repository [8] and in a fork of the git repository mentioned above [9].

With this implementation, it was possible to generate a trajectory with two hundred points in one to two milliseconds. The calculation time of a trajectory depends linearly on the number of points. For a shorter trajectory, the computational time could be reduced significantly. For the configuration of "DroGone" the NMPC needs a trajectory that's two seconds long, however a configuration of the NMPC that needs a shorter trajectory would be possible. In this implementation new trajectories are planned with 10Hz. Based on the calculation times, this frequency could be increased.

Algorithm 1: Calculating a Trajectory

input : position of carrier multicopter $\boldsymbol{r}_{cm}^{W}$
velocity of carrier multicopter $\boldsymbol{v}_{cm}^{W}$
target position in image $\boldsymbol{u}$ (from detector)
distance to target d (from detector)
output: trajectory in world frame $traj$

$\boldsymbol{r}_{tm}^{W} \leftarrow$ `tranform(`$\boldsymbol{r}_{cm}^{W}, \boldsymbol{u},\ d$`)`;
for $t \leftarrow 0.0$ *to* 2.0 **do**
  $\boldsymbol{u}, \dot{\boldsymbol{u}} \leftarrow$ `get(`$\boldsymbol{r}_{cm}^{W}, \boldsymbol{r}_{tm}^{W}, \boldsymbol{v}_{cm}^{W}, roll \leftarrow 0, pitch \leftarrow 0$`)`;
  $d, \dot{d} \leftarrow$ `get(`$\boldsymbol{r}_{cm}^{W}, \boldsymbol{r}_{tm}^{W}, \boldsymbol{v}_{cm}^{W}$`)`;
  $d2g, d\dot{2}g \leftarrow$ `get(`$\boldsymbol{r}_{cm}^{W}, \boldsymbol{v}_{cm}^{W}$`)`;
  $\boldsymbol{f}_{cm}^{W} \leftarrow$ `rmp(`$\boldsymbol{u}, \dot{\boldsymbol{u}}, d, \dot{d}, d2g, d\dot{2}g$`)`;
  $\boldsymbol{r}_{cm}^{W}, \boldsymbol{v}_{cm}^{W} \leftarrow$ `integrate(`$\boldsymbol{f}_{cm}^{W}$`)`;
  $traj \leftarrow$ `append(`$\boldsymbol{r}_{cm}^{W}, \boldsymbol{v}_{cm}^{W}, \boldsymbol{f}_{cm}^{W}, t$`)`;
  $t \leftarrow$ `increase(`0.01`)`;
end

Chapter 4

Experiments and Results

To evaluate the performance of the different flight modes, experiments were conducted. All the experiments were done in simulation, which was initially set up for "DroGone" and used throughout the whole thesis. In this chapter the experimental setup for the different modes is explained and results are shown.

4.1 Experimental Setup

4.1.1 Simulation

Simulations of multicopters had previously been set up in the ASL using Gazebo, an open source 3D simulator integrated with ROS [10, 11]. A lot of these simulations were adapted, like for example "firefly", a gazebo model of a hexacopter, which was used as the carrier multicopter. RVIZ [12], which is a 3D visualization tool for ROS, was used to visualize the simulations. In RVIZ, the target multicopter was visualized as a point and its path was generated with a seperate ROS node. To simulate detections, the target multicopters position is transformed from world into camera frame, which is again transformed to a detection of the form (u, v, d). For these transformations, the camera matrix is needed. For this thesis the following camera matrix was used in simulation.

$$K = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1140 & 0 & 0 & 0 \\ 0 & 1140 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

The width of the camera image used in simulation is $2048px$ and the height is $1536px$.
Since perfect detections are unrealistic, noise is added to the target position in the camera frame. For all three axes the noise is generated with a random walk on a normal distribution. For the x^C- and y^C-coordinates, the associated standard deviation is calculated from a constant value of three pixel. As the error in the distance estimation by the detector increases if the target is further

away, the standard deviation for the z^C-coordinate is calculated with a formula that considers the distance to the target. This formula was taken from the simulation implemented by "DroGone". The noise was added for all of the experiments done. An example for the evolution of the noise during a test run for follow mode can be seen in Fig. 4.1. For all experiments the maximum acceleration of the carrier multicopter, which is used in all of the three policies (see Section 3.3), is set to eight meters per second squared.

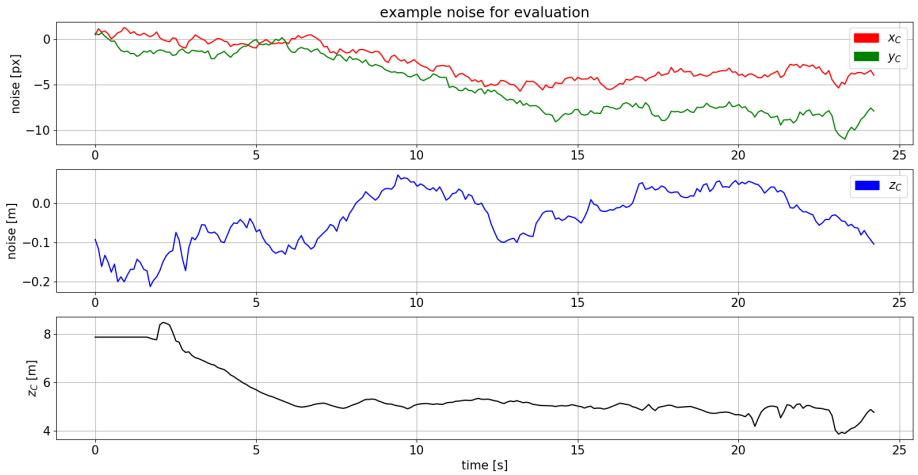

Figure 4.1: Noise during a Flight in Follow Mode

4.1.2 Follow Mode

To evaluate the performance of the follow mode, it was tested in different situations. These situations were created by varying the following parameters and for each combination fifteen tests were conducted.

