UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

A perception-aware NMPC for collision avoidance and control of a multi-rotor UAV with visual localization constraints

Author:
Andriy DMYTRUK

Supervisor:
Dr. Giuseppe SILANO

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Prague, Lviv 2021

Declaration of Authorship

I, Andriy DMYTRUK, declare that this thesis titled, "A perception-aware NMPC for collision avoidance and control of a multi-rotor UAV with visual localization constraints" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

A perception-aware NMPC for collision avoidance and control of a multi-rotor UAV with visual localization constraints

by Andriy DMYTRUK

Abstract

Multirotor Unmanned Aerial Vehicles (UAVs) are an extremely versatile and practical platform that can be used for automatization of many real-world tasks. Special interest at present time is shown to solutions requiring collaboration of multiple agents, which require precise relative localization and reactive control to avoid collisions, while performing high-level mission tasks (e.g., autonomous inspection, human-robot interaction) when UAVs are moving in close proximity with each other.

In this work, a perception-aware NMPC (Nonlinear Model Predictive Control) scheme for controlling a multi-rotor UAV while avoiding dynamic obstacles and maintaining visibility coverage of a target is set up. The proposed approach is meant to directly produce the rotor-level (torque) inputs of the platform motors, hence it does not require an intermediate unconstrained controller to work. It is also meant to be generic, by covering standard coplanar quad-rotors as well as tilted-propeller multi-rotors. A generic onboard sensor is taken into account to retrieve the 3-D pose of the target. The model can be applied to various types of sensors equipped on aerial vehicles, such as monocular cameras or laser scanners. Further, real actuator limitations are modelled to simulate platforms physical constraints.

Numerical simulations in MATLAB extensively test the proposed framework showing its validity and effectiveness.

Acknowledgements

I would like to express my deep and sincere gratitude to my research supervisor, Dr. Giuseppe Silano for providing invaluable guidance, help and advice. I am extremely thankful to him for showing me to this field and proposing to work on this challenging both theoretically and technically task. I am very grateful for motivation he gave and for the time he spent on explanations, discussions, reviews during my work.

Special thanks to the Multi Robotic Systems (MRS) laboratory at the Czech Technical University in Prague, all my colleagues there in particular, the head Prof. Dr. Martin Saska for introducing me to aerial robotics, and allowing to work on tasks and challenges related to this field.

I am also expressing my gratitude to the AERIAL-CORE EU project where the proposed framework finds an application for showing the importance and practicality of Unmanned Aerial Vehicles (UAVs).

Contents

List of Abbreviations

CoM	**C**enter **o**f **M**ass
FoV	**F**ield **o**f **V**iew
GNSS	**G**lobal **N**avigation **S**atelate **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
LiDAR	**Li**ght **D**etection **A**nd **R**anging
MAV	**M**icro **A**erial **V**ehicle
MPC	**M**odel **P**redictive **C**ontrol
NMPC	**N**onlinear **M**odel **P**redictive **C**ontrol
NLP	**N**on**l**inear **P**rogramming
OCP	**O**ptimal **C**ontrol **P**roblem
PID	**P**roportional **I**ntegrative and **D**erivative (Control)
QP	**Q**uadratic **P**rogramming
Radar	**Ra**dio **d**etection **a**nd **r**anging
RTI	**R**eal **T**ime **I**teration (scheme)
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
Sonar	**So**und **n**avigation **a**nd **r**anging
SQP	**S**equential **Q**uadratic **P**rogramming
UAV	**U**nmanned **A**erial **V**ehicle
UV	**U**ltra **V**iolet light
UVDAR	**U**ltra **Violet D**etection **A**nd **R**anging

Chapter 1

Introduction

1.1 Autonomous UAVs

Autonomous Unmanned Aerial Vehicles (or UAVs) have gotten an extremely large focus in the academics and commercially over the last decades. In the recent years, development in sensor technologies and progress in data processing techniques have led to a large amount of research that aims to achieve accurate and safe completion of various tasks with aerial platforms.

Small size, high maneuverability and versatility make UAVs, expecially multirotor copters, a perfect tool for a large variety of tasks, ranging from passive missions, like monitoring, surveillance and photographing (Gonçalves and Henriques, 2015) to interactive objectives like grasping (Spica et al., 2012), transportation, and autonomous exploration (Qin et al., 2019), mapping and search and rescue missions (Rouček et al., 2019), with the hardest tasks requiring cooperation between multiple UAVs.

These tasks encourage the development of more complex multirotor platforms (Rajappa et al., 2015, Franchi et al., 2018 and Brescianini and D'Andrea, 2016), control techniques with faster responses to changes in environment (Carlos et al., 2020) and more accurate perception techniques.

1.2 Power Line Inspection

AERIAL-CORE[1] is a collaborative research project between multiple EU universities funded by Horizon 2020 European Program. The project is developing a congnitive robotic system for inspection and maintenance of various infrastructures, with a focus on extended operational range and safety in the interaction with people. One particular task that received a lot of research interest within the project is power lines inspection.

As noted by Uzakov, Nascimento, and Saska, 2020 nowadays, teams of specialists have to climb on power towers for inspection, which often requires shutdown of the facility and comes at an extremely high cost. Using UAVs to autonomously collect high-resolution images which can be analysed to assess maintanance needs can be both more accurate and eliminate the dangers present in humans manually performing this work. Before reaching fully autonomous inspections, a human-supervised inspection can be performed.

In this case a number of UAVs could be performing tasks simultaneously to collect more data per a time period, which requires task and motion planning techniques for both efficient task distribution and specific safety requirements, for instance, using Signal Temporal Logic by Silano et al., 2021.

[1] http://aerial-core.eu/

Due to the danger of damaging high-voltage power lines and the possibility of harming nearby human workers, maintaining the safety requirements is extremely important. This is especially the case when multiple UAVs operate at the same time and in cluttered environments. High precision and ability to immediately react to potential changes is essential in this case to prevent crashes between the aerial platforms.

1.3 Objectives

It is evident that control of the multirotor UAV with collision avoidance and ability for fast reactions to disturbances and changes in the system and environment is a task of great importance nowadays (Jacquet et al., 2020, Carlos et al., 2020 and Falanga et al., 2018). Another limitations on the system can appear from objectives related to perception, and need to be appropriately handled by the aerial platform.

While most tasks, feature static obstacles, which need to be avoided, dynamic obstacles pose a greater challenge to not only be avoided, but also detected in time. In the tasks of target-tracking or cooperation of multiple aerial agents, maintaining visual contact may be required to avoid collisions and successfully track the location of nearby aerial vehicles. This can prove to be especially useful in real-world scenarios where Global Navigation Satelate System (GNSS) service is not present or limited.

The objective of this work is exactly to solve the problem of collision-free control of a multirotor with task or safety related visual constraints, in particular for perception of other UAVs in multi-agent tasks. All this needs to be done with certain guarantees about reaction time and ability to instantly react to changes in the environment. Furthermore, the solution presented here is not limited to a single configuration of the platform, but can work for any configuration in our knowledge, including fully-actuated multirotors with tilted propellers (Jacquet et al., 2020).

Nonlinear Model Predictive Control (NMPC) is gaining a lot of interest these years (Falanga et al., 2018, Jacquet et al., 2020, Lindqvist et al., 2020). Advances in optimization algorithms allow to get near-optimal solutions to constrained control problems with the ability to predict the state of the system for a relatively long time-period ahead (Carlos et al., 2020, Diehl, Bock, and Schlöder, 2005). All this computed online on an aerial platform. Due to its advantageous performance compared to non-predictive methods, the solution presented here will be based on NMPC.

Chapter 2

Related Works

2.1 Perception

Perception is a fundamental ability for autonomous vehicles. It allows to retrieve information from data generated by various types of sensors present on the platform. Perception is crucial for self-localization and also situational awareness. Both these objectives are often tackled at the same time in Simultaneous Localization and Mapping (SLAM) techniques.

Muitiple sensors can be processed to 'perceive' own state and surroundings, including Radio & Laser detection and ranging sensors (Radars & LiDARs), monocular and stereo cameras, GPSs, inertial measurement units (like gyroscopes, accelerometers, etc.) and sound navigation and ranging sensors (Sonars).

2.1.1 Self-Localization and Static Objects

The data from sensors is used to generate a map of the environment, that is saved in a occupancy grid (Moravec, 1989) or grid-like structures (e.g. octo-trees in Hornung et al., 2013). Furthermore, data from multiple sensors can be fused together to generate a more accurate map, while reducing the disadvantages of each of the sensors (Kocić, Jovičić, and Drndarević, 2018).

Further processing of maps or camera images leads to perception of individual objects present in the environment by detecting, classifying and tracking them. These tasks are often handled with computer vision methods, like convolutional neural networks (Gu et al., 2018) processing image data or even fused data from different sensors. For self-driving cars the objects could be people, other cars, signs present on the road (Kim et al., 2016, Gao et al., 2018c). In case of UAVs, the detection targets are extremely task-dependant and ranges from daily objects to people and sources of danger in search and rescue missions (Rouček et al., 2019).

2.1.2 Dynamical Objects

Dynamical objects often pose a different challenge from building static obstacle maps, because their fast movement complicates tracking between frames, which is especially the case for small objects. Thus more advanced algorithms for detection and tracking are required.

The task often requires estimating trajectory and prediction expected position. Approaches for tracking dynamical objects include Kalman filters (Watanabe, Calise, and Johnson, 2007), Particle filters (Catalin and Nedevschi, 2008) or combination of different methods (Miller and Campbell, 2007).

The detection of other aerial platforms, while cooperating with them for a successful completion of a task is even more challenging. A large number of experiments for multi-UAV scenarios is performed in laboratory conditions, which feature high-tech equipment for localizing self and other UAVs. As an example, in Riviere et al., 2020 all the UAVs fly inside motion capture space where each robot is equipped with a single marker and Crazyswarm (Preiss et al., 2017) is used for tracking them. Crazyswarm is a system that depends on a large number of static cameras that detect light emitted on board of the UAVs. Another example is presented in Mellinger and Kumar, 2011 who use Vicon[1] motion capture system to estimate position, orientation and velocity of the vehicle.

In practice, this equipment is not available in real-world scenarios. GNSS services, like Global Positioning System (GPS), may not be precise enough in many cases, and can be further limited or absent due to noise, communication interference and physical barriers (Langley, Teunissen, and Montenbruck, 2017, Karaim et al., 2018, Tang et al., 2015). Instead, computer vision techniques most commonly are executed on board to detect any neighbouring UAV. This can be achieved using neural networks (Gu et al., 2018), attaching labels with printed patterns to UAVs that can be detected (Krajník et al., 2014), or approaches for communicating ones presence visually, like UVDAR (Walter et al., 2019).

UVDAR (or Ultraviolet Detection and Ranging) is a technology, that relies on placing Ultraviolet (UV) light emitting diodes (LED) on UAVs blinking at a certain frequency. The blinking lights are then detected on other UAVs to achieve accurate localization up to 15 meters. Walter et al., 2019 have shown that placing 6 LEDs in a hexagon shape around the UAV is sufficient to estimate both its relative position and orientation. The main advantage is that this system is independent from lighting conditions of the surrounding environment and because of the lack of natural sources of UV light in the world, is not demanding in terms of computational resources. Even though this system is limited to usage in cooperative missions, where all the UAVs have UV blinking configured, and cannot be used for tracking a target without LEDs, it is highly practical in many real-world scenarios.

2.2 Motion Planning

Motion planning is a module with the main objective of computing the plan of future movement of the vehicle while respecting its dynamic constraints, avoiding collisions and optionally maintaining additional properties, like making the motion smooth.

Motion planning results in generation of a path or trajectory. A path is a sequence of points that the vehicle will move through, that can be defined as $(p_1, p_2, \ldots, p_N)$, where p_i is a vector that can include any combination of position, orientation, their derivatives and other variables. A trajectory is also a sequence of points, but parametrized over time. It can be created from path by assigning each point an exact timestamp, corresponding to the time when vehicle will be at that point.

2.2.1 Graph-Based Methods

Graph-based planning methods build the plan in a discretized to a graph version of the environment, where vertices correspond to possible configurations in the space

[1] http://www.vicon.com/

of the vahicle and edges between them correspond to a possibility of direct movement from one state to the other.

The easiest approaches use grid-like occupancy maps and build a path, ignoring the dynamic constraints and after that make sure the task is feasible in terms of dynamics. The most famous algorithm for building path is likely by Dijkstra et al., 1959. A faster algorithm that is based on the same idea is A* (Hart, Nilsson, and Raphael, 1968), which relies on a heuristic to initially choose paths, that are more likely to be optimal, but in the worst case has the same complexity as dijekstra's algorithm. There are also further optimized algorithms, like jump point search by Harabor and Grastien, 2011.

While being optimal in terms of the graph, that the planning is based on, the final solutions is not optimal due to the discretization of the space. To tackle this issue other sampling techniques were developed, like rapidly-exploring random trees (RRT by LaValle et al., 1998) and probabilistic roadmaps (Kavraki et al., 1996).

2.2.2 Optimization-Based Methods

Unlike graph based-methods, optimization-based methods do not discretize the environment. They allow the UAV to be in an infinite number of configurations in space and instead create bounds on it to avoid collisions. They also enforce the dynamical constraints of the vehicle directly into the problem, which results in a much better solution.

The desired trajectory is piece-wise function of N segments, each of which are polynomials $s_i(t)$. Then to guarantee dynamic feasibility, one has to ensure Lipschitz continuity in the points between two segments, thus the position, velocity, acceleration and further derivatives if required at the end of i-th polynomial need to match the values at the beginning of $(i+1)$-th polynomial:

$$\begin{aligned} s_i(t_i) &= s_{i+1}(t_i), & i &\in \{1, \dots, N-1\}, \\ \dot{s}_i(t_i) &= \dot{s}_{i+1}(t_i), & i &\in \{1, \dots, N-1\}, \\ &\dots & &\dots \end{aligned}$$

where t_i is the time when vehicle is expected to be at the end of i-th trajectory.

