Application of Control Barrier Functions to Collision Free Model Predictive Control

Robust UAV Trajectories with MPC-CBF and Euclidean Signed Distance Fields

Rinto Cees de Vries

Application of Control Barrier Functions to Collision Free Model Predictive Control

Robust UAV Trajectories with MPC-CBF and Euclidean Signed Distance Fields

Thesis report

by

Rinto Cees de Vries

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on October 10, 2023 at 13:00

Thesis committee:	
Chair:	Dr. Ir. C. de Wagter
Supervisors:	Ir. T.S. Horstink
	Dr. Ir. E.J.J. Smeur
External examiner:	Dr. A. Anisimov
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	November, 2022 - October, 2023
Student number:	4881699

Faculty of Aerospace Engineering · Delft University of Technology

Preface

It is with immense joy that I present this thesis report. The research on UAV (Unmanned Aerial Vehicle) navigation within these pages signifies a remarkable collaboration between Mainblades Inspections, a pioneer in aircraft inspections via UAVs, and the esteemed Technical University of Delft. On a high level, the research in this paper serves to improve the navigation capabilities of a UAV in complex environments. Specifically, by incorporating control barrier functions as collision constraints in model predictive control.
I would like to express my sincere gratitude to Ewoud, my supervisor from TU Delft, for his insights and our weekly engaging discussions, which added immeasurably to this project, making the learning process both enlightening and enjoyable.
Equally deserving of my gratitude is Thomas, my dedicated supervisor from Mainblades Inspections. His commitment to addressing challenges on a daily basis has been truly remarkable. And, of course, the delightful coffee breaks added a lot of fun to the journey.
Additionally I would like to thank Mathias and Bastiaan from the KU Leuven, for providing me with several valuable insights during our video calls throughout my thesis project.
Lastly, I would like to thank my colleagues Alessandro, Daan, Julia and Rick from Mainblades Inspections for providing me with practical help in the multiple testing sessions at Schiphol.

Thank you for reading this work.

Rinto de Vries
Delft, September 2023

Contents

Nomenclature

List of Abbreviations

APF	Artificial Potential Field
AUV	Autonomous Underwater Vehicle
CBF	Control Barrier Function
CC-SBC	Chance-Constrained Safety Barrier Certificate
CCNMPC	Chance-Constrained Nonlinear Model Predictive Control
CLF	Control Lyapunov Function
EKF	Extended Kalman Filter
ESDF	Euclidean Signed Distance Field
FOV	Field of View
GPS	Global Positioning System
HIL	Hardware In the Loop
IMU	Inertial Measurement Unit
IP	Interior Point
LKF	Linear Kalman Filter
MAV	Micro Aerial Vehicle
MD	Mission Duration
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
PD	Proportional Derivative
PrSBC	Probabilistic Safety Barrier Certificate
PSDK	Payload Software Development Kit
QCQP	Quadratically Constrained Quadratic Program
QP	Quadratic Programming
RRT	Rapidly-exploring Random Tree
SBC	Safety Barrier Certificate
SFP	Successful Flight Percentage
SITL	Software in the Loop
SQP	Sequential Quadratic Programming
TSDF	Truncated Signed Distance Field
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UKF	Unscented Kalman Filter

List of Symbols

Δt	Time step
δ	A slack term
$\ell_N(x_N)$	Terminal cost term for time step n
$\ell_n(x_n, u_n)$	Stage cost term for time step n
Γ	Covariance
γ	Extended class K function
$\mathbf{a}$	Acceleration vector in an inertial frame
$\mathbf{p}$	Position vector in an inertial frame
$\mathbf{R}(\psi, \theta, \phi)$	Rotation matrix R from global frame to body frame
$\mathbf{s}$	Slack vector
$\mathbf{u}_{t:t+N-1\|t}$	Control input vector of timestep t up and until timestep t+N-1 at iteration t
$\mathbf{v}$	Velocity vector in an inertial frame
$\mathbf{x}_i^{k+1}$	The state vector of robot i at timestep $k+1$
$\mathcal{U}$	Control input space
$\mathcal{X}$	State space
μ	Mean
ω_i^k	Uncorrelated noise process of robot i at timestep k
ϕ	Roll angle

ϕ_c	Commanded roll angle
ψ	Yaw angle
σ	Standard deviation
τ_ϕ	Time constant for roll angle control
θ	Pitch angle
A_x	Drag coefficient in global frame x direction
c_{d_x}	Drag coefficient along the x-axis
$d(\mathbf{p})$	Distance to nearest obstacle function
g	Gravitational acceleration constant
h	State constraint function
J	Cost function
K_ϕ	Gain of the inner roll loop
L_f	Lie derivative with respect to f
N	Number of steps in time horizon MPC
p	Terminal cost function
q	Stage cost function
r	Relative degree of a dynamical system
r	Slack cost function
T	Mass normalized thrust
t	Time
V	Lyapunov potential function
x_{pos}	X-axis position in world frame
x_{vel}	X-axis velocity in world frame

1 Introduction

This report provides the reader with an article and a literature review, focusing on the use of Control Barrier Functions (CBFs) to handle collision constraints in Model Predictive Control (MPC) for unmanned multicopters. The literature review identifies gaps in the current research and forms the basis for the article. The goal of the article is to answer the research questions arising from the literature review. The purpose of this introduction is to provide a concise motivation, define the research objective and questions, and outline the report's structure.

1.1. Motivation

The use of unmanned aerial vehicles (UAVs) has seen a significant growth in the field of inspection applications [1]. An example of this is an A-check, a visual aircraft inspection, which takes about 3-4 hours when performed manually. By using an automated drone this time can be decreased by up to 75%[1]. In their survey on fully autonomous drones Elmokadem and Savkin noted that the development of robust planning and execution algorithms for safe UAV trajectories, particularly in complex and dynamically changing environments, remains a challenge. This challenge is primarily caused by non-perfect sensors, non-perfect state estimation and limited hardware capabilities.

In the realm of trajectory generation and tracking, the conventional approach for UAVs involves a decoupled process. In this approach, a high-level trajectory planner plans a collision-free path, which is subsequently tracked by a controller. For instance, in [3], this involves employing an RRT* path planner to create a collision-free path, which is then tracked using an MPC. However, in unknown environments, the integration of these two steps can be more efficient (kinodynamic planning), enabling real-time trajectory planning and tracking. Yet, incorporating collision constraints can complicate real-time solutions and introduce convergence challenges [2].

In [4], the authors employ Nonlinear MPC (NMPC) with Euclidean Signed Distance Fields (ESDFs) for collision avoidance in dense, unknown environments, achieving real-time navigation within a forest environment without the need for obstacle segmentation. Addressing the limitations of previous deterministic kinodynamic planning, Zhu and Alonso-Mora propose a Chance Constrained Nonlinear MPC (CCNMPC) in [5]. This introduces probabilistic collision risk constraints based on state estimator mean and standard deviation values. While this approach improves robustness, it comes at the cost of computational complexity, mainly arising from integrating probability densities.

Another emerging area of research focuses on the fusion of MPC with Control Barrier Functions (CBFs). The primary objective here is to combine the non-greedy nature of MPC with the safety guarantees offered by CBFs. However, research in this field is, to the best of the authors' knowledge, limited to just a few papers. Two of these papers present simulation experiment results using MPC with a CBF collision constraint applied in the context of a racing car, i.e., [6] and [7]. Additionally, there is one paper that applies CBFs as an MPC constraint for foot placement on a walking robot on real hardware [8]. Recently Li et al. published the arXiv preprint [9]. This paper focused on incorporating CCNMPC with CBF for collision avoidance using a 2D double integrator model with moving obstacles. The authors aimed to harness the strengths of both MPC-CBF and the robustness of CCNMPC in their approach. Their findings indicate that

[1]`https://mainblades.com/wing-inspections/`

CC-MPC-CBF delivers significant enhancements over deterministic MPC-CBF when dealing with noise, although they do not discuss the potential increase in computational complexity. It is worth noting that all four papers assumed an analytical function for the safe set of the CBFs. This assumption might not hold in scenarios involving collision avoidance, for example when the environment is at least partially unknown.

Despite the limited amount of research and the dependency on analytical functions to describe the surroundings, the results look promising. In all three papers, the use of CBFs as constraints provided a significant improvement in performance.

1.2. Research Formulation

The goal of this research is to study the feasibility of MPC-CBF combined with ESDFs that encode the obstacles in the Optimal Control Problem (OCP). Specifically, the research objective is defined as follows:

Research Objective

The primary objective of this thesis is to develop and validate a control framework for UAVs that combines MPC and CBF collision constraints using ESDFs. The research aims to demonstrate the effectiveness, robustness, and practicality of this novel control approach in real-time trajectory planning, tracking, and obstacle avoidance scenarios, ultimately enhancing the capabilities of UAVs for complex missions in challenging environments.

This research objective has been made more concrete by subdividing it into three research questions. The three research questions are as follows:

Research Question 1

What is the effect of applying first-order and second-order CBFs to ESDF-based collision constraints in MPC for UAVs on the trajectory planning and tracking?

Research Question 2

What is the effect of changing the variable parameters γ on the trajectory planning and tracking when using ESDF-based CBF collision constraints for UAVs in MPC?