- target flight path
  - linear
  - random
- target velocity
  - $2\frac{m}{s}$
  - $4\frac{m}{s}$
- initial pixel error (see red line in Fig. 4.2)
  - $200px$
  - $400px$

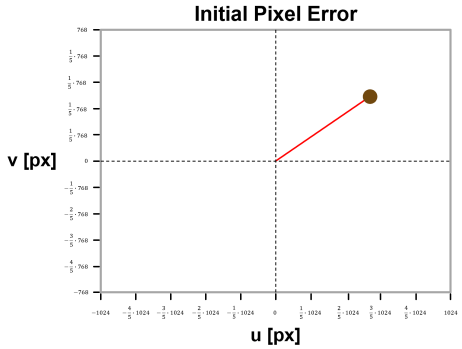

Figure 4.2: Visualization of the Initial Pixel Error

– $600px$

The direction of the linear flight path is defined through the θ and φ angles of a spherical coordinate system. Both angles were taken from a uniform distribution, φ from $Uniform[0, 2\pi]$ and θ from $Uniform[\frac{\pi}{3}, \frac{2\pi}{3}]$. This limitation of the θ angle was chosen, because for steeper flight paths a switch to catch mode would occur very early in the full autonomy pipeline. For the random path the direction of the vector from the current to the next point is also determined by the θ and φ angles, where the length is defined through the target velocity. The changes of the angles θ and φ are generated with a random walk on a uniform distribution. A new angle is calculated with a frequency of one hundred hertz.

$$\begin{aligned} \varphi_{new} &= \varphi_{old} + Uniform\left[-1.5°, +1.5°\right] \\ \theta_{new} &= \theta_{old} + Uniform\left[-1°, +1°\right] \end{aligned} \tag{4.2}$$

The initial position of the target multicopter is defined through the initial pixel error on the image and the initial distance between the target and the carrier multicopter. The latter is randomized and comes from a uniform distribution $Uniform\left[5m, 15m\right]$. The follow mode was defined to be successful, if the carrier multicopter could follow the target for twenty seconds without losing it. Switches to prevent ground crash and recover mode were possible. Switching to recover distance mode was not possible, because this mode was implemented after the evaluation of the follow mode.

4.1.3 Catch Mode

The catch mode was evaluated for five different flight manoeuvres of the target multicopter.

- stationary
- $2\frac{m}{s}$ linear

- $2\frac{m}{s}$ random
- $4\frac{m}{s}$ linear
- $4\frac{m}{s}$ random

For each flight manoeuvre, the simulation was run fifteen times. A catch was defined to be successful when the horizontal distance between the centres of carrier and target multicopter was smaller than sixty centimetres and the vertical distance smaller than twenty five centimetres. These values represent the size of the catching mechanism from "DroGone". In these experiments the whole pipeline was tested, so for a successful catch, the follow mode also had to be successful until the switch to catch mode. In catch mode switches to prevent ground crash and recover mode were possible.

4.1.4 Safety Modes

In the experiments for the follow and catch modes, switches to two of the safety modes were possible. To better see their effect, they were also evaluated separately. Also the lastly implemented recover distance mode was evaluated, to test its potential.

Prevent Ground Crash Mode

For the prevent ground crash mode, two experiments were conducted. For the first experiment, a world with a mountain was loaded into the simulation. Then the target was flown over the mountain with a small distance. The carrier multicopter that was following the target at a distance of five meters would have crashed into the mountain without the prevent ground crash mode. In the second experiment, the target multicopter flew a trajectory towards the ground. Without the prevent ground crash mode, the carrier multicopter would have flown into the ground. This mode was defined to be successful, when the carrier multicopter did not crash.

Recover Mode

A switch to recover mode had to be provoked, to analyze the performance. This was done by a target flying linearly with a velocity of four meters per second with a large initial pixel error. The target flew along the x^W-axis to be able to evaluate the recover mode by only looking at the u image coordinate.

Recover Distance Mode

The recover distance mode was triggered by a target flying linearly with four meters per second steeply upwards. The effect of this mode on the distance between the target and the carrier multicopter was evaluated.

4.2 Results

4.2.1 Follow Mode

To evaluate the performance of the follow mode, the pixel error on the actual image and the horizontal error between the target and the carrier multicopter were plotted. The horizontal error is the error in the x^W and y^W coordinates. In both cases the error is the difference from zero. Since there were fifteen successful tests available for every setup, the mean and standard deviation of these fifteen tests are plotted in Figs. 4.4 to 4.7.

In all plots, a notable behaviour can be seen at the beginning. There is a large drop in the pixel error plots. This is because the carrier multicopter starts following the target multicopter from a hovering position. In the beginning, the acceleration goes from zero to a high value very quickly. This causes the drop in the pixel error, since the multicopter has a big change in the roll and pitch angles to achieve such a high acceleration. A little bit later the horizontal error has its peak. This peak is shortly after the start, because the carrier multicopter needs some time to get to a velocity faster than the velocity of the target. In all experiments, the pixel error had oscillation. This was caused by oscillation in the roll and pitch angles of the carrier multicopter. Since no prediction of the target flight path is used, the target is assumed to be stationary for the calculation of the trajectory whereas in reality the target is moving. Therefore the position of the target on the image at the next replanning step differs from the position that was assumed in the previous trajectory at this point in time. With the positions, the accelerations differ and there is a jump in acceleration with every new trajectory as it can be seen in Fig. 4.3. Oscillations in roll and pitch are caused by these jumps. Because the distance gets smaller towards the end of the follow mode, even small roll and pitch angles cause high oscillations in the error on the image.

Linear, $2\frac{m}{s}$

For a target moving linearly with two meters per second, fifteen out of fifteen tests were successful. In the upper plot of Fig. 4.4 it can be seen that the pixel error approaches a value of approximately three hundred pixels. There are oscillations around this value. In the lower plot, one can see the horizontal error, which converges to a value of approximately one meter. Depending on the initial pixel error, it takes between three and ten seconds to get to this error.

Random, $2\frac{m}{s}$

In Fig. 4.5 one can see the plots for a randomly moving target with two meters per second. The follow mode was successful in all tests. As in the linear case, the pixel error approaches approximately three hundred pixels and the horizontal error approaches approximately one meter. The standard deviations were higher