The objective function can minimize the distance, minimize time or even more specific expressions, like the integral of squared snap (snap is the fourth derivative of position) which was pioneered by Mellinger and Kumar, 2011 and was shown to generate safer trajectories with less rapid changes in movement. The collision constraints are generated by sampling points along the trajectory and enforcing in each of them. This has been modified by Gao et al., 2018a, who proposes to use Bezier curves instead of splines, that have the convex hull property and allow to reduce the number of contraints for each component of the trajectory.

In these works the trajectory is generated from desired path points that need to be visited, and Gao et al., 2018b address the issue of time distribution between each of the segment of the trajectory. Gao et al., 2018a create a velocity field and use Fast Marching Method to create a path and distribute time between the segments.

2.3 Motion Control

The task of a controller is to guide the vehicle by sending commands to hardware or low-level modules to control actuators on the platform. While some of the motion

planning techniques mentioned above already can generate a trajectory that is feasible by vehicle dynamics, the control module has to handle various disturbances that affect the vehicle and correct its state back to desired trajectory. On aerial platforms the disturbances can significantly effect the system, because of the high spinning rates of its propellers and possible environmental disturbances like wind gust.

A number of control strategies for multi-rotor UAVs are designed specifically for trajectory tracking. The most common of these are PID (or Proportional, Integrative and Derivative) controllers which can be created by linearizing the system around hovering conditions as done by Michael et al., 2010 or by linearizing feedback as by Lee, Leok, and McClamroch, 2010. Other methods include back-stepping (Bouabdallah and Siegwart, 2005) and adaptive control (Dai, Lee, and Bernstein, 2014). Even more control strategies can be found in the review by Hua et al., 2013.

However, planning and optimization-based trajectory generation algorithms tend to be computationally demanding and take a relatively large amount of time for an aerial platform. The disadvantage of strictly following the trajectory generated by these algorithms is that the planner might not be able to react in time to external distrubances, like a detection of an obstacle, that wasn't previously seen.

The usage of NMPC (Bicego et al., 2020, Alexis, Nikolakopoulos, and Tzes, 2014) makes large advances to tackle this problem. The main advantage of NMPC is that it is predictive. It uses system's model to predict the behaviour of this system up to a certain time horizon, and allows to add additional constraints directly on the control solution.

In many works NMPC was used in the outer loop while the inner-loop attitude control task was solved by unconstrained controllers Bangura and Mahony, 2014 and Alexis et al., 2016). However, in works like Kamel et al., 2015 and Ligthart et al., 2017 it is emloyed directly in the inner loop, which makes the technique very powerful.

While works like by Alexis, Nikolakopoulos, and Tzes, 2012 use NMPC to improve the trajectory tracking, others utilize the advantages of its formulation to add constraints. Static obstacles were added to the NMPC contraints as described by Lin, Chen, and Liu, 2016, but the solution is generated offline and is only used as a reference trajectory. Carlos et al., 2020 uses NMPC to avoid dynamic obstacles online. This was possible due to the development of the RTI scheme (Diehl, Bock, and Schlöder, 2005) which trades speed for accuracy, but allows to solve complex optimization problems (Gros et al., 2020).

2.3.1 Perception-aware Control

While other approaches are used for control with perception requirements (Thomas et al., 2017), NMPC is the most common solution.

Perception-aware NMPC approach is proposed by Falanga et al., 2018, where a term is added to objective to maximize the visibility of a desired point in pinhole-model camera, together with minimizing its movement inside the image plane for further accurate detection.

Perception-induced constraints that are added directly into optimization task are proposed by Jacquet et al., 2020 and Jacquet and Franchi, 2020 to track a moving target. Penin, Giordano, and Chaumette, 2018 additionally adds the ability to prevent occlusions while performing the perception objective, but only generates a trajectory, that is later tracked by a different controller.

Chapter 3

Preliminaries

3.1 Rotation Representation

The most commonly used methods for the description of rotation (and orientation) in 3 dimensional space are euler angles, rotation matrices and quaternions.

The angular velocity is the most commonly used measure of rotation rate and is defined as a vector $\omega = (\alpha, \beta, \gamma)^\top$, where its components represent the rotation rate around each of the 3 axis of the euclidean space.

3.1.1 Euler Angles

The euler angles, often denoted and represent three rotations around axis applied one by one. The composed rotations are most commonly in order "ZYX", "XYZ" or "ZYZ", with a total of 12 combinations of the axis choice. "Z", "Y" and "X" meaning rotations around a z-, y- and x-axis in the Cartesian coordinates. In a group of euler angles, the rotations around different axis are defined as yaw, pitch and roll representing the rotations around z-, y- and x-axis, respectively (where the x-axis is pointing in the direction of the heading of the aircraft).

Although 3 independent variables define the rotation in euler angles, they have a major disadvantage due to the singularities that are said to arrise from the gimbal lock as described by Diebel, 2006. The lock can occur e.g. in the case of "ZYZ" proper euler angle with values $(0, 0, 0)$, the changes to first and third variables represent the same motion, which causes the singularity. In the case of "ZYX", the singularity appears at the values $(0, \frac{\pi}{2}, 0)$. As suggested by Diebel, 2006 a common approach to solve the problem is to change the representation whenever the system is near a singularity.

3.1.2 Rotation Matrices

Rotation matrices are a group of transformation matrices that are defined as the following set:

$$\mathcal{R} = \{R \in \mathbb{R}^{3\times 3} | R^\top R = I, |R| = 1\}, \tag{3.1}$$

where I is the identity matrix, and $|R|$ is the determinant of matrix R. A vector can be transformed with the use of rotation matrix from one coordinate system to another with $v' = R \cdot v$.

Because 3 degrees of freedom are associated with rotation, 6 constraints are required during differentiation and integration of the matrix, which can be deduced from its definition. During the computation of differentiation and integration, these constraints may not be met exactly, resulting in singularities, as suggested by Zhao and Van Wachem, 2013.

3.1.3 Unit Quaternions

Unit quaternions are becoming increasingly popular to represent rotation, in particular in robotic tasks.

A quaternion is is defined by:

$$\mathcal{Q} = q_w + q_x i + q_y j + q_z k \in \mathbb{H}, \tag{3.2}$$

where $\{q_w, q_x, q_y, q_z\} \in \mathbb{R}$ and $\{i, j, k\}$ are imaginary units defined by equations $i^2 = j^2 = k^2 = ijk = -1$ as described by Graf, 2008.

An alternative representation is in vector form:

$$q = \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} = \begin{pmatrix} q_w \\ q_v \end{pmatrix}. \tag{3.3}$$

The norm of a quaternion is defined as $\|q\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}$ and its conjugate is defined as $\bar{q} = \left(q_w, -q_v\right)^\top$.

A unit quaternion is a quaternion, for which $|q| = 1$. As noted by Zhao and Van Wachem, 2013 to represent rotation, quaternions are required to unitary.

A unitary quaternion can be decomposed as:

$$q = (q_w, q_v)^\top = \left(\cos\frac{\gamma}{2}, \sin\frac{\gamma}{2} \cdot \vec{n}\right)^\top, \qquad \|\vec{n}\| = 1, \tag{3.4}$$

where γ is the angle of rotation in radians, and $\vec{n}$ is the vector that is a base of rotation axis, axis that is perpendicular to rotation plane and is defined as one-dimensional subspace, all the points of which are left unaffected by the rotation.

Given this, the inverse of a quaternion - another quaternion representing rotation in the opposite direction, is its conjugate: $q^{-1} = \bar{q}$.

Composition of two quaternions (also called Hamilton product), that results into another quaternion that represents rotation with q_1 and then consecutive rotation with q_2 is defined as:

$$q_3 = q_2 \circ q_1 = \begin{pmatrix} q_{2w} q_{1w} - q_{2v}^\top q_{1v} \\ q_{2w} q_{1v} + q_{1w} q_{2v} + q_{2v} \times q_{1v} \end{pmatrix}, \tag{3.5}$$

where $q_{2v} \times q_{1v}$ is the cross product of 2 vectors.

A rotation, that is represented by quaternion q can be applied on a vector $v = (v_x, v_y, v_z)^\top$, by creating a quaternion with 0 scalar part: $w = (0, v_x, v_y, v_z)^\top$ and using the formula

$$w' = q \circ w \circ q^{-1}. \tag{3.6}$$

The resulting vector v' can be then be retrieved as the vector part from the resulting quaternion as $v' = w'_v$. For simplicity I will further denote the rotation of a 3d vector with quaternion as $v' = q \circ v \circ q^{-1}$.

A quaternion rotation can be converted to identical representation of rotation with rotation matrix as:

$$R(q) = \begin{pmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2. \end{pmatrix} \tag{3.7}$$

More details about quaternion algebra are presented by Graf, 2008 and Sola, 2017.

As given by Zhao and Van Wachem, 2013 quaternion derivative can be determined by:

$$\dot{q} = \frac{1}{2} \begin{pmatrix} 0 \\ \omega \end{pmatrix} \circ q, \tag{3.8}$$

where ω is the angular velocity vector and q is the quaternion itself.

There are many methods for quaternion integration, that vary depending on computational complexity and accuracy they achieve. A comparison of a large number of them is provided by Zhao and Van Wachem, 2013. The easiest is Newton method, that approximates a quaternion at a new time step $n+1$ with linearization up to first derivative and then normalizing to be a unit quaternion:

$$\hat{q}_{n+1} = q_n + \frac{1}{2} \begin{pmatrix} 0 \\ \omega_n \end{pmatrix} q_n \delta t, \tag{3.9}$$

$$q_{n+1} = \frac{\hat{q}_{n+1}}{\|\hat{q}_{n+1}\|}. \tag{3.10}$$

Because this method is not very accurate and has large numeric errors, more advanced methods, like direct multiplication and predictor-corector direct multiplication methods were created, with more details provided by Zhao and Van Wachem, 2013.

Compared to euler angles and rotation matrices, quaternions do not suffer from lack of stability, non-uniqueness and occurance of singularities, and seem to be the most attractive representation for the orientation of a body in space. With 3 degrees of freedom given by rotation, quaternions have one constraint of keeping them unitary. As non-unitary quaternions will perform scaling additionally to rotation, conservation of their unit length is required for an accurate description of orientation. Although numerical errors arise in different algorithms during the renormalization, some methods, like the predictor-corrector direct multiplication method presented in Zhao and Van Wachem, 2013 achieves much lower angle prediction errors, which makes it favourable for the representation of bodies orientation.

3.1.4 Rotation Vector

An alternative approach to integration is presented by Diebel, 2006, which relies on another orientation representation - rotation vector, that is defined as $v = n\gamma$, where γ is the rotation angle and n is the normalized vector spanning the axis of rotation.

They prove that as that as the norm of the vector v approaches to zero, its derivative

$$\lim_{\|v\| \to 0} \dot{v} = \omega. \tag{3.11}$$

Given this they define

$$v(t_i, t_{i+1}) = \int_{t_i}^{t_{i+1}} \omega dt \tag{3.12}$$

where ω is in the body frame and dependent on t. $v(t_i, t_{i+1})$ is a rotation vector that describes the change of attide over an interval of time. Then authors propose to calculate the orientation in quaternions over time with $q_{i+1} = q_v(v(t_i, t_{i+1})) \circ q_i$, where $q_v(v)$ is a function converting from rotation vectors to quaternions. Note, that q_i is defined in the world frame in this case.

Authors claim that this method is much more accurate than integrating the euler angles. This method also has the advantage of being very simple, as angular velocity can be integrated accurately with Newton or similar methods.

3.2 NMPC

Model Predictive Control (MPC) is an advanced technique for control of a system that is based on solving an Optimal Control Problem (OCP) using optimization methods. MPC controllers rely on dynamic models that simulate the systems that are being controlled.

The advantage of MPC is that it takes into account the behaviour of the system multiple time steps ahead when optimizing the controls for just the next time step. It also allows for introduction of various constraints on the system.

It has been used in various fields for controlling processes, and in recent years became popular in robotics community.

Nonlinear Model Predictive Control (NMPC) is MPC that can have a nonlinear system-description and nonlinear constraint formulations, and requires solving a nonlinear (and often non-convex) OCP.

General NMPC tracking problem can be formulated in the following way:

$$\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(x_i,u_i)-\tilde{\eta}\|^2_W + \frac{1}{2}\|\eta_N(x_N)-\tilde{\eta}_N\|^2_{W_N} \tag{3.13}$$

subject to

$$x_0 = \bar{x}_0, \tag{3.14}$$
$$x_{i+1} = F(x_i,u_i), \qquad i\in\{0,\dots,N-1\}, \tag{3.15}$$
$$g(x_i,u_i)\le 0, \qquad i\in\{0,\dots,N-1\}, \tag{3.16}$$
$$g_N(x_N)\le 0. \tag{3.17}$$

The symbols are defined as follows:

Notation	Description
$x_i \in \mathbb{R}^{n_x}$	The state of the system at time steps i - all the variables that are required to represent the system at a given time. n_x is the size of state vector.
$\bar{x}_0 \in \mathbb{R}^{n_x}$	The state of the real system at time step $i = 0$. The system is at this state at the beginning of optimization, thus this variable x_0 cannot be varied, while all future states can be optimized for.
$u_i \in \mathbb{R}^{n_u}$	The controls of the system - the variables that are inputs to the system for changing its state. n_u is the number of controls in the system.