Research Question 3

How do noise and delays influence trajectory planning and collision avoidance in UAVs, and how do these effects compare between baseline ESDF-based distance constraints and CBF collision constraints in MPC?

1.3. Structure of the Report

The report is structured as follows: Part I contains a stand-alone scientific article titled ”Application of Control Barrier Functions to Euclidean Signed Distance Field based Collision Constraints in Model Predictive Control”. This article covers key aspects including an introduction that sets the context, background information to provide essential context, the design of the controller, details of the experiment setup, presentation and analysis of results, and a concluding section that summarizes the insights gained. Part II contains the literature review, which has been assessed in $AE4020$.

Part I

Scientific Article

Application of Control Barrier Functions to Euclidean Signed Distance Field based Collision Constraints in Model Predictive Control

R.C. de Vries, Ir. T.S. Horstink, Dr. Ir. E.J.J. Smeur
Delft University of Technology, Faculty of Aerospace Engineering, Department of Control & Simulation, Micro-Air-Vehicle Laboratory (MAVLab)

Recent literature in real-time trajectory planning has proposed using Control Barrier Functions (CBFs) as collision constraints in Model Predictive Control (MPC) for efficient guidance, a concept referred to as MPC-CBF. This concept has been explored for both first and second-order CBFs. However, these approaches relied on an analytical description of the environment. Building upon this, we propose combining MPC-CBF with Euclidean Signed Distance Fields (ESDFs), eliminating the need for such an analytical model of the environment. Notably, we extend this approach to a new field by applying it to Unmanned Aerial Vehicles (UAVs). Through simulations, we compare flown trajectories and noise robustness for distance constraints, first-order CBF constraints and second-order CBF constraints. First-order CBF constraints outperform distance constraints, excelling in path planning and noise resilience. Second-order CBF constraints face challenges due to numerical approximations of the hessian of the ESDF and stricter dependency on an accurate acceleration model, limiting their practicality for UAVs. The proposed control framework was tested by safely maneuvering an enterprise inspection drone around a Boeing 787-9 aircraft inside an aircraft hangar, confirming its effectiveness in collision avoidance and real-world scenarios.

I. Introduction

The field of unmanned aerial vehicles (UAVs) has seen a significant growth in recent years due to their potential applications in various fields such as search and rescue [1], precision agriculture [2], and inspection purposes [3]. One of the key challenges in the development of UAVs is the ability to plan and execute safe and efficient trajectories in complex and dynamic environments [4].

Usually, this trajectory generation and tracking is achieved in a decoupled manner where a high-level trajectory planner is used to create a feasible collision-free path that is to be tracked by a controller. An example of such a workflow is [5], the approach initially involves devising a collision-free path using an RRT* path planner. Subsequently, it employs a minimum snap trajectory generator using the generated path that is then tracked by a Model Predictive Controller (MPC).

In unknown environments it can be more efficient to combine these two steps to achieve real-time trajectory planning and tracking. This is called kinodynamic planning. However, factoring in collision constraints in planning complicates real-time solutions and introduces possible convergence issues [4].

In [6] the authors use NMPC with Euclidean Signed Distance Fields (ESDFs) collision constraints for obstacle avoidance with local exploration to escape local minima. The authors manage to navigate in the unknown environment of a dense forest using fully onboard mapping and planning at 4 Hz. The use of ESDFs eliminates the need for segmenting the obstacles.

Zhu and Alonso-Mora note in [7] that trajectories generated by previous deterministic kinodynamic planning studies are not robust to state estimation or localization errors. The authors propose a Chance Constrained Nonlinear MPC (CCNMPC). Rather than using strict collision constraints, a constraint is put on the probability of the risk of a collision. This is done by taking into account the mean and standard deviation of the state estimator. The authors show in a position swap experiment between two quadcopters a significant improvement over a deterministic NMPC while using a motion capture system with injected noise on the state. However, the authors note that numerically integrating collision probabilities is computationally very expensive.

A novel promising subfield of kinodynamic planning is the combination of MPC and Control Barrier Functions (CBFs). In [8] the authors show that CBFs on distance are a more general case of Artificial Potential Fields (APFs) and solve the problem of oscillatory behavior common in APF methods. APF is inspired by the repulsive and attractive force of charged particles and a common technique for collision avoidance first introduced in 1986 by Khatib. Stastny et al. show the first combination of MPC and APF on a UAV for collision avoidance, repeated many times, e.g., in [11]. More background information about CBFs is provided in subsection II.B.

The general idea of combining MPC with CBF is to combine the safety guarantees of CBFs with the non-greedy nature of MPC to plan and track robust trajectories. In both [12] and [13] the authors combine MPC with a CBF constraint for a car simulation and show increased driving performance over conventional MPC. In [14] the authors demonstrate the performance improvement achieved by incorporating both a first-order CBF constraint in an MPC planner and a safety filter in the low-level tracking controller for a legged robot. Li et al. recently published an arXiv preprint [15] on chance constrained MPC with CBF for moving obstacle collision avoidance using a 2D double integrator model. The authors conclude that CC-MPC-CBF offers a significant improvement over deterministic MPC-CBF in the presence of noise, but make no mention of the added computational cost. Note that all these papers rely on an analytical description of the environment.

We aim to investigate and analyze the effects of this combined methodology specifically for trajectory planning and obstacle avoidance applied to quadcopter scenarios when combined with ESDFs. The contributions of this work are as follows:

- Novel Control Design: MPC-CBF Integration with ESDFs. This work presents, to the best of the authors' knowledge, the first combination of MPC-CBF with ESDFs. The use of ESDFs eliminates the need to segment obstacles and a specific obstacle model (e.g., modeling objects as spheres). Specifically, a comparison of the mission duration and collision avoidance performance between a distance constraint, first-order CBFs and second-order CBFs, all using ESDFs is given.
- Noise and Delay Analysis: This research also includes an investigation into the influence of delays and noise on the state of the deterministic MPC-CBF controller. By subjecting the controller to various challenging scenarios, the study evaluates its robustness, reaffirming its reliability and practicality in real-world applications.
- New Context: This work presents, to the best of the authors' knowledge, the first-ever combination of MPC and CBF for real-time trajectory planning and tracking onboard a UAV. The proposed controller is thoroughly tested on two simulators. Additionally, the resulting framework is validated using an aircraft inspection in a GPS-denied hangar using lidar inertial odometry.

The article is organized as follows: section II provides background information about the quadcopter model used and CBFs. In section III, the controller design is described. In section IV the methodology and results of the first simulation experiment are presented: flying a multi-waypoint trajectory around a machine hall. In section V the second simulation experiment is laid out: the inspection of an aircraft under the influence of a fixed delay and varying noise levels on the state. After that, section VI describes the practical experiment of an aircraft inspection, which serves to validate the previously acquired results. Finally, in section VII, the paper is concluded, and potential future work is discussed.

II. Background

This section provides the reader with background information on the quadcopter model used in subsection II.A and on CBFs in subsection II.B.

A. Quadcopter Model

Note that this model assumes a low-level controller for the roll angle, pitch angle, yaw rate and throttle. This is (among others) available for Parrot multicopters[1], DJI multicopters[2] and PixHawk controllers [16]. It is also assumed that the user performed a throttle-thrust mapping, e.g., as in [17].

Formally, the control input vector is represented as $\mathbf{u} = \left[\phi_c, \theta_c, T_c, \dot{\psi}_c\right]^T \in \mathbb{R}^4$, where ϕ_c, θ_c, T_c, $\dot{\psi}_c$ are the commanded roll angle, pitch angle, mass normalized thrust (total thrust divided by UAV mass) and yaw rate, respectively. The model is proposed by [18] and modified by adding first-order dynamics to the mass normalized thrust, introducing an extra state. The model is as follows:

$$\begin{aligned}
\dot{\mathbf{p}}(t) &= \mathbf{v}(t) \\
\dot{\mathbf{v}}(t) &= \mathbf{R}(\psi, \theta, \phi)\begin{pmatrix} 0 \\ 0 \\ T \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \\
&\quad - \begin{pmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{pmatrix}\mathbf{v}(t) \\
\dot{T}(t) &= \frac{1}{\tau_T}\left(T_c(t) - T(t)\right) \\
\dot{\phi}(t) &= \frac{1}{\tau_\phi}\left(\phi_c(t) - \phi(t)\right) \\
\dot{\theta}(t) &= \frac{1}{\tau_\theta}\left(\theta_c(t) - \theta(t)\right) \\
\dot{\psi}(t) &= \dot{\psi}_c(t),
\end{aligned}$$

where $\mathbf{p}$ and $\mathbf{v}$ are the three-dimensional position and velocity vectors in the global frame, respectively. The model assumes first-order dynamics for the roll angle, pitch angle[2] and the mass normalized thrust, no time delay for the commanded control inputs and a linear relationship between the velocity and the air resistance.

$\mathbf{R}(\psi, \theta, \phi)$ is the rotation matrix from the body frame of reference to the global frame of reference. g is the gravitational acceleration. A_x, A_y and A_z indicate the mass normalized drag coefficients, and τ_ϕ, τ_θ and τ_T indicate the time constants for the roll angle, pitch angle and mass normalized thrust, respectively.