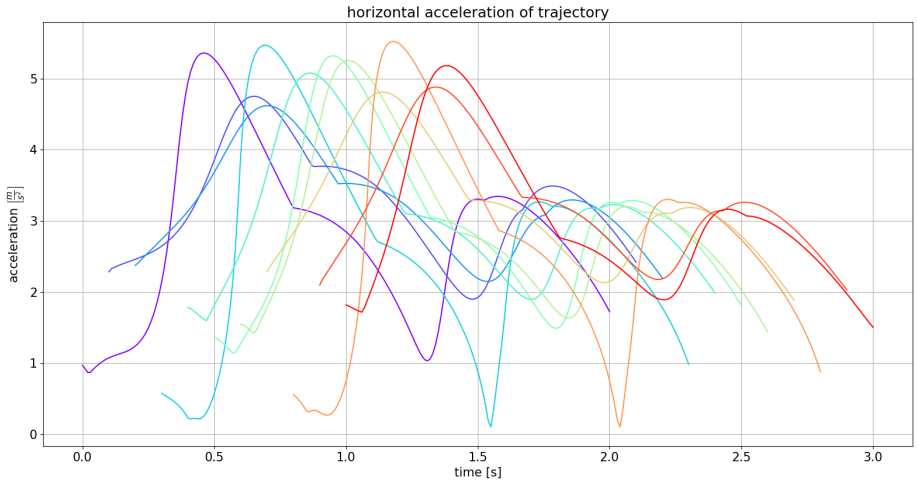

Figure 4.3: Horizontal Accelerations in Trajectories during 1 second in Follow Mode, following a Target moving with $2\frac{m}{s}$

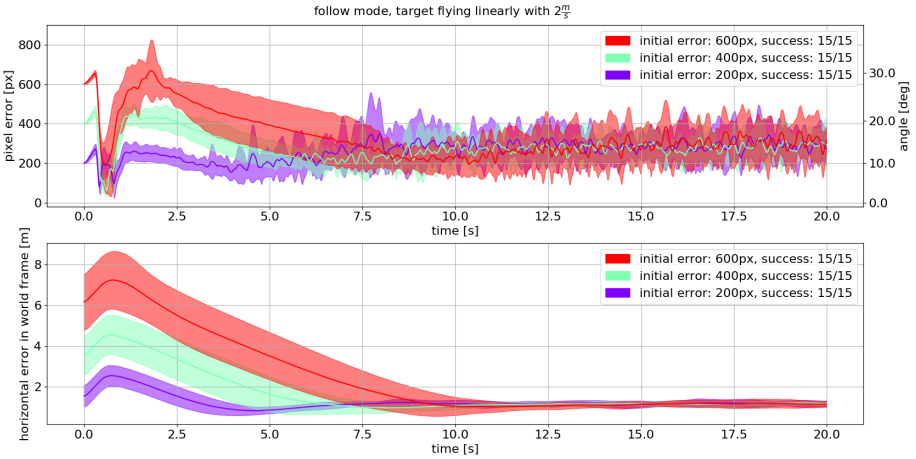

Figure 4.4: Mean and Standard Deviation of Pixel and Horizontal Errors of 15 successful Follow Mode Tests for a Target moving linearly with $2\frac{m}{s}$

and the horizontal error converged faster in the random case than in the linear case.

Linear, $4\frac{m}{s}$

In the case of a target moving linearly with four meters per second (Fig. 4.6) the follow mode was not successful every time. The success rate decreased with increasing initial pixel error. The pixel error converges to a value of about four

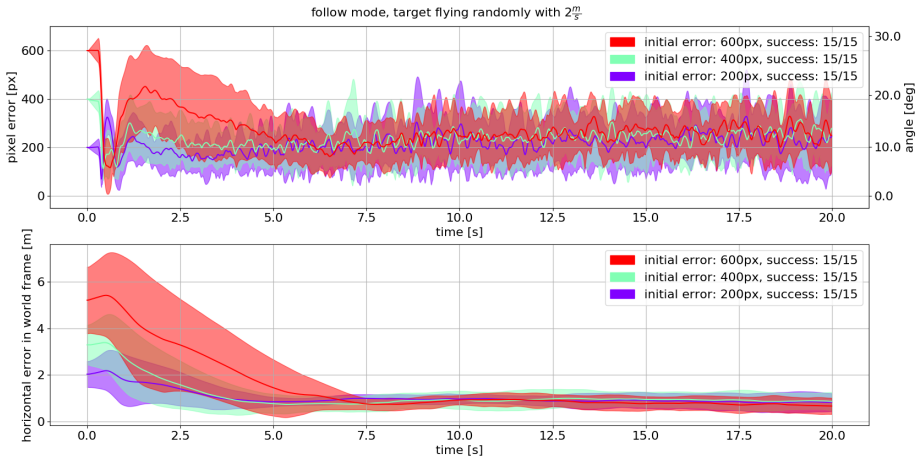

Figure 4.5: Mean and Standard Deviation of Pixel and Horizontal Errors of 15 successful Follow Mode Tests for a Target moving randomly with $2\frac{m}{s}$

hundred pixels, around which it oscillates. Especially for an initial error of six hundred pixels, the pixel error reached values noticeably bigger than in the case of a target flying with two meters per second. The horizontal error converges to a value of approximately two meters and it takes between five and ten seconds to converge.

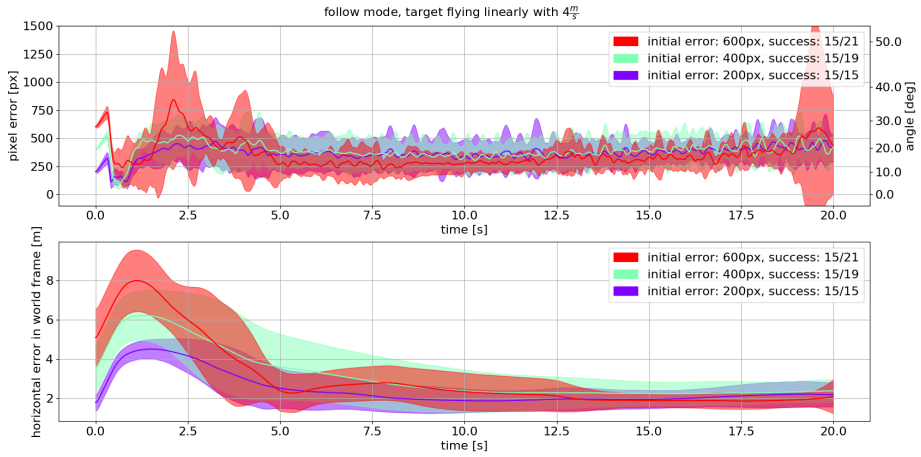

Figure 4.6: Mean and Standard Deviation of Pixel and Horizontal Errors of 15 successful Follow Mode Tests for a Target moving linearly with $4\frac{m}{s}$

Random, $4\frac{m}{s}$

The success rate of the follow mode for a target flying randomly with four meters per second was lower than in the linear case. For an initial pixel error of six hundred pixels, the success rate got so low that it did not make sense to evaluate that case. Therefore in Fig. 4.7 only the cases of an initial pixel error of two- and four hundred pixels are plotted. The pixel error oscillates around a value of about three hundred pixels. The horizontal error is around two meters for the successful tests, however, with a very large standard deviation.