$F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$	The function describing the dynamics of the system. The state-space system representation relies on the equation $x_{i+1} = F(x_i, u_i)$. Note that this system is discretized, as it calculates the time at each time step i. Real-world processes are continues and thus are likely better described by continues systems of from $\dot{x}(t) = F(x(t), u(t))$. However discrete approximations are simulated more easily on modern computers, and are very accurate provided that the sampling interval is small enough to react to be able to reproduce dynamical changes of the system
$\eta : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\eta}$	The output of the system. The function returns a vector of important for optimization variables.
$\tilde{\eta} \in \mathbb{R}^{n_\eta}$	The reference for the output of the system. These are parameters that can change over time and are used in the objective function of the optimization problem. The corresponding term in optimization problem is a least-squares between the output and this reference. It will be weighted by a symmetric positive-definite matrix W as: $\|\eta - \tilde{\eta}\|^2_W = (\eta - \tilde{\eta})^\top W(\eta - \tilde{\eta})$
$\eta_N : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{\eta_N}}$	The function returning output of the system at time step N. The function depends only on the state of the system x_N and not u_N. The reasoning behind this is most likely because $x_N = F(x_{N-1}, u_N)$, and there is no effect of input u_N on the states before $N+1$, so the objective function includes x_N and not u_N.
$\tilde{\eta}_N \in \mathbb{R}^{n_{\eta_N}}$	The reference for system output at time step N. The respective objective-function least-squares term is like-wise weighted by matrix W_N.

N is the horizon length - the number of time steps ahead that we want to optimize for. The optimization tries to find the best controls $u_i, i \in \{0, \ldots, N-1\}$, while all $x_i, i \in \{0, \ldots, N\}$ can be fully deduced from the values of u_i and $\bar{x}_0$.

The optimization is solved once per time step (which is often the same as the time interval between i-th and $(i+1)$-th states in the optimization task). Let me denote $x_i^k, i \in \{0, \ldots, N\}$ and $u_i^k, i \in \{0, \ldots, N-1\}$ the states and controls produced when solution of the k-th optimization problem. Once the optimization task at time step k is solved, only the controls that are for current time u_0^k are applied, while all others are ignored. After an interval of time passes, a new $(k+1)$-th optimization problem is solved and new controls will be applied. Note, that even though controls $u_i^k, i \in \{1, \ldots, N-1\}$ are not used, they are very important for the optimization problem, as they provide it with the ability to 'predict' the future state of the system and start reacting to possible changes that will happen in the future time steps. Additionally, even though $u_i^k, i \in \{1, \ldots, N-1\}$ already provide optimal feedback for the next $N-1$ time steps, performing additional optimizations are required in case the real system does not behave as predicted by the model and to respond to disturbances that effect the system.

Lots of methods can be used for solving the given problem, including nonlinear solvers and Sequential Quadratic Programming (SQP) methods. One of the main requirements for optimization of real-world tasks, especially in robotics, is its speed.

Because optimization result takes time to be computed and the system evolves in time in the meantime, the supplied command needs to be still relevant after optimization is performed. Also, the commands need to be provided at a high enough rate to the system, or otherwise delayed reactions to changes can appear.

3.3 Real Time Iteration Scheme

One of the methods created to tackle the speed requirements on NMPC for modern applications is Real Time Iteration Scheme (RTI) proposed by Diehl, Bock, and Schlöder, 2005. I will present an explanation of the main ideas of RTI here.

Let us define $L(x_i, u_i) = \frac{1}{2} \|\eta(x_i, u_i) - \tilde{\eta}\|^2_W$ and $E(x_N) = \frac{1}{2} \|\eta_N(x_N) - \tilde{\eta}_N\|$. Without the additional constraints, the optimization problem of a system is formulated as:

$$\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \sum_{i=0}^{N-1} L(x_i, u_i) + E(x_N) \tag{3.18}$$

subject to

$$\begin{aligned} x_0 &= \bar{x}_0, \\ x_{i+1} &= F(x_i, u_i), \qquad i \in \{0, \dots, N-1\}. \end{aligned} \tag{3.19}$$

Inequality constraints of form $g(x_i, u_i) \leq 0$ can be additionally introduced into the problem as in the Karush-Kuhn-Tucker conditions, which are a generalization of the Lagrange multipliers. Diehl, Bock, and Schlöder, 2005 however focus on proving the general convergence and note that inequality constraints do not pose any difficulty for algorithm's performance in practice.

RTI is a Newton-Type optimization method. To solve the problem, first Lagrange multipliers $\lambda_i, ..., \lambda_N$ are introduced to create the Lagrangian function:

$$\begin{aligned} \mathcal{L}(\lambda_i, x_i, u_i, \dots, \lambda_N, x_N, u_{N-1}) = \\ = \sum_{i=0}^{N} L(x_i, u_i) + E(x_N) + \\ + \lambda_0^\top (x_0 - \bar{x}_0) + \sum_{i=0}^{N-1} \lambda_{i+1}^\top (x_{i+1} - F(x_i, u_i)). \end{aligned} \tag{3.20}$$

Define $y = (\lambda_i, x_i, u_i, \dots, \lambda_N, x_N, u_{N-1})$ as vector of all the variables. The optimality conditions of first order (or Karush-Kuhn-Tucker conditions) are then defined as: $\nabla_y \mathcal{L}^k(y) = 0$.

To reach to such a solution, the exact Newton-Raphson method would start at an initial guess y_0 and compute a sequence of $y_1, y_2, ...$ as:

$$y_{j+l} = y_l + \Delta y_l, \tag{3.21}$$

where δy_i is a solution of the linearization of the initial problem:

$$\nabla_y \mathcal{L}(y_l) + \nabla^2_y \mathcal{L}(y_l) \Delta y_l = 0. \tag{3.22}$$

The authors of RTI scheme propose to perform only 1 iteration at each time step. Together with additional optimizations their algorithm can be summarized as:

1. Prepare the response as much as possible without the knowledge of x_0^k. Only the first component of the vector $\nabla_y \mathcal{L}(y^k)$ depends on x_0^k, while $\nabla_y^2 \mathcal{L}(y^k)$ is completely independent.

2. Perform the feedback response. Calculate $\Delta y_k = -\nabla_y^2 \mathcal{L}(y^k)^{-1} \cdot \nabla_y \mathcal{L}(y^k)$. For the calculation take the most recent x_0^k, as we need the control to be as up-to-date for the system as possible. Calculate the optimal control as $u_0^k + \Delta u_0^k$. Diehl, Bock, and Schlöder, 2005 proposes to approximate the Hessian matrix $\nabla_y^2 \mathcal{L}^k(y)$ d by a symmetric matrix to simplify calculations.

3. Project the solution y_k on the space of $y_{k+1} = P^{k+1}(y_k + \Delta y_k)$. Because at the time of $k+1$, the system would have evolved, the commands u_0^k are no longer relevant. But the value of u_1^k can be used in the next step of optimization for u_0^{k+1}. In the case the optimization intervals are the same as time between system updates inside the optimization problem, the transform from y_k to y_{k+1} needs to just shift the variables by one back. As described in Diehl, Bock, and Schlöder, 2005, the next optimization is then performed on a smaller number of variables, but additional controls, corresponding to u_N^{k+1} can be filled with some initialization value, like zeros, and the size of vector y will be the same for all time steps. Finally, finish the step - set $k = k + 1$ and continue from step 1.

Additional optimizations that are given by Diehl, Bock, and Schlöder, 2005 are leaving parts of the Hessian $\nabla_y^2 \mathcal{L}^k(y)$ pre-calculated from the previous step, and due to its block-diagonal structure, solving $\nabla_y \mathcal{L}(y_l) + \nabla_y^2 \mathcal{L}(y_l) \Delta y_l = 0$ using a Ricatti recursion proposed independently by De O. Pantoja, 1988 and Dunn and Bertsekas, 1989.

Carlos et al., 2020 propose an implementation with modern technologies, in particularly to linearize the problem, generate the RTI scheme definition using the tools of ACADOS library created by Verschueren et al., 2019 and solve it using modern QP solvers, like the the high performance interior point methods as described in Frison and Diehl, 2020. For the linearization, an approximation of Gauss-Newton Hessian approximation is calculated as described by Frison et al., 2018 and an exact gradients are calculated. Additionally, condensing and partial condensing can be used to convert the original OCP QP to a dense QP or an OCP QP with shorter horizon respectively.

The advantage of RTI compared to optimal feedback control, is that it is faster than computing exact solutions. Also, given a large shooting interval N, it is able to perform a large number of optimization steps likely giving a good estimate of optimal solution. The convergence of RTI as the number of iterations goes to infinity is proved in Diehl, Bock, and Schlöder, 2005.

Chapter 4

Control Architecture

4.1 System Model

4.1.1 Notations and Definitions

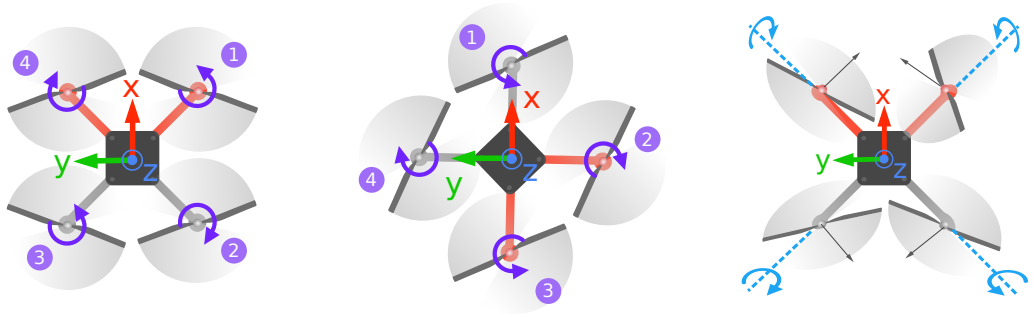
The system requires the definition of the following notation and variables:

Notation	Description
$\mathcal{W}$	The inertial (or world) frame.
$\mathcal{B}$	The body frame. The origin of the Cartesian Coordinates is located at the center of mass (CoM) of the body (UAV), and the axis of the coordinates rotate along with the UAV. (All the frames are depicted in Figure 4.2.)
$p \equiv p^{\mathcal{W}} \in \mathbb{R}^3$	The position of the CoM of the UAV in the world frame.
$q \equiv q^{\mathcal{W}} \in \mathbb{H}$	The orientation of the UAV represented as a quaternion.
$v \equiv v^{\mathcal{W}} \in \mathbb{R}^3$	The velocity of the UAV in the inertial frame.
$\omega \equiv \omega^{\mathcal{B}} \in \mathbb{R}^3$	The angular velocity of the UAV, that is in the UAV body frame to simplify system equations.
K	The number of actuators (or propellers) that are present on the UAV and will be controlled.
$\Omega = \begin{pmatrix} \Omega_1 \\ \vdots \\ \Omega_K \end{pmatrix} \in \mathbb{R}^K$	The rotor spinning rate of each of the rotors $\{1, \ldots, K\}$ that are on the UAV collected into a column vector. The rotor spinning rate represents the magnitude of angular velocity of the propeller, and is positive when the rotation is counter-clockwise and negative when it is clockwise.

We will be considering a generic aerial multirotor platform consisting of a physical body and $K \geq 4$ propellers.

The platform can be under-actuated or fully-actuated depending on the number of propellers and their orientation in space relative to the UAV body (Franchi et al., 2018).

A control system is defined to be fully-actuated if by changing the controls, it is



(A) Quadrotor in cross configuration. The rotors include numbering and their type σ_k - CW or CCW.

(B) Qudarotor in plus configuration

(C) Example of quadrotor with tilted propellers. The propellers are tilted in order to produce forces whose vectors are not aligned to give more freedom of movement.

FIGURE 4.1: Configurations of quadrotor

able to instantaneously change the direction of its acceleration (and rotational acceleration).

An under-actuated system cannot do that. An UAV has 6 degrees of freedom, where 3 define the position and 3 define the orientation of the system. A quadrotor is always underactuated, as the number of controls corresponding to 4 propellers is less than the number of degrees of freedom. Additionally, other standard multirotors (like hexarotors) are also underactuated, as they have collinear propellers, that generate forces aligned to one direction in body frame. In order to apply a force in a desired direction, such a multirotor would first need to change its orientation in space. Such an underactuation may negatively effect the interaction of the system with environment, since it is unable to exert force in an arbitrary direction.

To overcome this, fully-actuated multirotor systems have been developed, in which the rotors are slightly tilted (as the tilted rotors in Figure 4.1c), so that their thrust is not aligned in the same direction (Franchi et al., 2018).

I will use a general system model, that can simulate an under-actuated or fully-actuated UAV depending on parameter values.

4.1.2 Model Dynamics

The derivative of position over time is the linear velocity: $\dot{p} = v$. The derivative of quaternion is defined in equation 3.8 depending on angular velocity.

The derivative of velocity is acceleration $\dot{v} = a \in \mathbb{R}^3$. The acceleration of the UAV is dependent on the gravitational force and the rotation of propellers.