[1]https://www.parrot.com/assets/s3fs-public/2022-01/whitepaperanafiai.pdf

[2]https://developer.dji.com/onboard-sdk/documentation/introduction/homepage.html

B. Control Barrier Functions

Throughout this subsection the following dynamic control-affine system is assumed:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{1}$$

where f and g are locally Lipschitz functions, $\mathbf{x} \in X \subset \mathbb{R}^n$ is the state in the state space and $\mathbf{u} \in U \subset \mathbb{R}^m$ is the control input in the space of control inputs.

The goal is to be forward invariant inside a user defined safe set. I.e., starting in the set, means staying inside of it. This set is usually denoted by $S(\mathbf{x})$ for all $\mathbf{x} \in X$ where $h(\mathbf{x}) \geq 0$. An example of the function h in the context of collision avoidance is the distance to the nearest obstacle. This implies a safe set of every state with a non-collision position. For a system of relative degree 1, this can be guaranteed by:

$$\sup_{\mathbf{u}\in U}\left[\dot{h}(\mathbf{x})\right] = \sup_{\mathbf{u}\in U}\left[L_f h(\mathbf{x}) + L_g h(\mathbf{x})u\right] \geq -\gamma(h(\mathbf{x})). \tag{2}$$

If a function h satisfies this property, it is called a CBF. Intuitively this makes sense: $-\gamma(h(\mathbf{x}))$ is zero near the borders and is negative in the set. $L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u}$ is the Lie derivative of $h(\mathbf{x})$. In other words, the derivative is forced to be at least 0 at the borders but can decrease in the safe set S. Given the definition of the safe set as the region where the CBF is greater than zero, if a starting point is within this set, it implies that the trajectory remains within the safe set [19].

In the context of collision avoidance, where $d(\mathbf{p})$ is the distance to the nearest object given a certain position $\mathbf{p}$, the function γ dictates what the maximum velocity towards an object $\dot{d}(\mathbf{p})$ can be given a distance function d and a certain position $\mathbf{p}$. Note that γ can be seen as a trade-off between feasibility and safety. This is extensively described in [12].

These results can also be extended to systems with a higher relative degree by nesting CBFs, as described in [20]. For a system of relative degree 2 this results in applying the CBF condition Equation 2: $g(\mathbf{x}) = (\frac{d}{dt} + \gamma) \circ h(\mathbf{x})$ resulting in:

$$\dot{g}(\mathbf{x}) \geq -\gamma(g(\mathbf{x})). \tag{3}$$

Note that $\circ$ stands for function composition. This CBF condition guarantees $g(\mathbf{x}) \geq 0$, which on itself is a CBF that guarantees $h(\mathbf{x}) \geq 0$.

III. Control Design

This section describes the use of ESDFs, defines the optimal control problem, and finally lays out the control architecture.

A. Euclidean Signed Distance Field

ESDFs are data structures in computer graphics, enabling efficient distance calculations from any point to the nearest obstacle. Negative distance values signify that the point is located inside a closed shape.

The authors of [21] introduced Voxblox, an open-source library facilitating real-time 3D environment modeling and mapping through discrete voxel-based ESDF maps, with each voxel storing proximity distance. Voxblox's effectiveness lies in seamless voxel grid updating and querying, employing incoming depth maps, and calculating distance gradients using the finite difference method across adjacent voxels. In this paper, static ESDFs are created from available OctoMaps [22]. This process is described in Appendix A.

B. Optimal Control Problem

The optimal control problem for an iteration at timestep t is defined as follows:

$$\begin{aligned}
\min_{\mathbf{u}_{t:t+N-1|t}} \quad & J = \sum_{k=0}^{N-1} q\left(\mathbf{u}_{t+k|t}\right) \\
& + p\left(\mathbf{x}_{t+N|t}\right) + \sum_{k=0}^{N} r\left(\mathbf{s}_{t+k|t}\right) \\
s.t. \quad & \mathbf{x}_{t+k+1|t} = f\left(\mathbf{x}_{t+k|t}, \mathbf{u}_{t+k|t}\right), k = 0, \ldots, N-1 \\
& \mathbf{u}_{t+k|t} \in \mathcal{U}, k = 0, \ldots, N-1 \\
& \mathbf{x}_{t+k|t} + \mathbf{s}_{x,t+k|t} \in \mathcal{X}, k = 0, \ldots, N \\
& \mathbf{s}_{x,t+k|t} \geq 0, k = 0, \ldots, N \\
& h\left(\mathbf{x}_{t+k|t}\right) + \mathbf{s}_{h,t+k|t} \geq 0, k = 0, \ldots, N \\
& \mathbf{s}_{h,t+k|t} \geq 0, k = 0, \ldots, N,
\end{aligned}$$

where q is the control input stage cost, p is the terminal state cost, r is the slack cost, h is the collision constraint and f is the model (which was described in section II). The slack vectors $\mathbf{s}$ are used to soften the state and collision constraints at a high cost. Slightly violating the constraint is preferred over a controller that fails to solve the control optimization problem. The state $\mathbf{x} \in \mathbb{R}^{10}$ is as follows: $\mathbf{x} = [\mathbf{p}, \mathbf{v}, \phi, \theta, \psi, T]^T$.

After every iteration, the optimal first control input $\mathbf{u}_0$ is sent to the UAV interface, as shown in Figure 1.

In this paper three different collision constraints of increasing dynamic order will be compared, namely:

$$\begin{aligned}
h_1 &= d(\mathbf{p}) \\
h_2 &= \dot{h}_1 + \gamma_1 h_1 = \frac{dh}{d\mathbf{p}} \cdot \mathbf{v} + \gamma_1 d(\mathbf{p}) \\
h_3 &= \dot{h}_2 + \gamma_2 h_2 = \frac{dh}{d\mathbf{p}} \cdot \mathbf{a} + \frac{d^2 h}{d\mathbf{p}^2} \cdot \mathbf{v}^2 \\
&\quad + (\gamma_1 + \gamma_2) \frac{dh}{d\mathbf{p}} \cdot \mathbf{v} + \gamma_1 \gamma_2 d(\mathbf{p}),
\end{aligned}$$

where h_1 is a distance constraint, h_2 is a first-order CBF constraint on the distance and h_3 is a second-order CBF constraint on the distance. Note that h_1 is only dependent on the positions, h_2 is dependent on the positions and the velocities and h_3 is dependent on the accelerations, the velocities and the positions.

Please note that the distance to the nearest obstacle, denoted as $d(\mathbf{p})$, as well as its respective derivatives, are computed using ESDFs.

C. Control Architecture

The complete control architecture can be summarized in Figure 1. Details of each process (block in diagram) can be found below:

1. Optimizer

For the optimization solver in this study, we selected the widely used HPIPM solver implemented by the acados software package [23, 24].

2. Model

The model is described in subsection II.A.

3. UAV Interface

For both simulation experiments the 3DR Iris Quadrotor PX4 Gazebo Software In The Loop was used[3].

[3]https://docs.px4.io/v1.12/en/simulation/

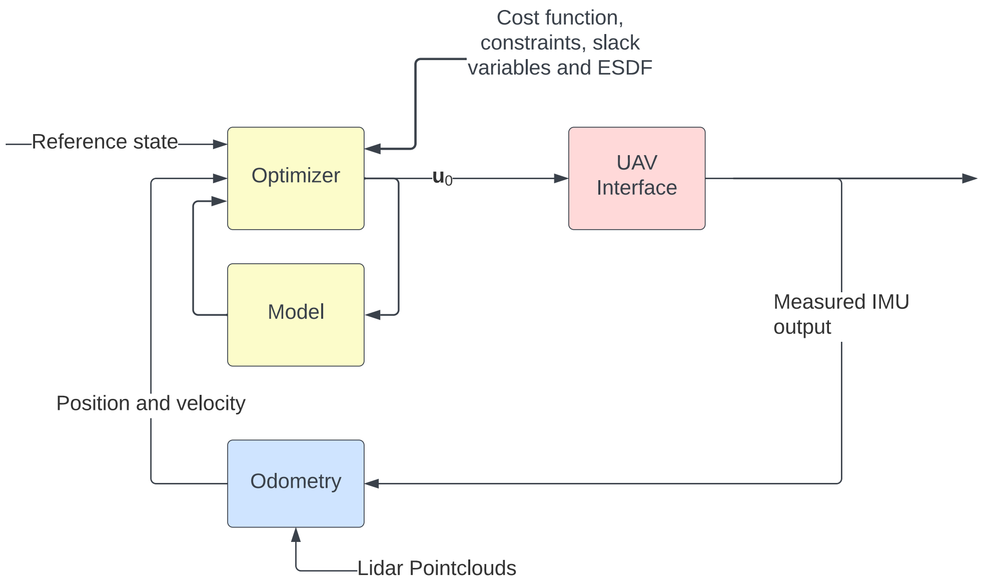


Figure 1. Control diagram of the proposed controller.

This open-source 3D simulator simulates both the physics and the resulting MAVLink messages[4]. The interfacing to the simulator is done via MAVROS[5].

For the practical experiment a DJI M300 RTK drone equipped with a custom companion computer was used. This is visualized in Figure 2. This computer features an Intel i5-8365UE processor with 8GB of RAM and is responsible for onboard control. The interaction with the drone is facilitated through the DJI PSDK. During the development phase of the project, DJI's Hardware In the Loop Flight Simulator[6] was employed.