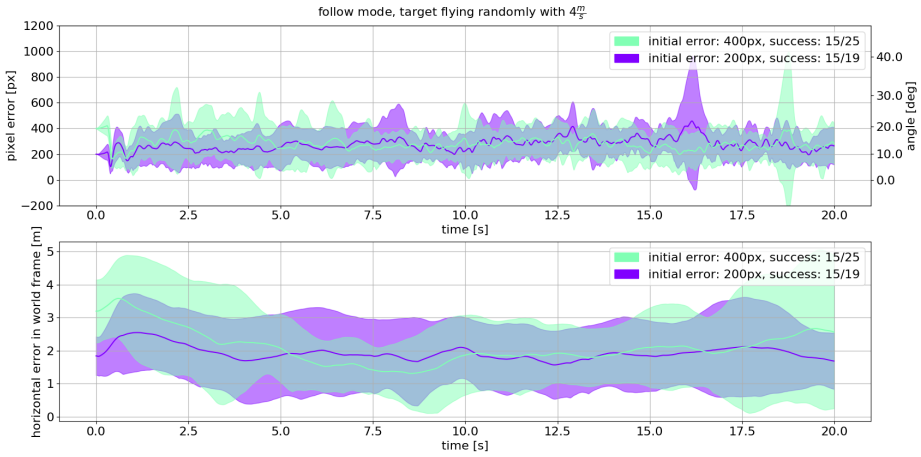

Figure 4.7: Mean and Standard Deviation of Pixel and Horizontal Errors of 15 successful Follow Mode Tests for a Target moving randomly with $4\frac{m}{s}$

Distance

The follow mode aims for a distance of five meters. In Fig. 4.8 one can see how the distance evolves during the twenty seconds in follow mode for the different flight manoeuvres. For this evaluation, all the tests of a certain flight manoeuvre were considered together. For the two linear cases, the distance clearly approaches the five meters. When the target is moving randomly with two meters per second, the distance still decreases and approaches a value that on average is bigger than five. For the four meters per second random case the distance does not decrease on average. The standard deviation in that case is very large.

4.2.2 Catch Mode

In Fig. 4.9 one can see how many out of fifteen times, the target could be caught for different flight manoeuvres. The success rate for a stationary or a linearly moving target was 80% or higher. For a target moving randomly with two

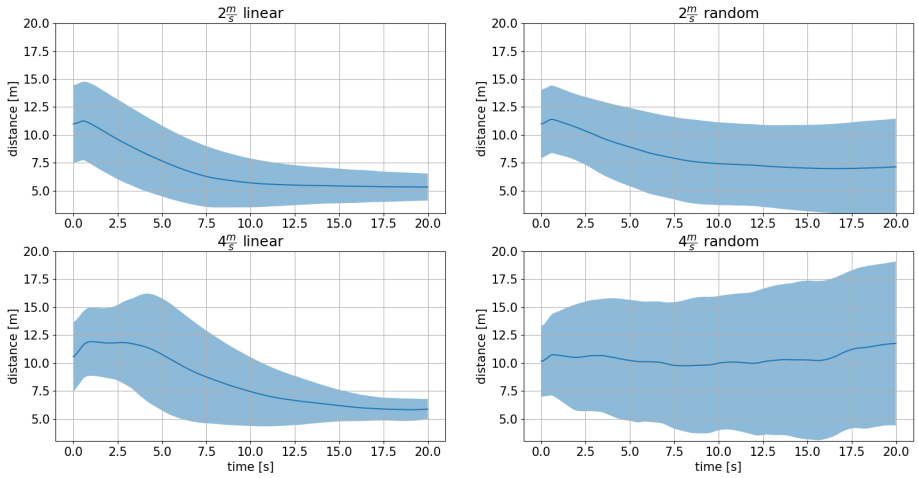

Figure 4.8: Absolute Distance from Carrier to Target Multicopter during Follow Mode for different Flight Manoeuvres

meters per second, still ten out of fifteen tries were successful. For a target moving randomly with four meters per second only two tests were successful.

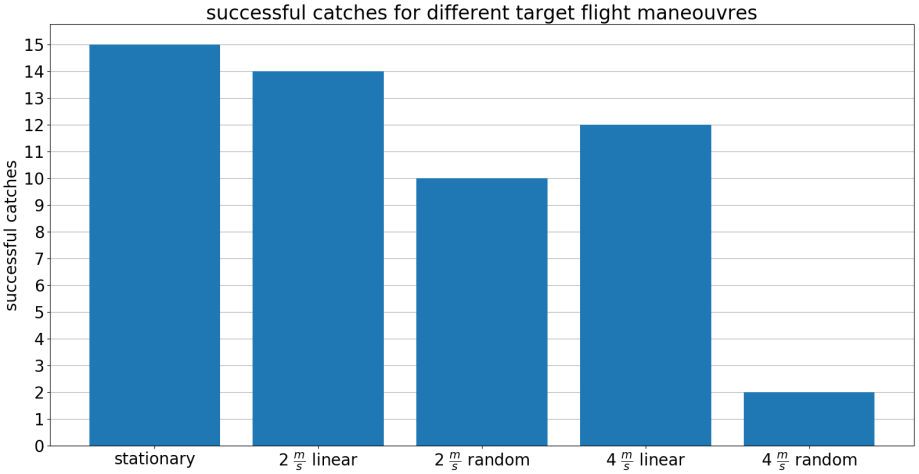

Figure 4.9: Number of successful Catches out of 15 for different Flight Manoeuvres

4.2.3 Safety Modes

Prevent Ground Crash Mode

In both experiments for the prevent ground crash mode, the carrier multicopter did not crash into the ground. In Fig. 4.10 the altitude is plotted over time and the modes are indicated by colours. In the experiment where the target flew over a mountain in a horizontal trajectory, the carrier multicopter was able to fly over the tip of this mountain with a minimum altitude larger than one meter. On the other side a switch back to follow was possible. In the other experiment, where a target multicopter flew towards the ground, the carrier multicopter recovered from flying into the ground twice. After the first time a switch back to follow was possible, but since the target was still flying towards the ground, another switch to prevent ground crash mode was necessary. After recovering for the second time, the distance to the target was so small that with the upwards velocity of the prevent ground crash mode, the target was caught. As the prevent ground crash mode only generates accelerations in the z^W-direction the catch was a coincidence.

