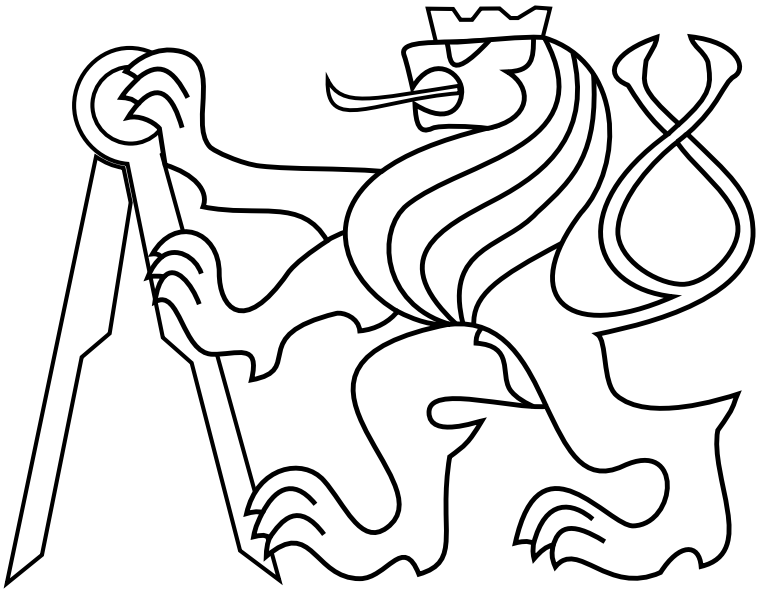
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Daniel Heřt

Autonomous Predictive Interception of a Flying Target by an Unmanned Aerial Vehicle

Department of Measurement

Thesis supervisor: **Ing. Tomáš Báča**

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24. 5. 2018 .....

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Heřt Daniel** Personal ID number: **406429**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Measurement**

Study program: **Cybernetics and Robotics**

Branch of study: **Air and Space Systems**

II. Master's thesis details

Master's thesis title in English:

Autonomous Predictive Interception of a Flying Target by an Unmanned Aerial Vehicle

Master's thesis title in Czech:

Automatické prediktivní zastavení letícího cíle s pomocí bezpilotní helikoptéry

Guidelines:

The thesis aims to develop a system to autonomously intercept a flying target with an unmanned aerial vehicle (UAV). Such approach provides the means of defense against foreign unmanned aerial vehicles and has a potential to be applied against off-the-shelf camera drones. The work will be structured as follows:
- Familiarize yourself with the UAV platform which is currently used in the MRS lab at FEE CTU.
- Integrate an existing marker-based computer vision system [1] for detection of the target into the UAV platforms.
- Design and implement a state observer for estimating and predicting the states of the target based on the computer vision system.
- Improve upon the current predictive control scheme in [2] to allow precise tracking along desired trajectories in the Robot Operating System (ROS).
- Design an interception strategy and implement it as a state machine in ROS.
- Demonstrate the performance of the system, when using the visual localization, in the realistic Gazebo simulator in ROS.
- Test the designed interception mechanism in experiments with a balloon attached to another UAV of the same platform which is localized with a precise GPS system. The experiments shall test the estimation, control and planning part of the work and shall provide requirements for a future relative localization system.

It should be stressed out that the development of a computer vision system (relative localization of the target) is not in the scope of this work.

Bibliography / sources:

[1] T. Krajnik, et al.: A practical multirobot localization system. Journal of Intelligent & Robotic Systems 76.3-4 (2014): 539-562.
[2] T. Baca, G. Loianno and M. Saska: Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles. In 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR). 2016.
[3] J. Mattingley, and S. Boyd: CVXGEN: A code generator for embedded convex optimization. Optimization and Engineering 13.1 (2012): 1-27.
[4] Eric A. Wan, and R. Van Der Merwe: The unscented Kalman filter for nonlinear estimation. Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000..

Name and workplace of master's thesis supervisor:

Ing. Tomáš Báča, , FEL

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **30.01.2018** Deadline for master's thesis submission: _____

Assignment valid until:
by the end of summer semester 2018/2019

Ing. Tomáš Báča Supervisor's signature	Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

8.2 2018 Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor Ing. Tomáš Báča for his great support and all the knowledge he passed to me. Furthermore, I want to thank all the members of the Multi-robot Systems group for their help with the real-world experiments. My final thanks go to my family, who supported and encouraged me throughout my studies.

Abstract

The goal of this thesis is to develop a system for autonomous interception of a small flying target by an unmanned multirotor aircraft. The main work lies in developing an interception strategy and a trajectory planner, improving upon the current model predictive control scheme to allow for aggressive maneuvers, and developing a state observer to estimate and predict the states of the target aircraft. The functionality of the developed system is verified in a series of simulations and outdoor experiments in different configurations and with different target localization techniques.

Abstrakt

Cílem této práce je vyvinout systém pro autonomní zastavení letícího cíle pomocí bezpilotní helikoptéry. Hlavní náplní této práce je vývoj strategie a plánování trajektorií pro zastavení cíle, vylepšení současného prediktivního řídícího systému (model predictive control) aby bylo možné provádět agresivní manévry, a vývoj stavového pozorovatele, který bude odhadovat a předpovídat stavy cíle. Funkcionalita systému bude ověřena v sérii simulací i venkovních experimentů s různými konfiguracemi a metodami pro lokalizaci cíle.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Small unmanned aerial vehicles (UAVs), sometimes also called drones or MAVs (micro aerial vehicles) are more and more popular and are nowadays used even by the general public. This is mainly caused by the advances in lithium polymer batteries, small and powerful motors and control electronics. All of these components are getting cheaper and more available, making it possible to build MAVs with a very low price tag. The most popular configuration for a MAV is a multicopter, usually equipped with four or six motors with propellers. The multicopter concept is very simple, as the only moving parts are the counter-rotating propellers, and control of the vehicle is achieved only by changing their relative speeds. Big advantages of the multicopter platform include the ability to perform vertical takeoff and landing, execute agile maneuvers, or the ability to hover in place. This has made the platform very popular for aerial photography and capturing videos, with many commercially available models equipped with high resolution stabilized cameras, ready to fly out of the box, like the DJI Phantom shown in figure 1.

Figure 1: DJI Phantom, a very popular quadcopter equipped with a stabilized high resolution camera.

The boom of multicopters also has a negative side. There is an increasing number of cases when people fly their multicopters into forbidden areas like the surroundings of airports, where they are directly endangering landing aircraft, or high-security areas like power stations or military installations. Multicopters equipped with cameras can also easily intrude into ordinary people's privacy by flying over private property and capturing photos and videos. Multicopters can also directly cause injuries and property damage, as they can be operated by untrained individuals or suffer breakdowns while flying over populated areas.

This led to the advent of anti-drone devices, like jammers [25], which can block the operator's control signal and the signal from satellite navigation [12], which is often used by drones. Drones can also be eliminated by hand-held launchers [2], which can launch projectiles or nets at the flying drones. Attempts have also been made to train eagles and other birds to eliminate drones, with mixed results. Military solutions then include various guns, missiles, and even lasers.

This thesis will approach the problem of disabling drones by other means, as goes the famous saying: "The best way to defeat a tank is with another tank", then surely the best way

to disable a drone is with another drone. The goal of this thesis is to develop algorithms and strategies for autonomous intercepting and disabling of unwanted drones by an interceptor drone. This includes control, estimation and strategic trajectory planning for the interceptor drone, to autonomously perform the task of eliminating other drones.

Chapter 1 of this thesis gives an introduction to the stated problem and an overview of state of the art. Chapter 2 describes a state observer, which can estimate and predict states of the target drone based on available measurements. In chapter 3, the current MPC tracker scheme, developed in the MRS lab at CTU in Prague, is improved to increase the performance of the whole system and allow for easier trajectory planning. Chapter 4 describes different possibilities of target elimination, as well as trajectory planning algorithms and interception strategies, based on different levels of information about the position of the target. Chapter 5 then introduces the hardware and software platform used for verification, and Chapter 6 describes experimental results in simulations and the real world.

1.1 State of the art

Many publications are addressing the issue of drone security and possible threats. In [28], the authors explore security flaws of the AR Drone, and the possibility of using them for malicious activities, like person tracking. Similarly, [34] describes various ways in which the commercially available multicopter can be used for criminal and terrorist acts and some possibilities of threat mitigation.

Large amount of publications focus on drone detection and tracking by various ground-based sensors. These systems are often paired with RF (radio frequency) jammers, to disable the link between the operator and the drone. For the purposes of target detection, low probability of intercept radar, combined with noise detection is used in [36], and as a counter-measure, different techniques are used to jam the remote control signal of the drone. In [10] a 35 GHz continuous wave frequency modulating radar was used to detect small drones and estimate their positions and velocities. Authors of [32] developed a drone detection system, based on acoustic sensing, optical cameras and RF sensors, paired with a jammer that can eliminate the RF link to the intruding drone. Inverse synthetic aperture radar is used to track an image flying drones in [18]. The technique of RF jamming can be effective in disabling or deterring some drones, but this approach is not consistent, as the drone can use various different frequencies for communication, or it can be semi or completely autonomous, which removes the need of the RF link and therefore make the jammer completely ineffective.

A different way of defending against an intruding drone is explored in [15], where spoofing of the GPS signal is used to take control of a drone, and send it outside of the defended area, or force it to crash. Authors of [6] are proposing a swarm of defending UAVs, with a task to envelop the intruding UAV in a tight formation and escort it outside of the defended area. This approach was however only tested in a simulation with simplified physics, and it would be very difficult to execute in the real world. In [20], a methodology is presented for assessing the design of an interceptor UAV, so-called "anti-drone drone", along with interception strategies and simulations of a scenario involving a defense of a nuclear power plant.

Most of the publications in the area of counter drone operations focus on drone detection, classification, and tracking, while the problem of incapacitation of the intruding drone is either tackled with jammers or GPS spoofing. While some authors are proposing to use specialized interceptor drones for target elimination, most of these solutions are verified only in simulations, not in real life. Commercially available systems for eliminating intruding drones are available, but these systems are mostly manually controlled, or they are ground-based. This thesis proposes an autonomous interceptor drone, with a task of capturing or eliminating an intruding target drone.

1.2 Mathematical notation

The mathematical notation used throughout this thesis is described in table 1.

Symbol	Meaning
Upper or lowercase letter (a, b, N, Q)	scalar
Bold lower case letter $(\mathbf{x}, \mathbf{y})$	column vector
Bold upper case letter $(\mathbf{A}, \mathbf{B})$	matrix
$\mathbf{x}^T, \mathbf{Q}^T$	vector and matrix transpose
$x_{[t]}, \mathbf{x}_{[t]}$	$x, \mathbf{x}$ at a time sample t
$\dot{x}, \ddot{x}$	first and second time derivative of x
$diag(a, b, ..., z)$	diagonal matrix with a,b,...,z on the diagonal
$\mathbb{N}$, $\mathbb{R}$	set of natural and real numbers

Table 1: Mathematical notation used in this thesis.

2 Target state estimation

Information about position of the target is critical for the interception scenario. This information can be obtained by a ground-based tracking station, or by onboard sensors. In both cases, positional information will most likely not be continuous, due to many factors, like drop-outs in communication or target flying out of range or out of the FOV (field of vision) of the tracking cameras. In all those cases, we need to maintain an estimate of the target's position, to be able to continue with the interception, or adjust the orientation and position of the intercepting drone to regain tracking of the target. Estimating target's future trajectory from the current and past states is also beneficial, as it can be directly used to optimize the trajectory of the intercepting drone using the MPC tracker, which will be described in the next chapter. The information about target position will also most likely contain noise, which can be filtered by the estimator. To address all these requirements, linear Kalman filter was implemented in this thesis, to serve as a state estimator.

2.1 Linear Kalman filter

Linear Kalman filter (LKF) is an algorithm for state estimation. It can be easily implemented on digital computers, and it is recursive - it does not need to keep the whole set of measured data. If all noise in the system is Gaussian, the LKF will minimize the mean square error, if the noise is not Gaussian, LKF is still the best linear estimator. LKF was first described in [14] and [13].

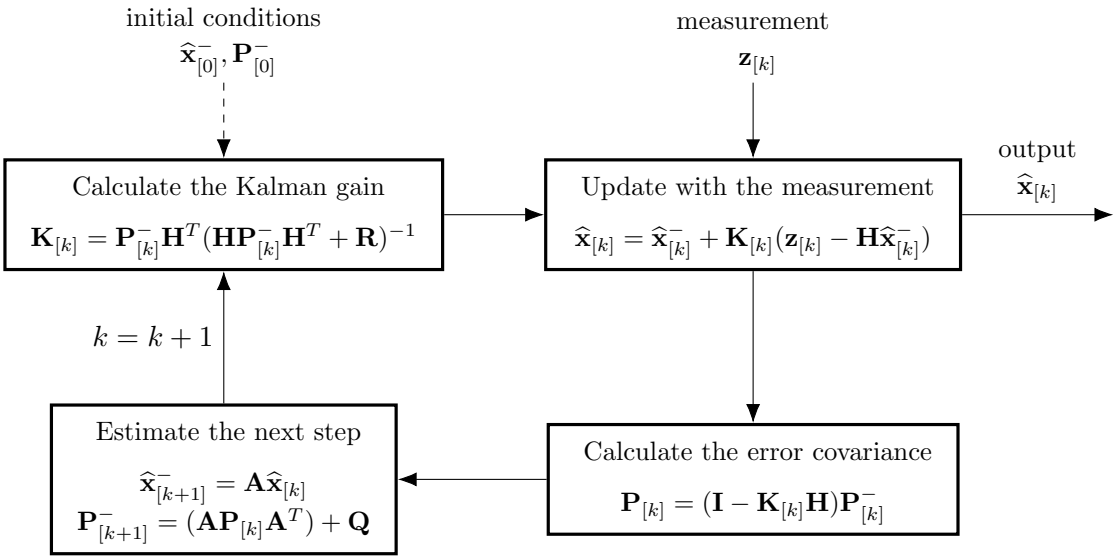


Figure 2: Diagram of the linear Kalman filter loop.

LKF contains a model of the estimated system, which is used to produce an estimate of the system state $\widehat{\mathbf{x}}_{[k]}$ and an estimate of an error covariance matrix $\widehat{\mathbf{P}}_{[k]}$ based on the previous estimate and new measured data. The error covariance matrix represents the accuracy of the

estimated state produced by the LKF. Diagram of the LKF loop with all equations is shown in figure 2 and all used symbols are described in table 2.

Symbol	Dimension	Meaning
n	$\mathbb{N}$	Number of states.
m	$\mathbb{N}$	Number of measured variables.
$\widehat{\mathbf{x}}_{[k]}$	$\mathbb{R}^{n\times 1}$	Estimate of the system state at a time sample k updated with a measurement.
$\widehat{\mathbf{x}}^{-}_{[k]}$	$\mathbb{R}^{n\times 1}$	Prior estimate of the system state at a time sample k.
$\widehat{\mathbf{x}}^{-}_{[0]}$	$\mathbb{R}^{n\times 1}$	Initial estimate of the system state.
$\mathbf{z}_{[k]}$	$\mathbb{R}^{m\times 1}$	Measurement at a time sample k.
$\mathbf{K}_{[k]}$	$\mathbb{R}^{n\times m}$	Kalman gain at a time sample k. Kalman gain decides how much the new measurement changes the estimated state.
$\mathbf{P}_{[k]}$	$\mathbb{R}^{n\times n}$	Error covariance matrix at a time sample k after the measurement update. This matrix represents the estimated accuracy of the state estimate.
$\mathbf{P}^{-}_{[k]}$	$\mathbb{R}^{n\times n}$	Prior error covariance matrix at a time sample k
$\mathbf{P}^{-}_{[0]}$	$\mathbb{R}^{n\times n}$	Initial error covariance matrix
$\mathbf{R}$	$\mathbb{R}^{m\times m}$	Measurement noise covariance matrix
$\mathbf{Q}$	$\mathbb{R}^{n\times n}$	Process noise covariance matrix
$\mathbf{A}$	$\mathbb{R}^{n\times n}$	State transition matrix.
$\mathbf{H}$	$\mathbb{R}^{m\times n}$	Observation matrix. This matrix defines a transformation between the measured values and states of the system.
$\mathbf{I}$	$\mathbb{R}^{n\times n}$	Identity matrix.

Table 2: Meaning of symbols used in the linear Kalman filter.

In a general case, matrices $\mathbf{R}, \mathbf{Q}, \mathbf{A}$ and $\mathbf{H}$ can change with each iteration, but in our case, their values remain the same.

2.2 Implementation

In the interception scenario with a camera tracking system, we can expect that only the position of the target can be measured, as the attitude is very hard to determine just from images, as the appearance of the target is not known apriori. With this assumption, we can

use a model of a point mass moving in 3-dimensional space, in our linear Kalman filter. The estimate of the state vector $\widehat{\mathbf{x}}_{[k]}$ will contain position, speed and acceleration in X, Y and Z ($n = 9$), and the A matrix will contain transitional dynamics for an object in 3 dimensional space:

$$\widehat{\mathbf{x}}_{[k]} = \begin{bmatrix} x_{[k]} \\ y_{[k]} \\ z_{[k]} \\ \dot{x}_{[k]} \\ \dot{y}_{[k]} \\ \dot{z}_{[k]} \\ \ddot{x}_{[k]} \\ \ddot{y}_{[k]} \\ \ddot{z}_{[k]} \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Measurement vector $\mathbf{z}_{[k]}$ will contain only information about position in X,Y and Z ($m = 3$) directions and $\mathbf{H}$ represents transformation between the estimate of the state vector $\widehat{\mathbf{x}}_{[k]}$ and $\mathbf{z}_{[k]}$:

$$\mathbf{z}_{[k]} = \begin{bmatrix} x_{[k]} \\ y_{[k]} \\ z_{[k]} \end{bmatrix}, \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2}$$

Finally, the matrices $\mathbf{Q}$ and $\mathbf{R}$ has to be set. The matrix $\mathbf{R}$ is the measurement noise covariance matrix, and it represents noise in the measured data. Values in this matrix can be determined by setting the system to a steady state and observing the noise in the measurements. The $\mathbf{Q}$ matrix represents the noise in the process. The goal of setting the $\mathbf{Q}$ and $\mathbf{R}$ is to get an output which is without noise but still reacts quickly to changes in states. Value of $\mathbf{R}$ was in this case initially set by observing the measurement data, and $\mathbf{Q}$ was then determined empirically along with adjustments to $\mathbf{R}$. Following values were used in case of using the WhyCon relative localization system (which will be described later) as a source of measurement data:

$$\mathbf{R} = diag(0.01, 0.01, 0.01), \qquad \mathbf{Q} = diag(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1). \tag{3}$$

If another source of target position measurement is used, the $\mathbf{Q}$ and $\mathbf{R}$ matrices need to be readjusted to give proper results.

The diagram of the linear Kalman filter shown in figure 2 assumes a constant stream of measurements and only outputs new estimates if new measurements are coming in. As stated

before, in this application it is not expected to get a constant stream of measurements, as the target can fly outside the range or FOV of the sensor. The frequency of the calculated estimates should be constant, so that it can be directly used in the 100 Hz MPC control loop, but the rate of incoming measurements is usually 30 Hz or lower, if an onboard camera is used.

To address both of these points, the frequency of the LKF loop is set at 100 Hz, in sync with the MPC control loop. If a new measurement from the tracking system is available, the full LKF loop, which is shown if figure 2 is executed. If however there is no new measurement, the whole update phase of the LKF loop is skipped and only the next step estimation is executed. This simplified loop is shown in figure 3.

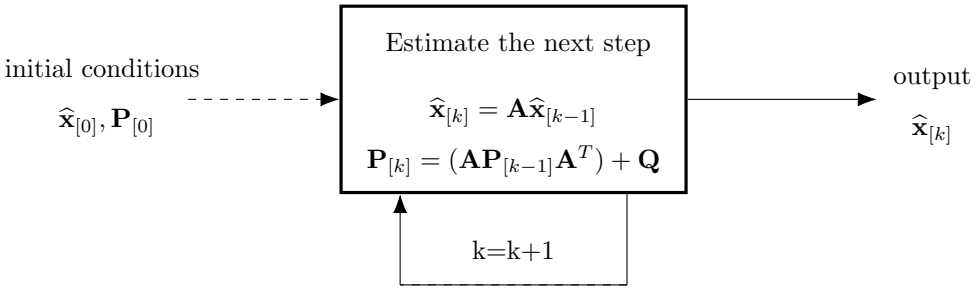


Figure 3: Diagram of the simplified LKF loop without measurement updates.

In case of the simplified loop, the current estimated state vector is only propagated to the future by the transition matrix $\mathbf{A}$, and the error covariance matrix increases, which corresponds to increasing uncertainty of the estimated state, as there are no measurement updates. This approach is used to bridge the gap between the 100 Hz control loop and slower measurements from the sensors, and also to provide an estimate of the target position if there is drop-out in measurements.

This estimate can be used to regain tracking of the target, but the accuracy of the estimate decreases rapidly. The accuracy of the estimate can be determined by checking the error covariance matrix. In our case, a threshold is set for values in the error covariance matrix. If this threshold is crossed, the estimate of the target state is declared as unusable and is ignored. The Kalman filter is then reset with the next available measurement and new initial value for the error covariance matrix.