4. Odometry

In simulation, both UAV interfaces provide ground truth odometry. In the experimental setting, odometry is generated by combining lidar and IMU data. The localization process uses a known static point cloud of the aircraft and an iterative closest point method for localization [25]. Scan-to-scan matching is performed using a tightly coupled lidar inertial odometry algorithm. Both the localization and scan-to-scan matching are run onboard on the companion computer and combined with the localization results to provide accurate UAV position and velocity information. The lidar used is an Ouster OS0, which has a $90° FOV$[7].

[4]https://mavlink.io/en/

[5]https://github.com/mavlink/mavros

[6]https://www.dji.com/nl/simulator

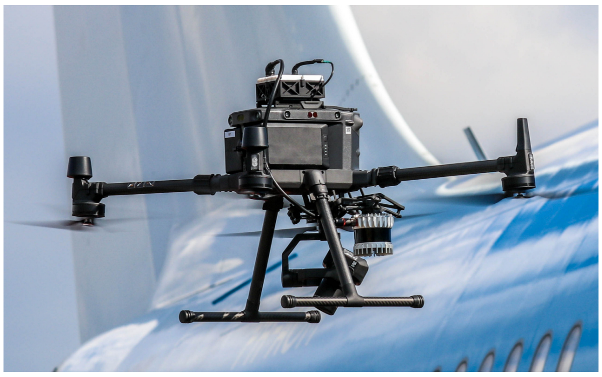


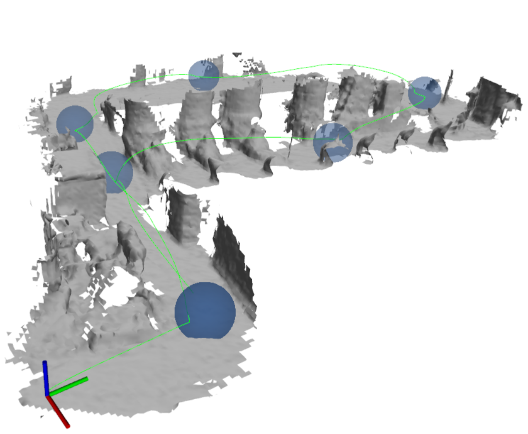
Figure 2. A DJI M300 RTK equipped with a camera, Ouster OS0 lidar and a custom companion computer.

IV. Simulation Experiment: Machine Hall

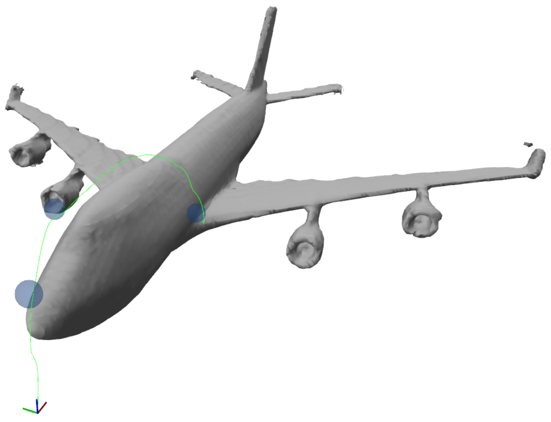
A. Methodology

The objective of this simulation experiment is to test and compare the behavior of the different constraints in a confined flight space. Additionally, the goal is to describe the effect of the choice of γ_1 and γ_2, which are the variable parameters in the CBF constraint.

[7]https://ouster.com/products/scanning-lidar/os0-sensor/



(a) The mesh of the machine hall visualized with the waypoints and one of the flown paths (constraint h_2, $\gamma_1 = 2$ s^{-1}).



(b) The mesh of a Boeing 747-400F visualized with the waypoints and one of the flown paths (constraint h_2, $\gamma_1 = 2$ s^{-1}) under 100 ms delay and no noise.

Figure 3. Visualized voxblox meshes and flown flight paths.

The experiment is to fly a multi-waypoint trajectory in a machine hall. The ESDF of the machine hall is generated onboard a micro aerial vehicle using a RealSense D415 depth camera[8] and the open-source Voxblox library [21]. The waypoints and one of the flown routes are visualized in Figure 3a.

The simulation experiment was repeated for the three different collision constraints (h_1, h_2 and h_3) for different values of γ. Specifically, $\gamma_1 \in \{0.5, 1, 2, 4, 6\}$ s^{-1} for constraint h_2. For constraint h_3, we will perform a cross-search on these same values for both γ_1 and γ_2. This results in fifteen unique combinations for the product and sum since the order of γ_1 and γ_2 is irrelevant for producing a product and a sum (as they appear in h_3).

The controller was run at 20 Hz with a time horizon of 1 s. The control inputs are constrained to 30° in roll and pitch angle, and 30°/s for the yaw rate. The different settings were compared based on their Mission Duration (MD) and the number of near crashes (number of data points within 5 cm of the lower threshold on the distance constraint). The minimum distance to keep is set to 30 cm from the center of the drone. For the CBF constraints, the minimum distance is 30 centimeters given that $\dot{h} = 0$ and $\ddot{h} = 0$. The simulation was stopped if the time between two waypoints is longer than 30 s.

[8]`https://www.intelrealsense.com/depth-camera-d415/`

B. Results

The average computational time of one iteration on a laptop (Intel i7-8750H processor, 32GB RAM) is 3.7 ms with a standard deviation of 2.1 ms across all runs. The maximum computational time observed is 12.1 ms. No significant difference in the computational time was observed between the three constraints.

The MD and the number of near crashes (within 5 cm of distance threshold) are provided in Table 1. For readability, only the configurations that reached the final waypoint (with maximum 30 s between the waypoints) and did not violate the distance constraint of the total 21 configurations are shown. The constraint h_1 violated the minimum distance constraint but was still provided since the constraint only has one configuration and forms a baseline for the other two constraints.

Firstly, it was observed that lower values of γ_1 for h_2 and lower values of $\gamma_1\gamma_2$ for h_3 resulted in an increased MD. This relationship aligns with intuition, as γ_1 governs the allowed velocity towards an obstacle based on the distance. Similarly, for h_3 with a given sum $\gamma_1 + \gamma_2$, the product $\gamma_1\gamma_2$ influences the maximum velocity/acceleration combination for a given distance. However, lower values of γ_1 for h_2 and lower values of $\gamma_1\gamma_2$ for h_3 do not necessarily result in less near crashes. This is because excessively low γ_1 (or $\gamma_1\gamma_2$) values led

			MD [s]	**Near Crashes**
h_1[a]			45.3	73
	γ_1		**MD** [s]	**Near Crashes**
h_2	0.5		67.75	0
h_2	1		57.55	0
h_2	2		53.7	45
h_2	4		47.75	16
h_2	6		45.95	31
	$\gamma_1 + \gamma_2$	$\gamma_1 \cdot \gamma_2$	**MD** [s]	**Near Crashes**
h_3	3	2	98.05	13
h_3	4.5	2	102.95	0
h_3	5	4	88.55	9
h_3	6.5	3	91.1	5
h_3	6	8	77.05	39
h_3	8	12	68.35	7
h_3	8	16	63.2	6
h_3	12	36	54.05	21

Table 1. Results for simulated machine hall experiment of MD and near crash incidents for constraints h_1, h_2 and h_3.

[a] Violated the distance constraint by 2.5 cm.

to a restricted feasibility space, causing the drone to get stuck near obstacles and increasing the likelihood of near crashes. Conversely, very high γ_1 (or $\gamma_1\gamma_2$) values compromise safety, as this approaches a simple distance-to-obstacle (h_1) constraint.

Secondly, for the second-order CBF h_3 it can be noted that the performance in general is lower than for h_2 (higher MD and 7/15 configurations not reaching the final waypoint successfully). Although arguably higher than for h_1 since some configurations can reach the waypoint without crashing. It can be noted that the sum of $\gamma_1 + \gamma_2$ dictates the weighing between first time derivative $\dot{h}$ and the second time derivative $\ddot{h}$ towards an obstacle. A higher ratio indicated a higher dependency on $\dot{h}$, effectively making it more like the first-order CBF. This can also be seen in the result, where $\gamma_1 + \gamma_2 = 12$ s^{-1} and $\gamma_1\gamma_2 = 36$ s^{-2} comes closest to the MD of a first-order CBF, where the contribution of $\ddot{h}$ is relatively small (only $\frac{1}{12}$ as compared to $\dot{h}$).

There are two items that make the use of $\ddot{h}$ challenging:

1) The acceleration vector is not a state in the optimal control value problem, and thus not updated to the measured values at every timestep. Rather, all the states are updated to their measured values. Since the thrust is not measured, the previous control input filtered with first-order dynamics is used (as described in the model in subsection II.A). This means the constraint is directly dependent on the correctness of the acceleration model including the throttle-thrust mapping.
2) The term $\ddot{h}$ depends upon the hessian of the ESDF. This hessian is calculated numerically on voxels of size 0.1 m, thus there will be an error there. Calculating the Hessian of the ESDF for voxelized structures is challenging due to the discrete nature of the grid, which can lead to inaccurate curvature estimates.

These hypotheses are validated using an analytical distance function and using a simulator where the model of the simulator is the same model of the MPC. The performance of constraint $h_3 = \ddot{h}(\mathbf{p}, \mathbf{v}, \mathbf{a}) + 10\dot{h}(\mathbf{p}, \mathbf{v}) + 24h(\mathbf{p})$ was the best performing controller in terms of MD. This indicates that the term $\ddot{d}$ can indeed improve the performance theoretically. The experiment is available in subsection IV.C.