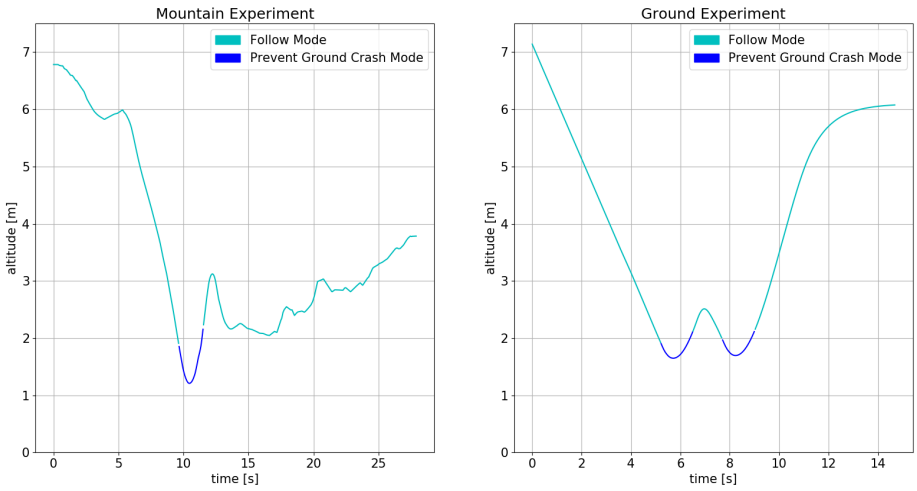

Figure 4.10: Altitude over Time for two different Experiments

Recover Mode

The effect of the recover mode is visualized in Fig. 4.11. The upper plot shows the actual u-position on the image, where the middle plot shows the u-position on the image if roll and pitch were zero. The lower plot shows the evolution of the horizontal distance. At the beginning the target moves closer to the edge of the image. Based on the u-position for the roll and pitch zero assumption, a switch to recover mode occurs. After this switch the u-position of the target gets smaller in both cases, which means the target gets closer to the center of

the image. This causes a switch back to follow mode. Back in follow mode, the planner generates an acceleration away from the target, because the error in u is small, but the velocity high. This causes the carrier multicopter to pitch away from the target, which can be seen in the upper plot as the peak in u. After this peak, the behaviour goes back to the follow mode behaviour as already seen in Section 4.2.1. In the lower plot it is visible that the horizontal distance was significantly reduced in recover mode.

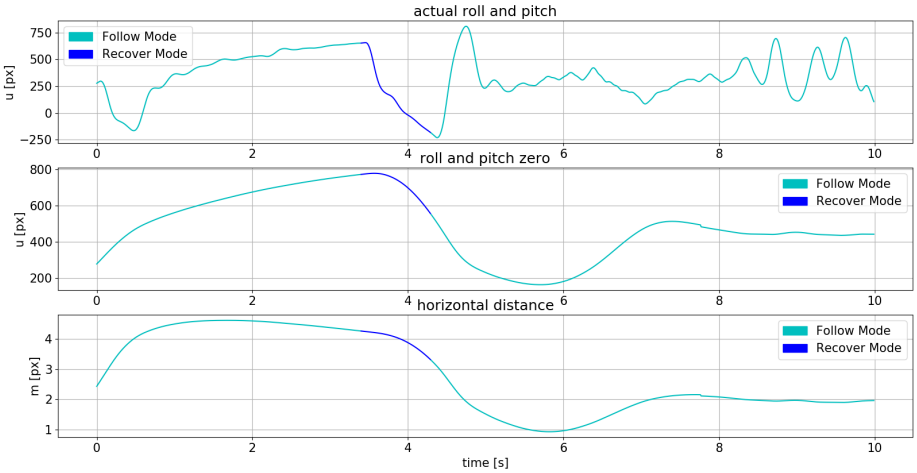

Figure 4.11: Change of the u Pixel Coordinate in Recover Mode

Recover Distance Mode

In Fig. 4.12 it can be seen how the distance between the target and the carrier multicopter behaves in recover distance mode. When the distance increases, the recover distance mode is activated. Its effect can be seen in the fast reduction of the distance. When the distance is close to the target again, a switch back to follow mode occurs, where the fast reduction of the distance stops.

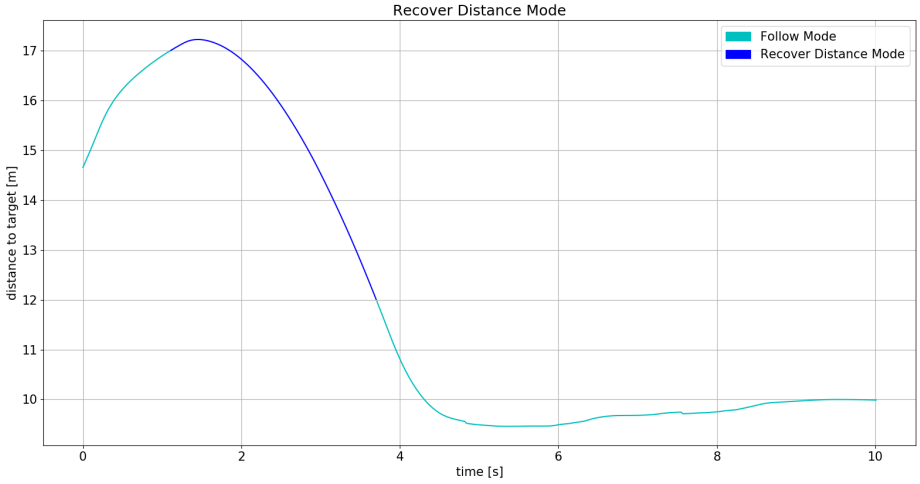

Figure 4.12: Distance from Carrier to Target multicopter over Time, during Recover Distance Mode

Chapter 5

Discussion

In this chapter the results of the experiments presented in Section 4.2 are discussed. The Discussion starts with the follow mode, which was the focus of this thesis. Afterwards the results of the catch and the safety modes are discussed.