3 Model Predictive Control

Model predictive control (MPC), also called the Moving or Receding Horizon Control (MHC, RHC), is an advanced control method, which first found its use in slow industrial processes. These processes usually operate with long time constants and thus computational power was not a limiting factor. The main advantage of the MPC approach is that the current control action is being optimized with regard to predicted future states of the system, and not only the current state. This allows the system to anticipate future events, and react to them with proper control action before they happen.

To achieve this behavior, MPC uses a dynamical model of the controlled system. Measured states of the real system are injected into the dynamical model. Series of control inputs is then applied to the dynamical model, which produces a prediction of future system states over a certain time horizon (often called the prediction horizon). The goal of the MPC approach is to optimize the series of control inputs with respect to a certain cost function. This function, also called the objective function, usually penalizes the difference between predicted and desired states, and the magnitude of the control input itself. However, it can also penalize other aspects of control, like control input slew rate. The output of the objective function is a scalar value, which represents how well is the system being controlled. The goal is to find the global minimum of this function, which represents the optimal control action with regards to our definition of the objective function. Once the optimization process is completed, the first of the series of optimized control inputs is applied to the real system. Then the new state of the real system is measured, and the whole process repeats.

Since we predict the system states and optimize the control action for the whole prediction horizon in each control step, significant computational power is needed. But with advancements in hardware performance, MPC started to spread to systems with faster dynamics, and nowadays it can be used to drive systems with control loops running at tens or hundreds of hertz, like multirotor UAVs. Since MPC is usually implemented using computers or embedded hardware, we will only consider the time to be discrete.

3.1 UAV Model

To produce useful predictions about the future states of the system, we need a model that represents our system. Based on the initial state $\mathbf{x}_{[0]}$ we can calculate the future states of the system $\mathbf{x}_{[1]}, \mathbf{x}_{[2]}, ..., \mathbf{x}_{[M]}$, where M is the length of the prediction horizon, by applying a series of control inputs $\mathbf{u}_{[0]}, \mathbf{u}_{[1]}, ..., \mathbf{u}_{[M-1]}$ to the model. In general, any kind of model can be used for the purpose of MPC, but in practice, it is best to use a linear model since its behavior facilitates optimization. In this thesis, we assume a linear, discrete, time-invariant system with n states, n outputs, and k inputs:

$$\begin{aligned} \mathbf{x}_{[t+1]} &= \mathbf{A}\mathbf{x}_{[t]} + \mathbf{B}\mathbf{u}_{[t]}, \\ \mathbf{y}_{[t]} &= \mathbf{C}\mathbf{x}_{[t]} + \mathbf{D}\mathbf{u}_{[t]}, \end{aligned} \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the state matrix, $\mathbf{B} \in \mathbb{R}^{n \times k}$ is the input matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$ is the output matrix, $\mathbf{D} \in \mathbb{R}^{n \times k}$ is the direct transition matrix and t is a discrete time sample. We also

assume that $\mathbf{C} = \mathbf{I}$, the output states consist of all the system states, and $\mathbf{D} = \mathbf{0}$, there is no direct transfer from the input to the output.

3.2 Objective function

As stated before, the objective function outputs a single scalar value that represents the quality of the found solution. During the optimization process, we are trying to minimize the objective function with the goal of finding the global minimum, which represents optimal control action with respect to our objective function. A quadratic objective function is often used in MPC as it penalizes larger deviations from reference more than small deviations. By using a quadratic objective function, the whole problem of solving MPC reduces to solving a quadratic programming (QP) problem. There are many solvers for QP problems, for example in MATLAB we can use the quadprog function. Solver used in our case is discussed in section 3.4.

The objective function can penalize and evaluate any of the aspects of the system, but for basic functionality, we penalize the difference between predicted states and reference states, which forces the system to follow a provided reference. In addition, the magnitude of the control input itself is also penalized, to reduce control action to a necessary minimum. We can define the reference state as follows:

$$\mathbf{x}_{r[t]} = \begin{bmatrix} x_{r1} \\ x_{r2} \\ \vdots \\ x_{rn} \end{bmatrix}, \tag{5}$$

where n is the number of states and $\mathbf{x}_{r[t]}$ is the reference state at a time sample t. We can then define tracking error as follows:

$$\mathbf{e}_{[t]} = \mathbf{x}_{[t]} - \mathbf{x}_{r[t]} = \begin{bmatrix} x_1 - x_{r1} \\ x_2 - x_{r2} \\ \vdots \\ x_n - x_{rn} \end{bmatrix}, \tag{6}$$

where $\mathbf{e}_{[t]}$ is the tracking error and $\mathbf{x}_{[t]}$ is predicted state at a sample time t. Since we want to use a quadratic objective function, we need to formulate it as a sum of squares of tracking errors and magnitudes of the control input. We define the objective function as follows:

$$V_{(\mathbf{x},\mathbf{u})} = \frac{1}{2} \sum_{i=1}^{M} (\mathbf{e}_{[i]}^T \mathbf{Q} \mathbf{e}_{[i]}) + \mathbf{u}_{[i-1]}^T \mathbf{R} \mathbf{u}_{[i-1]}), \tag{7}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{[1]} \\ \mathbf{x}_{[2]} \\ \vdots \\ \mathbf{x}_{[M]} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \mathbf{u}_{[0]} \\ \mathbf{u}_{[1]} \\ \vdots \\ \mathbf{u}_{[M-1]}, \end{bmatrix},$$

where M is the length of the prediction horizon, $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is the state weighting matrix and $\mathbf{R} \in \mathbb{R}^{k \times k}$ is the input weighting matrix. Matrices $\mathbf{Q}$ and $\mathbf{R}$ are required to be positive semi-definite.

MPC implementations also often include penalization of the final state and control input ($\mathbf{e}_{[M]}$, $\mathbf{u}_{[M-1]}$) with higher weights, to force the system to converge faster to the reference. This is however not needed in case of this thesis, because the prediction horizon is long enough (8 seconds). We can also extend the objective function to penalize more aspects of the system. As an example, if we want smoother system behavior with less stress to the actuators, we can introduce slew rate penalization:

$$V_{(\mathbf{x},\mathbf{u})} = \frac{1}{2}\sum_{i=1}^{M}(\mathbf{e}_{[i]}^T\mathbf{Q}\mathbf{e}_{[i]} + \mathbf{u}_{[i-1]}^T\mathbf{R}\mathbf{u}_{[i-1]}) + \frac{1}{2}\sum_{i=1}^{M-1}[(\mathbf{u}_{[i]}^T - \mathbf{u}_{[i-1]}^T)\mathbf{S}(\mathbf{u}_{[i]} - \mathbf{u}_{[i-1]})], \tag{8}$$

where $\mathbf{S}$ is the slew rate weighting matrix. This new term penalizes fast changes in control inputs, to ensure smoother system behavior, which however comes at a cost of worse reference tracking. In the case of this thesis, we need a fast system with aggressive reactions, so this term is omitted and only serves as an example of MPC possibilities.

3.3 Constraints

One of the main advantages of MPC is its inherent property of constraint handling. Constraints can be imposed not only to the control input, which is a feature that most controllers have but also to the output states, which means that solutions produced by MPC will lie in a certain region of state variables. Control input constraints represent the maximum (or maximum acceptable) signal that the actuator is capable of handling, this can represent for example the maximum current flowing through a motor. Output or state constraints can limit the states themselves, for example with multirotor UAVs, we can limit the maximum speed and acceleration of the UAV according to its capabilities and MPC will drive it only in this safe region. We can even limit the position of the UAV, which will create an artificial "cage" which the UAV can't leave. These constraints are represented as a set of linear inequalities:

$$\begin{aligned} \mathbf{u}_{min} \leq \mathbf{u}_{[t]} \leq \mathbf{u}_{max}, \\ \mathbf{x}_{min} \leq \mathbf{x}_{[t]} \leq \mathbf{x}_{max}. \end{aligned} \tag{9}$$

Constraints can be changed between MPC control iterations. For example, if we need to avoid an obstacle quickly, we can temporarily relax constraints on maximum acceleration and velocity to allow for better maneuverability.

3.4 Solving the quadratic programming problem

To solve the QP problem, we can either develop a custom solver or use a third party solution. CVXGEN is a software tool developed by Jacob Mattingley and Stephen Boyd [22], which can generate a custom solver based on a high-level description of a QP-representable convex optimization problem. The generated solver is specifically tailored to the described problem, it is not a general solver. The generated code is an ANSI C program, which does not require any additional libraries. It is almost branch free and has a predictable run-time behavior. All these attributes make it an ideal solution for onboard real-time applications. For this reason, it is used for example by SpaceX for the landing of first stages of their Falcon 9 rockets [5]. CVXGEN works best for small and medium-sized problems, up to approximately 4000 total coefficients in the constraints and the objective function (non-zero Karush–Kuhn–Tucker (KKT) matrix entries).

CVXGEN uses a custom problem specification language, which ensures validity and convexity of described optimization problems. First, dimensions of the problem are specified:

```
dimensions
  m = 1   # inputs
  n = 3   # states
  T = 40  # length of prediction horizon
end
```

Then we specify the parameters, which stay constant during the optimization process:

```
parameters
  A (n,n)  {1,1 2,2 3,3 1,2 2,3}  # dynamics matrix
  B (n,m)  {3,1}   # input matrix
  Q (n,n) psd    # state cost
  R psd    # input cost
  x[0] (n) # initial state
  x_max_2 nonnegative
  x_max_3 nonnegative   # state constraints
  x_r[t] (n), t=1..T    # reference
end
```

In vectors and matrices, we can declare only certain elements to be non zero, which is represented by the numbers in {} brackets. This helps the generator to reduce the complexity of the problem. We can also define matrix properties (symmetric, positive semidefinite (psd), etc.). After the parameters, we specify variables which can change during the optimization process:

```
variables
  x[t] (n), t=1..T  # states
  u[t] (m), t=0..T  # inputs
end
```

Then we specify our objective function, which can be either minimized or maximized. To represent our objective function described in (7), we define the following expression:

```
minimize
  sum[t=1..T](quad(x[t]-x_r[t], Q)) + sum[t=0..T](quad(u[t], R))
```

Finally, we can introduce constraints to inputs, outputs or the dynamics of the system. For example, we can define these constraints:

```
subject to
  x[t+1] == A*x[t] + B*u[t], t=0..T-1 # dynamics constraints
  abs(x[t][2]) <= x_max_2, t=1..T
  abs(x[t][3]) <= x_max_3, t=1..T  #state constraints

end
```

The problem is defined through an online interface at https://cvxgen.com/. A custom solver is then generated in a matter of minutes, depending on the complexity of the problem.

Performance of the solver generated by CVXGEN was compared to other solvers in [9], by generating 500 random QP problems from a constrained MPC controller. Generated problems were similar in size to the problem in this thesis. The results are shown in table 3.

Solver	Average Time [ms]
QPC qpas	0.341
CVXGEN	0.463
QPC qpip	0.660
CLP	0.798
CPLEX	3.123
quadprog(MATLAB)	5.348
OOQP	6.163
IPOPT	8.665
SCIP	32.175

Table 3: Performance comparison of different QP solvers, from [9].

We can conclude from the results that performance of the solver generated by CVXGEN is very good, it is an order of magnitude faster than MATLAB's quadprog.

3.5 MPC Implementation

The pipeline for UAV control used at the Multi-robot Systems group has a three-layer structure. At the lowest level is an embedded attitude controller, which maintains desired pitch, roll, and yaw $(\phi_d, \theta_d, \psi_d)$ as well as total thrust (T_d). This controller is part of the PixHawk flight controller, and it directly outputs control signals which set the speeds of the motors.

One layer above is a non-linear SO(3) state feedback controller, which was developed at University of Pennsylvania, based on the work described in [23] and [19]. The following model is used to represent the UAV:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v}, \\ m\dot{\mathbf{v}} &= f\mathbf{R}\mathbf{e}_z + mg\mathbf{e}_z, \\ \dot{\mathbf{R}} &= \mathbf{R}\hat{\mathbf{\Omega}}, \\ \mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} &= \mathbf{M}, \end{aligned} \tag{10}$$

where $\mathbf{R}(\phi, \theta, \psi)$ represents attitude, $\mathbf{r}(x, y, z)$ is the position, g is the force of gravity, f is the total thrust force of the propellers, m is the mass of the UAV, $\mathbf{\Omega}(p, q, r)$ represents body-axis angular rates, hat map $\hat{\cdot} : \mathbb{R}^3 \to SO(3)$ is defined by the condition $\hat{x}y = x \times y$ for all $x, y \in \mathbb{R}^3$, $\mathbf{J}$ is the inertia matrix and $\mathbf{M}$ is the total moment exerted by the propellers on the UAV. This controller takes as an input the desired state of the UAV $\mathbf{x}_d(x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z})$ and outputs desired attitude $(\phi_d, \theta_d, \psi_d)$ and total thrust (T_d) for the embedded controller.

The MPC forms a third layer of the control pipeline and works in this case as a tracker. The outputs of the MPC are not directly used for UAV control. The non-linear SO(3) controller is much better suited for this task, as it allows for more aggressive maneuvers, which are needed in the interception scenario. The outputs of the MPC tracker are instead used to drive a virtual model with transitional dynamics that represents the UAV. The states of this model are then sampled and used as a reference for the non-linear SO(3) controller. This approach combines the predictive nature of the MPC with agile control from the SO(3) controller into a package that keeps the advantages of both its subsystems. The whole control pipeline is shown in figure 4. The MPC tracker, simulated control loop and the SO(3) controller all run on 100 Hz.

3.5.1 Current MPC shortcomings

The MPC tracker described in this thesis is designed to replace the MPC tracker that is currently in use on the hexacopter platform of Multi-robot Systems group. This MPC tracker, that was developed from the work described in [8] has several shortcomings. To improve computational performance, the X Y and Z axes of the UAV model are considered as decoupled, and each is optimized independently. To reduce complexity even more, move blocking technique is used. This technique reduces the size of the problem by assuming that the control input at the end of the prediction horizon takes the form of a constant function. With this assumption, we can, for example, define that the last five control inputs in the prediction horizon will have the same value and optimize them together as one variable,

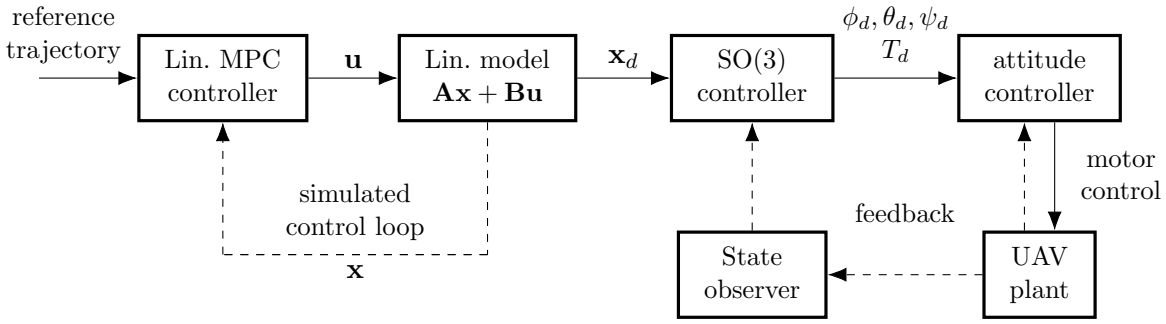


Figure 4: Diagram of the control pipeline

therefore reducing the size of the problem. Another problem are state constraints, which are not solved optimally by the current MPC and can sometimes be exceeded. The new MPC tracker addresses all of these issues.

Current MPC tracker solves the X, Y, and Z axes independently, which introduces a problem with state constraints. As an example, let us consider the case for horizontal velocity. Range of possible velocities is constrained by a box, since the constraints are simply defined as follows:

$$|v_x| \leq v_{hmax}, \qquad |v_y| \leq v_{hmax}, \tag{11}$$

where v_x, v_y are velocities in the X and Y directions and v_{hmax} is the maximum horizontal velocity constraint. With this set of constraints, the UAV can achieve maximum velocity in both X and Y directions simultaneously. If this happens, the UAV will be traveling at $\sqrt{2}v_{hmax}$. This situation is visualized in figure 5. Exceeding the maximum velocity by more than 40% while flying at the edge of the UAV capabilities, as is necessary for the interception scenario, is unacceptable.

With the current MPC implementation, this problem is solved by either using only a feasible reference trajectory, sampled at a reasonable velocity, or by setting the state constraints lower, which leaves headroom for the controller. In the interception scenario, the UAV has to fly at the edge of its capabilities, so reducing the state constraints is not desirable. The new MPC implementation presented in this thesis addresses the box constraint problem, which in turn allows infeasible trajectories to be set as a reference, without exceeding the maximal possible velocity of the UAV, which is beneficial in the interception scenario, as it facilitates trajectory planning.

3.5.2 MPC state constraints

In an ideal case, again assuming the horizontal speed situation, the constraint should be defined as follows:

$$\sqrt{{v_x}^2 + {v_y}^2} \leq v_{hmax}, \tag{12}$$

thus forming a circle in the horizontal velocity space and limiting the velocity in any direction to exactly the same value. This is visualized in figure 5. However, this constraint is no longer

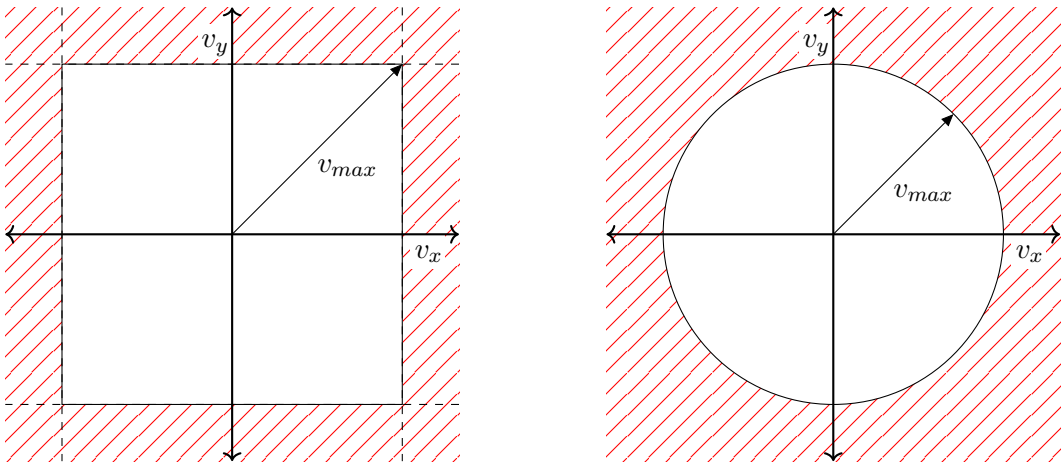


Figure 5: Constrains in horizontal velocity - current implementation (left), ideal constrains (right).

linearly constrained QP representable and therefore, suitable solver can't be generated by CVXGEN. To circumvent this problem, the ideal circular constraint can be approximated by a set of linear constraints, as follows:

$$|v_x| \leq v_{hmax}, \qquad |v_y| \leq v_{hmax}, \qquad \left|\frac{\sqrt{2}v_x}{2} + \frac{\sqrt{2}v_y}{2}\right| \leq v_{hmax}, \qquad \left|\frac{\sqrt{2}v_x}{2} - \frac{\sqrt{2}v_y}{2}\right| \leq v_{hmax}. \tag{13}$$

Let us define this set of constraints as the first order approximation. With this approximation, the maximal possible velocity is only 8.2% higher than v_{hmax}, much less than the original 41.4%. We can continue to add more linear constraints in a similar fashion, to better approximate the circular constraint. With second order approximation:

$$\begin{aligned} &|v_x| \leq v_{hmax}, \qquad \left|\frac{\sqrt{3}v_x}{2} + \frac{v_y}{2}\right| \leq v_{hmax}, \qquad \left|\frac{\sqrt{3}v_x}{2} - \frac{v_y}{2}\right| \leq v_{hmax}, \\ &|v_y| \leq v_{hmax}, \qquad \left|\frac{\sqrt{3}v_y}{2} + \frac{v_x}{2}\right| \leq v_{hmax}, \qquad \left|\frac{\sqrt{3}v_y}{2} - \frac{v_x}{2}\right| \leq v_{hmax}, \end{aligned} \tag{14}$$

the maximal possible velocity is only 3.5% higher than v_{hmax}, with third order approximation, it is only 1.9% higher. First and second order approximations are visualized in figure 6. A drawback of this approach is that it increases the size of the original problem with each added linear constraint, and therefore a reasonable compromise between approximation accuracy and computational complexity has to be made. In case of this thesis, the second-order approximation was chosen for the case of horizontal velocity and first-order approximation for the case of horizontal acceleration.