Thirdly, although not directly visible in this data, it is interesting to look at the impact speed. The impact speed is defined as the velocity towards the obstacle, which is a projection of the velocity on the normal vector of the obstacle, given that the distance constraint is violated. The impact damage scales quadratically with the impact speed [26].
Constraint h_1 had a maximum impact speed of 0.47 m/s during a constraint violation. Constraint h_2 did not crash but had a maximum speed of $0.09 - 0.22$ m/s for $\gamma_1 = 2$ s^{-1} up to $\gamma_1 = 6$ s^{-1} within 5 cm of the distance threshold. The impact speed for constraint h_3 is maximum for the flight with $\gamma_1 = \gamma_2 = 2$ s^{-1}, which had a maximum impact speed of 0.47 m/s.

The results can be summarized as follows:

- h_2 is in the limit equal to h_1 given γ_1 is infinitely high. Since h_1 offered efficient path planning (lowest MD) but bad safety (violated distance constraint with a high crashing speed), lowering γ (from infinity) improved the performance.
- The use of $\ddot{h}$ is challenging since the term depends on the modeled acceleration vector and implies dependence on a numerically calculated Hessian of the ESDF.
- One might argue that constraint h_1 can be better than h_2 if there was more margin in the distance constraint, due to the lower MDs. However, this would involve tuning the size of such a margin. Additionally, this would make the controller less flexible, since it cannot pass through smaller openings anymore.

Selecting appropriate values for γ is crucial to ensure safety and feasible trajectories, striking a careful balance between obstacle avoidance and efficient path planning.

C. Simulation Experiment: Investigating Performance of Second-Order CBF

The goal of this simulation experiment is to investigate the performance of constraint h_3. Where the motivation of constraint h_2 was to limit $\dot{h}$ when the distance is small, the motivation behind constraint h_3 is to limit a combination of $\ddot{h}$ and $\dot{h}$ when the distance is small. This is achieved by nesting CBFs.

The two hypotheses described in subsection IV.B are tested in an experiment for an obstacle where an analytical solution for the ESDF is available, namely a vertical cylinder with a radius of 2 m and infinite height. This implies that the hessian can be calculated exactly. Furthermore, the PX4 simulator is replaced by a simulator that employs the same model as the MPC, albeit integrated with a higher level of precision. This implies that the acceleration vector in the constraint h_3 is the actual acceleration of the simulated drone. Apart from these two changes, the same configurations as for the machine hall experiment are used. The results for the fastest versions are as follows:

- Constraint h_1 reached the waypoint successfully in 6.15 s.
- $h_2, \gamma_1 = 4$ s^{-1}, the fastest version of h_2, reached the waypoint successfully in 5.9 s.
- $h_3, \gamma_1 + \gamma_2 = 10$ s^{-1}, $\gamma_1\gamma_2 = 24$ s^{-2}, the fastest version of h_3, reached the waypoint successfully in 5.75 s.

Thus, there is an indication that h_3 can outperform h_2 and h_1, but a difference between the actual acceleration of the drone and the model-based acceleration and the numerical integration of the hessian make this challenging in a practical scenario. However, this is only based on hypotheses and one experiment. In general, more research into the role of $\ddot{h}$ is required.

V. Simulation Experiment: Aircraft Inspection

A. Methodology

The objective of this second simulation experiment was to test and compare the behavior of the different constraints in a more realistic scenario, i.e., under the influence of noise and a control input delay. The experiment is to fly an inspection flight around a static aircraft. The ESDF is generated based on an OctoMap, as described in Appendix A. The selected voxel size is 0.3 m. The waypoints, the mesh and a flight route are visualized in Figure 3b.

The simulation experiment was repeated for constraints h_1, h_2 and h_3. The values for γ_1 for h_2 are as follows: $\gamma_1 \in \{1, 2\}$ s^{-1}. For h_3 the combinations of $\gamma_1 + \gamma_2, \gamma_1\gamma_2$ were $(5, 4)$,$(8, 12)$ and $(8, 16)$ (s^{-1}, s^{-2}). These were chosen since they were all successful configurations in the previous experiment. The control input delay has been fixed to 100 ms, which is deemed a realistic upper bound for off-board control.

As for the noise, two different intensities of random walk noise were applied to the position, velocity, and orientation of the simulated drone trajectories. The noise comprises two white noise components: a high-frequency component, which was not integrated, and a low-frequency component, which was integrated. The low-frequency component is designed to mimic the drift of the odometry, while the high-frequency component represents the random noise. The low-frequency noise is clamped to a minimum and maximum value.

The noise on the state vector that is injected into the state is as follows:

$$\begin{aligned}\Delta\mathbf{x}^{(i)} =& \text{clamp}\left(\Delta\mathbf{x}^{(i-1)} + \boldsymbol{\omega}_{\text{lf}}^{(i)}(\mu = 0, \boldsymbol{\sigma}_{\text{lf}}), \text{min}, \text{max}\right) \\ &+ \boldsymbol{\omega}_{\text{hf}}^{(i)}(\mu = 0, \boldsymbol{\sigma}_{\text{hf}}),\end{aligned} \tag{4}$$

where $\Delta\mathbf{x}^{(i)}$ is the total noise for timestep i and $\boldsymbol{\omega}$ is a sample from a from a (multidimensional) normal distribution with mean μ and standard deviation $\boldsymbol{\sigma}$.

The following three noise models are used:

1) Zero Noise ($\boldsymbol{\sigma} = 0$): No additional noise is added to the drone trajectories.
2) Low Intensity Noise ($\boldsymbol{\sigma} = \boldsymbol{\sigma}_1$): The low-intensity noise model is designed with the following parameters:
   - Low-frequency component ($\boldsymbol{\sigma}_{lf}$): 1 cm for position, 0.5 cm/s for velocity, and 0.1° for orientation.
   - High-frequency component ($\boldsymbol{\sigma}_{hf}$): 0.5 cm for position, 0.25 cm/s for velocity, and 0.05° for orientation.
   - Minimum and maximum clamping values (min and max): (minus) 10 cm for position, 5 cm/s for velocity, and 1° for orientation.
3) High Intensity Noise ($\boldsymbol{\sigma} = \boldsymbol{\sigma}_2$): The high-intensity noise model incorporates the following parameters:
   - Low-frequency component ($\boldsymbol{\sigma}_{lf}$): 2 cm for position, 1 cm/s for velocity, and 0.2° for orientation.
   - High-frequency component ($\boldsymbol{\sigma}_{hf}$): 1 cm for position, 0.5 cm/s for velocity, and 0.1° for orientation.
   - Minimum and maximum clamping values (min and max): (minus) 20 cm for position, 10 cm/s for velocity, and 2° for orientation.

The output was the MD and the percentage of successful runs. The minimum distance to be kept was set to 1 m. The collision check is based on the ground truth position, which is unknown to the controller. A successful run is defined as a run that did not violate the distance constraint by more than 5 cm and that reached the final waypoint. The MD is the time from take-off till reaching the final waypoint, provided it is reached. The simulation was stopped if the time between two waypoints is longer than 30 s.

Since the experiment involves random variables, every setting is repeated 20 times. From the results of the MD a mean and standard deviation are obtained.

B. Results

The results for the MD and Successful Flight Percentage (SFP) are summarized in Table 2 for the varying noise levels.

Firstly, it can be noted that the CBF constraints h_2 and h_3 exhibit somewhat similar zero-noise performance compared to distance constraint h_1. However, when subjected to noise and delays, it becomes evident that h_2 and h_3 are more robust than h_1. The decrease in performance for an increase in noise levels is smaller for h_2 and h_3 than it is for h_1.

Secondly, it can be concluded that in this experiment a lower value of γ_1 for h_2 or $\gamma_1\gamma_2$ for h_3 tends to achieve a better SFP. These values dictate the allowed velocity towards the obstacle for h_2 or a combination between acceleration and velocity towards and obstacle for h_3. A lower value increases safety, but at the cost of a smaller feasible control input space. For the limited number of tested values, the odds of crashing were higher than of getting stuck. This does come at the cost of an increased MD.

Thirdly, it can be concluded that the extra term for $\ddot{h}$ does not give a performance gain in this experiment. Again, constraint h_3 has less succesful flights than h_2 for a higher MD. In subsection IV.C it is explained how this term can make the performance worse.

			$\mathbf{MD}_{\sigma=0}\,[s]$	$\mathbf{MD}_{\sigma=\sigma_1}\,[s]$	$\mathbf{MD}_{\sigma=\sigma_2}\,[s]$	$\mathbf{SFP}_{\sigma=0}$	$\mathbf{SFP}_{\sigma=\sigma_1}$	$\mathbf{SFP}_{\sigma=\sigma_2}$
h_1			22.4 ± 0.2	22.7 ± 0.4	23.6 ± 0.8	80%	20%	10%
	γ_1 [s^{-1}]							
h_2	1		25.4 ± 0.1	26.0 ± 0.8	26.8 ± 1.5	100%	100%	70%
	2		24.4 ± 0.1	24.5 ± 0.5	25.6 ± 1.1	100%	85%	20%
	$\gamma_1 + \gamma_2$ [s^{-1}]	$\gamma_1\gamma_2$ [s^{-2}]						
h_3	5	4	25.6 ± 0.1	26.1 ± 0.5	27.0 ± 1.2	95%	45%	30%
	8	12	24.5 ± 0.1	24.6 ± 0.4	25.2 ± 1.3	100%	35%	25%
	8	16	23.8 ± 0.1	24.0 ± 0.4	24.5 ± 0.6	95%	30%	15%

Table 2. Results for simulated aircraft inspection of MD and SFP under a fixed delay and varying levels of noise.