5.1 Follow Mode

In general the follow mode was successful in keeping the target in the centre of the image. In Fig. 5.1 the frequencies of the target occurring in different regions of the image are visualized with shadings, where a dark shading indicates a high frequency. As a base for this plot, all of the successful follow mode tests were taken into account. All of these tests sum up to fifty five minutes of follow mode that were evaluated for a variety of different flight manoeuvres. In this plot it is clearly visible that for the majority of the time, the target was within the inner five hundred pixels of u and v or within an angle of view of approximately twenty five degrees. In Fig. 5.2 an example of the motion of the target on the image during a single test of the follow mode is shown.

For a target multicopter flying with two meters per second, all tests of the follow mode were successful. For a linearly moving target, different planning approaches depending on a prediction of the future flight path could have gotten the same result with a very simple tracker. For a randomly moving target, the future path of the target multicopter is very hard to predict, so a planning approach depending on this prediction is not optimal. With this approach, where no prediction was used, the same success rate as in the linear case was possible. In both cases the horizontal distance between the target and the carrier multicopter converged to the same value. In the random case this happened even faster, which makes sense, as the carrier drone can take shortcuts since the flight direction of the target changes. Where the planner performed remarkably better in the linear case is in the distance to the target. In random mode, the target multicopter can also fly steep paths upwards, which was limited in the

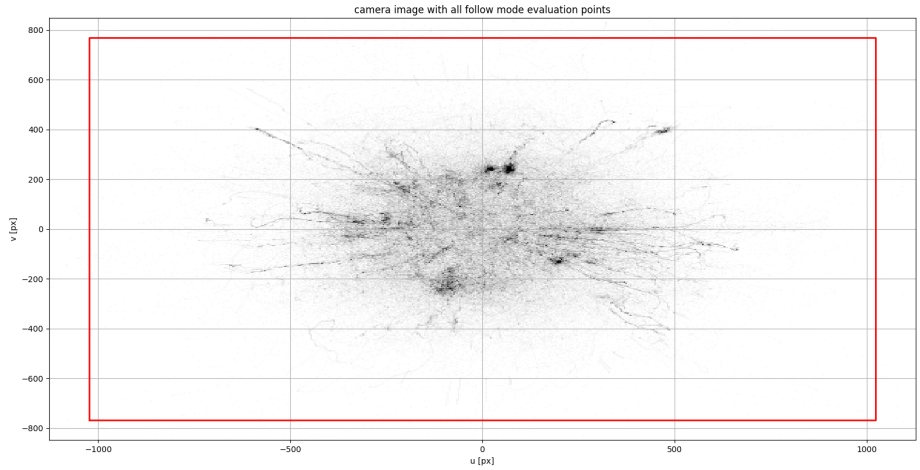

Figure 5.1: Position of the Target on the Image averaged over all Follow Mode Tests

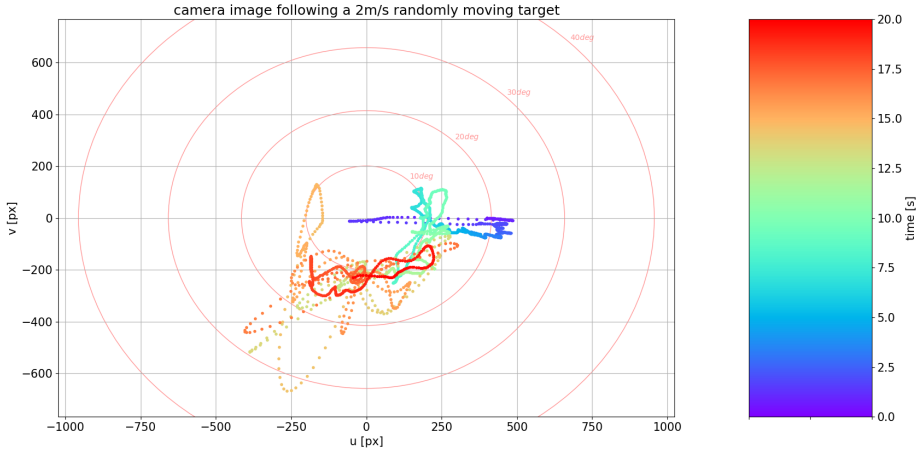

Figure 5.2: Image during a Follow Mode test with a randomly moving Target at $2\frac{m}{s}$

linear case. When the target flew steeply upwards for a longer time, the carrier multicopter could not keep up with the rising velocity of the target multicopter. In that case the distance got bigger and could not be kept at five meters. If the target did not fly steeply upwards, the distance could be kept at five meters very well. The high standard deviations in the distance for the random paths in Fig. 4.8 back up these observations. Considering this problem, the recover distance mode was implemented, which is evaluated separately.

With the target moving randomly at four meters per second, the planner came close to its limits. For the successful tests, which were a lot rarer in this scenario, the mean still seemed to be around the same value for the horizontal error, however, with a very high standard deviation. It was very difficult for the planner to match the fast changes of direction and during successful follows often multiple switches to recover mode took place. When the target was lost, the planner often did not even switch into recover mode. This indicates that the target is moving too fast for the planner to react. This reaction could however be improved with a higher replanning rate because then it is less likely that the recover mode is skipped.

The oscillations that were observed in the image for all cases, did not pose a problem in simulation, where detections were calculated from the world frame positions of the target and the carrier multicopter. For the actual detection, these oscillations could be a problem. The big jumps on the image and possibly also motion blur from the fast changes in orientation could make detection a lot harder. By including a tracker, these oscillations could be reduced. With a prediction as short as the time between to replanning steps, the flight path of the target could be considered in the image policy and the jumps in acceleration described in Section 4.2.1 could be minimized.

5.2 Catch Mode

Catching a target is even harder than following a target without a prediction of the flight path. For a catch the flight path has to be intersected, which is difficult when only the current position of the target multicopter is available. This was solved by an intentional overshoot, which worked quite well for a linearly moving target. For randomly moving targets, this method was less successful. Once the damping is deactivated, the planner cannot react fast enough to sudden changes of direction of the target. Even though the damping term is only deactivated once the distance between the target and the carrier multicopter is very small, sudden changes of direction often lead to a miss of the target, especially for the four meters per second random case. Even with a short prediction of the flight path, the overshoot necessary to catch the target could be reduced, which means that a faster reaction to the change of directions in the flight path would be possible.

5.3 Safety Modes

Prevent Ground Crash Mode

The prevent ground crash mode is very efficient for flat ground. It can be shown that for a flat ground and this particular tuning of the distance to ground policy, the carrier multicopter does not crash into the ground as long as it has a downwards velocity smaller than eight meters per second at the moment of the

switch to prevent ground crash mode. Even though higher velocities are unlikely, in such a case the parameters of the policy could be adjusted. Where this mode could fail as a safety feature is if the ground has a large slope, due to the fact that it only reacts to the current altitude. This can be seen in Fig. 4.10, as the carrier multicopter comes closer to the ground in the mountain experiment than in the ground experiment, even though it did not have a downwards velocity.