The model used for the horizontal situation is a linear model of a point mass in 2

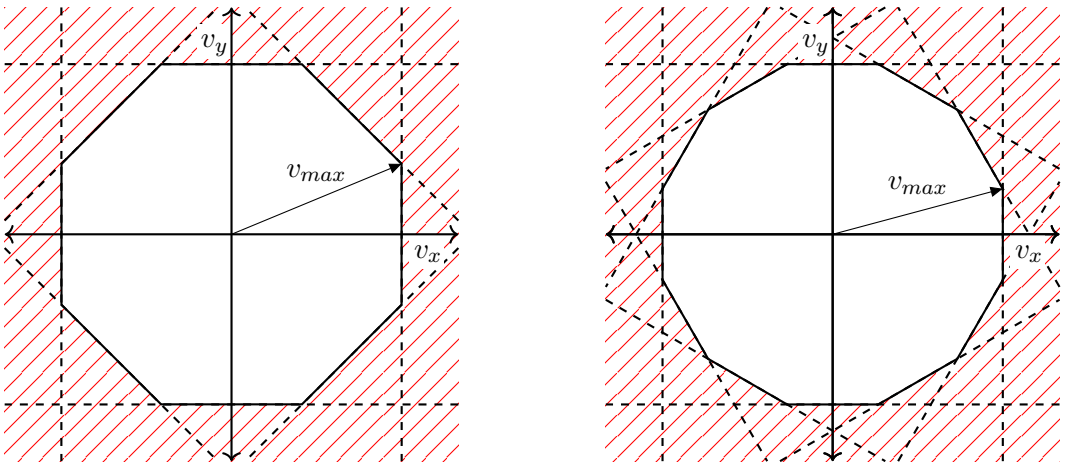


Figure 6: Constrains in horizontal velocity - first order circle linear approximation (left), second order circle linear approximation (right).

dimensions. It is written in matrix form as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \Delta t \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{15}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix, $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the input vector. The input vector in this case represents jerk, or $\dddot{x}$ and $\dddot{y}$. Constraints can be imposed on the input to limit the rate of change in acceleration.

3.5.3 Variable lateral constraints based on Z axis dynamics

So far, only X and Y axes were considered, but movements in the Z axis also significantly affect the lateral dynamics. Especially while climbing, the UAV has to use higher thrust, which means that there is much less thrust in reserve to sustain higher horizontal velocities. The Z axis could be incorporated into the solver for X and Y axes with all the appropriate constraints, but the complexity of such problem would be too high to be handled by solvers generated by CVXGEN. Instead, separate solver just for the Z axis is run before the XY solver. The Z axis solver will generate a prediction of the UAV states in the prediction horizon. Based on the predicted velocities and accelerations in the Z axis, we can limit velocities and accelerations in the X and Y axes, because we can set different state constraints for each step of the prediction horizon. By doing this, climbing in the Z axis will take priority and limit horizontal velocity, according to the capabilities of the UAV.

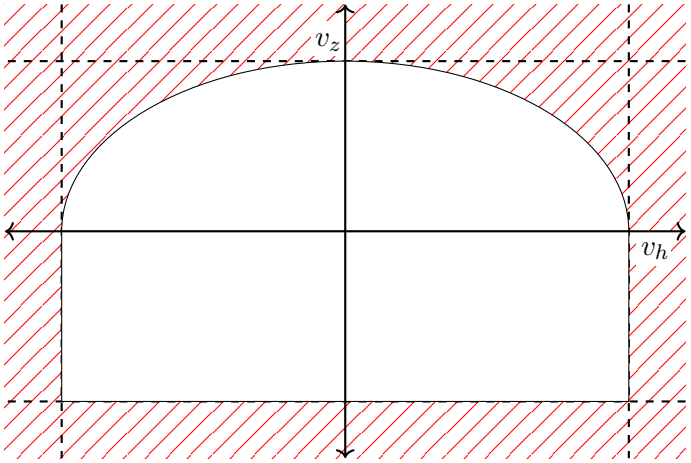


Figure 7: Constrains in horizontal velocity (v_h) with regard to previously calculated velocity in the Z direction (v_z).

Since we are setting these constraints outside of the optimization process, we do not need to approximate with linear functions. Assuming v_{hmax} is the maximal velocity in the horizontal direction, v_{zmax} is the maximal velocity in the Z direction, $v_{z_{[t]}}$ is estimated velocity in the Z direction at a time sample t, the horizontal velocity constraint $v_{hmax_{[t]}}$ at a time sample t is calculated as:

$$v_{hmax_{[t]}} = v_{hmax}\sqrt{1 - \left(\frac{v_{z_{[t]}}}{v_{zmax}}\right)^2}. \tag{16}$$

This constraint is only applied when $v_{z_{[t]}}$ is positive (the UAV is climbing), if the UAV is descending, the horizontal velocity constraint is kept at maximum. The range of possible velocities in the Z and horizontal directions is shown in figure 7. Acceleration in the horizontal plane is not intrinsically constrained to still allow for necessary changes in velocity while climbing, but positive acceleration in the Z axis will limit velocity in the horizontal plane as:

$$v_{hmax_{[t]}} = v_{hmax}\sqrt{1 - \left(\frac{a_{z_{[t]}}}{a_{zmax}}\right)^2}. \tag{17}$$

The model used for the Z axis movement follows the same principle as the model for the horizontal situation. It is again a linear model a point mass in 1 dimension. It is written in matrix form as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} z \\ \dot{z} \\ \ddot{z} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \end{bmatrix}, \tag{18}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix, $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the input vector, which represents jerk ($\dddot{z}$) and can be limited with an input constraint.

3.5.4 MPC yaw tracker

Last part of the puzzle which is needed for successful interception is control of the UAV yaw angle. This is only necessary if the UAV is carrying a tracking camera or other devices, which has to be pointed at the target. Otherwise, no change in yaw is necessary, as the UAV is symmetric. Current implementation on the MRS platform uses a PD (proportional-derivative) yaw tracker, which is not ideal for the case of interception. Since an estimate of the future trajectory of the intercepting UAV is known, thanks to the MPC tracker and estimate of the future trajectory of the target UAV is also known thanks to the estimator, it is possible to calculate desired yaw angle $\phi_{d_{[t]}}$ at which the interceptor will be pointed at the target UAV, so that the target is visible to the camera of the interceptor for each time sample t of the prediction horizon, as follows:

$$\phi_{d_{[t]}} = atan2(x_{t_{[t]}} - x_{e_{[t]}}, y_{t_{[t]}} - y_{e_{[t]}}), \tag{19}$$

where $x_{t_{[t]}}$ and $y_{t_{[t]}}$ are the estimated position of the target UAV at a time sample t, and $x_{e_{[t]}}$ and $y_{e_{[t]}}$ are the estimated position of the interceptor at a time sample t. The values of $\phi_{d_{[t]}}$ will then serve as a reference for an MPC yaw tracker. The yaw tracker has constraints on yaw rate and yaw acceleration, similar to the other MPC trackers, and it is used in the same way to generate a reference for the non-linear SO(3) controller. The MPC yaw tracker allows for predictive tracking of the target UAV with an onboard camera, improves tracking quality and reduces the chance of losing track of the target UAV.

The model used for the yaw tracker is the same model used for the Z axis, only the state values, in this case, represent rotational movement instead of a translational movement. It is written in matrix form as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \phi \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} u_1 \end{bmatrix}, \tag{20}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix, $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the input vector, which represents jerk ($\dddot{\phi}$) and can be limited with an input constraint.

3.5.5 Advantages and comparison of the new implementation

With all the new constraints for horizontal and vertical velocity, the new implementation of the MPC tracker can safely operate with reference trajectories which are infeasible. This is very important for the interception scenario, as we can easily plan infeasible trajectories based on an estimate of the target position and trajectory and let the MPC tracker optimize the interceptor trajectory with respect to the UAV capabilities. For example, the simplest implementation of target interception would be just to set the estimated position of the target as the only reference for the entire prediction horizon of the MPC. The new implementation of the MPC tracker would then plan an optimal interception trajectory with regards to the objective function and state and input constraints while assuring that the interceptor will not exceed its maximal capabilities in velocity and acceleration.

Performance of the new MPC tracker was compared with the original implementation. It should be noted that the original MPC tracker was designed for smooth and safe flying, it is not suitable for tracking infeasible trajectories and flying at the edge of the UAV capabilities. This comparison was made in the Gazebo robotic simulator (which will be described later), but it would have yielded the same results if it ran on an actual UAV, as the software is the same. Figure 8 shows a response of the tracker outputs to a step reference in the X direction going from 0 m to 100 m. The original MPC implementation overshoots the maximum velocity and acceleration constraints. In the original implementation, the maximum velocity and acceleration were considered more as safety limits, not actual state constraints, which is one of the reasons why this tracker is not suitable for the interception scenario. Both the original and the new MPC trackers overshoot their setpoints, which is, in fact, correct behavior, given that the quadratic objective function penalizes larger deviations from the setpoint more.

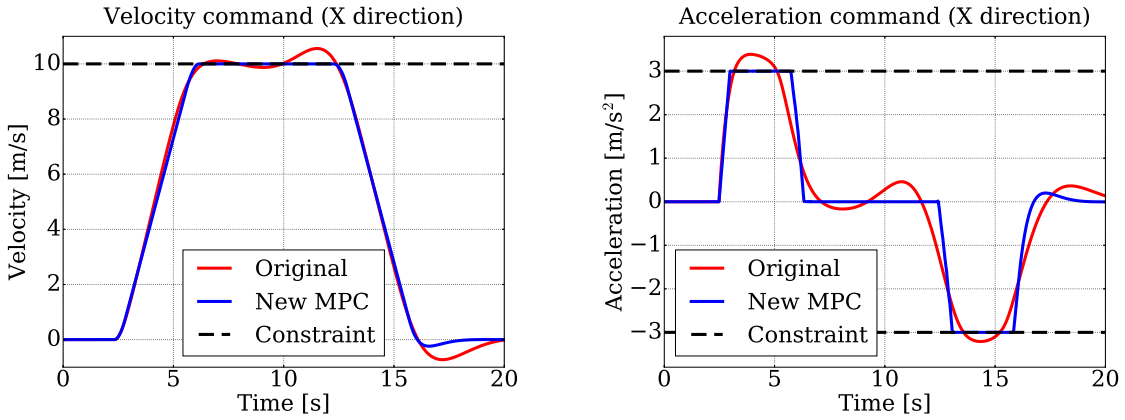


Figure 8: Comparison between the velocity (left) and acceleration (right) outputs of the new MPC tracker and the original MPC tracker, with state constraints $v_{hmax} = 10$ m/s, $a_{hmax} = 3.0$ m/s^2 and $\dot{a}_{hmax} = 6.0$ m/s^3. The reference is a step from 0 m to 100 m in the X position.

If this behavior is not desirable, **Q** matrix can be adjusted to eliminate the overshoots. In the case with overshoots, the **Q** matrix is set up to only penalize the positional error. To reduce or eliminate overshoots, another term can be added to the **Q** matrix, which will penalize the magnitude of the velocity. Values in the **R** matrix can also be increased to penalize the input. This will make the system behavior a little less aggressive, but the difference is very subtle. System responses with and without overshoots are visualized in figure 9.

Figure 10 shows trajectory tracking performance comparison. The reference trajectory is sampled at a constant velocity of 5 m/s and is infeasible, as it contains sharp turns. This demonstrates the predictive nature of the MPC, as the UAV reacts to the sharp turns in advance, with regards to the known dynamical limitations. The new MPC tracker shows more accurate reference trajectory tracking than the original MPC tracker.

The effect of circular approximation in the horizontal velocity constraint, which was discussed section 3.5.2 is demonstrated in figure 11. In this scenario, the reference was a step

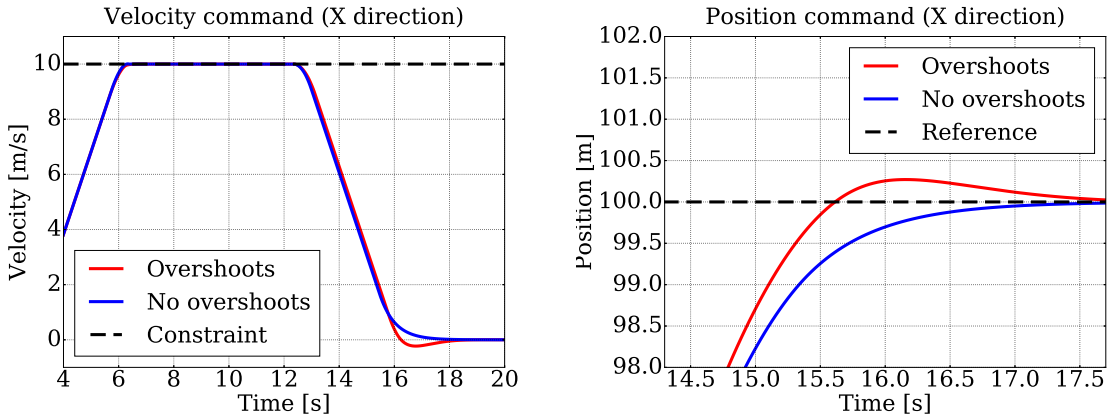


Figure 9: Comparison between the velocity (left) and position (right) overshoot behavior of the new MPC tracker while penalizing only position error (red) and with added penalization of the velocity (blue). All other parameters were the same as in figure 8.

in position from 0 to 100 in both the X and Y axes, with horizontal velocity constraint of 10 m/s. The original MPC tracker was not designed to operate under such conditions, which is demonstrated by the large overshoot, but the new MPC implementation can still operate safely even with this extremely infeasible reference.

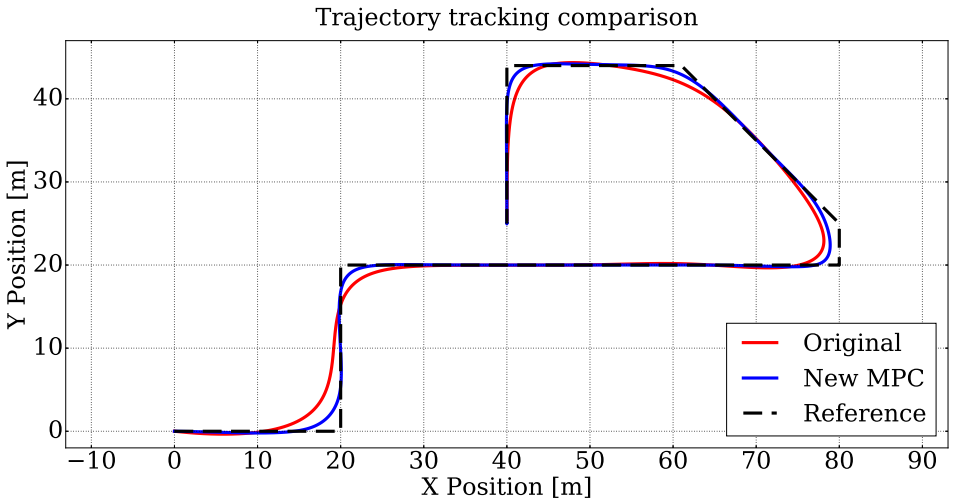


Figure 10: Comparison between the outputs of the new MPC tracker and the old MPC tracker while tracking a trajectory. State constraints are $v_{hmax} = 10$ m/s, $a_{hmax} = 3.0$ m/s^2 and $\dot{a}_{hmax} = 10.0$ m/s^3.

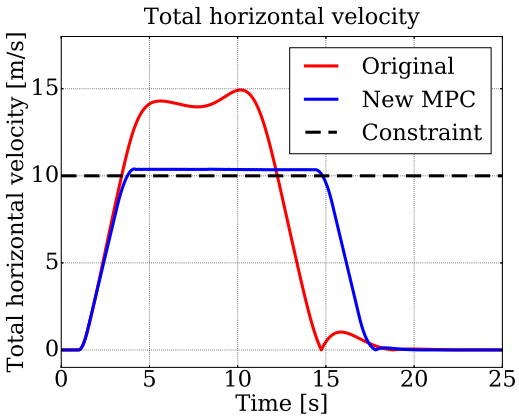


Figure 11: Demonstration of the circular approximation in the horizontal velocity constraint, shown in figures 5 and 6.

4 Interception

This chapter begins with a discussion of various methods of stopping an intruder UAV. Trajectory planning and interception algorithms based on different situations and configurations are then presented, and finally, possible solutions for target tracking are evaluated.

4.1 Target elimination

In this section, different approaches to stopping an intruder UAV will be discussed. It is assumed that the target and the interceptor are both multirotor UAVs, also called drones. The target can be neutralized in several different ways, three of which will be considered in this thesis.

4.1.1 Kinetic attack

The mode of attack which requires almost no additional equipment is a kinetic attack, during which the interceptor tries to disable the target by crashing into it at high velocity. The interceptor can be equipped with protective elements which can cover the delicate electronics and shield the propellers and motors, but it would be still probable that the interceptor itself could get damaged during this scenario. If the attack is successful, the target, and in some cases, even the interceptor, will fall from the air and might damage property on injure somebody on the ground.

In terms of control, localization, and planning, this mode of attack is the most difficult to implement, as it requires a direct hit of a relatively small target with a relatively small interceptor. The interceptor needs to be able to perform agile maneuvers and preferably be able to fly faster than the target. This mode of attack can be used as a backup if other methods fail and the neutralization of the target is critical, the kinetic attack can be deployed as a last resort.

4.1.2 Passive net

In this mode of attack, a large passive net is mounted to the bottom of an interceptor. The interceptor then flies over the target UAV, which then tangles into the net with its propellers, legs or arms and is captured. This scenario is easier in terms of control and localization, as the net can cover a much larger area.

In this case, the interceptor must be able to carry a much higher payload, not only the additional large net but also the weight of the captured target. The large net also introduces a significant amount of aerodynamic drag, and can be susceptible to wind and will negatively affect the maneuverability of the interceptor. There is also a risk that the net can self-tangle, or interfere with the propellers of the interceptor, but these risks can be minimized by good design and construction of the net. Multirotor UAV with a passive net is shown in figure 12,

in use by the Tokyo Metropolitan Police Department.

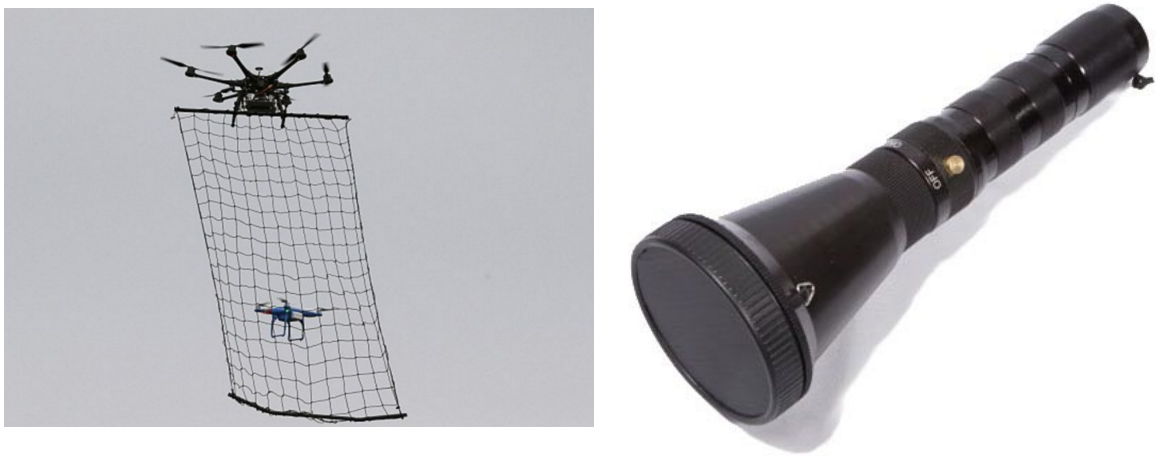


Figure 12: Devices for capturing target UAV. A passive net (left) in use by the Tokyo Metropolitan Police Department and a commercially available CO_2 powered net launcher (right).

4.1.3 Net/projectile launcher

The last mode of attack discussed in this section is a use of a net or projectile launcher. Commercially available net launchers (an example is shown in figure 12) are usually powered by a compressed CO_2 cartridge, which is used to launch a large (usually 3×3 meters) net with attached weights in the corners. These net launchers are designed for small animal capture, but their construction is also suitable for drone capture. The effective range is up to 10 meters, with an initial velocity of the net of 10 m/s.

In this mode of attack, the interceptor must be able to carry the net launcher itself, which weighs approximately 1 kg, considering the commercially available models. The launched net can remain attached to the launcher, allowing the interceptor to carry the captured target away, for which the interceptor again needs sufficient payload capacity. The net can also be detached after launching, leaving the target falling to the ground. A small parachute can be attached to the net, reducing the velocity of the falling captured target. The net launcher is much more compact than the passive net and has much less aerodynamic drag. The disadvantage of this approach is the fact that the net launcher has only one shot, after which the launcher needs to be reloaded manually. Instead of the commercially available solution, a custom net launcher could be developed, with the main goal of reducing the weight of the launcher to make it more suitable for UAVs.

The net launcher idea could be simplified, as the interceptor could just release material like long strings, which would tangle into the propellers of the target, bringing it down. The release could be powered by compressed gas, or the material could be simply released and blown down on the target by the air stream coming from the propellers of the interceptor. In

this case, the target would again hit the ground and may damage property or cause injuries.