Each propeller rotates with spinning rate Ω_k. The position of propeller relative to the UAV body position is $p_k^{\mathcal{B}} \in \mathbb{R}^3$. $z_{p_k}^{\mathcal{B}} \in \mathbb{R}^3$ is a unit vector parallel to the rotor spinning axis. Then according to the most commonly acknowledges model, the following equations hold (Michieletto et al., 2020 or Michieletto, Ryll, and Franchi, 2018):

$$f_k = \sigma_k c_{f_k} |\Omega_k| \Omega_k z_{p_k}^{\mathcal{B}}, \tag{4.1}$$

$$\tau_k = -c_{\tau_k} |\Omega_k| \Omega_k z_{p_k}^{\mathcal{B}}, \tag{4.2}$$

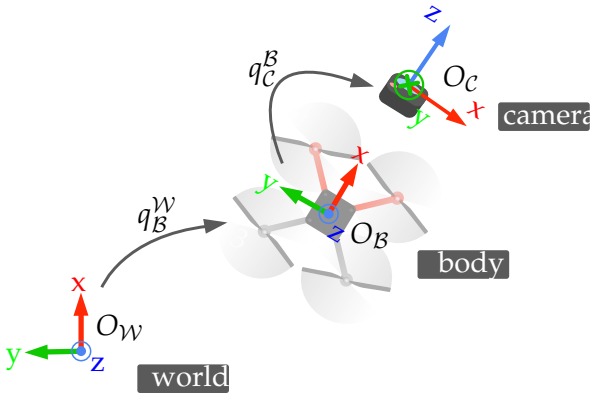


FIGURE 4.2: The three frames of world $\mathcal{W}$, body $\mathcal{B}$ and camera $\mathcal{C}$ and transitions between them.

where $f_k \in \mathbb{R}^3$ is the thrust force generated by the rotor, and $\tau_k \in \mathbb{R}^3$ is the drag moment. c_{f_k} and c_{τ_k} and $\sigma_k, k \in \{1, \ldots, K\}$ are constants that depend on various factors, like rotor shape. The value of σ_k is:

$$\sigma = \begin{cases} 1 & \text{if the propeller is of counter-clockwise (CCW) type} \\ & \text{(meaning it will generate upward force by turning CCW)} \\ -1 & \text{if the propeller is of clockwise (CW) type} \end{cases} \tag{4.3}$$

The sum of all the propeller forces creates the control force, which can also be rewritten in matrix form as:

$$f_c^{\mathcal{B}} = \sum_{k=1}^{K} f_k = \sum_{k=1}^{K} \sigma c_{f_k} z_{p_k}^{\mathcal{B}} |\Omega_k| \Omega_k = F_c \begin{pmatrix} |\Omega_1|\Omega_1 \\ \vdots \\ |\Omega_K|\Omega_K \end{pmatrix} = F_c \cdot \gamma, \tag{4.4}$$

where γ is a vector of angular rates squared multiplied by their sign.

The control torque can be written as:

$$\tau_c^{\mathcal{B}} = \sum_{k=1}^{K} (\tau_k + p_k^{\mathcal{B}} \times f_k) = \sum_{k=1}^{K} (-c_{\tau_k} z_{p_k} + c_{f_k} p_k^{\mathcal{B}} \times z_{p_k}^{\mathcal{B}}) |\Omega_k| \Omega = M_c \cdot \gamma, \tag{4.5}$$

where the $p_k \times f_k$ is the torque generated by thrust, and τ_k is torque generated by drag moment.

As a result we get two matrices $F_c, M_c \in \mathbb{R}^{3 \times K}$, that are control force input matrix and control torque input matrix respectively. Note, that because p_k and z_{p_k} that are used in the creation of these matrices are in body frame $\mathcal{B}$, the matrices will create the control force and body force in body frame, too. To convert the control force into world frame, one can use $f_c^{\mathcal{W}} = q \circ (F_c \cdot \gamma) \circ q^{-1}$.

Finally, as described by Michieletto et al., 2020, ignoring second order-effects we end up with the system dynamics that is described by the following equations:

$$\begin{aligned}
\dot{p} &= v, \\
\dot{q} &= \frac{1}{2} q \circ \begin{pmatrix} 0 \\ \omega \end{pmatrix}, \\
\dot{v} &= -g e_3 + m^{-1} q \circ (F_c \cdot \gamma) \circ q^{-1}, \\
\dot{\omega} &= J^{-1}(-\omega \times J\omega + M_c \cdot \gamma),
\end{aligned} \tag{4.6}$$

where m is the mass of the body, g is the gravitational constant, e_3 is the third column of the identity matrix I_3. The positive definite constant matrix $J \in \mathbb{R}^{3\times3}$ is the vehicle inertial matrix in the body frame $\mathcal{B}$.

Note that all rotation-related calculations are performed with quaternions to avoid any possible computational errors while converting it to an alternative framework for representing rotation.

To ensure the continuity of control action (and thus Ω), the derivative of angular rate $\dot{\Omega}$ of the propellers is added to the system. Its relation to γ is $\dot{\gamma} = 2|\Omega|\dot{\Omega}$.

The state of the system is defined as $x = (p, q, v, \omega, \gamma)^\top \in \mathbb{R}^{9+K} \times \mathcal{H}$. The control inputs are $u = \dot{\gamma} \in \mathbb{R}^K$.

The whole system is described by $\dot{x} = F(x, u)$, where F is deduced from equation 4.6 and

$$\dot{\gamma} = u. \tag{4.7}$$

Together with an appropriate objective function and possibly additional constraints, the whole system can be used to create an NMPC formulation as in equations 3.13 and 3.14-3.17.

4.1.3 Dynamics Constraints

Bicego et al., 2020 propose to add constraints on the angular rate of rotors Ω, in particular a lower bound $\underline{\Omega}$ and an upper bound $\bar{\Omega}$. The upper bound is based on the maximal current limitation of the motors used, while the lower bound is also based on the capabilities of the motors and the capability to estimate the angular velocity with a required precision.

Additionally they the lower bound of angular acceleration $\dot{\underline{\Omega}}$ and a corresponding upper bound $\bar{\dot{\Omega}}$ are introduced. They are based on practically measuring the physical limits of the rotors to accelerate and depend on the value of angular velocity.

Because it does not make sense to change the direction of angular rotation midflight (especially based on the fact, that stopping and starting motors takes a reasonable amount of time in practice), the constraints also assume that $\Omega > 0$.

In order to get realistic physical values and maintain certain safety limits, I will also introduce into the problem formulation upper bounds on the velocity $\bar{v}$ and angular velocity $\bar{\omega}$ of the UAV.

4.2 Collision Avoidance

The collision avoidance task is to avoid from coming to close to L obstacles. The position of each of the obstacles is $p^{\text{obs}_l}, l \in \{1, \ldots, L\}$ and is parametrized over time: $p_i^{\text{obs}_l}$, where i is the number of time increments in NMPC scheme that we want to look ahead. For dynamic obstacles, the future position needs to be predicted

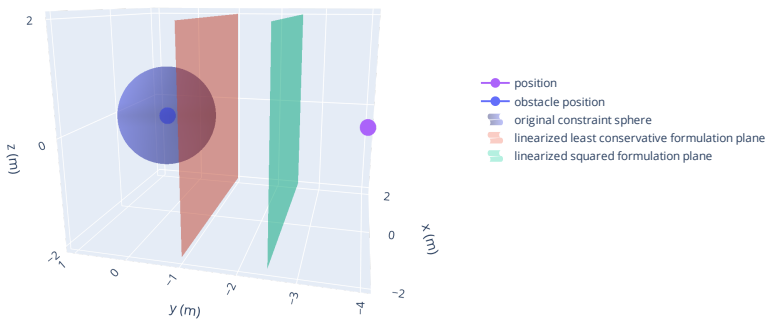


FIGURE 4.3: An example of obstacle constraints. The sphere represents how the original formulation splits the space. The planes represent how the constraints appear after linearization. As can be seen, the squared constraint appears to be further from obstacle.

over time. Falling obstacles would require estimating their position and velocity to predict future trajectory, while a Kalman filter can be used to predict the position of other UAVs, but researching these methods will not be the focus of this work.

The obstacles are modelled as points in 3D the space, with which a minimal distance $d_{\min}$ needs to be maintained in order to be safe. This distance should consider the size of the robot, the size of obstacle and a safety margin due to possible inaccuracies in position estimation or tracking.

collision avoidance is added to the system as a constraint:

$$\left\| p_i - p_i^{\text{obs}_l} \right\| > d_{\min}, \qquad l \in \{1, \dots, L\},\ i \in \{1, \dots, N\}. \tag{4.8}$$

The constraint needs to be added separately for each of the obstacles and each of the time-steps.

4.2.1 Constraint Linearization

As described in Section 3.3, the RTI optimization scheme performs a linearization of the constraints in order to convert the optimization problem to QP problem.

Reformulating the constraint from inequality 3.16 as

$$g(p) \leq 0, \tag{4.9}$$

where $p = (x, u)$ are all the parameters of constraint. I will formulate the linearized version of the constraint as:

$$g_{\text{lin}}(p) = g(p_0) + \nabla_p g(p_0) \cdot (p - p_0), \tag{4.10}$$

where p_0 is the point of linearization.

The constraint in inequality 4.8 is a nonlinear function, thus performs differently than its linearized version. It has been shown by Carlos et al., 2020 that the formulation as in inequality 4.8 is the least conservative linearized formulation. This formulation will restrict the solution space the least when linearized compared to others.

An alternative formulation of the constraint can be achieved by squaring both terms of the inequality 4.8:

$$\left\| p_i - p_i^{\text{obs}_l} \right\|^2 > d_{\min}^2, \qquad l \in \{1, \dots, L\},\ i \in \{1, \dots, N\}. \tag{4.11}$$

It can be proven that formulation 4.8 constraint splits the space into 2 subspaces (feasible and unfeasible) by a plane, whose normal vector is collinear to the vector $(p_0 - p^{\text{obs}})$ and which at the distance $d_{\min}$ from the obstacle. This means that the plane of constraint touches the sphere of $\|p - p^{\text{obs}}\| = d_{\min}$. Note that a plane is created for each of the obstacles and each of the time steps $i \in \{1, \dots, N\}$ but this is omitted here.

On the other hand, the formulation as in inequality 4.11 creates a plane that has the same normal vector, but is at a distance

$$\frac{d_{\min} + \|p_0 - p^{\text{obs}}\|}{2}.$$

In the case that $\|p_0 - p^{\text{obs}}\| > d_{\min}$ the plane is at a greater distance than the plane in formulation 4.8 and thus removing a larger subset from the feasible solutions. A visualization of the difference between formulations is presented in Figure 4.3. For more details, refer to Carlos et al., 2020.

Despite being undifferentiated at one point due to the modulus operation, the constraint formulation as in inequality 4.8 will thus be preferred due to the better resemblance of its linearized version.

4.2.2 Soft Constraints

A hard constraint is of form $g(x_i, u_i) \leq 0$ like the inequality 3.16 and 3.17. There is no possibility to violate such constraint, as it is restricting the feasible region.

On the other hand, the idea behind a soft constraint is to add an option of constraint violation. However, the violation of the constraint is highly penalized inside the objective function, instead. A soft constraint is created by first adding a slack variable to the control variables and then reformulating the hard constraint:

$$g(x_i, u_i) - s_i \leq 0, \qquad i \in \{1, \dots, N-1\}, \tag{4.12}$$

$$s_i > 0, \qquad i \in \{1, \dots, N-1\}. \tag{4.13}$$

The s_i are added on each time step $i \in \{1, \dots, N\}$ and will require to be a non-zero value in case the original constraint 3.16 is violated.

Additional terms will be added to the objective function 3.13 for each of the slack variables:

$$\sum_{i=1}^{N-1} w_{\text{obs}} \cdot s_i^2, \tag{4.14}$$

where w_{obs} is a constant that defines the amount of penalization on the slack variable. Note that this additional term can be incorporated to the $\|\eta(x_i, u_i) - \tilde{\eta}_i\|_W^2$

expression. If u_i contains the s_i, and because term 4.14 is a squared norm, the term can be added by changing $\eta(x_i, u_i)$, $\tilde{\eta}$ and W accordingly.

The hard constraint could be also formulated for final state as in inequality 3.17, but soft constraint is only available for intermediate states $i \in \{1, \ldots, N-1\}$ because it requires a slack variable which is a control variable and is not available for final states of NMPC. However, this is a minor disadvantage and its effect is neglected in case the shooting interval N is sufficiently high.

While one ideally wants to respect all the constraints, it is not essentially possible in real-world scenarios. For example, in case a UAV is travelling at a high speed and an obstacle is detected in front of it, there might be no feasible maneuver to avoid violating minimal distance constraint. In this case, a reasonable behaviour would be to try to minimize the depth of the overstep into the unfeasible set and the time of this violation. This is exactly what a soft constraint is achieving. On the other hand, a hard constraint would not be able to find any solution which results in not being able to control the UAV at all.

While hitting an obstacle is very dangerous for a UAV, but minimizing the impact may result in the UAV recovering from it and successfully continuing on its trajectory. Thus I will add the obstacle violation constraint as a soft constraint:

$$\begin{aligned} \left\| p_i - p_i^{\text{obs}_l} \right\|_2 + s_i^{\text{obs}_l} &> d_{\min}, & l \in \{1, \ldots, L\};\ i \in \{1, \ldots, N\}, \\ s_i^{\text{obs}_l} &> 0, & l \in \{1, \ldots, L\};\ i \in \{1, \ldots, N\}. \end{aligned} \tag{4.15}$$

The weight for the penalization of slack variables that correspond to obstacles is inside the W and needs to have a high value comparing to all the other weights, because respecting the constraint is more important than tracking other references.

Note, that not for all constraints, performance is improved by making them soft. Unfeasibility can happen to constraints, that are task-related and whose current status may depend on a number of time steps before. On the other hand, constraints on the controls and states that can be immediately changed that are based on system dynamics can always be respected. Thus I will not reformulate the constraints on Ω and $\dot{\Omega}$ which are described in section 4.1.3 as soft constraints.

4.3 Perception

4.3.1 Camera Definition

We will define notations, parameters and variables that are related to camera:

Notation	Description
$\mathcal{C}$	The camera frame $\mathcal{C}$. The camera is considered to be of pinhole model. The last vector of basis (or z-axis) is pointing in the direction of the center of camera view. The x-axis corresponds to horizontal pixels in the image and y-axis corresponds to vertical pixels from top to bottom. The camera frame can be seen in Figure 4.2.
$O_{\mathcal{C}}^{\mathcal{B}}$	The camera frame origin, that is also the position of the camera.