In summary, CBF constraints h_2 and h_3 are robuster to noise than the baseline distance constraint h_1. Constraint h_2 outperforms h_3 since the term $\ddot{h}$ makes performance worse. Lastly, a lower value of γ_1 (or $\gamma_1\gamma_2$) works better for this experiment.

VI. Practical Experiment: Aircraft Inspection

A. Methodology

The objective of a real-life aircraft inspection is to validate the controller's performance in a practical, real-world scenario.

The controller's aim was to start beside the aircraft and reach an infeasible waypoint on top of the fuselage. The desired behavior of the controller would be to stop directly on top of the aircraft since that is the closest point to the waypoint, while maintaining at least 3 m to the aircraft.

This is also visualized in Figure 4a. This simple path is chosen to validate the flown routes using back-of-the-envelope calculations. Additionally, the path can be visualized in two dimensions, making it relatively easy to compare the different configurations.

The flight velocity had been limited to 0.5 m/s in every direction and the control input has been limited to 0.1 rad $\approx$ 6 ° in roll and pitch and 0.5 rad/s $\approx$ 3 °/s for the yaw rate for safety. No artificial delay or noise is added.

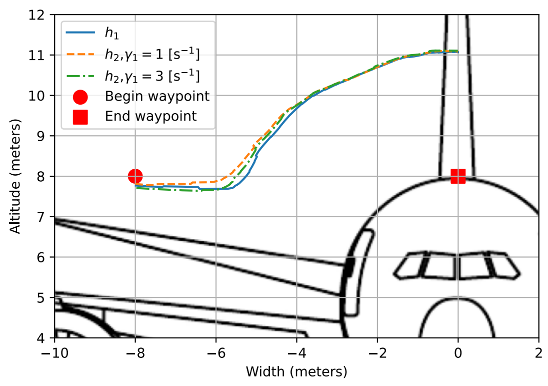
This flight was performed on a Boeing 787-9, since that aircraft was available in the hangar the day of testing. The ESDF's used are pre-generated offline from OctoMap models, as described in Appendix A.

The experiment was repeated for the configurations h_1, h_2 with $\gamma_1 \in \{0.5, 1, 2, 3\}$ s^{-1}. Constraint h_3 is not tested in real life since the extra term $\ddot{h}$ did not increase any performance in simulation. The output is a validation of the performance based on a simple back-of-the-envelope calculation and a comparison of the trajectories flown. The experiment is repeated two times per configuration (10 runs in total). The results are cherry-picked to keep the final output readable.

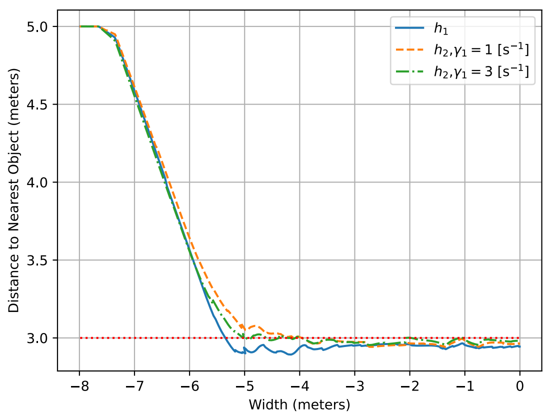
B. Results

The average computational time on the custom companion computer (Intel i5-8365UE processor with 8GB of RAM) is 5.4 ms with a standard deviation of 3.4 ms and a maximum of 24 ms. No significant difference in the computational time was observed between the different constraints. The trajectories flown and the distance to the aircraft are visualized for configurations h_1, h_2 with γ_1 = 1 s^{-1} and h_2 with γ_1 = 3 s^{-1} Figure 4.

Firstly, it can be noted from Figure 4a that the con-



(a) Trajectories flown for constraints h_1, $h_2, \gamma_1 = 1$ s^{-1} and $h_2, \gamma_1 = 3$ s^{-1}. [a]



(b) Distance to nearest obstacle for trajectories flown for constraints h_1, $h_2, \gamma_1 = 1$ s^{-1} and $h_2, \gamma_1 = 3$ s^{-1}.

[a]Drawing from Wikipedia user Julien.scavini `https://en.wikipedia.org/wiki/Boeing_787_Dreamliner`

Figure 4. Results for practical aircraft inspection around a Boeing 787-9 Dreamliner.

troller with constraint $h_2, \gamma_1 = 1$ s^{-1} starts avoiding the aircraft approximately half a meter earlier than distance constraint h_1. The controller with $h_2, \gamma_1 = 3$ s^{-1} starts avoiding the aircraft approximately 20 cm earlier than h_1.

All these results match with a back-of-the-envelope prediction. Assuming a velocity towards the aircraft $\dot{d}(\mathbf{p}) = -0.5$ m/s (maximum flying velocity), constraint $h_2, \gamma_1 = 1$ s^{-1} should start avoiding the aircraft at 3.5 m before the aircraft. Compare this behavior to constraint h_1, that should start avoiding the aircraft at 3 m from the aircraft. For $h_2, \gamma_1 = 3$ the controller should avoid the aircraft at ≈ 3.2 m.

Secondly, in Figure 4a it can also be seen that all three controllers start below their beginning waypoint. This is because the throttle-thrust mapping used was static and did not account for changing battery voltages. Thus, all controllers suffer from a mismatch between their model and reality, which is equivalent to a disturbance in the z-direction. This leads to a static offset from the waypoint and decreased performance in obstacle avoidance. The offset at the end waypoint is lower than at the beginning waypoint, due to the extra cost associated with violating the distance constraint.

Thirdly, in Figure 4b it can be seen that all controllers are somewhat successful at keeping 3 m distance from the aircraft, although none of the controllers managed to keep the 3 m exactly. The maximum violation of h_1 is 11.5 cm. The maximum violation for $h_2, \gamma_1 = 1$ s^{-1} is 6 cm and for for $h_2, \gamma_1 = 3$ s^{-1} it is 4.5 cm. It can be seen the maximum violation of h_1 happens at $y \approx -4$ m, which is in front of the aircraft. For both h_2 constraints, the maximum violation occurs at $y \approx -1.5$ m, which is on top of the aircraft. In other words, all controllers violate the constraint on top of the aircraft due to the throttle-thrust mapping that is off, but the controllers with a first-order CBF constraint manage to avoid the violation in front of the aircraft.

Finally, one last interesting aspect to look at is the maximum impact speed. This results in a maximum impact speed of ≈ 0.17 m/s for h_1 and ≈ 0.11 m/s both for $h_2, \gamma_1 = 1$ s^{-1} and $\gamma_1 = 3$ s^{-1} towards the aircraft.

The difference in impact speed between h_1 and h_2 is to be expected, since a violation of a distance constraint with a higher velocity towards the obstacle is an even bigger violation for CBF constraints.

In summary, CBF constraints h_2 have a higher performance than the baseline h_1. Specifically, the distance violation is smaller and the impact speed during the distance violation is lower. Additionally, the trajectories of the controllers are validated based on simple calculations.

VII. Conclusion & Recommendations

We proposed the use of an ESDF-based CBF constraint in MPC with the goal of improved obstacle avoidance for UAVs.

Simulation experiments show that first-order CBF constraints offer a significant improvement in terms of trajectory planning & tracking efficiency and safety over the baseline distance constraint when using ESDFs to encode obstacles. The results have been validated on a real drone, reaffirming the real-life applicability of the proposed control framework.

In contrast, second-order CBFs suffer from practical limitations like the stricter dependency on an accurate acceleration model and the numerical approximation of the hessian of the ESDF.

The authors conclude that using first-order CBFs combined with ESDFs as a collision constraint is a viable method to improve MPC for navigating complex environments for UAVs. By wrapping the distance constraint in a first-order CBF the controller achieves better safety and scales better with noise, making the controller better suitable for safety-critical applications, such as inspections purposes and search & rescue missions.

Recommendations include finding a method to tune the CBF dynamically depending on the surroundings, overcoming the practical limitation for the second-order CBF and studying the theoretical properties of the proposed controller.

References

[1] Scherer, J., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., Vukadinovic, V., Bettstetter, C., Hellwagner, H., and Rinner, B., "An Autonomous Multi-UAV System for Search and Rescue," Association for Computing Machinery, New York, NY, USA, 2015, p. 33–38. doi: 10.1145/2750675.2750683, URL `https://doi.org/10.1145/2750675.2750683`.

[2] Ruwanpathirana, P., Perera, N., Jayasinghe, G., and Priyankara, P., "Unmanned Aerial Vehicles (UAV) in Precision Agriculture: Applications, challenges, and Future Perspectives," Vol. 7, 2022, p. 36.