4.2 Trajectory planning

In this section, various methods of trajectory planning for the interceptor will be discussed. These methods will later be combined to produce a robust interception trajectory planner. At first, the kinetic attack scenario, described in section 4.1.1, will be considered, as it is the most challenging in terms of control and trajectory planning. Trajectory planner for kinetic attack can be later relatively easily modified for the other scenarios, and it can also be used as a backup if other approaches fail. It is also assumed for now that the measurement of the target position is available at all times to the interceptor.

4.2.1 Stationary target

In the simplest scenario, the target is hovering and staying in the same position. This behavior is not uncommon with operators of camera-equipped drones who are surveying an area. The estimator, described in section 2, produces an estimate of the position of the target

$$\mathbf{x}_e = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix}, \tag{21}$$

which is used for interceptor trajectory planning. First, yaw angle ϕ_t at which the target is "seen" by the interceptor is calculated as follows:

$$\phi_t = atan2(y_e - y, x_e - x), \tag{22}$$

where x and y is the position of the interceptor. The interception trajectory can be then planned as a straight line from the position of the interceptor through the position of the target. Let us call this the direct attack trajectory. The process for direct attack trajectory planning is shown in algorithm 1, and visualisation of the planned trajectory is shown in figure 13.

The trajectory planning algorithm can run at a high frequency, up to the frequency of the MPC loop itself, to keep the planned trajectory up to date with the latest target position estimate. Notice that the Z coordinate of every sample of the planned trajectory is the same as the current estimated Z coordinate of the target, to force the Interceptor to fly at the same altitude level. This not only forces the interceptor to prioritize changes in altitude, which are more energy demanding than changes in X and Y, it will also keep the target in the FOV of a potential onboard tracking camera. The produced trajectory will probably be infeasible, not only because of the discontinuity in the Z coordinate but also because the current horizontal velocity vector of the interceptor may not be in line with the planned trajectory. But since the new implementation of the MPC tracker can safely handle infeasible trajectories, this is

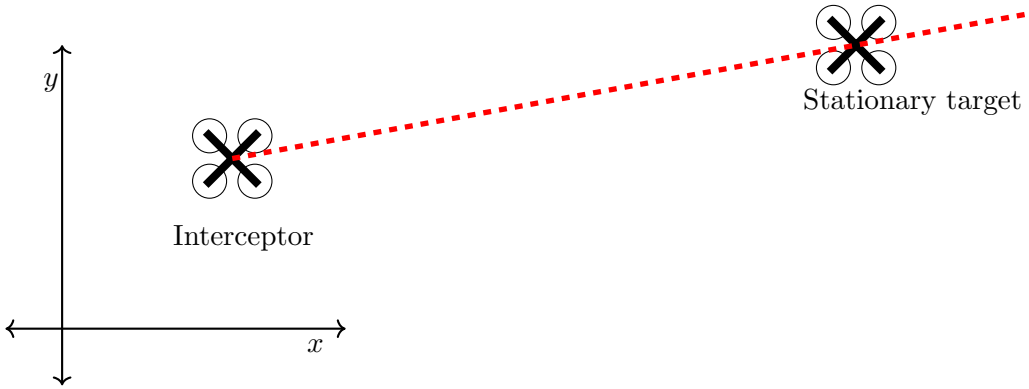


Figure 13: Direct attack trajectory planning in case of a stationary target.

not a problem, and the MPC tracker will plan an optimal trajectory given the dynamical limitations of the interceptor and the objective function.

input : $x_{tgt}, y_{tgt}, z_{tgt}$ - position of the target
x_{int}, y_{int} - position of the interceptor
v_{int} - horizontal velocity of the interceptor
v_{max}, a_{max} - maximum velocity and acceleration of the interceptor
ϕ_t - yaw angle to the target
dt - trajectory sample time (equivalent to MPC horizon sample time)
output: $traj$ - trajectory for the interceptor

```
1  begin
2      x ← x_int; y ← y_int; z ← z_tgt
3      i ← 0
4      vel ← v_int
5      while i < mpc_horizon_length do
6          vel = vel + a_max * dt
7          if vel > v_max then
8              vel ← v_max
9          end
10         x ← x + vel * cos(φ_t)
11         y ← y + vel * sin(φ_t)
12         traj.append(x, y, z)
13     end
14 end
```

Algorithm 1: Direct attack trajectory planning.

If the interceptor misses the target during the first attack run, there is no need to change anything in the planning algorithm, as it will simply plan the new trajectory from the new position of the interceptor. The interceptor will get carried away by its residual velocity from the previous attack run. This will allow it to start the next attack run from a sufficient distance, to pick up enough horizontal velocity to eliminate the target.

4.2.2 Dynamic target

A similar approach in trajectory planning can be used to intercept moving targets, but the algorithm has to be modified to plan interception trajectories not through the current position of the target, but through an estimated future target position. The future trajectory of the target can be estimated by taking the current estimated state of the target, produced by the estimator, which was described in chapter 2 and applying the state transition matrix **A**, shown in equation 1, to it. We can obtain estimates of the future states as follows:

$$\mathbf{x}_{e_{[t+1]}} = \mathbf{A}\mathbf{x}_{e_{[t]}}, \tag{23}$$

where $\mathbf{x}_{e_{[t]}}$ is the estimated state of the target at a time sample t. The time interval Δt in the **A** matrix can be changed, to produce estimates at different sampling frequencies. By applying the state transition matrix **A** over and over, future states can be estimated and combined into an estimated future trajectory of the target. With this estimate, an interception point can be calculated given the current position of the interceptor and its known dynamic limitations. The process for calculating the estimated point of interception is shown in algorithm 2. Once the estimated interception point is calculated, it can be used to plan a straight trajectory from the current position of the interceptor through the estimated point of interception. To plan this trajectory, algorithm 1 is used, only the estimated position of the target is substituted with the estimated point of interception. The resulting interception trajectory for dynamic target is visualized in figure 14.

4.2.3 Target following

This approach will not produce an intercepting trajectory, but it will plan a trajectory for the interceptor to follow the flying target. The interceptor can then switch to a direct attack, after reaching a position behind the target and matching velocities. Attacking from behind of the target can be very beneficial, as the target might be equipped with an onboard camera, which is used by the operator for video piloting (also called FPV - first person view). This camera is usually pointed in the direction of the flight, and it will therefore not capture the interceptor approaching from behind. If the interceptor is using an onboard camera to detect the target, the position of the target in the camera picture can be measured with a relatively good accuracy, however the distance between the interceptor and the target is much harder to determine, mostly because the actual size of the target is not know, therefore it cannot be compared to the size of the target in the camera picture do determine the distance. This means that the vector from to the interceptor to the target can be measured with better accuracy than the distance to the target. Attacking from behind of the target will reduce the influence of the error in the distance measurement, as the planned interception trajectory is close to being in line with the vector from the interceptor to the target and therefore reducing the influence of the error. This is visualized in figure 15. In a similar fashion, error in the estimated velocity of the target can also be partially compensated for by attacking from behind.

To plan a target following trajectory, the past and the estimated future trajectory of the target can be used. The reference trajectory for the interceptor can be then produced by

input : $x_{tgt[t]}, y_{tgt[t]}, z_{tgt[t]}$ - future estimated positions of the target at time samples t
x_{int}, y_{int} - position of the interceptor
v_{int} - horizontal velocity of the interceptor
v_{max}, a_{max} - maximum velocity and acceleration of the interceptor
dt - time between samples of the estimated target trajectory
output: n - number of the estimation sample closest to the interception point

begin
  $x \leftarrow x_{int}$; $y \leftarrow y_{int}$; $z \leftarrow z_{tgt}$
  $i \leftarrow 0$
  $vel \leftarrow v_{int}$
  $d_{target} \leftarrow distance2D(x, y, x_{tgt[0]}, y_{tgt[0]})$
  $d_{flown} \leftarrow 0$
  while $i < number\_of\_estimated\_samples$ **do**
    $i \leftarrow i + 1$
    $vel \leftarrow vel + a_{max} * dt$
    if $vel > v_{max}$ **then**
      $vel \leftarrow v_{max}$
    end
    $d_{flown} \leftarrow d_{flown} + vel * dt$
    $d_{target} \leftarrow distance2D(x, y, x_{tgt[i]}, y_{tgt[i]})$
    if $d_{flown} > d_{target}$ **then**
      $n \leftarrow i$
      $break$
    end
  end
end

Algorithm 2: Point of interception estimation in case of a dynamic target.

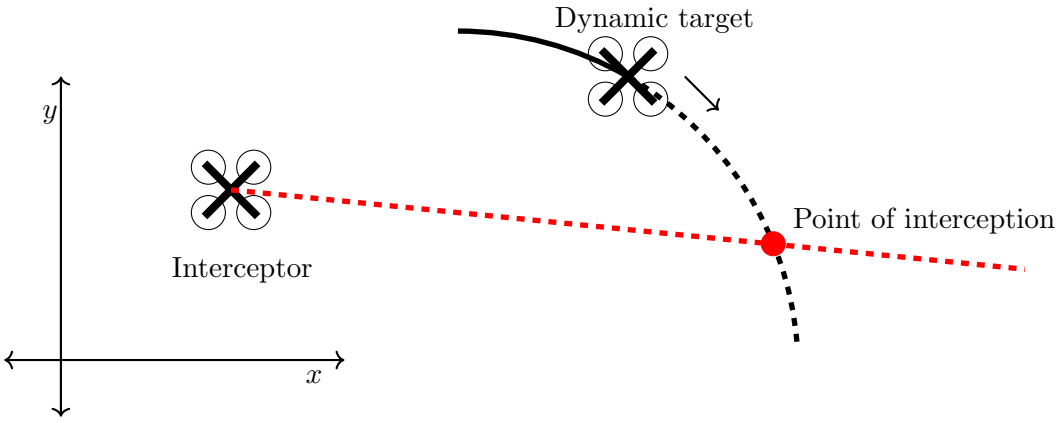


Figure 14: Direct attack trajectory planning in case of a dynamic target.

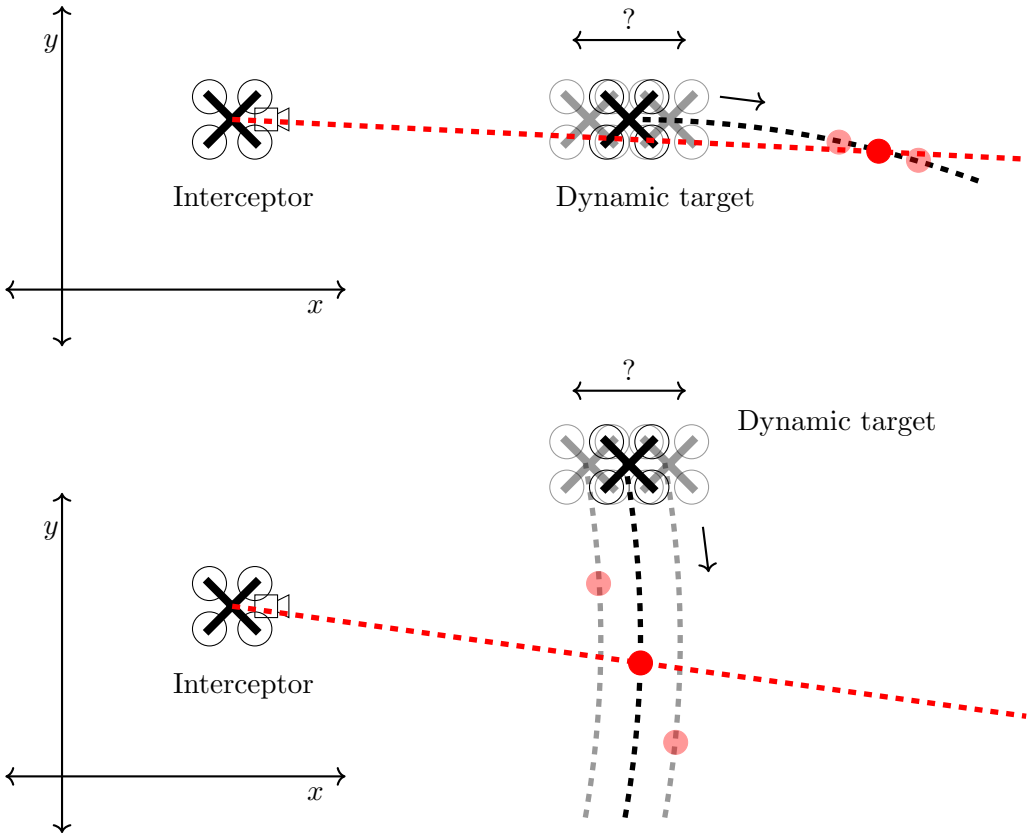


Figure 15: The uncertainty in distance measurement caused by the onboard tracking camera is compensated for by attacking from behind. The interceptor is able to fly much closer to the optimal point of interception, even when there is high uncertainty in the distance measurement.

combining these trajectories:

$$\mathbf{x}_r = \left[\mathbf{x}_{p[m]}, ..., \mathbf{x}_{p[-1]}, \mathbf{x}_{e[0]}, \mathbf{x}_{e[1]}, ..., \mathbf{x}_{e[n]}\right]^T, \tag{24}$$

where $\mathbf{x}_r$ is the reference trajectory for the interceptor, $\mathbf{x}_{p[t]}$ is a past position of the target at a time sample t, $\mathbf{x}_{e[0]}$ is the current estimated position of the target and $\mathbf{x}_{e[t]}$ is the estimated position of the target at a time sample t. Parameter n is the length of the estimated future trajectory and parameter m determines the time delay at which the interceptor follows the trajectory of the target. Following the target with a constant time delay means that the distance between the target and the interceptor is changing based on the velocity of the target. To follow at a constant distance, the parameter m has to be dynamically changed, which is shown in algorithm 3. The resulting trajectory is shown in figure 16. At first, this trajectory is infeasible, as there is a discontinuity between the initial position of the interceptor and the first sample of the reference trajectory, but this is not a problem for the new implementation of the MPC tracker.

input : $x_{tgt}, y_{tgt}, z_{tgt}$ - current estimated position of the target
$x_{p[t]}, y_{p[t]}, z_{p[t]}$ - past estimated positions of the target at time samples t
$dist$ - desired distance between the target and the interceptor
output: m - number of the first sample to be used as the interceptor reference

begin
  $i \leftarrow -1$
  $tmpdist \leftarrow dist3D(x_{tgt}, y_{tgt}, z_{tgt}, x_{p[i]}, y_{p[i]}, z_{p[i]})$
  while $tmpdist < dist$ **do**
    $i \leftarrow i - 1$
    $tmpdist = tmpdist + dist3D(x_{p[i+1]}, y_{p[i+1]}, z_{p[i+1]}, x_{p[i]}, y_{p[i]}, z_{p[i]})$
  end
  $m \leftarrow i$
end

Algorithm 3: Determining the first sample to be used as a reference for target following.

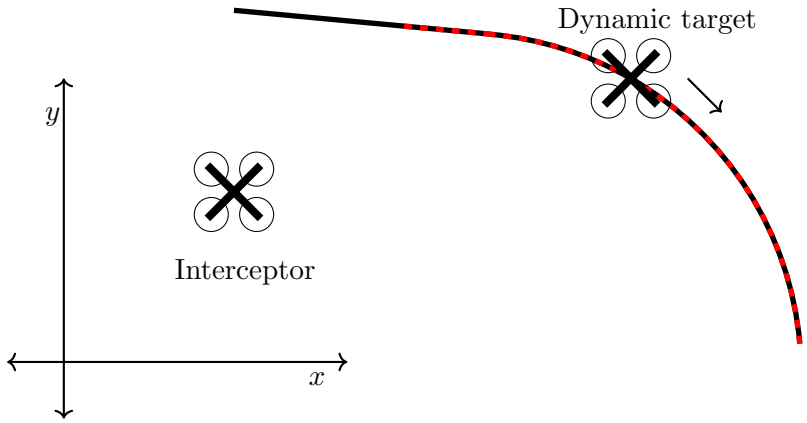


Figure 16: Reference trajectory for target following (red).

4.2.4 Head-on attack

Sometimes, the target is already flying towards the interceptor, which is often the case if the interceptor misses its attack from behind and ends up in front of the target. This fact can be used to plan a trajectory with a head-on collision. The process of planning is similar to the target following planning, but the trajectory is planned in reverse:

$$\mathbf{x}_r = \left[\mathbf{x}_{e[m]}, ..., \mathbf{x}_{e[1]}, \mathbf{x}_{e[0]}, \mathbf{x}_{p[-1]}, ..., \mathbf{x}_{p[m]}\right]^T, \tag{25}$$

where $\mathbf{x}_r$ is the reference trajectory for the interceptor, $\mathbf{x}_{p[t]}$ is a past position of the target at a time sample t, $\mathbf{x}_{e[0]}$ is the current estimated position of the target and $\mathbf{x}_{e[t]}$ is the estimated position of the target at a time sample t. Parameter n is the length of the estimated past trajectory and parameter m marks the sample of the future estimated target trajectory which is the closest to the current position of the interceptor. The trajectory of the interceptor can also be resampled at a particular velocity, as the estimated target velocity in reverse may not be ideal for the interceptor. The process of head-on collision trajectory planning is described

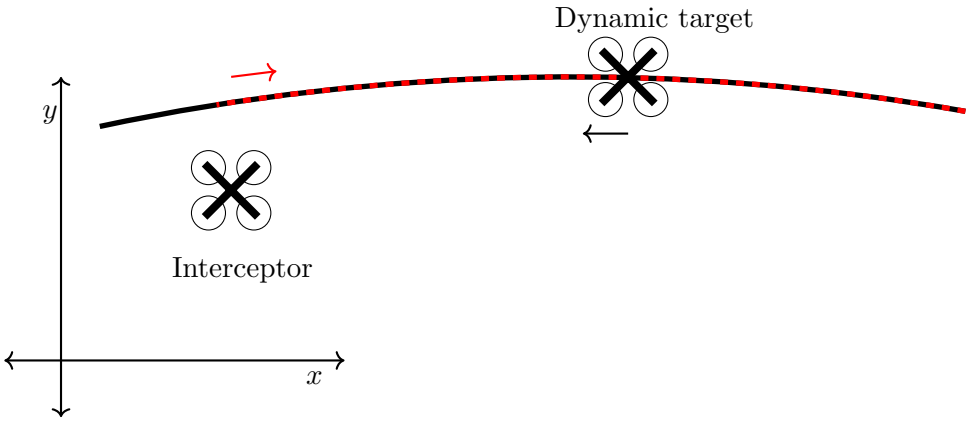


Figure 17: Head on attack trajectory (red).

in algorithm 4 and the resulting trajectory is visualized in figure 17.

A similar process could also be used for planning an interception trajectory from behind the target. But compared to the head-on scenario, there are some problems. As the target is flying away from the interceptor, the interception trajectory should be resampled at a high or even maximum possible velocity (according to the velocity constraints of the MPC tracker), to allow for the interceptor to catch up with the target. But if the target is maneuvering at a slower velocity, it can turn with a sharper turn radius than the fast flying interceptor. This can be resolved by resampling the interception trajectory dynamically, based on the turns of the target, but it is much easier and robust to just use the direct attack approach.

4.3 Interception algorithm

In the previous section, three main flight and attack modes were presented, the direct attack, target following, and the head-on attack. For a successful and robust interception, a supervising algorithm is used to switch between the different modes of attack. Firstly, the algorithm is formulated for a kinetic attack with continuous information on target position. This algorithm is shown in figure 18. The flowchart contains green boxes, these are considered as states, and diamonds, which mark decisions with a positive or negative outcome. The white box is the initial state. In each iteration, the flowchart is evaluated until a state is reached. Depending on what state is reached, different actions are taken. In the next iteration, the flowchart is evaluated again, beginning from the last reached state.

Decision variable v_t represents the total velocity of the target. If this velocity is lower than a set threshold v_s, the target is considered as stationary. The decision *In front of target* signifies that the interceptor is in front of the target, and the target is flying towards the interceptor. This is determined by finding the sample from vector $\mathbf{x}_r$, described in equation 24, which is closest to the current position of the interceptor. This found sample can be either a past position of the target, the current position of the target, or future estimated position of the target. If it is the future estimated position, the interceptor is considered to be in front of the target.

input : $x_{e[t]}, y_{e[t]}, z_{e[t]}$ - future, past and present estimated positions of the target at a time sample t, where $t = 0$ is the current position of the target
n - (negative number) number of past estimated target trajectory samples
p - (positive number) number of future estimated target trajectory samples
des_vel - desired velocity at which the trajectory is resampled
dt - time between the samples of the estimated target positions
$x_{int}, y_{int}, z_{int}$ - current position of the interceptor
output: $traj$ - reference trajectory of the interceptor

```
1  begin
2      dist = dist3D(x_e[p], y_e[p], z_e[p], x_int, y_int, z_int)
3      for i = p − 1; i == 0; i = i − 1 do
4          tmpdist = dist3D(x_e[i], y_e[i], z_e[i], x_int, y_int, z_int)
5          if tmpdist < dist then
6              dist ← tmpdist
7              m = i
8          end
9      end
10     traj.append(x_e[m], y_e[m], z_e[m])
11     tmpdist ← 0
12     for i = m − 1; i == n + 1; i = i − 1 do
13         tmpdist = dist3D(x_e[i], y_e[i], z_e[i], x_e[i−1], y_e[i−1], z_e[i−1])
14         if tmpdist > dt ∗ des_vel then
15             traj.append(x_e[i], y_e[i], z_e[i])
16             tmpdist = tmpdist − dt ∗ des_vel
17         end
18     end
19 end
```

Algorithm 4: Head on collision trajectory planning.