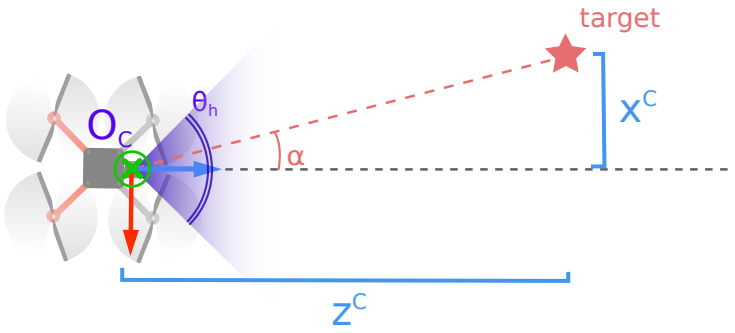


FIGURE 4.4: Visulalization of the FoV horizontal constraint. The camera frame $\mathcal{C}$ has origin at $O_\mathcal{C}$ and axes present in the picture. Tangent of horizontal angle the target α is relative to camera axis is equal to $\tan(\alpha) = \frac{x^\mathcal{C}}{z^\mathcal{C}}$. The angle α needs to be less than half of horizontal FoV $\frac{\theta_h}{2}$ and thus $|\tan(\alpha)| < \tan\left(\frac{\theta_h}{2}\right)$
.

$q^\mathcal{B}_\mathcal{C}$	The quaternion that represents rotation from camera frame $\mathcal{C}$ to the body frame $\mathcal{B}$. The rotation matrix $R(q^\mathcal{B}_\mathcal{C})$ (equation 3.7) will perform transformation from the camera to body frames.
θ_h, θ_v	The horizontal and vertical field of view of the camera respectively; in radians. This value should be $< \pi$, which is the case for most cameras.
$p_{\text{tgt}} = (x_{\text{tgt}}, y_{\text{tgt}}, z_{\text{tgt}})^\top$	The position of the target as a vector and its x, y, z components, respectively. To get the target in camera view, the equation 3.6 is used: $p^\mathcal{B}_{\text{tgt}} = q^\mathcal{B}_\mathcal{C} \circ p^\mathcal{C}_{\text{tgt}} \circ (q^\mathcal{B}_\mathcal{C})^{-1} + O^\mathcal{B}_\mathcal{C}$.

4.3.2 Perception-Based Constraints

The main perception-based task that I want to solve is keeping the target UAV in the field of view (FoV) of camera.

The constraints propsed by Jacquet and Franchi, 2020 to achieve this are of this form:

$$\left|\frac{x^\mathcal{C}_{\text{tgt}}}{z^\mathcal{C}_{\text{tgt}}}\right| \leq \tan\left(\frac{\theta_h}{2}\right), \tag{4.16}$$

$$\left|\frac{y^\mathcal{C}_{\text{tgt}}}{z^\mathcal{C}_{\text{tgt}}}\right| \leq \tan\left(\frac{\theta_v}{2}\right). \tag{4.17}$$

They are based on the fact, that tangent of angle is also a quotient of opposite side to adjacent side in a right triangle, which is valid in this case because the basis of the camera space is orthogonal. The first inequality is to keep the target in the horizontal FoV, while the second is for vertical FoV.

We will now focus in more detail on constraint in inequality 4.16. It can reformulated as

$$-\tan\left(\frac{\theta_h}{2}\right) \leq \frac{x^{\mathcal{C}}_{\text{tgt}}}{z^{\mathcal{C}}_{\text{tgt}}} \leq \tan\left(\frac{\theta_h}{2}\right). \tag{4.18}$$

The value of $z^{\mathcal{C}}_{\text{tgt}}$ is positive in the case that the object is in front of the camera (or the plane whose basis are x- and y-axis), and is negative if it is behind it. Then, the value of $z^{\mathcal{C}}_{\text{tgt}}$ should be always positive, which is not considered by inequlity 4.18. However, in practice if θ_h is not close to the value of π, the constraint will keep the angle from becoming greater than $\frac{\pi}{2}$ or lower than $-\frac{\pi}{2}$ and thus $z^{\mathcal{C}}_{\text{tgt}}$ from becoming negative.

The fact, that the object will be in front of the camera, allows to rewrite the inequality 4.18 as a system

$$\begin{cases} -\tan\left(\frac{\theta_h}{2}\right) z^{\mathcal{C}}_{\text{tgt}} - x^{\mathcal{C}}_{\text{tgt}} \leq 0 \\ -\tan\left(\frac{\theta_h}{2}\right) z^{\mathcal{C}}_{\text{tgt}} + x^{\mathcal{C}}_{\text{tgt}} \leq 0 \end{cases}. \tag{4.19}$$

This reformulation has the advantage, that in case of negative $z^{\mathcal{C}}_{\text{tgt}} < 0$, the first term of both equations becomes positive and there is no value of $x^{\mathcal{C}}_{\text{tgt}}$ that can be a solution of the system. This means that the constraint really has the correct feasible set for FoV θ_h compared to formulation 4.16.

Similarly to as described in section 4.2.2, the camera objective might be violated in certain cases and thus it will be reformulated as a soft constraint. Even though target-tracking is our main objective, we do not want to stop controlling the UAV after losing it, as this will likely result in a fall or collision. Additionally, we might recover back to tracking the target, if our prediction of its position is reliable enough.

The soft constraint from inequality 4.19 is:

$$\begin{cases} -\tan\left(\frac{\theta_h}{2}\right) z^{\mathcal{C}}_{\text{tgt}} - x^{\mathcal{C}}_{\text{tgt}} - s^{\text{tgt}_h} \leq 0 \\ -\tan\left(\frac{\theta_h}{2}\right) z^{\mathcal{C}}_{\text{tgt}} + x^{\mathcal{C}}_{\text{tgt}} - s^{\text{tgt}_h} \leq 0 \\ s^{\text{tgt}_h} > 0 \end{cases}. \tag{4.20}$$

Only one slack variable is needed because there is no possibility for both inequalities in 4.19 to be unfulfilled.

The additional term that will be added to objective, like in formulation 4.14 is the weighted square of the slack variable

$$w_{\text{tgt}_h} \cdot \left(s^{\text{tgt}_h}\right)^2. \tag{4.21}$$

Note that all the above formulations, including constraints 4.20 and objective term 4.21, need to applied to each of the time steps $i \in \{1, \ldots, N\}$ in the NMPC optimization task formulation.

All the same reasoning can be applied to the vertical FoV constraint 4.17 to reformulate it with slack variable s^{tgt_v} and weight parameter w_{tgt_v}.

4.3.3 Linearization of Perception-based Constraints

The left inequality parts in system 4.20 are linear w.r.t. the variable p. This means that the linearization w.r.t. position will result in the same exact constraint formulation.

The linearity is verified after transforming the equation to world frame $\mathcal{W}$, as that is where the optimized variable of UAV position is, and where variable p_{tgt} is independent from all the optimized variables (state of the multirotor). Note that rotation using quaternion does not affect the linearity of position, neither does addition of a constant, e.g., camera position.

The two planes that divides the three-dimensional space of positions into 1 feasible and 3 unfeasible sets both pass through the target and are at the angle $tan(\theta_h)$ w.r.t. the vector between multirotor and its target. This is the case for constraints 4.16, 4.19 and the linearization of 4.19. On the other hand, linearization of constraint 4.16 produces planes that are at different angle and not always contain the target position.

The gradient of the constraint w.r.t. the orientation and its effect on the is not studied in detail, but because the constraint does not include division, the linearization is likely to be more accurate.

An additional problem that arises in formulation 4.16 is that when it is linearized, the modulus operation is replaced as a non-linear operation and in case the inner term $\frac{x^{\mathcal{C}}_{\text{tgt}}}{z^{\mathcal{C}}_{\text{tgt}}}$ is of different sign than at the point of linearization, it can reach values that are larger than $\tan(\frac{\theta_h}{2})$ in magnitude, as the value is not bounded from both sides like in inequality 4.18. This might be especially a problem when the target is nearby and values of $z^{\mathcal{C}}_{\text{tgt}}$ and $x^{\mathcal{C}}_{\text{tgt}}$ are small.

4.3.4 Camera Centering Objective

Perception based-constraints are supposed to keep the target UAV in the view of camera. However, the NMPC formulation is predicting only until N time steps into the future, and there is still a possibility of the target moving from the FoV further in the future. This is especially relevant because the change of orientation in practice takes more time and requires more variations in rotors speed than a change of position.

Moreover, this is possible because the constraint does not limit where the target needs to be in the image plane. The target can move to the edge of the camera image, and make an unpredicted motion out of the view which results in losing track of the target. To tackle this problem, I will additionally add a term to the objective to try keeping the target in the center of the camera.

Similar to Penin, Giordano, and Chaumette, 2018 we will define β_{tgt} as the angle between the vector aligned with camera's z-axis and the vector of $p^{\mathcal{C}}_{\text{tgt}}$.

Because cosine is equal to the quotient of adjacent side and the hypotenuse of a right triangle, this equation holds:

$$\cos(\beta_{\text{tgt}}) = \frac{z^{\mathcal{C}}_{\text{tgt}}}{\left\| p^{\mathcal{C}}_{\text{tgt}} \right\|}. \tag{4.22}$$

In order to achieve robust tracking, the angle β_{tgt} should preferrably be close to zero, and $\cos \beta_{\text{tgt}}$ should be close to 1.

Thus the term, that will be added to objective will be of form

$$w_{\text{tgt}} \left\| \cos(\beta_{\text{tgt}}) - 1 \right\|^2, \tag{4.23}$$

that should be summed for all the $i \in \{1, \ldots, N\}$. The w_{tgt} is the corresponding weight parameter that is a component of the W matrix. This can be introduced into

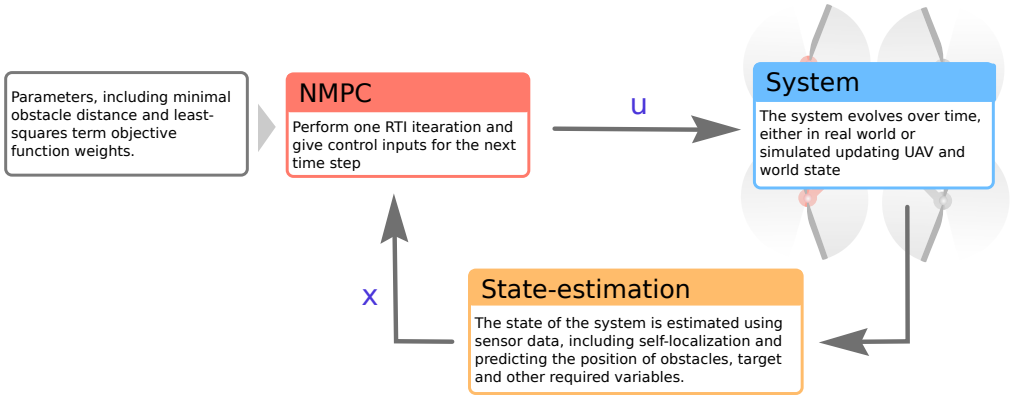


FIGURE 4.5: The schematics of system with NMPC control, that produces control vector u based on the current state of the sytem, that can be estimated or directly produced by simulation.

objective function 3.13 by setting the appropriate component of $\tilde{\eta}_i$ to the value of 1 and setting the correponding component of η_i to the expression $\cos(\beta_{\text{tgt}})$.

Additionally, the derivative of cosine can be added to the objective in order to minimize the movement of target in the image plane thus likely reducing the noise in the view and resulting in a better prediction of position. The term is formulated in the following way:

$$w_{\text{tgt-dot}} \left\| \dot{\cos}(\beta_{\text{tgt}}) \right\|^2 . \tag{4.24}$$

4.3.5 Target-Following Objective

In case of a required following that is not enforced by a planning module with a trajectory, an additional objective to the NMPC formulation is added.

The distance to the target will be defined as:

$$d_{\text{tgt}} = \left\| p^{\mathcal{B}}_{\text{tgt}} \right\| . \tag{4.25}$$

The additional term to the objective will be of form

$$w_{\text{tgt-dist}} \left\| d_{\text{tgt}} - \tilde{d}_{\text{tgt}} \right\|^2 , \tag{4.26}$$

and it encourages the optimization to minimize the difference between distance and desired distance.

To avoid collisions, target is also added to the obstacles that need to be avoided. I will not describe the formulation, as it is the same as in inequality 4.15.

The desired distance to target $\tilde{d}_{\text{tgt}}$ can be zero, which would result in trying to keep as close as possible, while avoiding collisions. However, for safety reasons I will set it to positive value (larger than minimal obstacle distance $d_{\min}$).

4.4 NMPC Formulation

We will add tracking of position p, velocity v and acceleration $a = \dot{v}$ (from equation 4.6) to the objective function. This may be used in a case a separate planner is

present, that orders UAV to perform other tasks while tracking a neighbouring aerial vehicle.

The weights that will be added to the least squares term in the objective function are w_p, w_v and w_a for position, velocity and acceleration, respectively.