Recover Mode

The recover mode proved to be very helpful in terms of reducing the pixel error for the roll and pitch zero assumption. Also the horizontal error is clearly reduced in recover mode. Based on these results, the recover mode is very successful. However, when looking at the pixel error in the actual image (Fig. 4.11), there is a peak when switching back to follow mode that is caused by slowing down. This peak oftentimes gets bigger than the error that caused a switch to recover mode in the first place. Since the switch to recover mode is triggered by the image with zero roll and pitch assumption and the carrier multicopter is facing towards the target in follow mode, the target position on the actual image is not critical yet at the moment the recover mode starts. Therefore, there exists a safety margin to have a pixel error larger than the one at the start of the recover mode when switching back to follow. So at the peak the target will go to the edge of the image, however, it is unlikely that it will leave the image. Even if it would leave the image, it would only leave the image for a very short time causing a miss of maximally two to three detections. In this case the carrier multicopter would continue following the last trajectory received. Because of that and the small horizontal error at the end of the recover mode, the target would most likely get back into the image.

Recover Distance Mode

This mode was implemented to improve the follow mode when it is standing on its own and the target is flying steeply upwards. This mode has not yet been tested extensively in follow mode. However it showed promising results as can be seen in Fig. 4.12. Within three seconds it was possible to reduce the distance by five meters. Because the target multicopter is far away, the risk of the target going out of the image is small. Also the image policy is active and weighted with not negligible importance. Therefore, fast reactions in the horizontal direction are still possible. If it would get critical in this mode, a switch to recover mode is possible.

Chapter 6

Conclusion

6.1 Summary of the Results

In this thesis an alternative planner for "DroGone" to follow and catch a target multicopter was successfully implemented. The planner is aware of about the position of the target on the image and is able to react to the target being at the edge of it. With RMPs it is possible to decouple the planning challenges and solve them separately in their respective spaces. With the weighting of the policies via the metrics, an importance can be given to the accelerations calculated in the different manifolds depending on the situation. The goal of having a planner that is less dependent on a prediction of the future flight path of the target was achieved. No tracker was used at all throughout the thesis. The planner only uses the current position of the target.

For a target velocity of two meters per second, the planner is not only able to follow a linearly, but also a randomly moving target multicopter with a success rate of 100%. This is noteworthy, as planners relying on a prediction of the target flight path would need an accurate prediction to be able to follow a randomly moving target for twenty seconds. Following a target moving with four meters per second is also possible, however the planner gets to its limits at that velocity, which can be seen in the lower success rates.

When it comes to catching a target multicopter, not having a prediction of the flight path makes it very difficult, as the target has to be intersected. With the solution proposed in this thesis, catching stationary and linearly moving targets is possible with a success rate of 80% or higher. Randomly moving targets with two meters per second, can be caught two thirds of the time, however for a velocity of four meters per second, the target can almost never be caught.

The use of no prediction at all is probably not the most efficient approach. Even though there is an advantage for following randomly moving targets, as

predicting their flight path can be difficult, there are also some disadvantages. By including a prediction, the oscillations in the follow mode could be reduced and the catch mode could be made more successful. Those improvements would already be possible with a prediction that is much shorter than the prediction necessary for Dijkstra's algorithm that is currently used at "DroGone".

6.2 Outlook

Due to limitations in time and ressources, the planner was only tested in simulation for this thesis. In the future, the planner should be tested on the real hardware. The real detections and the dynamics of the carrier multicopter can differ from the simulation, so it is of interest to see how the planner performs.

A necessary future improvement is to make the parameters of the policies dynamic. As of now the parameters are constant during a flight. In reality, threatening multicopters will not fly with constant speed and therefore this change is essential. Making these parameters dynamic could be done by fitting a function.

As mentioned before, the use of a short prediction of the flight path could possibly improve the planner. For the calculation of each trajectory the target is assumed to be stationary. With a short prediction, the flight path of the target could be considered. It would also be useful for the calculation of an intersection point in catch mode.

Thanks to the modular structure of Riemannian Motion Policies, more policies could easily be added. In the future of "DroGone", policies such as collision avoidance or following two potential target multicopters may be useful.

Bibliography

[1] A. Balestra, M. Baumgartner, J. Becker, I. Boschung, L. Hänsli, S. Laasch, N. Naimi, F. Stadler, S. Steiner, and L. Strässle, "Autonomous detection, tracking and interception of a multicopter," 2020.

[2] R. Bähnemann, M. Pantic, M. Popović, D. Schindler, M. Tranzatto, M. Kamel, M. Grimm, J. Widauer, R. Siegwart, and J. Nieto, "The eth-mav team in the mbz international robotics challenge," *Journal of Field Robotics*, 2017.

[3] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," 2018.

[4] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor uav," 2017.

[5] N. D. Ratliff, J. Issac, and D. Kappler, "Riemannian motion policies," *CoRR*, vol. abs/1801.02854, 2018.

[6] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot Operating System (ROS) The Complete Reference, Volume 2*, A. Koubaa, Ed. Springer, 2017.

[7] M. Pantic, "rmpcpp," https://github.com/ethz-asl/rmpcpp, 2020, last accessed 15 May 2020.

[8] L. Strässle and F. Stadler, "drogone-rmp," https://gitlab.ethz.ch/lucastr/drogone-rmp, 2020.

[9] L. Strässle and M. Pantic, "drogone-rmpcpp," https://github.com/lucastr98/rmpcpp, 2020.

[10] gazebosim.org, "Gazebo," http://gazebosim.org/, 2020, last accessed 02 October 2020.

[11] ASL, "Rotors," https://github.com/ethz-asl/rotors_simulator, 2019, last accessed 02 October 2020.

[12] ROS, "Rviz," http://wiki.ros.org/rviz, 2018, last accessed 02 October 2020.

Appendix A

Appendix

A.1 Derivation Image Jacobian

For the derivation of the Jacobian from world frame to image frame one needs the image coordinates (u, v) expressed in terms of the world coordinates (x, y, z). For this purpose, the position of the target multicopter in world frame $(x_{tm}^W, y_{tm}^W, z_{tm}^W)$ is transformed into image frame, depending on the carrier multicopters position in world frame (x, y, z).