The decision *At follow position* signifies that the interceptor has reached its following position behind the target and it is now following with matched velocity. This condition is met only if the interceptor is closer than a certain threshold to the desired following point ($\mathbf{x}_{p[m]}$, described in equation 24), and the velocity of the interceptor is matched with the current velocity of the target, in a certain threshold.

The direct attack is stopped after a certain time, which is represented by the decision *Out of time*. This is to prevent the interceptor from attempting successive direct attacks if the first one fails and instead switch to the head-on attack or target following. If the previous state was *Target follow*, then the attack time is calculated as the time needed for the interceptor to reach the point of interception plus two seconds. If *Direct attack* state was reached because the target is considered stationary, then the attack time is set to two seconds. The direct attack continues after these two seconds if the target is still considered stationary.

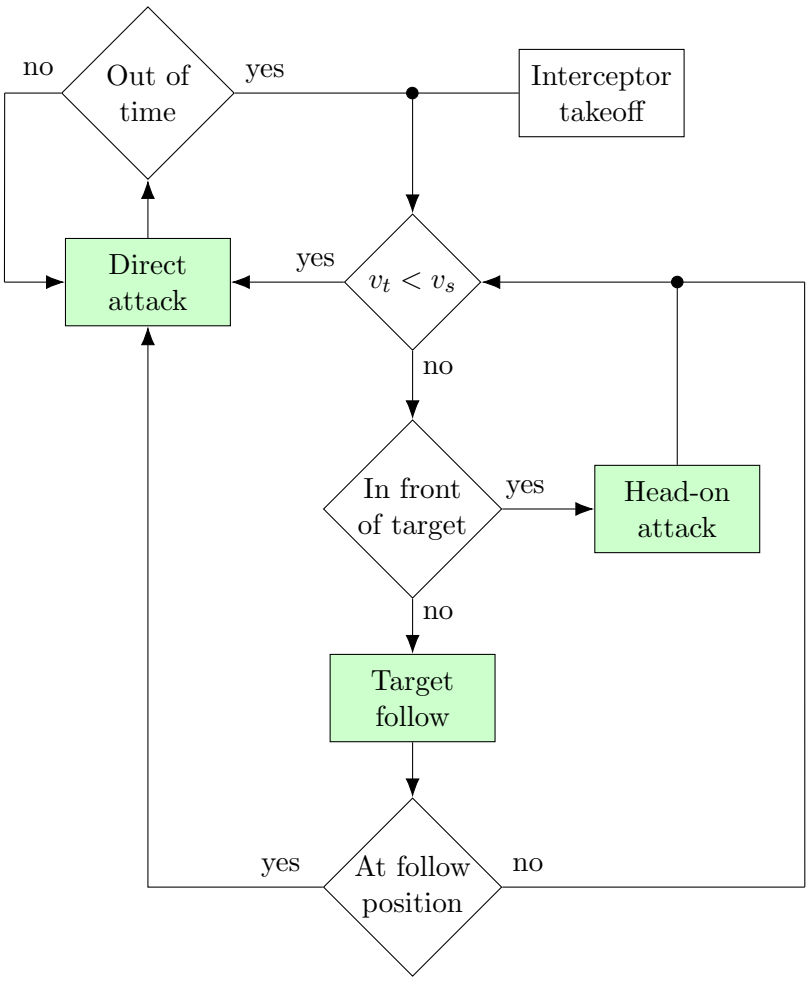


Figure 18: Flowchart of the kinetic attack interception algorithm.

4.3.1 Other approaches to target elimination

The above-described interception algorithm is designed for kinetic attack, but it can be modified for the passive net or the net launcher. If the interceptor is using the passive net, the only required change is to add an altitude offset to all the planned trajectories, depending on the height and mounting point of the net. The yaw angle of the interceptor has to be also considered, to ensure that the net will hit the target with its broad side. The yaw angle for each time sample t of the trajectory is calculated as follows:

$$\phi_{[t]} = atan2(y_{e[t]} - y_{i[t]}, x_{e[t]} - x_{i[t]}), \tag{26}$$

where $y_{e[t]}$ and $x_{e[t]}$ is the estimated position of the target at a time sample t and $x_{i[t]}$ and $y_{i[t]}$ is the planned position of the interceptor at a time sample t. If the interceptor is less than 1 meter from the estimated position of the target, the yaw angle is kept at the last calculated value, to prevent it from jumping 180° as the interceptor flies over the target.

If the interceptor is using the net launcher, the interception algorithm has to be significantly modified. The position of the following point ($\mathbf{x}_{p[m]}$, described in equation 24) is

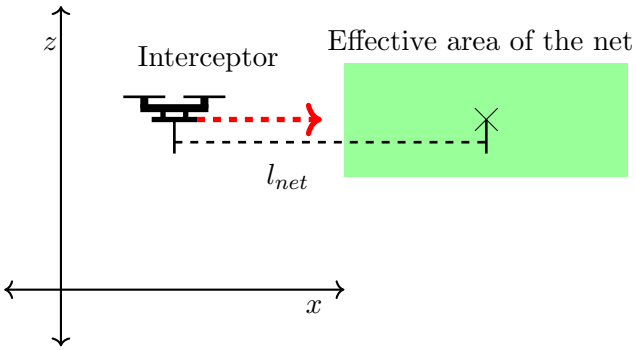


Figure 19: Effective space of the launched net (green) and ideal distance from the target l_{net}.

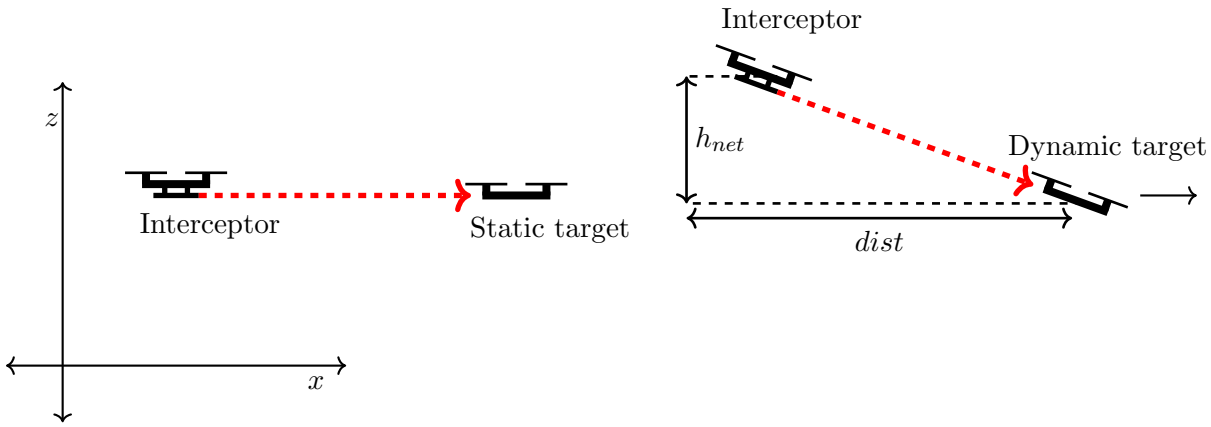


Figure 20: Interceptor equipped with a statically mounted net launcher has to compensate for different pitch angles θ at different velocities by changing the relative altitude to the target h_{net}.

adjusted so that the following point is an ideal point from which the net should be launched at the target. The net should be launched from a sufficient distance to allow the net to unravel itself to its full size. We can define a 3-dimensional space in front of the interceptor, where the net can effectively hit the target. For example, the typical net launcher can fire a 3×3 meter net at a distance of up to 10 meters, so the effective space can be defined as a cylinder with a diameter of 2 meters, spanning from three to seven meters in front of the interceptor. In this space, the net is fully unraveled, and it is still flying at high velocity, ensuring target hit. Therefore, if the target is inside of this defined effective space, the net should be launched. In the ideal case, the target should be in the center of this space, at an ideal distance l_{net} from the interceptor, which is shown in figure 19 Similarly, with the passive net scenario, the yaw angle of the interceptor has to be set to aim the launcher at the target, using the same equation 26. Additionally, if the launcher is mounted directly to the frame of the interceptor, the pitch angle θ has to be considered when aiming the net launcher, which is illustrated in figure 20. To compensate for the pitch angle θ of the interceptor, the following point $\mathbf{x}_{p[m]}$ has to be offset both in altitude (h_{net}) and in horizontal distance (parameter *dist* in algorithm 3),

to keep the 3D distance between the target and the interceptor close to the optimal l_{net}. The offsets are calculated as follows:

$$\begin{aligned} h_{net} &= cos(\theta)dist, \\ dist_c &= sin(\theta)dist, \end{aligned} \tag{27}$$

where $dist_c$ is the new desired horizontal following distance behind the target, used as an input in algorithm 3. The pitch angle of the interceptor is mainly influenced by three factors: interceptor's velocity, it's air resistance and wind. Pitch angle needed to fly at a certain velocity can be determined beforehand, by calculation or experimentation. Those values can be stored in a look-up table, and desired offsets h_{net} and $dist_c$ can be added to the reference trajectory in the planning phase, based on the desired velocity of the interceptor. This allows the MPC tracker to fly the trajectory with the offsets in mind from the start. During flight, the actual pitch angle of the interceptor will vary from the predetermined value, because of wind and other factors. This is compensated for by adding another correction on top of the predetermined one, based on the current actual pitch angle. The magnitude of this correction will be much smaller as most of the correcting is already accounted for by the predetermined correction.

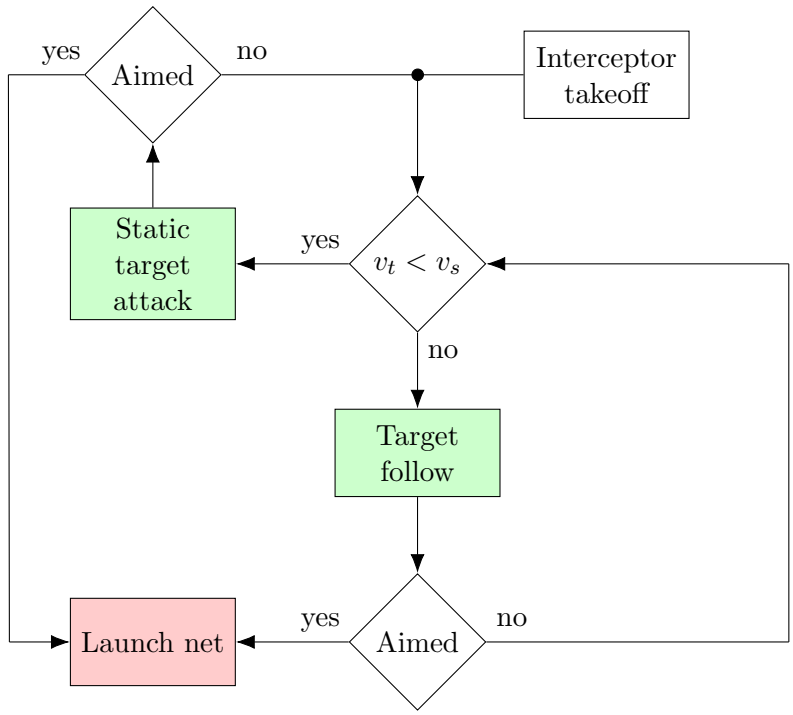


Figure 21: Flowchart of the net launcher interception algorithm.

In case of a static target, the interceptor will assume a position at the optimal distance l_{net} for launching, that is closest to the current position of the interceptor. Coordinates of this position x_a and y_a and the yaw angle θ of the interceptor are calculated as follows:

$$\begin{aligned} x_a &= x_e - cos(atan2(y_e - y_i, x_e - x_i))l_{net}, \\ y_a &= y_e - sin(atan2(y_e - y_i, x_e - x_i))l_{net}, \\ \theta &= atan2(y_e - y_a, x_e - x_a), \end{aligned} \tag{28}$$

where x_e and y_e are the estimated position of the target and x_i and y_i is the current position of the interceptor. The Z coordinate is set to be either the same as the Z position of the target or offset by a constant, depending on the offset of the net launcher mounting point. The algorithm for attack with a launcher is visualized in figure 21.

Decision variable v_t represents the total velocity of the target. If this velocity is lower than a set threshold v_s, the target is considered as stationary. The decision *Aimed* means that the target is inside of the effective space of the launched net. In this case, the head-on attack mode is not utilized. If the interceptor misses the target with the launched net, it can either abort the attack or switch to the kinetic attack mode.

4.4 Target tracking

So far, it was assumed that the information about the position of the target was available at all times. However, in the real world situations, target tracking is a very complex problem. Ground-based tracking station equipped with powerful cameras, radio receiver or radars can be used to detect and track the target. Such equipment is commercially available, as shown in figure 22, and it can cover a much larger area then sensors onboard of the interceptor, but the tracking information may not be precise enough to serve as the only mean of target tracking. This tracking station can provide a rough estimate of the target's position, and the interceptor can then search for the target in the specified area.

Another method that can be used for initial target localization is being developed at the Multi-robot Systems Group [7]. This method relies on a person, who can notice and identify an intruding drone. The person then simply points a smartphone at the target drone. The position and attitude of the phone are then determined with GPS and onboard inertial measurement unit, which defines a line in space, on which the target is located. The interceptor can then follow this line and search for the target with its onboard sensors. This approach relies on a person in the loop for the initial target detection, as humans are still better in this regard than computers.

As the interceptor is much closer to the target than the ground tracking station or the person, it can track the target more precisely. For target localization, the interceptor is equipped with some form of onboard tracking equipment. This equipment can consist of radars, lidars, microphones, and cameras. The visible light camera offers a great combination of low weight, low price, and availability of many different models. Drones can be detected in the images from the visible light camera by using deep learning algorithms, mainly convolutional neural networks (CNN). This approach was discussed and tested in [29], [3] and in [31], showing promising results. A similar system based on the CNN approach is being developed at the Multi-robot systems group [33]. However, it was not ready for integration at the time of writing this thesis, so it could not be used in the experiments.

Figure 22: Commercially available tracking station from Dedrone.

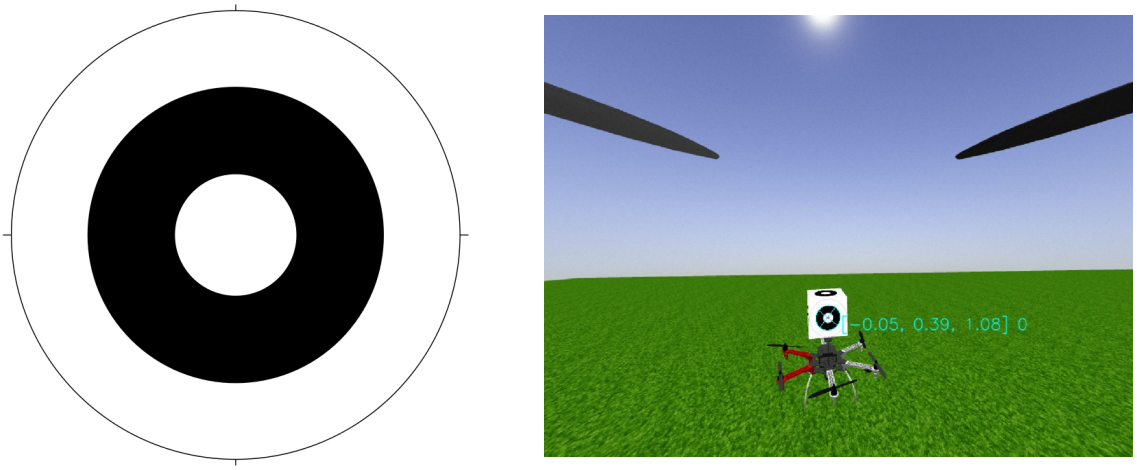


Figure 23: WhyCon marker (left). WhyCon detecting the marker in the Gazebo robotic simulator (right).

The goal of this thesis is not to develop a system for drone localization, but algorithms for drone interception. To simulate target interception with an onboard computer vision target localization system, a marker-based system is used as a substitute. This system is described in [16] and [17], is also called WhyCon, and it offers fast and precise marker detection while requiring only low computational power. The marker used by WhyCon (shown in figure 23) was added to the target in the Gazebo robotic simulator, to test interception strategies with visual target localization. WhyCon has similar disadvantages to other computer vision approaches, like limited range, limited field of vision, delays caused by the camera and image processing or camera lens distortions, and therefore it serves as a good analog to a computer vision target localization system based on CNNs.

The interception algorithm has to be modified when using any kind of visual target localization system like WhyCon or CNNs, to account for their properties. Most notably their limited range and field of view, which can be mitigated by using both more number of

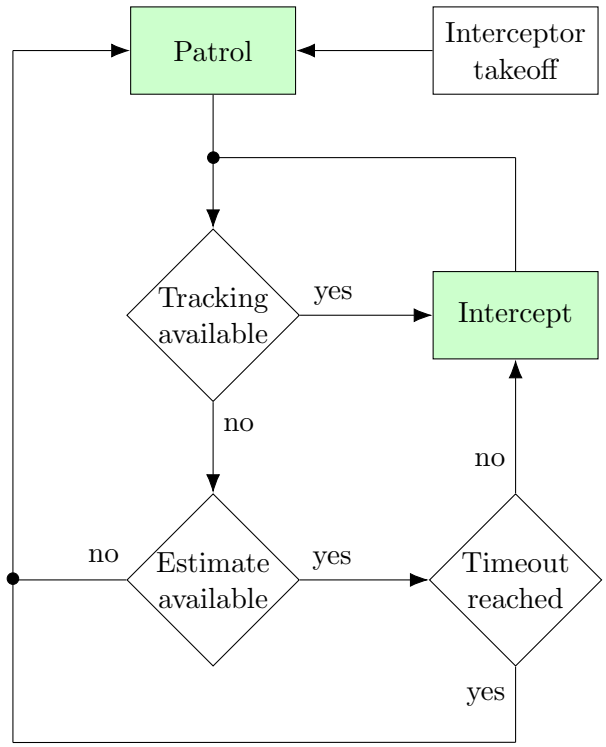


Figure 24: Flowchart of the target tracking algorithm.

cameras and more sophisticated cameras, but certain compromise will have to be made in regards to the available onboard computing power and payload of the interceptor. In case of this thesis, the interceptor will be only equipped with one statically mounted camera for the purposes of target tracking.

In the scenario with onboard visual target localization system, the interceptor can either obtain an initial estimate of the target's position, or it can stay in a selected position or guard path, and look for nearby targets. Once a target is detected with the onboard camera, the yaw angle reference of the interceptor is set in a way that the camera is keeping the target in the middle of the image frame, for best tracking results. The yaw angle reference is calculated as follows:

$$\theta_{[t]} = atan2(y_{e[t]} - y_{i[t]}, x_{e[t]} - x_{i[t]}), \tag{29}$$

where $\theta_{[t]}$ is the yaw angle reference at time sample t, $x_{e[t]}$ and $y_{e[t]}$ is the estimated position of the target at a time sample t and $x_{i[t]}$ and $y_{i[t]}$ is the X and Y position reference of the interceptor at time sample t. The task of keeping the target centered in the camera image is much more difficult in case of a moving and maneuvering target. If the target tracking is lost due to the target flying out of range or out of the field of view, the estimator can still produce new estimates of the target's position, as shown in figure 3. These estimates can be used to regain target tracking, but the uncertainty of the estimate rises quickly, which means that the estimate can be reliably used only for several seconds after the tracking is lost.

If an onboard visual target localization system is used, another algorithm, shown in figure 24, is used to deal with target tracking. The state *Patrol* can include many different in-

terceptor behaviors, the interceptor can, for example, hover on a defined position while slowly rotating around its Z axis, to scan the surrounding area with its onboard camera, or it can fly on a predefined patrol route, or if a rough initial estimate of the target's position is available, the interceptor can fly to this position and rotate around its Z axis to scan for the possible target. If the state *Intercept* is reached, the interceptor continues with desired interception algorithm, shown for example in figures 18 or 21. The decision *Tracking available* gives a positive result if the onboard camera is currently tracking the target. The decision *Timeout reached* gives a positive result if the available estimate is based on tracking information that is older than a certain time t_t, which signifies that the estimate is not accurate anymore.

5 Tools for verification

Work presented in this thesis was verified in two phases. Initial experiments and algorithm verification was done in Gazebo robotic simulator [24] and was followed by experiments with real hardware. In this chapter, the Gazebo simulator and the software platform used will be described, followed by a description of the Multi-robot Systems group's hardware platform (MRS platform), which was used for experimental verification.

5.1 Software platform

The main component of the used software platform is Robot Operating System (ROS, [27]), as it greatly facilitates system development. ROS is in fact not an operating system, but a framework and a set of tools designed to control robots. It enables to run and manage multiple processes in parallel as different nodes and provides a standardized communication link between them which works on the publisher-subscriber model. ROS also provides low-level device control, as many sensors have their official ROS packages, with appropriate drivers and nodes. These qualities make ROS a good link between hardware and software, while the node structure and standardized communication also allow for parallel software development.