As a result the system output η and its reference $\tilde{\eta}$ will be of form

$$\eta_i = \begin{pmatrix} p \\ v \\ a \\ d_{\text{tgt}} \\ \cos(\beta_{\text{tgt}}) \\ \dot{\cos}(\beta_{\text{tgt}}) \\ s^{\text{tgt}_h} \\ s^{\text{tgt}_v} \\ s^{\text{obs}_1} \\ \vdots \\ s^{\text{obs}_L} \end{pmatrix}, \qquad \tilde{\eta}_i = \begin{pmatrix} \tilde{p} \\ \tilde{v} \\ \tilde{a} \\ \tilde{d}_{\text{tgt}} \\ \cos\tilde{\beta}_{\text{tgt}} \\ \dot{\cos}(\tilde{\beta}_{\text{tgt}}) \\ \tilde{s}^{\text{tgt}_h} \\ \tilde{s}^{\text{tgt}_v} \\ \tilde{s}^{\text{obs}_1} \\ \vdots \\ \tilde{s}^{\text{obs}_L} \end{pmatrix} = \begin{pmatrix} \tilde{p}_i \\ \tilde{v}_i \\ \tilde{a}_i \\ \tilde{d}_{\text{tgt}} \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{4.27}$$

The final state the system output (which is independent of controls) will be

$$\eta_i = \begin{pmatrix} p \\ v \\ a \\ d_{\text{tgt}} \\ \cos(\beta_{\text{tgt}}) \\ \dot{\cos}(\beta_{\text{tgt}}) \end{pmatrix}, \qquad \tilde{\eta}_i = \begin{pmatrix} \tilde{p} \\ \tilde{v} \\ \tilde{a} \\ \tilde{d}_{\text{tgt}} \\ \cos\tilde{\beta}_{\text{tgt}} \\ \dot{\cos}(\tilde{\beta}_{\text{tgt}}) \end{pmatrix} = \begin{pmatrix} \tilde{p}_i \\ \tilde{v}_i \\ \tilde{a}_i \\ \tilde{d}_{\text{tgt}} \\ 1 \\ 0 \end{pmatrix}. \tag{4.28}$$

The weights of the least squares terms will be accordingly:

$$W = \text{diag}\begin{pmatrix} w_p \\ w_v \\ w_a \\ w_{\text{tgt-dist}} \\ w_{\text{tgt}} \\ w_{\text{tgt-dot}} \\ w_{\text{tgt}_h} \\ w_{\text{tgt}_v} \\ w_{\text{obs}} \\ \vdots \\ w_{\text{obs}} \end{pmatrix}, \qquad W_N = \text{diag}\begin{pmatrix} w_p \\ w_v \\ w_a \\ w_{\text{tgt-dist}} \\ w_{\text{tgt}} \\ w_{\text{tgt-dot}} \end{pmatrix}. \tag{4.29}$$

The whole NMPC formulation is deduced from formulas 3.13-3.17, where the system is substituted from 4.6, 4.7, the perception-based constraints are formulated as 4.20, collision-avoidance constraints are formulated in 4.15 and output, its reference and least-square term weights are formulated in this section.

A schematics of how NMPC control feedback loop can be seen in Figure 4.5

Chapter 5

Simulation Results

5.1 Setup

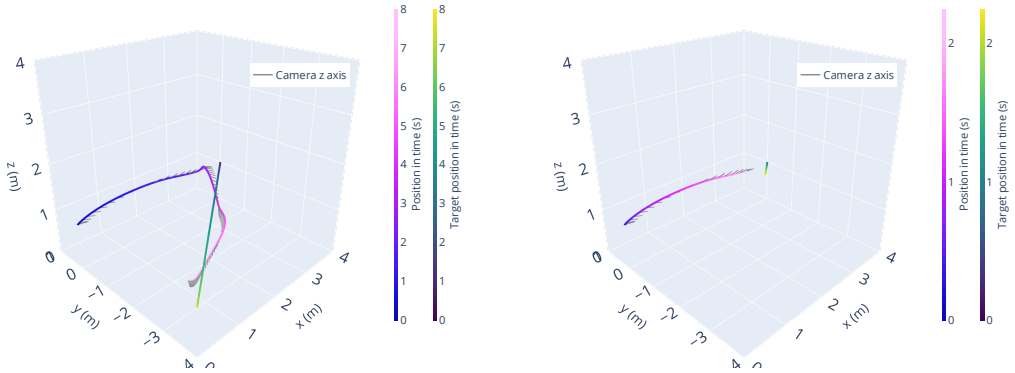
For numerical simulations I used a quadrotor in a cross configuration (cross configutation can be seen in Figure 4.1a). All the parameters are summarised in the table below and are the same as used in simulations by Bicego et al., 2020.

Notation	Value
m	$1.042kg$
g	$9.81\frac{m}{s^2}$
K (num propellers)	4
c_f	$5.95e-4\frac{N}{Hz^2}$
c_τ	$1e-5\frac{N}{Hz^2}$
σ	$\begin{pmatrix}1 & -1 & 1 & -1\end{pmatrix}^\top$
d (arm length)	$0.23m$
$\begin{pmatrix}p_1^\mathcal{B} & p_2^\mathcal{B} & p_3^\mathcal{B} & p_4^\mathcal{B}\end{pmatrix}$	$\begin{pmatrix}1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 0 & 0 & 0 & 0\end{pmatrix}\cdot\frac{d}{\sqrt{2}}$
$\begin{pmatrix}z_{p_1}^\mathcal{B} & z_{p_2}^\mathcal{B} & z_{p_3}^\mathcal{B} & z_{p_4}^\mathcal{B}\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 1\end{pmatrix}$
J	$\text{diag}\begin{pmatrix}0.015 & 0.015 & 0.070\end{pmatrix}^\top kgm^2$

The simulation is based on the formulations in 4.1 section and is run at 200Hz. The integration is performed with ERK4 (Explicit Runge-Kutta 4) method. The quaternion is integrated as described in 3.1.4 section. It is implemented with the MATLAB Simulink programming environment.

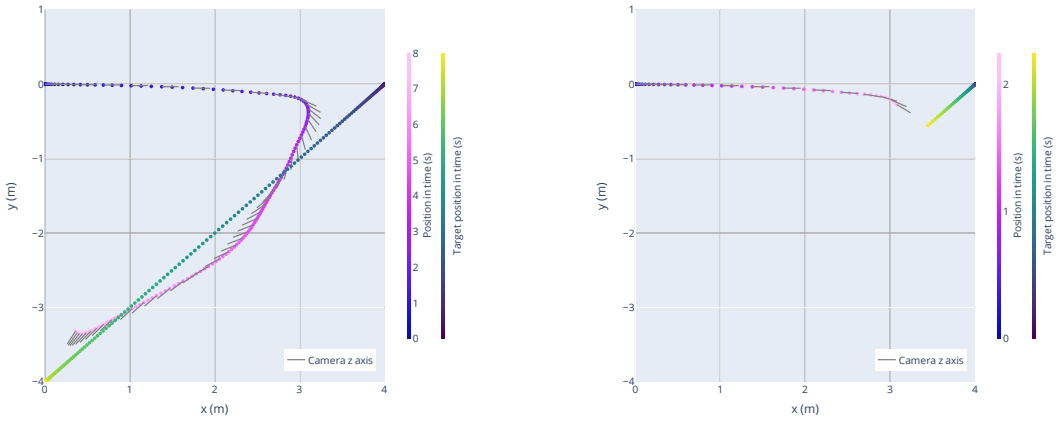
The NMPC problem is formulated with CASADI (Andersson et al., 2019) optimization tool which is supplied to the MATMPC (Chen et al., 2019) framework. The RTI solution is computed using a QP solver from the QPOASES (Ferreau et al., 2014) library with full condensing. The integrator is ERK4.

The NMPC is running with $N = 50$ and $0.015s$ time step which results in a time horizon of 0.75 seconds.



(A) Linear FoV constraint (B) Fraction FoV constraint

FIGURE 5.1: 3D trajectory of UAV (pink) and of its target (green) varied in time in the follower scenario. The camera direction is denoted as grey lines along the trajectory. The target is moving from $(4, 0, 1)^\top$ to $(0, -4, 1)^\top$, while the UAV starts its movement from $(0, 0, 1)^\top$ facing its target.



(A) Linear FoV constraint (B) Fraction FoV constraint

FIGURE 5.2: 2D trajectory of UAV and target in the follower scenario. The camera direction is denoted as grey lines along the trajectory (It is a vector aligned with camera z-axis).

(A) Linear FoV constraint (B) Fraction FoV constraint

FIGURE 5.3: The value of the slack variables in the follower scenario

5.2 Follower Scenario

The follower scenario requires following another UAV.

For simulation, I will use the camera position $O_C^{\mathcal{B}} = (0.1, 0, 0)^\top$. The rotation from camera to the body frame will be the same as a rotation by euler angles of form "XYZ" by $(0, \frac{\pi}{2}, -\frac{\pi}{2})^\top$. This is a camera that is facing in the direction of x-axis of the body frame. The rotation represented by quaternion will be $q_C^{\mathcal{B}} = (0.5, -0.5, 0.5, -0.5)^\top$.

The parameters are summarized in the following table:

θ_h	θ_v	$d_{\min}$	$\bar{v}$	$\bar{\omega}$	$\underline{\Omega}$	$\bar{\Omega}$	$\dot{\underline{\Omega}}$	$\dot{\bar{\Omega}}$
$\frac{\pi}{2}$	$\frac{\pi}{2}$	0.5	10	1	16	110	-180	180

Additionally, the distance $d_{\text{tgt}} = 1$.

The weight matrix parameters used are:

w_p	w_v	w_a	$w_{\text{tgt-dist}}$	w_{tgt}	$w_{\text{tgt-dot}}$	w_{tgt_h}	w_{tgt_v}	w_{obs}
0	0.1	0.1	2	10	10	10^3	10^3	10^4

Because no trajectory is supplied, the $\tilde{v} = 0$ and $\tilde{a} = 0$ to reduce the amount of motion of the UAV. The UAV also starts in a hovering state, so that there is no need to take-off or counteract the gravity to stop its fall.

I compared the formulation of constraint 4.18 with an addition of a slack variable to the formulation of constraint 4.20. I will call the first constraint as fraction FoV constraint and the second as linear FoV constraint corresponding to the nature of their formulation with respect to UAV position.

5.2.1 Perception Constraints Comparison

The experiment is visualized in Figures 5.1 and 5.2, where the target is performing a rapid movement and the UAV is supposed to track while also moving towards it. The NMPC with Linear FoV constraint formulation of the constraint performed a fast maneuver, first moving to the initial position of the target and then turning right to continue following it. At 2 seconds the constraints were broken as can be seen in Figure 5.3. This happened due to the requirement to continue following and make a rapid turn to keep the UAV in FoV at the same time. Nevertheless, the target was restored in FoV and followed successfully.

The NMPC with Fraction FoV constraint formulation failed during the RTI scheme iteration. This was likely caused by a division by zero (or close to zero values), which happens when the camera is close to the target. As NMPC predicts a few steps ahead, the camera could be very close to target in any of them - in this case up to 0.75s. Far ahead states would be further improved by the RTI scheme before reaching them, and the collision could be avoided, but optimization terminated.

As can be seen in Figure 5.3, the slack variable values also grew very fast in the case of fraction FoV constraint. This might be another disadvantage as the tangent grows unlinearly and is infinite at $\frac{\pi}{2}$, which results in uneven treatment of violation depending on the angle.

Due to this and other advantages presented in section 4.3.3, we will use linear FoV constraint in further simulations.

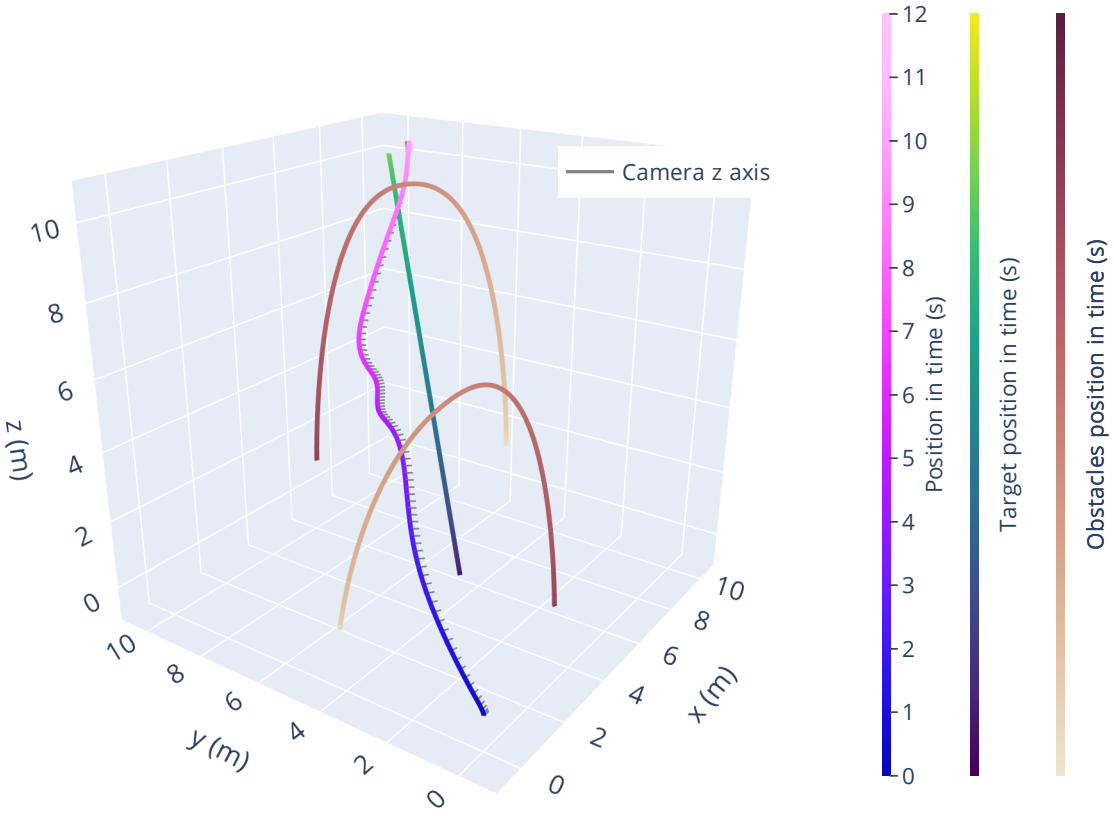


FIGURE 5.4: Following a target while avoiding obstacles. The target starts at $(2,2,0)^\top$ and UAV starts at $(0,0,0)^\top$. 2 Obstacles appear on the way, which are successfully avoided. Simulation takes 12 seconds. Target goes into hovering state at 10 seconds.