[3] Jordan, S., Moore, J., Hovet, S., Box, J., Perry, J., Kirsche, K., Lewis, D., and Tse, Z. T. H., "State-of-the-art technologies for UAV inspections," *IET Radar, Sonar & Navigation*, Vol. 12, No. 2, 2018, pp. 151–164.

[4] Elmokadem, T., and Savkin, A. V., "Towards Fully Autonomous UAVs: A Survey," *Sensors*, Vol. 21, No. 18, 2021. doi: 10.3390/s21186223.

[5] Iskander, A., Elkassed, O., and El-Badawy, A., "Minimum Snap Trajectory Tracking for a Quadrotor UAV using Nonlinear Model Predictive Control," *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Giza, 2020, pp. 344–349. doi: 10.1109/NILES50944.2020.9257897.

[6] Oleynikova, H., Taylor, Z., Siegwart, R., and Nieto, J., "Safe Local Exploration for Replanning in Cluttered Unknown Environments for Microaerial Vehicles," *IEEE Robotics and Automation Letters*, Vol. 3, No. 3, 2018, pp. 1474–1481. doi: 10.1109/lra.2018.2800109.

[7] Zhu, H., and Alonso-Mora, J., "Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments," *IEEE Robotics and Automation Letters*, Vol. 4, No. 2, 2019, pp. 776–783. doi: 10.1109/LRA.2019.2893494.

[8] Singletary, A., Klingebiel, K., Bourne, J., Browning, A., Tokumaru, P., and Ames, A., "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Prague, 2021, pp. 8129–8136.

[9] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, Vol. 5, No. 1, 1986, pp. 90–98.

[10] Stastny, T. J., Garcia, G. A., and Keshmiri, S. S., "Collision and obstacle avoidance in unmanned aerial systems using morphing potential field navigation and nonlinear model predictive control," *Journal of dynamic systems, measurement, and control*, Vol. 137, No. 1, 2015.

[11] Tran, N., Prodan, I., Grøtli, E., and Lefèvre, L., "Potential-field constructions in an MPC framework: application for safe navigation in a variable coastal environment," *IFAC-PapersOnLine*, Vol. 51, No. 20, 2018, pp. 307–312. doi: https://doi.org/10.1016/j.ifacol.2018.11.049, 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[12] Zeng, J., Zhang, B., and Sreenath, K., "Safety-Critical

Model Predictive Control with Discrete-Time Control Barrier Function," *2021 American Control Conference (ACC)*, Online, 2021, pp. 3882–3889. doi: 10.23919/ACC50511.2021.9483029.

[13] Son, T. D., and Nguyen, Q., "Safety-Critical Control for Non-affine Nonlinear Systems with Application on Autonomous Vehicle," *2019 IEEE 58th Conference on Decision and Control (CDC)*, Nice, 2019, pp. 7623–7628. doi: 10.1109/CDC40024.2019.9029446.

[14] Grandia, R., Taylor, A. J., Ames, A. D., and Hutter, M., "Multi-layered safety for legged robots via control barrier functions and model predictive control," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Xi'an, 2021, pp. 8352–8358.

[15] Li, M., Sun, Z., Liao, Z., and Weiland, S., "Moving Obstacle Collision Avoidance via Chance-Constrained MPC with CBF," , 2023.

[16] Meier, L., Tanskanen, P., Heng, L., Lee, G., Fraundorfer, F., and Pollefeys, M., "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, Vol. 33, 2012. doi: 10.1007/s10514-012-9281-4.

[17] Bos, M., Theys, B., Swevers, J., and Pipeleers, G., "Modeling and Identification of Multirotor Drone Dynamics for Onboard MPC Motion Planning," *12th International Micro Air Vehicle Conference*, The International Micro Air Vehicle Conference; Mexico, Puebla, 2021.

[18] Kamel, M. S., Stastny, T., Alexis, K., and Siegwart, R., *Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System*, 2017. doi: 10.1007/978-3-319-54927-9_1.

[19] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., and Tabuada, P., "Control Barrier Functions: Theory and Applications," *2019 18th European Control Conference (ECC)*, Naples, 2019, pp. 3420–3431. doi: 10.23919/ECC.2019.8796030.

[20] Wu, G., and Sreenath, K., "Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds," *2015 American Control Conference (ACC)*, IEEE, Chicago, 2015, pp. 2038–2044.

[21] Oleynikova, H., Taylor, Z., Fehr, M., Nieto, J. I., and Siegwart, R., "Voxblox: Building 3D Signed Distance Fields for Planning," *CoRR*, Vol. abs/1611.03631, 2016.

[22] Hornung, A., Wurm, K., Bennewitz, M., Stachniss, C., and Burgard, W., "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, Vol. 34, 2013. doi: 10.1007/s10514-012-9321-0.

[23] Frison, G., and Diehl, M., "HPIPM: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, Vol. 53, No. 2, 2020, pp. 6563–6569. doi: https://doi.org/10.1016/j.ifacol.2020.12.073, 21st IFAC World Congress.

[24] Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Novoselnik, B., Frey, J., Albin, T., Quirynen, R., and Diehl, M., "acados: a modular open-source framework for fast embedded optimal control," 2019.

[25] Koide, K., Yokozuka, M., Oishi, S., and Banno, A., "Voxelized GICP for Fast and Accurate 3D Point Cloud Registration," EasyChair Preprint no. 2703, EasyChair, 2020.

[26] Sobhani, A., Young, W., Logan, D., and Bahrololoom, S., "A kinetic energy model of two-vehicle crash injury severity," *Accident Analysis & Prevention*, Vol. 43, No. 3, 2011, pp. 741–754. doi: https://doi.org/10.1016/j.aap.2010.10.021.

A. Creating Euclidean Signed Distance Fields from Octomaps

The authors had access to OctoMaps [22] from various aircraft. OctoMap is a probabilistic 3D mapping framework used in robotics and autonomous systems to represent and manage spatial environments with voxel-based octrees. For these collision constraints, these octomaps need to be converted to Voxblox [21] ESDFs to be compatible with the controller. The goal was to make these conversions with minimal changes to the Voxblox source code. OctoMaps have the advantage that they can be ray traced fast, since OctoMaps use octrees as underlying data structure. The general idea is to cast many rays on the octomap and use this mocked lidar data to send to Voxblox. Voxblox will create an ESDF from this mocked lidar data. Specifically:

1) The first step is to create a strategy to cast rays to the OctoMap. The following strategy is proposed. Rays are cast from a big sphere in which the object fits. The object is centered at the origin of the sphere. From evenly spaced points on the sphere, rays are cast inward to the origin (so the direction is the normal of the sphere). Random noise is applied to the direction of the normal vector and several rays are cast from every point. To loop over the sphere with evenly spaced points, we can use spherical coordinates and loop over the angles. The radius of the sphere and the amount of noise is tweaked by hand.
   This results in two pointclouds: A pointcloud with all the raycasted points (on the aircraft), which is visualized in Figure 5 and a pointcloud with all the origin points (on the sphere).
2) The origin pointclouds are moved towards the aircraft in the direction of the original beam. This makes the origin pointcloud an inflated version of the aircraft. This is to limit the integration time of Voxblox. From a practical standpoint, it is already established that we will never exceed a distance of 10 meters from the aircraft in our flights. Therefore, we can factor in this information here. This is visualized in Figure 6.
3) The Voxblox function `integratePointcloud` takes a depth pointcloud in the robot frame of reference and a transformation between the inertial frame and the robot frame T_{GC}, where C is the robot frame of reference and G is the inertial frame of reference. Based on the transformation and the depth pointcloud in the robot frame of reference the ESDF is updated.
   This function is modified slightly so that it can take two pointclouds, the origin pointcloud and the raycasted pointcloud, both already in the inertial frame of reference.

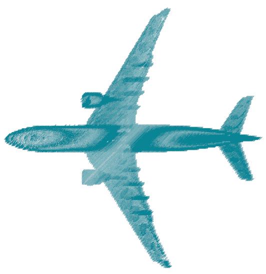


Figure 5. Visualized pointcloud of the casted points of raycasting on an Airbus A330-200 OctoMap.

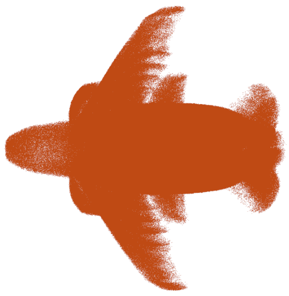


Figure 6. Visualized pointcloud of the origins of raycasting on an Airbus A330-200 OctoMap.

The result is visualized in Figure 7. This figure contains a mesh generated by Voxblox (to be used for visualization) and a slice of the ESDF at a height of $4[m]$. Especially at the tips of the wings some points are missing, but since the distance to the nearest neighbor for a non-filled voxel is set to zero, the controller will never go there. Thus, in general, the result is satisfactory for this use-case.

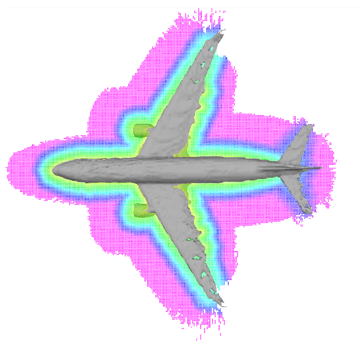


Figure 7. Visualized Airbus A330-200 mesh created by Voxblox and ESDF slice at $z = 4$ m.