5.1.1 Gazebo robotic simulator

ROS can also be easily connected with the Gazebo robotic simulator, which can be used together with firmware from the PixHawk autopilot for Simulation In The Loop (SITL). The Gazebo simulator provides a powerful physics engine, which can simulate real-world systems, as well as 3D visualization. The software running on the simulated UAV is the same software that runs on the real UAV, in the same ROS environment. All sensors used on the real UAV can be simulated as well and can have the same interface and characteristics as their real counterparts. This is very useful for evaluation and testing of new algorithms and software, and it allows for faster development and integration with real hardware. Visualization provided by the Gazebo simulator is shown in figure 25.

5.2 Hardware platform

The hardware platform used for experimental verification in this thesis is the multicopter platform developed by the Multi-robot Systems Group. It is designed for autonomous aerial experiments, and is highly modular, with predominant use of commercially available parts and sensors. This platform was initially developed for the MBZIRC 2017 robotic competition([21], [4]), and it now serves for research and experimental purposes ([30], [26], [35]). The base of the platform is a DJI F-550 hexacopter frame and 2312E motors. Six motors are used to provide additional payload capacity over the more common four motor quadcopters of similar size. PixHawk flight controller is used for low-level motor control, as well as a basic localization unit. This controller includes an internal inertial measurement unit, barometer and an external module with a magnetic compass and GNSS receiver. Based on the measured data, PixHawk

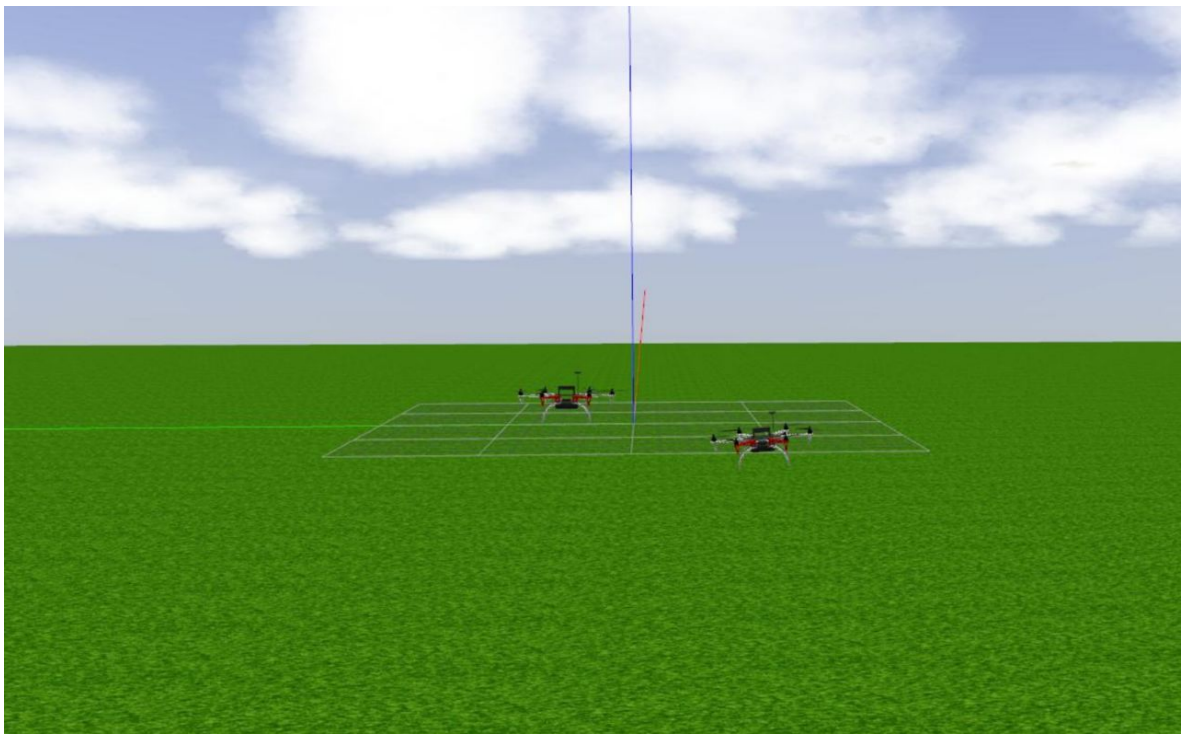


Figure 25: Two UAVs in the Gazebo robotic simulator.

uses an Extended Kalman Filter (EKF, [11]) to estimate states of the UAV, however additional sensors are used to increase the precision of this estimate.

5.2.1 Additional sensors

Garmin LIDAR-Lite v3 serves to determine the exact altitude over variable terrain features, by mounting it facing directly downwards on the UAV platform. This sensor provides measurements with frequency up to 500 Hz, at a range of 5 cm to 40 m with an accuracy of ± 2.5 cm. Since the sensor is mounted firmly, the tilt of the UAV introduces cosine error into the altitude measurement. However, this can be compensated for easily, as the attitude of the UAV is measured by the PixHawk flight controller.

To further increase positional accuracy, Tersus BX305 RTK (real-time kinematics) differential GNSS is used. This system requires a stationary GNSS receiver, called a base station, which is placed at a known location. The position of this location can be obtained by averaging regular GPS positions for a longer period of time. The base station then broadcasts information about its position and data about carrier phase measurements for all visible satellites to mobile rovers (the UAVs in our case). The mobile rovers can then use the information received from the base station to compensate for errors in the GNSS solution. Using these corrections, the rovers can obtain several different solutions, based on the number of visible satellites, their position in the sky and environmental variables like weather. Those are DGPS, RTK Float, and RTK Fix, in order of increasing accuracy. The positional accuracy

with the best solution, RTK Fix, is ±15 mm. Note that this is only a relative accuracy, as the whole coordinate system is offset by the error in the defined position of the base station versus its actual position. If this error is sufficiently reduced, the coordinates from the RTK will correspond to absolute coordinates. The BX-305 supports GPS, GLONASS, and Beidou satellite constellations and provides positional and velocity information on frequency up to 20 Hz. Measurements from the lidar and RTK GNSS are fused with the measurements from Pixhawk to provide accurate state estimates for the UAV.

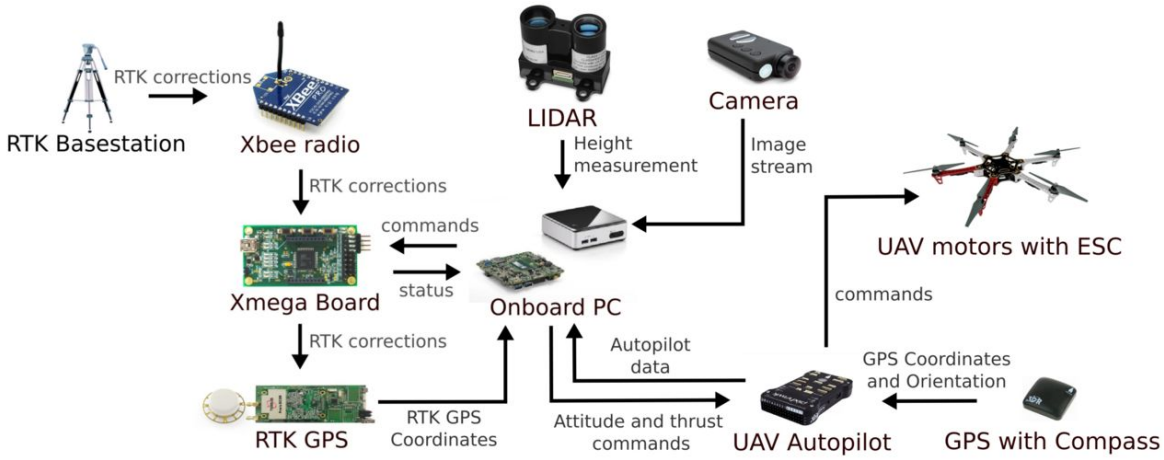


Figure 26: Architecture of the Multi-robot Systems group UAV platform.

Figure 27: Hardware platform used at the Multi-robot Systems group in MBZIRC configuration.

The hexacopters are equipped with an Intel NUCi7RYH, which serves as the main computer. With a fully featured Intel Core i7 dual-core x86-64 processor, it provides enough computing performance for difficult applications like computer vision or complex planning, to run onboard the platform. A custom board was designed by the author of this thesis to provide a low-level interface between the main Intel NUC computer and other devices

and sensors. This board uses an 8-bit ATxmega 128A4U microcontroller, which is running FreeRTOS, a real-time operating system, which facilitates running of multiple tasks in parallel. To communicate with the high-level computer, FT232RL serial to USB converter is used.

The board is fitted with a socket to accommodate an XBee Pro radio module, which provides wireless communication between the RTK base station and the RTK receiver on the UAV. The XBee modules offer a reliable low-speed communication link and can also be used to relay small packets of information between different UAVs. The board also features two additional UART interfaces with selectable voltage levels, one of which is used to relay RTK corrections directly to the RTK receiver board, an I^2C bus, which is used for communication with the Garmin LIDAR, Two PWM pins with selectable voltage levels and four general purpose pins with AD converters.

The platform is also fitted with two cameras, an mvBlueFOX global shutter, high framerate camera and a full HD Mobius ActionCam. Image streams from both of these cameras can be used by the main computer for purposes of computer vision, localization or others. Diagram of the platform architecture is shown in figure 26, and the whole robotic platform is shown in figure 27.

6 Experimental verification

In this chapter, the experimental verification with the Gazebo robotic simulator and real hardware is presented. The kinetic attack scenario simulated in Gazebo will be presented first, followed by the same scenario but with the use of the WhyCon system for target localization. Then two experiments with real hardware will be presented, the kinetic attack scenario with static and dynamic targets, and the net launcher scenario with static and dynamic targets.

6.1 Kinetic attack scenario in simulations

The Multi-robot Systems Group platform can serve both as the interceptor and as the target in the simulations, for the sake of not needing to implement a different multicopter model for the simulations. As the newly implemented MPC tracker, described in this thesis, allows for detailed settings of maximum velocities, accelerations, and jerks in all axes, the dynamical parameters of the target can be changed as needed. The target is then set to either stay in position in case of the static target attack or fly through randomly selected waypoints in an assigned 3-dimensional area. The goal of the interceptor was then to disable the target by colliding with it, using the algorithm shown in figure 18. The information about target's position was available to the interceptor at all times, and therefore it could be fed to the estimator continuously. Based on the estimated position and future trajectory of the target, the interceptor planned it's trajectories and collided with the target.

UAV	Parameter	Value	UAV	Parameter	Value
Interceptor	v_{hmax}	9 m/s	Target	v_{hmax}	4 m/s
	a_{hmax}	3 m/s^2		a_{hmax}	2 m/s^2
	$\dot{a}_{hmax}$	6 m/s^3		$\dot{a}_{hmax}$	3 m/s^3
	v_{vmax}	3 m/s		v_{vmax}	2 m/s
	a_{vmax}	2 m/s^2		a_{vmax}	2 m/s^2
	$\dot{a}_{vmax}$	3 m/s^3		$\dot{a}_{vmax}$	2 m/s^3

Table 4: Parameters for the MPC tracker of the interceptor and the target in the kinetic attack scenario.

One instance of this scenario is shown in figure 28. In this case, both the target and the interceptor are starting at an altitude of 2 meters, and the interceptor is 100 meters away from the target. The target starts following random waypoints in a 100×100×10 m area, and the interceptor begins its attack after 10 seconds, to give the target time to pick up velocity in a random direction. The parameters of the MPC tracker for the interceptor and the target are shown in table 4

Reference tracking in the X and Y coordinates is shown in figure 29. The reference is generated by the MPC tracker and it is tracked by the non-linear SO(3) controller. The gains

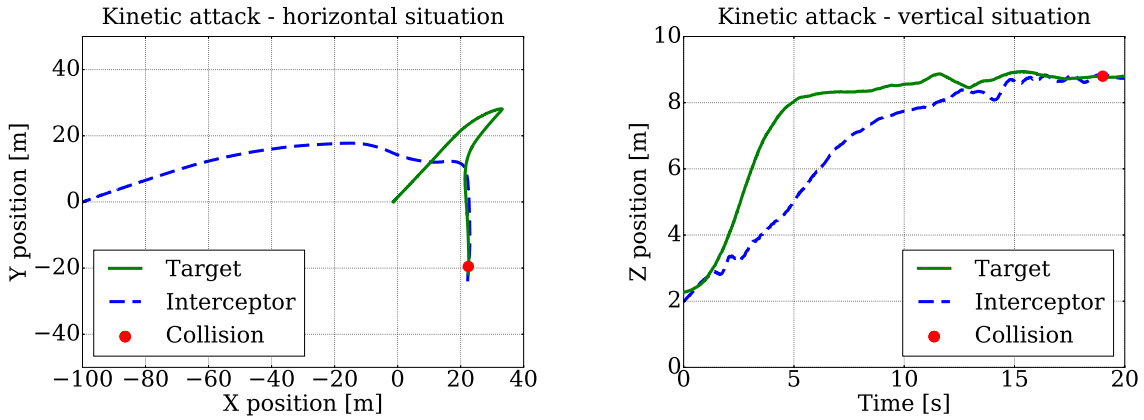


Figure 28: Simulation of the kinetic attack scenario, with the horizontal trajectory (left) and altitude in time (right).

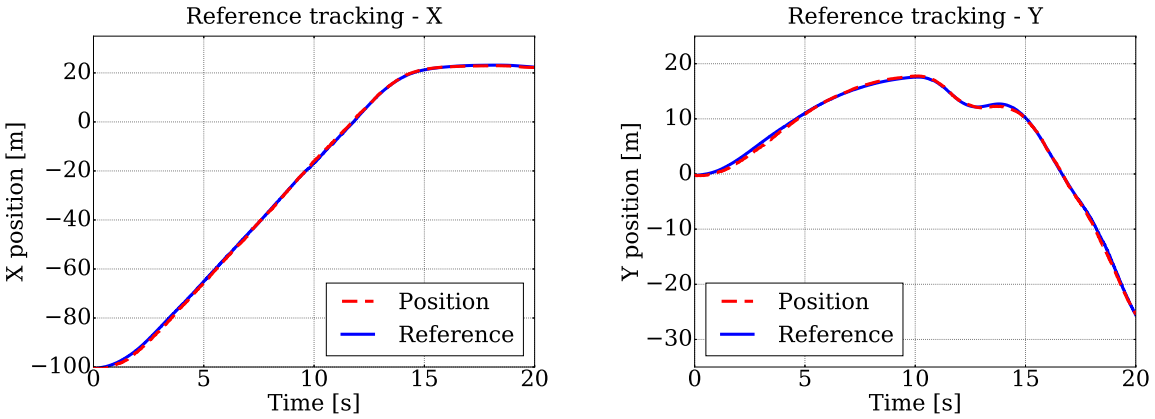


Figure 29: Reference tracking in the X (left) and Y (right) coordinates.

of the non-linear SO(3) controller have to be tuned accordingly, to ensure stable behavior and good reference tracking. Figure 30 shows trajectory and altitude estimates produced by the LKF estimator at various parts of the flight.

Snapshots from the Gazebo simulator visualization are shown in figure 31. They capture the interceptor following the target and subsequently attacking it and colliding with it.

6.2 Required interceptor performance

To effectively eliminate the target, the interceptor should be capable of flying faster than the target. Successful interception is possible even if the target is flying faster than the interceptor, but then the target has to fly in a favorable direction and not try to avoid interception actively. Influence of different speeds of the target and intercepting drone was examined in a series of simulations. The target is flying through random waypoints in a 100×100×20 m

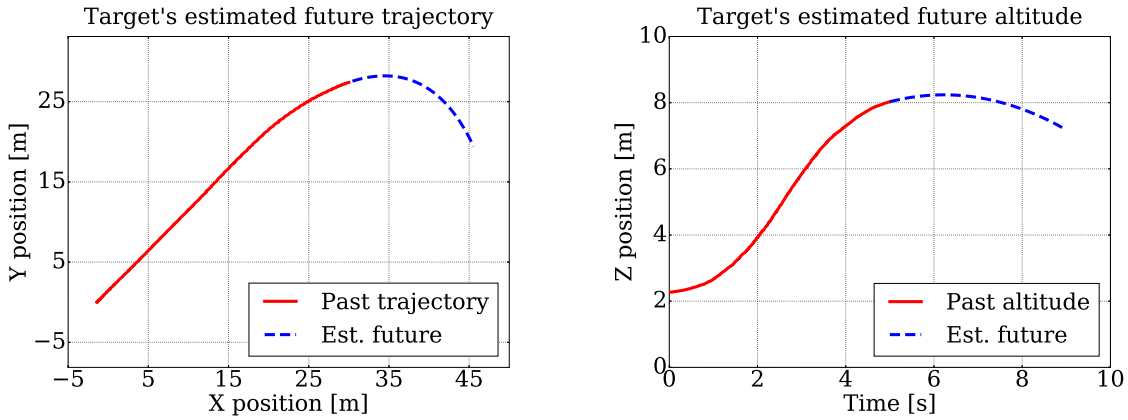


Figure 30: Estimates of the future position of the target. Horizontal situation (left) and altitude in time (right).

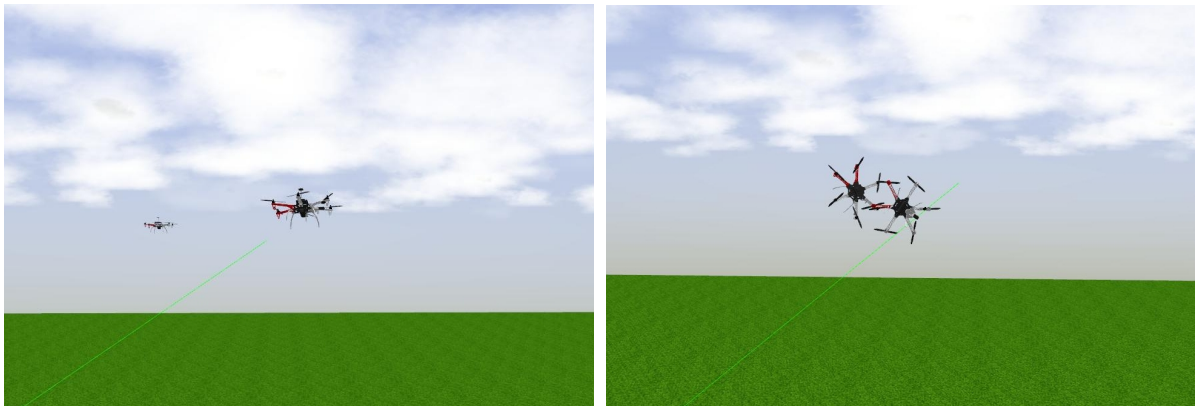


Figure 31: Interceptor approaching the target from behind (left) and interceptor colliding with the target (right).

area, while the interceptor is trying to catch up with the target. The interceptor is starting at a position that is 50 meters away from the initial position of the target. The parameters of the MPC tracker for interceptor and target are shown in table 5. The only parameter that was changed was the target's maximum horizontal speed v_{hmax}. We can define interceptor's velocity advantage as follows:

$$\Delta v = v_{hmax_i} - v_{hmax_t}, \tag{30}$$

where v_{hmax_i} is the interceptor's maximum horizontal velocity constraint and v_{hmax_t} is the target's maximum horizontal velocity constraint.

Ten different simulations were conducted for six different values of Δv and time to interception was recorded. Time was measured from the initiation of the interception algorithm, to the point at which the interceptor is less than 1 meter away from the target. The recorded times are shown in table 6

As expected, if the target is flying slowly, the interceptor can reach it earlier and with much more consistent times. If the target can fly at the same speed as the interceptor, the

UAV	Parameter	Value	UAV	Parameter	Value
Interceptor	v_{hmax}	10 m/s	Target	v_{hmax}	variable
	a_{hmax}	3 m/s^2		a_{hmax}	2 m/s^2
	$\dot{a}_{hmax}$	6 m/s^3		$\dot{a}_{hmax}$	3 m/s^3
	v_{vmax}	3 m/s		v_{vmax}	2 m/s
	a_{vmax}	2 m/s^2		a_{vmax}	2 m/s^2
	$\dot{a}_{vmax}$	3 m/s^3		$\dot{a}_{vmax}$	2 m/s^3

Table 5: Parameters for the MPC tracker of the interceptor and the target while testing the required performance.

interceptor more or less relies on luck and needs the target to fly in a favorable direction or to stop. Good and consistent results start to show up only when the interceptor can fly more than 2 m/s faster than the target.

Δv (m/s)	**0**	**1**	**2**	**3**	**4**	**5**
Time of interception	21.8	39.5	35.4	10.5	24.6	14.3
for each attempt (s)	74.1	35.8	20.9	37.1	18.1	16.2
	42.1	72.2	34.1	18.5	15.7	17.8
	32.6	22.1	32.4	15.9	15.5	15.1
	74.8	25.8	24.3	15.8	20	18.7
	23.8	46.6	20.7	36.9	20.2	16.9
	95.8	29.5	16.3	17.3	22.6	17.5
	52.8	37.4	19.4	31.6	20.3	13.1
	58.3	46.6	42.4	14.8	14.9	17.1
	33.2	39	26.2	22.6	15.2	12.4
Avg. time (s)	**50.9**	**39.5**	**27.2**	**22.1**	**18.7**	**15.9**

Table 6: Time to interception with different target maximum horizontal velocities.