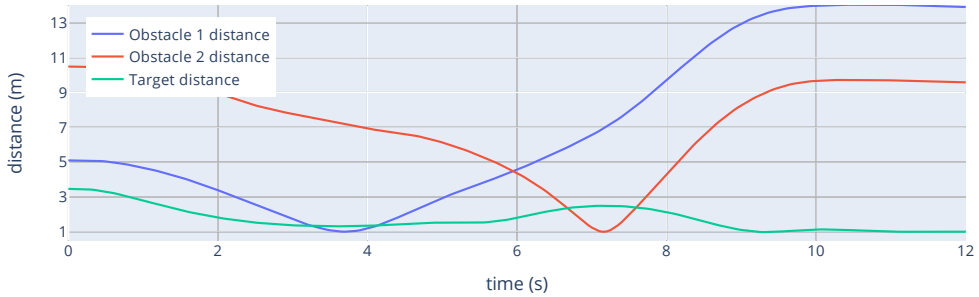
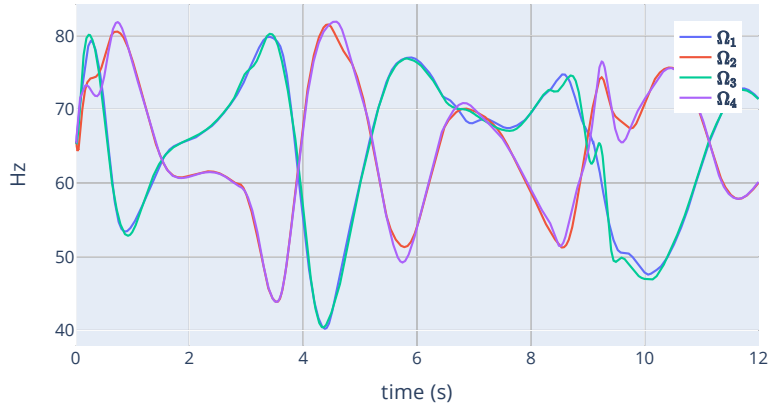
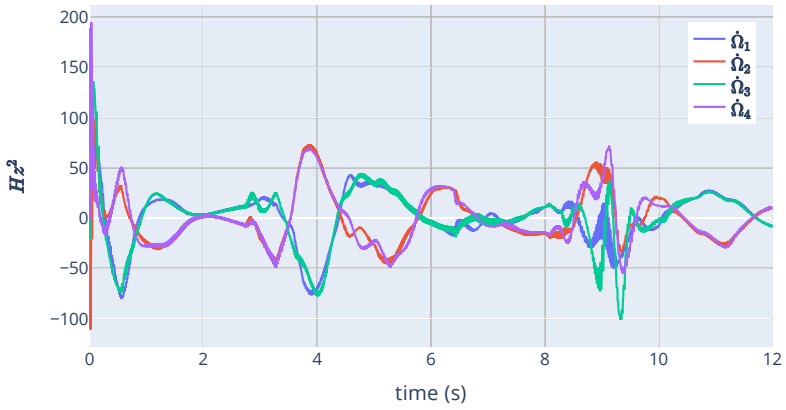


FIGURE 5.5: The distances to obstacles and target in the following with obstacles scenario



(A) The values of Ω_k, with $k \in \{1, 2, 3, 4\}$



(B) The values of $\dot{\Omega}_k$, with $k \in \{1, 2, 3, 4\}$

FIGURE 5.6: The values of rotors angular velocity Ω and its derivative $\dot{\Omega}$ when following a target while avoiding obstacles

5.2.2 Following with Obstacle Avoidance

The experiment has the same parameters as the previous one, except for the minimal distance to obstacle $d_{\min} = 1$. The number of obstacles $L = 2$.

The simulation can be seen in Figure 5.4. Target was successfully followed while keeping it in the field of view and avoiding obstacles. The distances to obstacles and target during the whole simulation are presented in Figure 5.5. The target stopped at the end point and UAV started hovering around it which is the expected behaviour.

The values of Ω and $\dot{\Omega}$ are presented in Figure 5.6. They remain in reasonably low values. Also, rotors that are located diagonally to each other seem to have similar values. This is likely because they contribute to changing the yaw of the UAV in the same way, and changing roll and pitch is physically easier than changing yaw.

5.3 Trajectory-Tracking Scenario

The second scenario that can often appear in practice is trajectory-tracking. The trajectory can be provided by higher-level module, like planner, which is aware of task specifics to a greater extent. The trajectory may be be related to the target, or completely independent.

To adjust for this scenario, the objective term that is encouraging movement to target is neglected. Also, position, velocity and acceleration weights are changed accordingly to their importance in this task. The parameters of the whole weight matrix are:

w_p	w_v	w_a	$w_{\text{tgt-dist}}$	w_{tgt}	$w_{\text{tgt-dot}}$	w_{tgt_h}	w_{tgt_v}	w_{obs}
1	0.1	0.005	0	10	1	10^3	10^3	10^4

All other system parameters are not changed.

In this case we will simulate the scenario, where 2 UAVs are performing different tasks when their trajectories overlap. The desired minimal distance to obstacles is $d_{\min} = 1$. The other UAV will be represented as perception target. The target needs to stay in the FoV of camera for relative localization. This requirement may appear due to low accuracy of location given by other sensors or communication. The objective in this case is to continue following given trajectory, while also avoiding collisions and maintaining accurate relative localization.

An example of this scenario is presented in Figure 5.7. The controlled UAV moves from $(0, 0, 0)^\top$ to $(4, 2, 0)^\top$. However the minimal distance constraint is violated near the point $(2, 2, 0)^\top$ of the proposed trajectory. Because of this, NMPC orders the UAV to perform a maneuver in direction of positive y-axis, which prevents the collision. The trajectory-following is later restored and the UAV finishes at $(4, 2, 0)^\top$.

During the whole simulation, the target was kept in the FoV, which can be seen in the figure. The trajectory is not followed exactly in the beginning, as the UAV is also turning to center the target in the camera image, and this affects trajectory-following, as the UAV is not fully-actuated.

The states of the system relative to reference is presented in Figure 5.8. The main deviations are due to the collision avoidance and are even more evident for velocity and acceleration. The plot also presents the predictions of the state, that NMPC made. The prediction of N time steps ahead has very high variation with the real state.

Note, that the variation between NMPC prediction and real values does not mean that the problem formulation predictions the dynamics of the system badly. The variation is because the controls differ for different iterations of RTI scheme.

This can occur because the predicted state is not close to the optimal - in RTI scheme a larger amount of iterations may be needed to reach near-optimum states. Due to this also a large variation of the predicted last state can be seen between iterations (There are regions on the plot where the prediction jumps up to the value of 0.5 between iterations). Another reason, which is related to this one is that NMPC only predicts N time steps ahead, and thus cannot know about changes that will happen further in the future. After N iterations, the optimal solution may be far from the predicted one. This can be seen on the plots in areas where the variation between NMPC prediction and real state are the largest.

On the other hand, the predictions of $\frac{N}{2}$ time steps ahead are much closer to the real state of the system. More iterations have occurred and its is easier to predict the state in smaller time periods ahead. The prediction of 1 time step ahead is not shown there, but the difference would be barely noticeable, as it highly resembles the current state of the system.

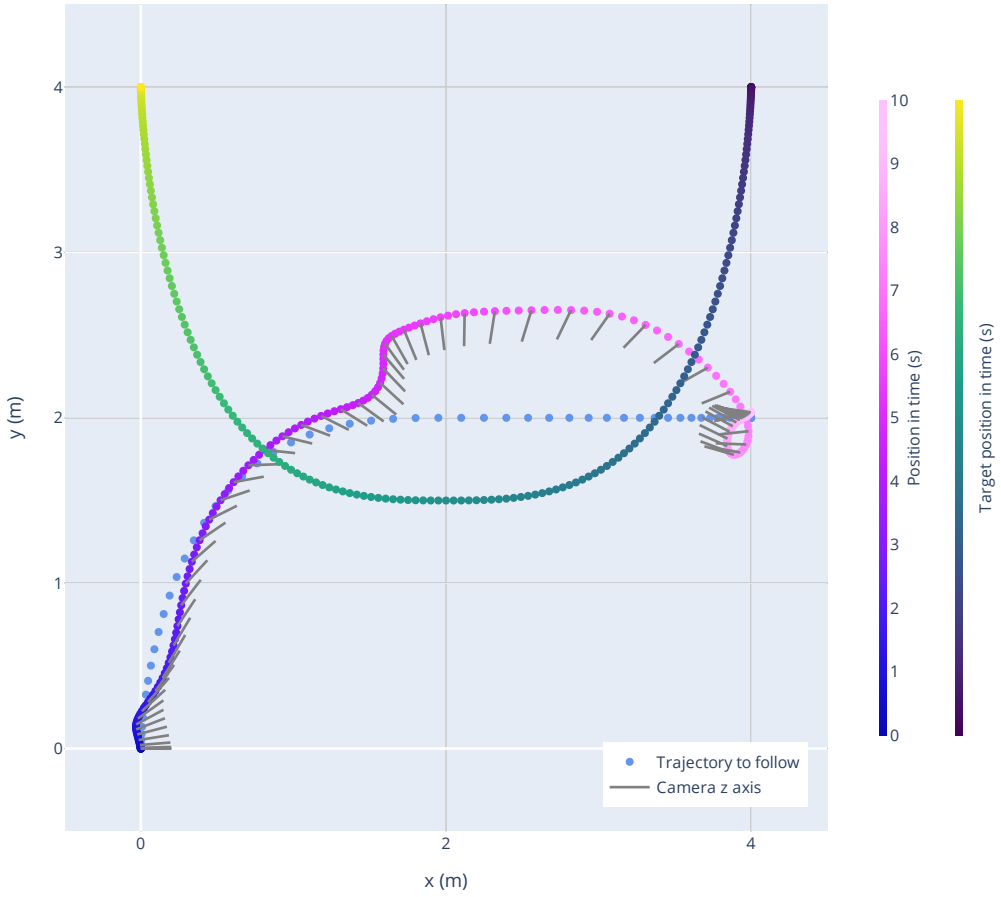


FIGURE 5.7: The position of UAV and target in time. The orientation of the UAV is given by the grey lines, which represent vectors aligned with the z-axis of UAV's camera. Blue points represent the given trajectory that UAV should follow.

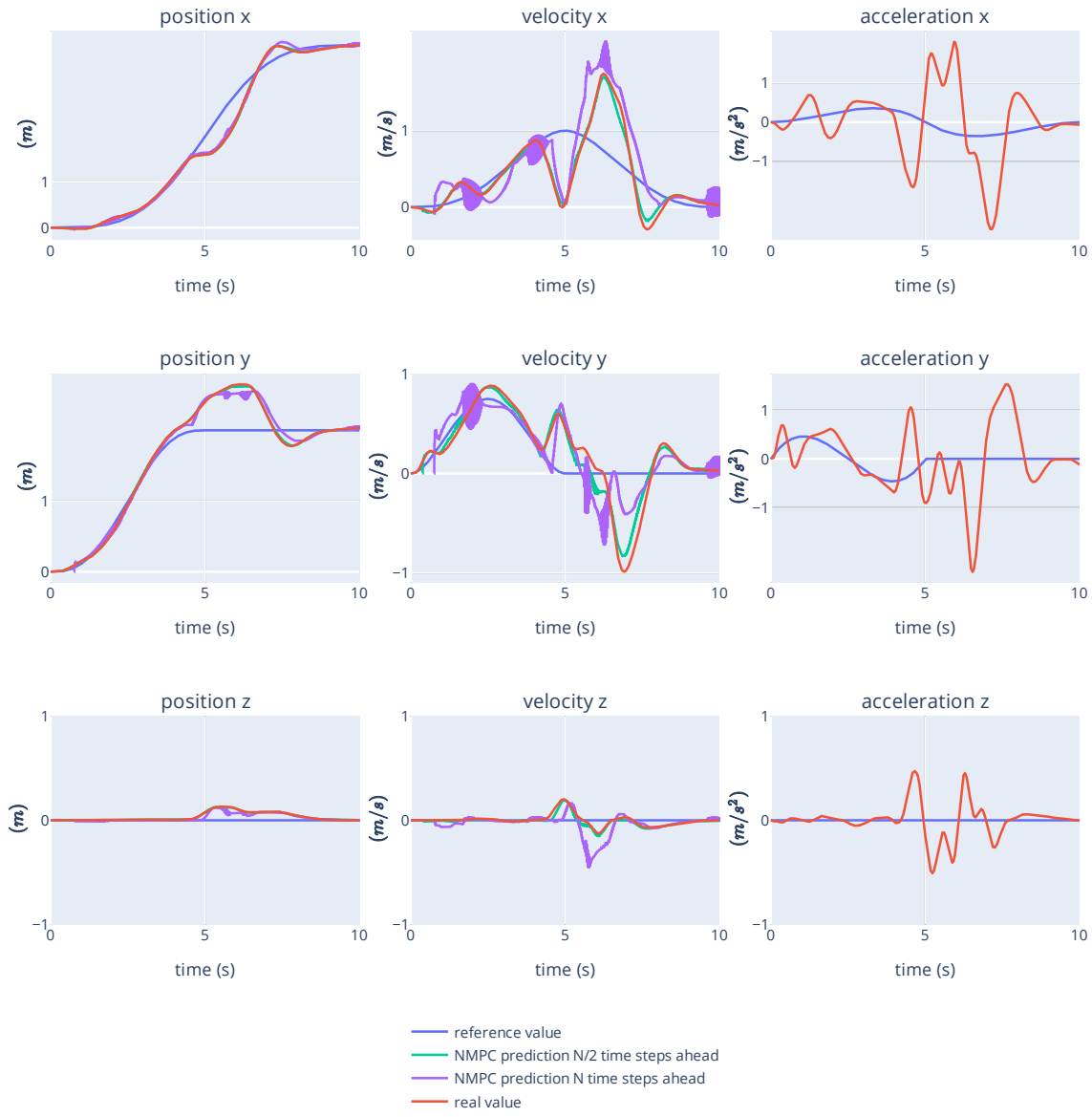


FIGURE 5.8: The desired trajectory and state of the system in time. The NMPC prediction is the predicted states of the system $N/2$ or N time steps ahead. In the cases when NMPC prediction curves are not visible - they are aligned with the real state of the system. Acceleration plots don't include NMPC predictions due to a larger amount of noise.