A.1.1 World to Body

First the position of the target is transformed from world into body frame.

$$\boldsymbol{p}_{tm}^B = T_{B\leftarrow W} \cdot \boldsymbol{p}_{tm}^W \tag{A.1}$$

This transformation consists of a rotation and a translation.

$$T_{B\leftarrow W} = \begin{bmatrix} R_{B\leftarrow W} & -R_{B\leftarrow W} \cdot \boldsymbol{t}_{B\leftarrow W} \\ \mathbf{0} & 1 \end{bmatrix} \tag{A.2}$$

The rotation matrix is defined through roll, pitch and yaw angles of the orientation of the carrier multicopter with respect to the world frame.
A rotation matrix defined through roll ϕ, pitch θ and yaw ψ angles is generally defined as:

$$R = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{A.3}$$

For the derivation of the image Jacobian, the rotation matrix from world to body frame assumes the roll and pitch angles to be zero $\theta_B^W = \phi_B^W = 0$. This is further explained in Section 3.2.2. With this assumption, the rotation matrix only

depends on the yaw orientation. With the yaw angle of the carrier multicopter in world frame $\psi = \psi_B^W$ the rotation matrix is defined as:

$$R_{B\leftarrow W} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

The translation vector is defined as the position of the carrier multicopter in world frame.

$$\boldsymbol{t}_{B\leftarrow W} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.5}$$

With the transformation matrix applied to the target multicopters position in homogeneous coordinates $\boldsymbol{p}_{tm}^W = \left(x_{tm}^W, y_{tm}^W, z_{tm}^W, 1\right)^T$ it is transformed from world to body frame.

$$\boldsymbol{p}_{tm}^B = \begin{bmatrix} \left(x_{tm}^W - x\right)\cos\psi + \left(y_{tm}^W - y\right)\sin\psi \\ -\left(x_{tm}^W - x\right)\sin\psi + \left(y_{tm}^W - y\right)\cos\psi \\ z_{tm} - z \\ 1 \end{bmatrix} \tag{A.6}$$

A.1.2 Body to Camera

From body frame the target position has to be transformed to camera frame.

$$\boldsymbol{p}_{tm}^C = T_{C\leftarrow B} \cdot \boldsymbol{p}_{tm}^B \tag{A.7}$$

The transformation again consists of a rotation and a translation.

$$T_{C\leftarrow B} = \begin{bmatrix} R_{C\leftarrow B} & -R_{C\leftarrow B} \cdot \boldsymbol{t}_{C\leftarrow B} \\ \mathbf{0} & 1 \end{bmatrix} \tag{A.8}$$

The rotation matrix is defined through roll, pitch and yaw angles of the orientation of the camera mounting on the carrier multicopter. “DroGone” has its cameras mounted facing upwards and therefore the axes of the camera frame are aligned with the axes of the body frame. Therefore roll, pitch and yaw are zero $\theta_B^W = \phi_B^W = \psi_B^W = 0$. With (A.3) one can see, that in this case the rotation matrix is the identity matrix.
The translation is defined as the position of the camera mounting with respect to the body frame. As this thesis was entirely done in simulation this translation was also assumed to be zero. Therefore the position of the target multicopter in camera frame is the same as in body frame.

$$\boldsymbol{p}_{tm}^C = \boldsymbol{p}_{tm}^B = \begin{bmatrix} \left(x_{tm}^W - x\right)\cos\psi + \left(y_{tm}^W - y\right)\sin\psi \\ -\left(x_{tm}^W - x\right)\sin\psi + \left(y_{tm}^W - y\right)\cos\psi \\ z_{tm} - z \\ 1 \end{bmatrix} \tag{A.9}$$

A.1.3 Camera to Image

The final transformation is the transformation from camera to image frame.

$$\boldsymbol{p}_{tm}^{I} = K \cdot \boldsymbol{p}_{tm}^{C} \tag{A.10}$$

For this transformation one needs the camera matrix. For the camera matrix one needs the focal lengths f_x, f_y and the coordinate of the principal point on the image u_0, v_0.

$$K = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

This gives the target multicopters position in image frame in homogeneous coordinates.

$$\boldsymbol{p}_{tm}^{I} = \begin{bmatrix} u' \\ v' \\ \lambda \end{bmatrix} = \begin{bmatrix} f_x \left(\left(x_{tm}^{W} - x \right) \cos\psi + \left(y_{tm}^{W} - y \right) \sin\psi \right) + u_0 \left(z_{tm}^{W} - z \right) \\ f_y \left(- \left(x_{tm}^{W} - x \right) \sin\psi + \left(y_{tm}^{W} - y \right) \cos\psi \right) + v_0 \left(z_{tm}^{W} - z \right) \\ z_{tm}^{W} - z \end{bmatrix} \tag{A.11}$$

To get the target multicopters position in image frame, these homogeneous coordinates have to be normalized in a last step by dividing through λ.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f_x}{z_{tm}^{W} - z} \left(\left(x_{tm}^{W} - x \right) \cos\psi + \left(y_{tm}^{W} - y \right) \sin\psi \right) + u_0 \\ \frac{f_y}{z_{tm}^{W} - z} \left(- \left(x_{tm}^{W} - x \right) \sin\psi + \left(y_{tm}^{W} - y \right) \cos\psi \right) + v_0 \end{bmatrix} \tag{A.12}$$

A.1.4 Jacobian

Now that the position of the target multicopter in image frame depending on the world frame position of the carrier multicopter is available, the Jacobian can be calculated.

$$\begin{aligned} J_{uv} &= \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \end{bmatrix} \\ &= \begin{bmatrix} \frac{f_x \cos\psi}{z - z_{tm}^{W}} & \frac{f_x \sin\psi}{z - z_{tm}^{W}} & -\frac{f_x}{\left(z - z_{tm}^{W} \right)^2} \left(\left(x - x_{tm}^{W} \right) \cos\psi + \left(y - y_{tm}^{W} \right) \sin\psi \right) \\ -\frac{f_y \sin\psi}{z - z_{tm}^{W}} & \frac{f_y \cos\psi}{z - z_{tm}^{W}} & -\frac{f_y}{\left(z - z_{tm}^{W} \right)^2} \left(\left(x - x_{tm}^{W} \right) \sin\psi - \left(y - y_{tm}^{W} \right) \sin\psi \right) \end{bmatrix} \end{aligned} \tag{A.13}$$