6.3 Visual target localization in simulation

To test the interception algorithms in simulation, a box with WhyCon markers was placed on the target. The interceptor is equipped with a forward facing camera and can detect the pattern at a maximum distance of 8 meters. The outer diameter of the used pattern is 122 mm, and the inner diameter is 50 mm. The size of the pattern can be increased, which also increases the maximum detection distance, but larger patterns are not suitable for real-world

UAV	Parameter	Value	UAV	Parameter	Value
Interceptor	v_{hmax}	9 m/s	Target	v_{hmax}	4 m/s
	a_{hmax}	3 m/s^2		a_{hmax}	2 m/s^2
	$\dot{a}_{hmax}$	6 m/s^3		$\dot{a}_{hmax}$	3 m/s^3
	v_{vmax}	3 m/s		v_{vmax}	2 m/s
	a_{vmax}	2 m/s^2		a_{vmax}	2 m/s^2
	$\dot{a}_{vmax}$	3 m/s^3		$\dot{a}_{vmax}$	2 m/s^3

Table 7: Parameters for the MPC tracker of the interceptor and the target in the kinetic attack scenario with onboard visual target localization system.

experiments, as they can interfere with the airflow form the propellers and are susceptible to wind. The field of view of the camera is 116 degrees, matching the field of view of the real Mobius camera used on the MRS platform. The resolution is 1280×720 pixels, also matching the Mobius camera.

In this case, if the interceptor was not tracking the target, it returned to its takeoff position, at an altitude of 4.5 meters and started slowly rotating around its Z axis to scan for the target. The target was following random waypoints in a 40×40×5 m area around the takeoff point of the interceptor. Once the target was detected, the interceptor attempted to collide with the target.

The parameters of the MPC tracker for the interceptor and target are shown in table 7. The trajectories of the interceptor and the target are visualized in figure 32. The output of the estimator is shown in figure 33. Snapshots from the simulation are shown in figure 34.

This experiment also starts to encounter delays in the positional information of the target, which are caused by the image capture and image processing. This accounted for about 100 ms of delay between the true position of the target and the position that was available to the interceptor's trajectory planning algorithm. To compensate for this delay, the estimator produces an estimate of the target's position 100 ms in the future, which is then used for planning the interceptors trajectory. In the real world, this issue is even more noticeable, as the delays caused by a real camera are even longer, as the image has to be captured, transmitted to the computer by USB and then processed.

6.4 Kinetic attack scenario with real UAVs

Since the kinetic attack scenario is the hardest in terms of control and planning, as described in section 4.1.1, it was tested with real hardware. The Multi-robot Systems Group's F550 platform served both as the interceptor and as the target. Since this platform is very expensive and hard to replace, and there is a high probability of serious damage being inflicted on both the target and the interceptor during the kinetic attack scenario, a long, thin wooden bar was fitted to the target. This bar extended upwards from the body of the target, and

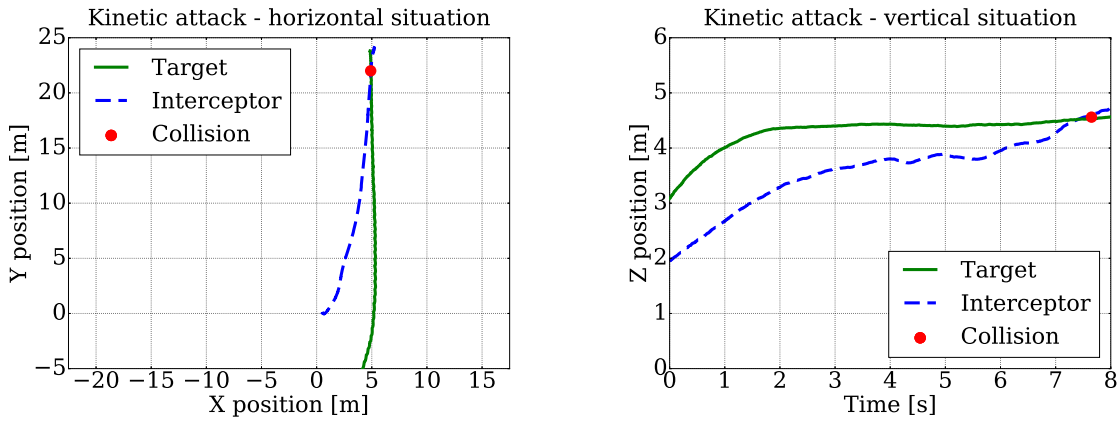


Figure 32: Simulation of the kinetic attack scenario with onboard visual target localization system. Horizontal trajectory (left) and altitude in time (right).

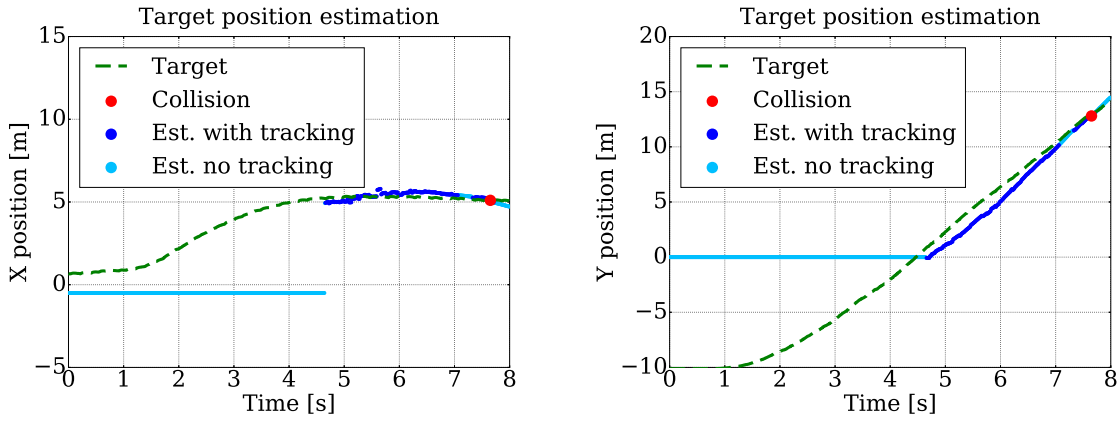


Figure 33: Estimates of the target position in X and Y axes. Dark blue denotes that the estimate was produced while a measurement was available, and light blue denotes that no measurement was available at that time (target was not detected by the camera).

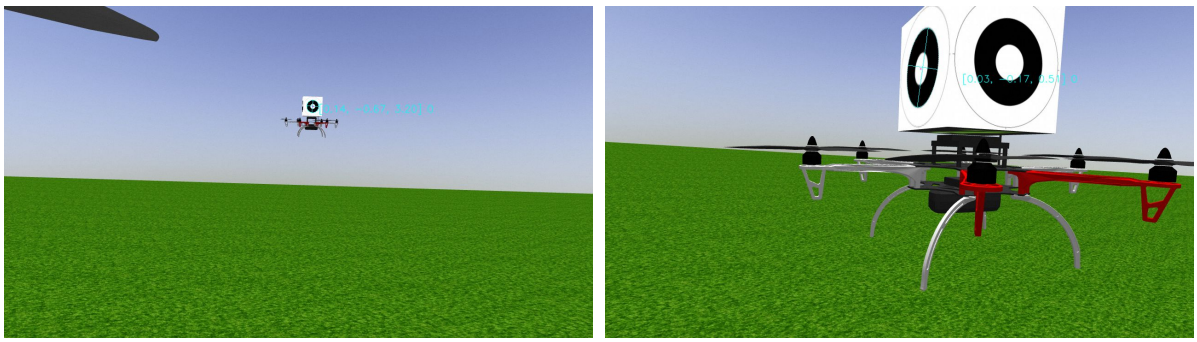


Figure 34: Detection of the WhyCon pattern (left) and interceptor moments before colliding with the target (right).

Figure 35: The target drone fitted with a balloon (left) and the interceptor (right).

an inflated balloon was attached to the top. The balloon on the rod was approximately 1 meter higher than the actual body of the drone, to allow for the interceptor to hit it and pop the balloon, without damaging the target drone or itself. The target fitted with the balloon and the interceptor are shown in figure 35. The only modification to the trajectory planning algorithm of the interceptor was offsetting all the planned trajectories by 1 meter in the Z direction.

Since the point of this experiment was to test the trajectory planning and control at high speeds, visual target localization system was not used. Instead, the target relayed information about its current position directly to the interceptor through Wi-Fi. This information was then used to produce an estimate of the states of the target by the estimator. The Multi-robot Systems Group's platform is equipped with an RTK capable GNSS receiver, which allows for the positional information of the target and the interceptor to be precise enough to perform the kinetic attack scenario. Normal GNSS receivers usually operate with an error of up to several meters, which is not precise enough for the kinetic attack scenario. That is why the balloon is mounted to the expensive platform equipped with the precise RTK GNSS receiver, and not to a cheaper and smaller drone.

If a visual target localization system is used for target localization, there is no need for the interceptor to be equipped with the expensive RTK capable GNSS receiver, as the position of the target is calculated relatively to the position of the interceptor, therefore eliminating problems with inaccurate localization of the interceptor itself.

6.4.1 Static target

The scenario with a static target was tested first, to verify the functionality of all parts of the system. The target was hovering in place, as the interceptor approached it and popped the balloon with its propellers, which is shown in figure 36 and 37. The experiment was repeated three times with the same setup, and the balloon was successfully popped every time. The wooden rod, which held the balloon in position, was also hit by the propellers of the interceptor, but it did not inflict any damage to them while being cut. This experiment

demonstrated that the system worked repeatedly with real hardware, and it validated the static target attack part of the interception algorithm.

Figure 36: The interceptor approaching the target with the balloon (left) and the interceptor popping the balloon (right).

Figure 37: Interceptor attacking the balloon mounted to a static target.

6.4.2 Dynamic target

The kinetic attack scenario with a dynamic target was tested in a similar fashion as the scenario with a static target. A long wooden rod with a balloon was again attached to the target, but this time the target was flying through randomly selected waypoints in a 50×20 m area. For safety reasons, the altitude of the target was kept at a constant value.

UAV	Parameter	Value	UAV	Parameter	Value
Interceptor	v_{hmax}	8.33 m/s	Target	v_{hmax}	4 m/s
	a_{hmax}	2.33 m/s^2		a_{hmax}	2 m/s^2
	$\dot{a}_{hmax}$	6 m/s^3		$\dot{a}_{hmax}$	3 m/s^3
	v_{vmax}	2 m/s		v_{vmax}	1 m/s
	a_{vmax}	2 m/s^2		a_{vmax}	1 m/s^2
	$\dot{a}_{vmax}$	3 m/s^3		$\dot{a}_{vmax}$	2 m/s^3

Table 8: Parameters for the MPC tracker of the interceptor and the target in the real-world kinetic attack scenario with balloons.

Since the MRS platform uses lidar to measure its altitude, the Z coordinate represents the actual altitude over terrain, which was in case of this experiment significantly uneven. As the interceptor is tilting while flying, the lidar is not pointing directly below the interceptor, but to the side at an angle. The difference in measured values is compensated for, as the attitude of the interceptor is known, but the physical level of the terrain is not even, which cannot be compensated for easily. This fact, combined with the target changing altitude while flying, would increase the risk of the interceptor unintentionally colliding with the target drone, causing a lot of damage. For this reason, the altitude of the target drone was kept constant. The interceptor also started its attack run at an altitude of 9 meters, to prevent it from crossing the flight level of the target drone.

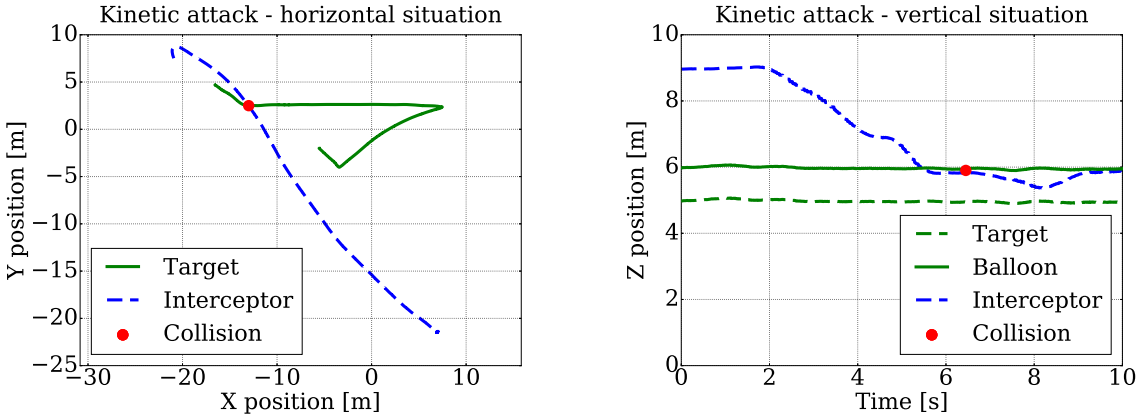


Figure 38: Kinetic attack scenario with real drones. Horizontal trajectory (left) and altitude in time (right).

The target took off first and started flying through random waypoints in the designated area. The interceptor took off second and then proceeded with intercepting the target. A camera was mounted on the interceptor, and the interceptor controlled its yaw angle to keep the camera pointed at the target, to simulate the usage of a visual target localization system or the passive net mounted on the bottom of the intercepting drone, as described in section 4.1.2.

Parameters of the MPC tracker for the target and the interceptor are shown in table 8. The new references for the target drone are set as steps, so the target drone will reach its horizontal velocity constraint while flying between waypoints. The initial experiments were done with a simplified interception algorithm, omitting the *Target follow* state (figure 18), and replacing it with *Direct attack*. One of these experiments is shown in figure 39, with the horizontal situation and altitude in time shown in figure 38. This set of experiments is also shown in a video `https://www.youtube.com/watch?v=RTzac8PLpkY` and on the enclosed DVD.

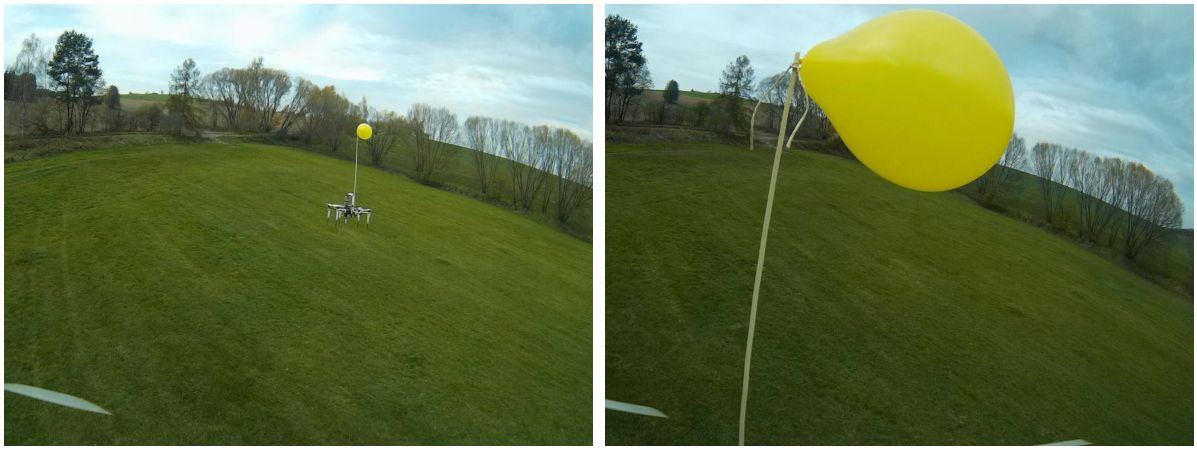


Figure 39: Interceptor approaching the target (left) and interceptor is about to hit the balloon (right).

Another set of kinetic attack experiments was conducted with the full interception algorithm, as shown in figure 18. The flight zone for the target was reduced to 30 × 15 meters, because the experimentation area was smaller. This led to the target changing its direction of flight more often, making the interception task harder.

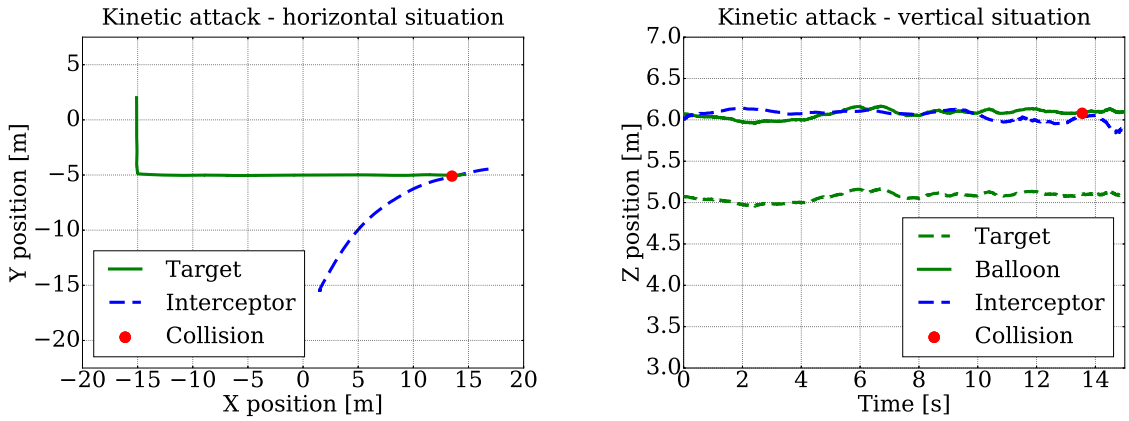


Figure 40: Kinetic attack scenario with real drones and the full interception algorithm. Horizontal trajectory (left) and altitude in time (right).

The same parameters (table 8) were used for the MPC tracker. One of these experiments

is shown in figure 41, with the horizontal situation and altitude in time shown in figure 40. Compilation video from this set of experiments is located on `https://www.youtube.com/watch?v=D1lygdYu0m0` and also on the enclosed DVD.

Figure 41: Interceptor approaching the target (left) and interceptor is about to hit the balloon (right).

6.5 Net launcher scenario with real UAVs

To further validate the system capabilities, the interceptor was fitted with a net launcher. The used launcher is originally intended for capturing wild animals, but its performance is suitable even for capturing drones. It is powered by a single-use, 16 g CO_2 cartridges, and it launches a 3×3 m net with an initial velocity of up to 10 m/s. The construction of the launcher is very rugged, which is not ideal for aerial use, as the mass of the loaded launcher is 1090 g.

Figure 42: Interceptor equipped with a net launcher (left), and net launcher servo triggering mechanism (right).

As the net launcher is intended for hand-held usage, the triggering mechanism is a mechanical push-button. To activate the trigger remotely, a servo motor was fitted to the launcher on a 3D printed mount. The servo motor can push the trigger button, and it is controlled by the custom ATxmega control board, designed by the author of this thesis, which is mentioned in 5.2.1. The board interfaces with the main computer, meaning that the launcher can be triggered directly from ROS (5.1). The interceptor equipped with the net launcher and the servo triggering mechanism is shown in figure 42. If higher payload capacity was available, the launched net could stay attached to the interceptor, allowing it to carry the captured target away.

The MRS platform itself has a mass of approximately 3 kg, and if the launcher is mounted, the mass exceeds 4 kg, which is much higher than the maximum recommended takeoff mass of 2.4 kg [1]. The interceptor is still capable of flying with a payload as high as this, but its thrust reserves are much lower, meaning that it cannot perform any aggressive maneuvers. The current draw from the battery is also substantial in this configuration. While hovering, the drone draws approximately 50 A of current, which is very close to the 60 A rated current of the used XT60 connectors. Any aggressive maneuvers would require an even higher current draw, meaning the connectors could melt, or weld together. This meant that the maximum velocities and accelerations of the interceptor had to be lowered, to prevent any damage. The used parameters are shown in table 9.

Figure 43: The target in flight, equipped with the WhyCon pattern (left), and size comparison between the target and the interceptor (right).

The target for this scenario was not another MRS platform, but a simple small quadcopter. This quadcopter was not equipped with any form of GNSS or Wi-Fi, which meant that it could not relay any information to the interceptor. The target drone was equipped with a WhyCon marker (figure 23), which could be detected by the interceptor's onboard camera. In this scenario, the WhyCon visual target localization system served as the only means for target localization. The target drone and its comparison to the interceptor is shown in figure 43.

The target was controlled manually, as it is not capable of autonomous flight. Since the target does not possess any assistance mode, like altitude or position keeping, its flight

UAV	Parameter	Value
Interceptor	v_{hmax}	5.00 m/s
	a_{hmax}	1.5 m/s^2
	$\dot{a}_{hmax}$	2 m/s^3
	v_{vmax}	1 m/s
	a_{vmax}	1 m/s^2
	$\dot{a}_{vmax}$	2 m/s^3

Table 9: Parameters for the MPC tracker of the interceptor in the net launcher scenario.

was much more erratic than that of a standard commercially available drone, like the DJI Phantom. This made the interception task harder, to test the system properly. The interceptor was positioned to an initial position at an altitude of 4.5 m and waited for the target to appear. As the WhyCon pattern was detected, the interceptor started to follow the target, and after it reached a suitable position, the net was launched, and the target was captured.