Chapter 6

Conclusion

A control solution for a multirotor with collision-avoidance and visual localization constraints is proposed in this work. The solution is based on NMPC control technique with RTI scheme used for optimization. This results in the ability to predict future states of the system directly inside the control loop while also allowing to achieve online optimization on-board of a multirotor.

The performance of different formulations of constraints has been analyzed in this work, including the impact of linearization performed by the RTI scheme. The least conservative formulation of obstacle avoidance constraint was used, which under linearization has minimal restrictions on the feasible set of the optimization problem while preventing the violations of the original constraint.

A linear version of FoV constraints is compared to the fractional one, both theoretically and practically, in order to achieve the best performance for keeping a target in the center of camera image. The linear version is used due to its better resemblance of original version under linearization and the absence of close-to-zero division which can occur for the fractional constraint in some real-world cases.

Additional terms are added to the objective in order to improve tracking of the target and optionally following it. Together with changes in constraint formulations, the solution accomplished to successfuly retain the target in camera's field of view, while safely moving in space.

The solution is tested in different scenarios that can arise in practical tasks. It is shown, that NMPC control standalone can be used in order to achieve target tracking and following together with collision avoidance. It is also shown that proposed solution can perform rapid movements for target-tracking and collision avoidance, while following a given trajectory. Good performance and variety of applications makes the proposed NMPC problem formulation highly practical.

The solution is generic and can be applied to both standard coplanar multirotos and multirotors with tilted propellers. It also produces rotor-level (torque) control inputs, hence it doesn't require an intermediate unconstrained controller to work. The UAV dynamics are modelled with high accuracy inside the control scheme, including actuator dynamical limitations.

Bibliography

Alexis, Kostas, George Nikolakopoulos, and Anthony Tzes (2012). "Model predictive quadrotor control: attitude, altitude and position experimental studies". In: *IET Control Theory & Applications* 6.12, pp. 1812–1827.

— (2014). "On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances". In: *Asian Journal of Control* 16.1, pp. 209–224.

Alexis, Kostas et al. (2016). "Robust model predictive flight control of unmanned rotorcrafts". In: *Journal of Intelligent & Robotic Systems* 81.3-4, pp. 443–469.

Andersson, Joel AE et al. (2019). "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1, pp. 1–36.

Bangura, Moses and Robert Mahony (2014). "Real-time model predictive control for quadrotors". In: *IFAC Proceedings Volumes* 47.3, pp. 11773–11780.

Bicego, Davide et al. (2020). "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs". In: *Journal of Intelligent & Robotic Systems* 100.3, pp. 1213–1247.

Bouabdallah, Samir and Roland Siegwart (2005). "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor". In: *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, pp. 2247–2252.

Brescianini, Dario and Raffaello D'Andrea (2016). "Design, modeling and control of an omni-directional aerial vehicle". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 3261–3266.

Carlos, Bárbara Barros et al. (2020). "Least conservative linearized constraint formulation for real-time motion generation". In: *IFAC-PapersOnLine* 53.2, pp. 9384–9390.

Catalin, Golban and Sergiu Nedevschi (2008). "Object tracking from stereo sequences using particle filter". In: *2008 4th International Conference on Intelligent Computer Communication and Processing*. IEEE, pp. 279–282.

Chen, Yutao et al. (2019). "Matmpc-a matlab based toolbox for real-time nonlinear model predictive control". In: *2019 18th European Control Conference (ECC)*. IEEE, pp. 3365–3370.

Dai, Shicong, Taeyoung Lee, and Dennis S Bernstein (2014). "Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass". In: *53rd IEEE Conference on Decision and Control*. IEEE, pp. 6149–6154.

De O. Pantoja, JFA (1988). "Differential dynamic programming and Newton's method". In: *International Journal of Control* 47.5, pp. 1539–1553.

Diebel, J (2006). *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*.

Diehl, Moritz, Hans Georg Bock, and Johannes P Schlöder (2005). "A real-time iteration scheme for nonlinear optimization in optimal feedback control". In: *SIAM Journal on control and optimization* 43.5, pp. 1714–1736.

Dijkstra, Edsger W et al. (1959). "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1, pp. 269–271.

Dunn, Joseph C and Dimitri P Bertsekas (1989). "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems". In: *Journal of Optimization Theory and Applications* 63.1, pp. 23–38.

Falanga, Davide et al. (2018). "Pampc: Perception-aware model predictive control for quadrotors". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1–8.

Ferreau, Hans Joachim et al. (2014). "qpOASES: A parametric active-set algorithm for quadratic programming". In: *Mathematical Programming Computation* 6.4, pp. 327–363.

Franchi, Antonio et al. (2018). "Full-pose tracking control for aerial robotic systems with laterally bounded input force". In: *IEEE Transactions on Robotics* 34.2, pp. 534–541.

Frison, Gianluca and Moritz Diehl (2020). "HPIPM: a high-performance quadratic programming framework for model predictive control". In: *arXiv preprint arXiv:2003.02547*.

Frison, Gianluca et al. (2018). "BLASFEO: Basic linear algebra subroutines for embedded optimization". In: *ACM Transactions on Mathematical Software (TOMS)* 44.4, pp. 1–30.

Gao, Fei et al. (2018a). "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 344–351.

Gao, Fei et al. (2018b). "Optimal time allocation for quadrotor trajectory generation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4715–4722.

Gao, Hongbo et al. (2018c). "Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment". In: *IEEE Transactions on Industrial Informatics* 14.9, pp. 4224–4231.

Gonçalves, JA and Renato Henriques (2015). "UAV photogrammetry for topographic monitoring of coastal areas". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 104, pp. 101–111.

Graf, Basile (2008). "Quaternions and dynamics". In: *arXiv preprint arXiv:0811.2889*.

Gros, Sébastien et al. (2020). "From linear to nonlinear MPC: bridging the gap via the real-time iteration". In: *International Journal of Control* 93.1, pp. 62–80.

Gu, Jingjing et al. (2018). "Multiple moving targets surveillance based on a cooperative network for multi-UAV". In: *IEEE Communications Magazine* 56.4, pp. 82–89.

Harabor, Daniel and Alban Grastien (2011). "Online graph pruning for pathfinding on grid maps". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 25. 1.

Hart, Peter E, Nils J Nilsson, and Bertram Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Hornung, Armin et al. (2013). "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3, pp. 189–206.

Hua, Minh-Duc et al. (2013). "Introduction to feedback control of underactuated VTOLvehicles: A review of basic control design ideas and principles". In: *IEEE Control systems magazine* 33.1, pp. 61–75.

Jacquet, Martin and Antonio Franchi (2020). "Motor and Perception Constrained NMPC for Torque-controlled Generic Aerial Vehicles". In: *IEEE Robotics and Automation Letters*.

Jacquet, Martin et al. (2020). "Perception-constrained and Motor-level Nonlinear MPC for both Underactuated and Tilted-propeller UAVS". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4301–4306.

Kamel, Mina et al. (2015). "Fast nonlinear model predictive control for multicopter attitude tracking on so (3)". In: *2015 IEEE Conference on Control Applications (CCA)*. IEEE, pp. 1160–1166.

Karaim, Malek et al. (2018). "GNSS error sources". In: *Multifunctional Operation and Application of GPS*, pp. 69–85.

Kavraki, Lydia E et al. (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4, pp. 566–580.

Kim, Huieun et al. (2016). "On-road object detection using deep neural network". In: *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, pp. 1–4.

Kocić, Jelena, Nenad Jovičić, and Vujo Drndarević (2018). "Sensors and sensor fusion in autonomous vehicles". In: *2018 26th Telecommunications Forum (TELFOR)*. IEEE, pp. 420–425.

Krajník, Tomáš et al. (2014). "A practical multirobot localization system". In: *Journal of Intelligent & Robotic Systems* 76.3, pp. 539–562.

Langley, Richard B, Peter JG Teunissen, and Oliver Montenbruck (2017). "Introduction to GNSS". In: *Springer handbook of global navigation satellite systems*. Springer, pp. 3–23.

LaValle, Steven M et al. (1998). "Rapidly-exploring random trees: A new tool for path planning". In:

Lee, Taeyoung, Melvin Leok, and N Harris McClamroch (2010). "Geometric tracking control of a quadrotor UAV on SE (3)". In: *49th IEEE conference on decision and control (CDC)*. IEEE, pp. 5420–5425.

Ligthart, Jeroen AJ et al. (2017). "Experimentally validated model predictive controller for a hexacopter". In: *IFAC-PapersOnLine* 50.1, pp. 4076–4081.

Lin, Penghong, Songlin Chen, and Chang Liu (2016). "Model predictive control-based trajectory planning for quadrotors with state and input constraints". In: *2016 16th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, pp. 1618–1623.

Lindqvist, Björn et al. (2020). "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles". In: *IEEE Robotics and Automation Letters* 5.4, pp. 6001–6008.

Mellinger, Daniel and Vijay Kumar (2011). "Minimum snap trajectory generation and control for quadrotors". In: *2011 IEEE international conference on robotics and automation*. IEEE, pp. 2520–2525.

Michael, Nathan et al. (2010). "The grasp multiple micro-uav testbed". In: *IEEE Robotics & Automation Magazine* 17.3, pp. 56–65.

Michieletto, Giulia, Markus Ryll, and Antonio Franchi (2018). "Fundamental actuation properties of multirotors: Force–moment decoupling and fail–safe robustness". In: *IEEE Transactions on Robotics* 34.3, pp. 702–715.

Michieletto, Giulia et al. (2020). "Hierarchical nonlinear control for multi-rotor asymptotic stabilization based on zero-moment direction". In: *Automatica* 117, p. 108991.

Miller, Isaac and Mark Campbell (2007). "Rao-blackwellized particle filtering for mapping dynamic environments". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3862–3869.

Moravec, Hans P. (1989). "Sensor fusion in certainty grids for mobile robots". In: *Sensor devices and systems for robotics*. Springer, pp. 253–276.

Penin, Bryan, Paolo Robuffo Giordano, and François Chaumette (2018). "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions". In: *IEEE Robotics and Automation Letters* 3.4, pp. 3725–3732.

Preiss, James A et al. (2017). "Crazyswarm: A large nano-quadcopter swarm". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3299–3304.

Qin, Hailong et al. (2019). "Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments". In: *IEEE Transactions on Vehicular Technology* 68.2, pp. 1339–1350.

Rajappa, Sujit et al. (2015). "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers". In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 4006–4013.

Riviere, Benjamin et al. (2020). "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning". In: *IEEE Robotics and Automation Letters* 5.3, pp. 4249–4256.

Rouček, Tomáš et al. (2019). "Darpa subterranean challenge: Multi-robotic exploration of underground environments". In: *International Conference on Modelling and Simulation for Autonomous Systesm*. Springer, pp. 274–290.

Silano, Giuseppe et al. (2021). "Power Line Inspection Tasks with Multi-Aerial Robot Systems via Signal Temporal Logic Specifications". In: *IEEE Robotics and Automation Letters* 6.2, pp. 4169–4176.

Sola, Joan (2017). "Quaternion kinematics for the error-state Kalman filter". In: *arXiv preprint arXiv:1711.02508*.

Spica, Riccardo et al. (2012). "Aerial grasping of a moving target with a quadrotor UAV". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4985–4992.

Tang, Jian et al. (2015). "LiDAR scan matching aided inertial navigation system in GNSS-denied environments". In: *Sensors* 15.7, pp. 16710–16728.

Thomas, Justin et al. (2017). "Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor". In: *IEEE Robotics and Automation Letters* 2.3, pp. 1762–1769.

Uzakov, Timur, Tiago P Nascimento, and Martin Saska (2020). "Uav vision-based nonlinear formation control applied to inspection of electrical power lines". In: *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, pp. 1301–1308.

Verschueren, Robin et al. (2019). "Acados: A modular open-source framework for fast embedded optimal control". In: *arXiv preprint arXiv:1910.13753*.

Walter, Viktor et al. (2019). "Uvdar system for visual relative localization with application to leader–follower formations of multirotor uavs". In: *IEEE Robotics and Automation Letters* 4.3, pp. 2637–2644.

Watanabe, Yoko, Anthony Calise, and Eric Johnson (2007). "Vision-based obstacle avoidance for UAVs". In: *AIAA guidance, navigation and control conference and exhibit*, p. 6829.

Zhao, F and BGM Van Wachem (2013). "A novel Quaternion integration approach for describing the behaviour of non-spherical particles". In: *Acta Mechanica* 224.12, pp. 3091–3109.