The launching of the net was triggered only if three specific conditions were met at the same time, defining the effective space of the net (figure 19). The conditions were specified as follows:

$$\begin{aligned} T_{x1} < dist3D(\mathbf{x}_{int}, \mathbf{x}_{tgt[t]}) < T_{x2}, \\ T_{z1} < (z_{int} - z_{tgt[t]}) - o_w < T_{z2}, \\ T_{\Psi 1} < (\Psi_{int} - atan2(y_{tgt[t]} - y_{int}, x_{tgt[t]} - x_{int})) < T_{\Psi 2}, \end{aligned} \tag{31}$$

where $dist3D$ is the three dimensional distance, $\mathbf{x}_{int}$ is the current position of the interceptor (consisting of x_{int}, y_{int} and z_{int}), $\mathbf{x}_{tgt[t]}$ is the estimated target position at a time sample t (consisting of $x_{tgt[t]}$, $y_{tgt[t]}$ and $z_{tgt[t]}$), while the time sample t is set to correspond with the estimated system delay caused by image processing and other factors. Ψ_{int} is the interceptor's yaw angle. Variables $T_{x1}, T_{x2}, T_{z1}, T_{z2}, T_{\Psi 1}$ and $T_{\Psi 2}$ specify the thresholds for the conditions, and o_w is the Z offset caused by camera and WhyCon pattern placement. This set of condition defines that the target has to be at a certain distance and relative altitude to the interceptor, while the interceptor's front is pointed at the target. If all three conditions are met at the same time, the net is launched. For the experiments, following values were used:

$$\begin{aligned} T_{x1} = 3 \text{ m}, \quad T_{x2} = 5 \text{ m}, \quad T_{z1} = -0.3 \text{ m}, \quad T_{z2} = 0.3 \text{ m}, \\ T_{\Psi 1} = -0.06 \text{ rad}, \quad T_{\Psi 1} = 0.06 \text{ rad}, \quad o_2 = 0.7 \text{ m}. \end{aligned} \tag{32}$$

Two experiments were conducted with a static target, although static, in this case, meant that the pilot was trying to keep the target in the same position, still resulting in a lot of movement. The target was captured in both cases, which is shown in figures 44 and 45. In the second case, the target was not hit in an ideal way, which was caused by wrongly estimating the delay in the system. The whole process of the camera capturing an image, relaying it to the main computer, image processing, commanding the net launcher to launch and the servo motor pushing the trigger takes approximately 350 milliseconds. During these experiments, this delay was not estimated correctly, causing the interceptor to launch the net

later and almost missing the target. The same issue caused a miss in the next scenario with a moving target, which prompted a reevaluation of the estimated delay and correction of the issue.

Figure 44: Target being captured by the launched net, aerial perspective.

Figure 45: The interceptor launching a net and hitting the target.

6.5.1 Dynamic target

The final experimental scenario involved the same setup as before, with the only difference that the target was now moving away from the interceptor. The target overflew the interceptor, entering the camera frame, and continued flying away from the interceptor at a speed of 2-3 m/s, again with side to side movement due to manual control. In the first instance of this scenario, the target was missed because of the mentioned delay. The delay issue was fixed, and in the next attempt, the target was hit square on. The entire interception, from the first detection of the WhyCon pattern to the launching of the net, took less than five seconds. Figure 51 shows the entire scenario from the interceptor's point of view, while figures 46 and 47 show the position of the interceptor and the estimated position of the target in 3D. Figures 48 and 49 show the estimated position of the target in X, Y and Z directions, as well as the total estimated velocity. Figure 50 the shows the status of the decision variables for net launching, and the yaw angle of the interceptor compared to the yaw angle at which the target is "seen" by the interceptor. The experiments with the net launcher are also shown in a video located on `https://www.youtube.com/watch?v=Z4lSam-lpN8` or on the enclosed DVD.

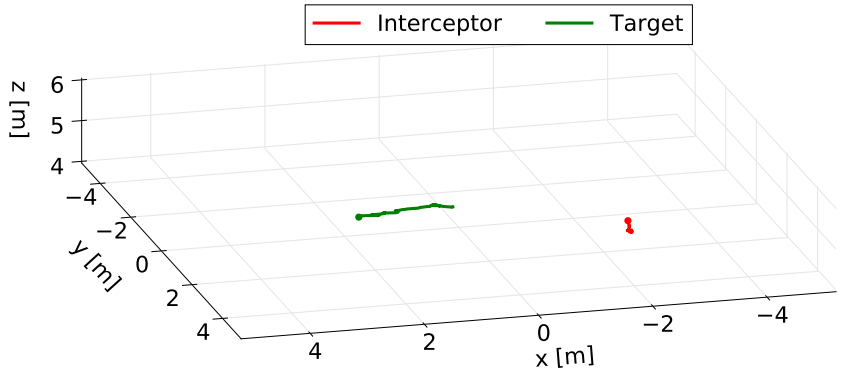


Figure 46: The interceptor detects the target with its onboard camera and begins the interception.

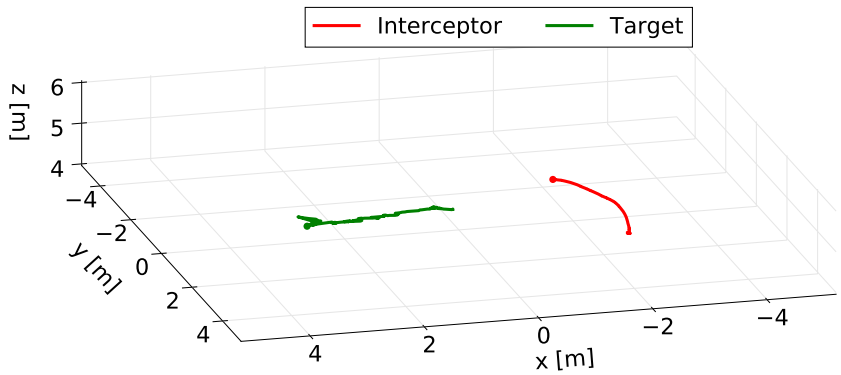


Figure 47: The interceptor aligns with the target and launches the net at this point.

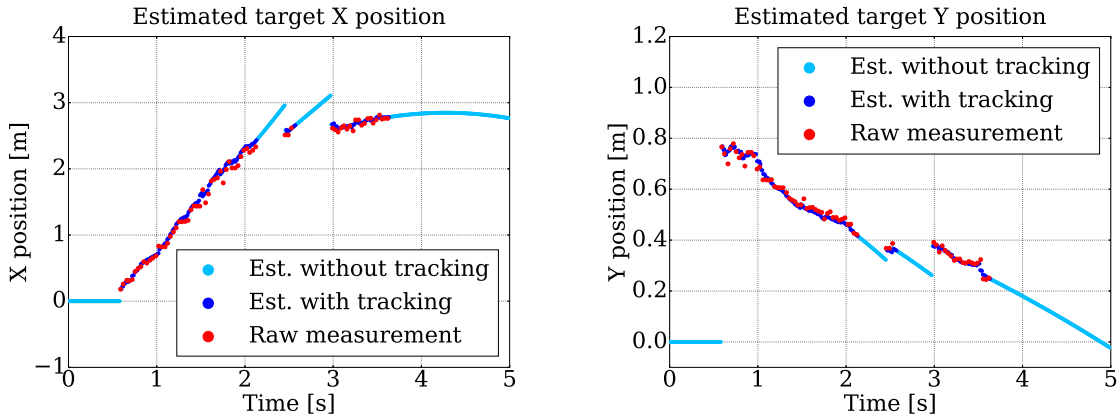


Figure 48: Estimated target position in X (left) and in Y directions (right).

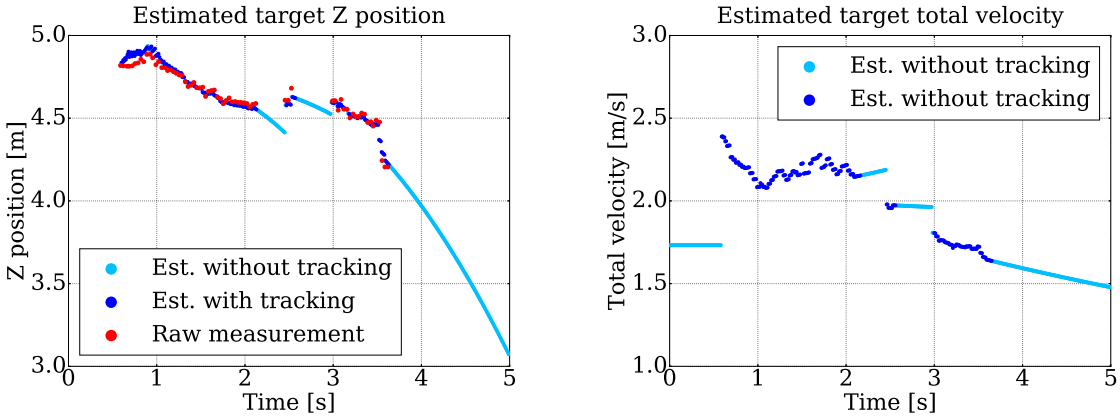


Figure 49: Estimated target position in Z (left) and estimated total velocity (right).

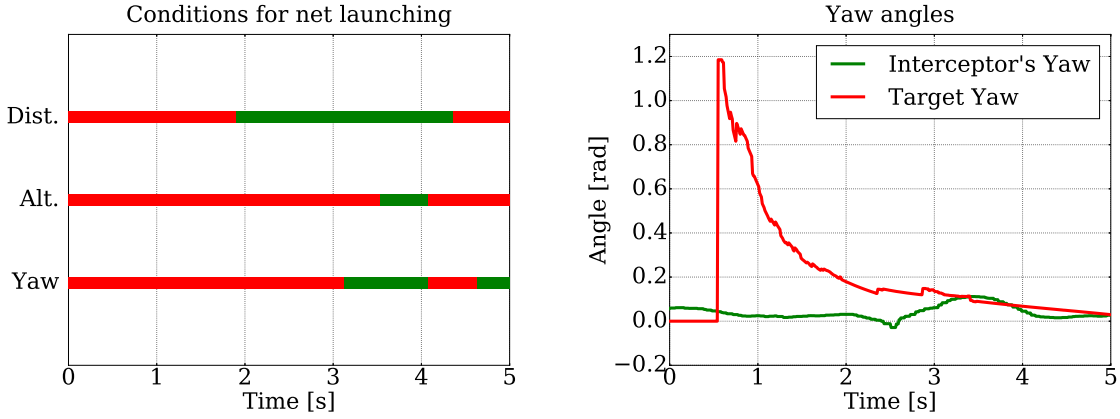


Figure 50: Conditions for net launching, as shown in equation 32 (left) and yaw of the interceptor, compared to the yaw at which the target is "seen" by the interceptor (right).

Figure 51: Dynamic target scenario from the point of view of the interceptor, including the output of WhyCon. Images have been cropped for better visibility.

6.6 Real-world utilization

The experiments described in this chapter verified the performance of the system in different scenarios. If precise and reliable target localization is available (as was the case in the kinetic attack scenario and localization with RTK GNSS), the system is able to consistently hit the randomly flying target. As described before, the kinetic attack scenario is the hardest in terms of control and trajectory planning. Therefore, we can assume that if the system is able to hit the target in a kinetic attack scenario, it will also be able to hit it with a passive net or with a net launcher. This means that the system would be able to intercept other drones if a good target localization information was available.

The big hurdles in a real-world deployment of this system are reliability and the target localization. The MRS platform is intended for scientific experiments, with frequent changes in hardware and software configuration, which sometimes leads to unexpected problems. In a real-world application, the interceptor drone would have to be very reliable and work in a wide range of environmental conditions. This could be achieved by selecting a reliable and proven hardware platform, along with a lot of testing. The target localization system would

have to be precise, with a low probability of false detections, for it to be usable in real-world applications, which is a very hard requirement. The work presented in this thesis successfully demonstrates techniques for target interception, but a working real-life commercially viable system would require a large amount of further work.

7 Conclusion

The goal of this thesis was to develop an autonomous system for intercepting small flying aircraft with an unmanned multirotor drone. The developed system consists of a strategic trajectory planner, which plans different trajectories according to the configuration of the intercepting drone and the used localization system, an improved MPC control scheme, which relies on a custom solver generated by CVXGEN and allows for aggressive maneuvers and usage of an infeasible reference trajectory, and a linear Kalman filter working as a state observer, which estimates and predicts the states of the target, based on available measurements. Many experiments were conducted in the Gazebo robotic simulator, as well as the real world to verify the performance of the system in different configurations of the intercepting drone and different target tracking approaches.

All of the tasks were successfully fulfilled, according to the assignment:

- The author familiarized himself with the Multi-robot Systems Group's UAV platform, along with the Robot Operating System.
- State observer was designed and implemented in chapter 2, to estimate and predict states of the target UAV, based on measurements from a computer vision system or other sources.
- The current model predictive control scheme was improved in chapter 3, by introducing a new solver generated using CVXGEN, with improved state constraints. This enabled the usage of infeasible trajectories as references, which facilitates trajectory planning for interception.
- Various interception strategies based upon different scenarios were designed and implemented in chapter 4.
- WhyCon marker based computer vision system was integrated with the system, for the purposes of target detection.
- The system was tested in chapter 6 during various scenarios in the Gazebo robotic simulator.
- Performance of the system was tested in the real world by hitting balloons attached to another UAVs. An additional scenario with a net launcher was also successfully tested, to demonstrate the flexibility of the system. This scenario was beyond the original thesis assignment.

The improved MPC tracker designed and implemented in this thesis replaced the older implementation on the Multi-robot Systems Group's platform and, it is being used since as the default tracker in most of the experiments conducted by the Multi-robot Systems Group.

7.1 Future work

To further improve the system and make it usable in the real world by third parties, the intercepting drone needs more payload capacity to carry the net launcher or a similar device for target capture. If the system is to be used over a populated area, the interceptor should also be able to carry the captured target away, without the target hitting the ground, to prevent damage. If the passive net is used, the target will tangle into it and remain attached, and if the net launcher is used, the launched net can remain attached to the interceptor with a tether, or the net can be equipped with a small parachute, to reduce the velocity of the falling target. The WhyCon computer vision system has to be replaced with another localization system, which can detect targets without markers. A possible candidate is computer vision system based on deep learning. To improve the coverage of the system, it should be integrated with a form of initial localization system, either a ground-based tracking station or a human operator, which will provide an initial estimate of the target's position for the intercepting drone.

References

[1] Dji flame wheel arf kit specifications. `https://web.archive.org/web/20171027032020/https://www.dji.com/flame-wheel-arf/spec`. Snapshot: 2017-08-30.

[2] Sky wall 100 hand-held net launcher. `https://web.archive.org/web/20180510134143/https://openworksengineering.com/skywall/100`. Snapshot: 2018-05-10.

[3] Cemal Aker and Sinan Kalkan. Using deep networks for drone detection. In *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017.

[4] Tomas Baca, Petr Stepan, and Martin Saska. Autonomous landing on a moving car with unmanned aerial vehicle. In *European Conference on Mobile Robotics (ECMR)*. IEEE, 2017.

[5] Lars Blackmore. Autonomous precision landing of space rockets. *The Bridge*, 4(46), 2016.

[6] Matthias R Brust, Gregoire Danoy, Pascal Bouvry, Dren Gashi, Himadri Pathak, and Mike P Gonçalves. Defending against intrusion of malicious uavs with networked uav defense swarms. In *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, pages 103–111. IEEE, 2017.

[7] Filip Bursík. Control and navigation of an unmanned helicopter using sensors in mobile phones. Bachelor's thesis, Czech Technical University in Prague, expected in June 2018.

[8] Tomáš Báča. Model predictive control of micro aerial vehicle using onboard microcontroller. Master's thesis, Czech Technical University in Prague, 2015.

[9] Jonathan Currie. Practical applications of industrial optimization: from high-speed embedded controllers to large discrete utility systems. Phd thesis, Auckland University of Technology, 2014.

[10] Jedrzej Drozdowicz, Maciej Wielgo, Piotr Samczynski, Krzysztof Kulpa, Jaroslaw Krzonkalla, Maj Mordzonek, Marcin Bryl, and Zbigniew Jakielaszek. 35 ghz fmcw drone detection system. In *2016 17th International Radar Symposium (IRS)*. IEEE, 2016.

[11] Keisuke Fujii. Extended kalman filter. *Refernce Manual*, 2013.

[12] Sait Murat Giray. Anatomy of unmanned aerial vehicle hijacking with signal spoofing. In *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, pages 795–800. IEEE, 2013.

[13] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. *Journal of basic engineering*, 83(1):95–108, 1961.

[14] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[15] Andrew J Kerns, Daniel P Shepard, Jahshan A Bhatti, and Todd E Humphreys. Unmanned aircraft capture and control via gps spoofing. *Journal of Field Robotics*, 31(4):617–636, 2014.

[16] Tomáš Krajník, Matías Nitsche, Jan Faigl, Petr Vaněk, Martin Saska, Libor Přeučil, Tom Duckett, and Marta Mejail. A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*, 76(3-4):539–562, 2014.

[17] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil. External localization system for mobile robotics. *2013 16th International Conference on Advanced Robotics (ICAR)*, Nov 2013.

[18] Kee-Woong Lee, Kyoung-Min Song, Jung-Hwan Song, Chul-Ho Jung, Woo-kyung Lee, Myeong-jin Lee, and Yong-Kyu Song. Implementation of radar drone detection based on isar technique. *The Journal of Korean Institute of Electromagnetic Engineering and Science*, 28:159–162, 02 2017.

[19] T. Lee, M. Leoky, and N. H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). *49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425, Dec 2010.

[20] Thierry Lefebvre and Thomas Dubot. Conceptual design study of an anti-drone drone. In *16th AIAA Aviation Technology, Integration, and Operations Conference*, page 3449, 2016.

[21] Giuseppe Loianno, Vojtech Spurny, Justin Thomas, Tomas Baca, Dinesh Thakur, Daniel Hert, Robert Penicka, Tomas Krajnik, Alex Zhou, Adam Cho, et al. Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments. *IEEE Robotics and Automation Letters*, 3(3):1576–1583, 2018.

[22] Jacob Mattingley and Stephen Boyd. Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[23] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[24] Johannes Meyer, Alexander Sendobry, Stefan Kohlbrecher, Uwe Klingauf, and Oskar Von Stryk. Comprehensive simulation of quadrotor uavs using ros and gazebo. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 400–411. Springer, 2012.

[25] Thomas Multerer, Alexander Ganis, Ulrich Prechtel, Enric Miralles, Askold Meusling, Jan Mietzner, Martin Vossiek, Mirko Loghi, and Volker Ziegler. Low-cost jamming system against small drones using a 3d mimo radar based tracking. In *2017 European Radar Conference (EURAD)*, pages 299–302. IEEE, 2017.

[26] Robert Pěnička, Jan Faigl, Petr Váňa, and Martin Saska. Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217, 2017.

[27] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3. Kobe, Japan, 2009.

[28] Fred Samland, Jana Fruth, Mario Hildebrandt, Tobias Hoppe, and Jana Dittmann. Ar. drone: security threat analysis and exemplary attack to track persons. In *Intelligent Robots and Computer Vision XXIX: Algorithms and Techniques*, volume 8301. International Society for Optics and Photonics, 2012.

[29] M. Saqib, S. Daud Khan, N. Sharma, and M. Blumenstein. A study on detecting drones using deep convolutional neural networks. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017.

[30] Martin Saska, V Kratky, V Spurny, and Tomáš Báca. Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control. *IEEE ETFA*, 2017.

[31] A. Schumann, L. Sommer, J. Klatte, T. Schuchert, and J. Beyerer. Deep cross-domain flying object classification for robust uav detection. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017.

[32] Xiufang Shi, Chaoqun Yang, Weige Xie, Chao Liang, Zhiguo Shi, and Jiming Chen. Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges. *IEEE Communications Magazine*, 56(4):68–74, 2018.

[33] Matouš Vrba. Relative localization of helicopters from an onboard camera image using neural networks. Master's thesis, Czech Technical University in Prague, expected in June 2018.

[34] Ryan J Wallace and Jon M Loffi. Examining unmanned aerial system threats & defenses: A conceptual analysis. *International Journal of Aviation, Aeronautics, and Aerospace*, 2(4), 2015.

[35] V. Walter, M. Saska, and A. Franchi. Fast mutual relative localization of uavs using ultraviolet led markers. In *2018 International Conference of Unmanned Aircraft System (ICUAS)*, 2018. accepted to ICUAS 2018.

[36] Yan Rockee Zhang, Yih-Ru Huang, and Charles Thumann. Noise and lpi radar as part of counter-drone mitigation system measures. In *Radar Sensor Technology XXI*, volume 10188. International Society for Optics and Photonics, 2017.

8 DVD Content

Table 10 lists names of all root directories on the enclosed DVD.

Directory name	**Description**
thesis	this thesis in pdf format
src/thesis	LaTeX source codes of this thesis
src/interception	source codes for the interception ROS nodes
videos	videos from the experiments

Table 10: DVD Content