Part II

Literature Review

3 Introduction

Autonomous unmanned aerial vehicle's (UAV) are increasingly used, demonstrating their capabilities as a safety inspection tool [1], for cinematography [10], precision agriculture [11], surveillance [12], military purposes [12], 3D mapping [12] and search-and-rescue missions [13]. These applications share the challenge of reaching goal locations/poses while ensuring no collisions without human intervention. This is defined as the navigation problem. Limitations on the vehicle (e.g., onboard sensing capabilities, computational power, and actuation limits) make the navigation problem complex to solve. To solve the navigation problem, designing efficient navigation methods is crucial.

An often-used approach for navigation is performed in a decoupled manner where a high-level control system, the motion planner, plans a safe and feasible trajectory, which is tracked by a low-level control system [2]. For example [3], which first plans a collision-free path using an RRT* path planner, followed by a minimum snap trajectory generator phase that is is tracked by an MPC. Usually there is a second sensor-based obstacle- and collision avoidance backup strategy, e.g., using stereo vision cameras, used to avoid unforeseen obstacles.
A novel research area is real-time trajectory planning techniques, which is made possible by improved hardware and efficient navigation algorithms. This effectively combines the motion planner and the obstacle- and collision avoidance system. Despite being more computationally intensive, this method can handle obstacles and collisions more intelligently. This is especially important for dynamic environments, since it would be impossible to plan a safe trajectory offline. An example of an online trajectory planner, a model predictive controller with dynamic collision constraints, can be seen in [14].

This literature review specifically focuses on using model predictive control for real-time trajectory planning with collision avoidance. Model predictive controllers are considered since they optimize over a time horizon and can include constraints explicitly. The result is an optimal control problem (OCP) that needs to be solved for each time step. This review also highlights the use of control barrier functions to ensure safety, as used in [15] for example. Safety is defined as forward-set invariance, or informally: Starting in the safe set equals staying in the safe set. The aim of this review is to answer the following research question: *How can model predictive control be used for real-time trajectory planning under the constraints of not colliding and staying within predefined state limits?* This question is subdivided into three sub-questions:

- What are the considerations in using a probabilistic method for MPC trajectory planning?
- Can forward-invariance of a safe set be guaranteed?
- How can a numerical solver be selected for the resulting OCP?

This review will be presented in the following structure: Chapter 4 will describe relevant background information. Chapter 5 will provide the reader with an overview of papers published on model predictive controllers (MPC) for online trajectory planning. This chapter is split into a deterministic and probabilistic section and a discussion. After that, using the same structure, Chapter 6 describes the result of several papers on control barrier functions. Chapter 7 describes the results from papers describing a combination between MPC's and CBFs. Chapter 8 will go in more depth on numerical solvers. Lastly, Chapter 9 concludes the report by answering the research questions.

4 Background Information

This chapter will describe relevant background information for this literature review. Firstly, background information on control barrier functions is described. Secondly, a discrete stochastic model is presented. This model (or a similar model) is used for all probabilistic navigation methods, that require a model, presented in this review. Lastly, a multirotor model will be presented. All navigation methods requiring a model use this (or a similar) model.

4.1. Control Barrier Functions

The background information presented in this section is borrowed from [15] and is placed here for completeness's sake.

Throughout this section the following dynamic control-affine system is assumed:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \tag{4.1}$$

with f and g locally Lipschitz, $\mathbf{x} \in X \subset \mathbb{R}^n$ is the state in the state space and $\mathbf{u} \in U \subset \mathbb{R}^m$ is the control input in the space of control inputs.
The motivation for control barrier functions (CBF) is based on control Lyapunov functions (CLF). CLFs are based on Lyapunov stability theorem, that states that if the derivative of the positive definite Lyapunov potential function is always negative, the system is asymptotically stable. The equilibrium point is the point where the Lyapunov potential function is zero. The authors of [16] provide the reader with a review on methods to numerically compute Lyapunov potential functions. Concretely, a function is a CLF if it satisfies:

$$\inf_{\mathbf{u} \in U} [L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u}] \leq -\gamma(V(\mathbf{x})) \tag{4.2}$$

Where V is the Lyapunov potential function. Intuitively this makes sense: $L_f V(\mathbf{x}) + L_g V(\mathbf{x})\mathbf{u}$ is the Lie derivative of V, which is forced to be lower than something negative definite: namely $-\gamma(V(\mathbf{x}))$. This term is zero at the equilibrium point and lower than zero everywhere else.

However, the restriction on a system to always be stable can be too restrictive, leading to empty control spaces. CBFs tackles exactly this problem. The goal is to be forward invariant inside a user defined safe set. I.e., starting in the set, means staying inside of it. This set is usually denoted by $S(\mathbf{x})$ for all $\mathbf{x} \in X$ for which $h(\mathbf{x}) \geq 0$. This can be guaranteed by:

$$\sup_{\mathbf{u} \in U} [L_f h(\mathbf{x}) + L_g h(\mathbf{x}) u] \geq -\gamma(h(\mathbf{x})). \tag{4.3}$$

If a function satisfies this property, it is called a control barrier function. Intuitively this makes sense: $-\gamma(h(\mathbf{x}))$ is zero near the borders and is negative in the set. $L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u}$ is the Lie derivative of $h(\mathbf{x})$. In other words, the derivative is forced to be at least 0 at the borders but can decrease in the set. Since the safe set was defined where the CBF is bigger than zero, starting in the set means staying in the set.
Without going in too much detail, this assumes a relative degree of one between the state constraint and the input. For higher relative degrees more derivatives are needed. This is because the condition on h

needs to be dependent on **u**. E.g., in the case of a position function of h, to always satisfy a condition on the derivative of h, direct velocity control is needed to guarantee safety. This is extensively described in [17].

CBFs can be used as a safety filter on a nominal controller in a minimal invasive way. CLFs and CBFs can also be combined in one framework, namely a CLF-CBF, as for example in [18]. Usually CLFs (ensuring stability) are implemented as a soft constraint, while CBFs (guaranteeing safety, i.e., forward invariance) are implemented as a hard constraint. The resulting framework would then look as follows:

$$\mathbf{u}^*(\mathbf{x}) = \underset{\mathbf{u}\in U, \mathbf{x}\in X}{\operatorname{argmin}} ||\mathbf{u} - \mathbf{u}_{ref}||_2 \tag{4.4}$$

$$\text{s.t. } L_g V(\mathbf{x})\mathbf{u} + L_f V(\mathbf{x}) + cV(\mathbf{x}) - \gamma_1(V(\mathbf{x})) - \delta \le 0, \tag{4.5}$$

$$L_g h(\mathbf{x})\mathbf{u} + L_f h(\mathbf{x}) + \gamma_2(h(\mathbf{x})) \ge 0 \tag{4.6}$$

Where $\mathbf{u}_{ref}$ is the reference input from a nominal controller and δ is a slack variable used to make the CLF a soft constraint.
Note the nominal controller can also be omitted in this scenario taking the CLF as cost function. I.e., guaranteeing safety while optimizing stability.

Technically speaking there are two types of control barrier functions, namely zeroing control barrier functions (the function h as described previously and reciprocal control barrier functions B:

$$B(\mathbf{x}) = 1/h(\mathbf{x}) \tag{4.7}$$

However, in recent literature, only zeroing control barrier functions are used. This literature review will not consider reciprocal control barrier functions, but only zeroing control barrier functions. Note that zeroing control barrier functions are not actually barrier functions, since they do not approach infinity at the borders of the domain.

4.2. Discrete Stochastic Model

The following stochastic model is described in [5], but used exactly or in very comparable form for all probabilistic motion planning strategies in this review that require a model. The model is repeated here for completeness.
Assume n UAV's operating in a three-dimensional environment. Each UAV i is modeled as follows:

$$\mathbf{x}_i^{k+1} = \mathbf{f}_i\left(\mathbf{x}_i^k, \mathbf{u}_i^k\right) + \omega_i^k, \quad \mathbf{x}_i^0 \sim \mathcal{N}\left(\hat{\mathbf{x}}_i^0, \Gamma_i^0\right), \tag{4.8}$$

Where $\mathbf{x}_i^k = \left[\mathbf{p}_i^k, \mathbf{v}_i^k, \phi_i^k, \theta_i^k, \psi_i^k\right]^T \in X_i \subset \mathbb{R}^{n_x}$ denotes the state of the UAV (global position, global velocity and orientation) and $\mathbf{u}_i^k \in U_i \subset \mathbb{R}^{n_u}$ denotes the control input of the robot at time k. X_i and U_i are the state space and control space respectively. The dynamic equations (i.e., the function **f**) are presented in the next section for a multicopter model. The initial state $\mathbf{x}_i^0$ is defined to be a Gaussian random variable with mean $\hat{\mathbf{x}}_i^0$ and covariance Γ_i^0. This mean and covariance is typically given by a state estimator.

4.3. Multirotor Modeling

In this section the multirotor model proposed in [19] is described. This model or a similar model is used by all motion planning strategies in this review. The model is repeated here for completeness. A picture of the coordinate frame used can be seen in Figure 4.1. Note that this model assumes a low-level controller of roll, pitch, yaw rate and thrust. This is common for Parrot multicopters[1], DJI multicopters[2] and PixHawk controllers [20]. In other words $\mathbf{u} = \left[\phi_c, \theta_c, T_c, \dot{\psi}_c\right]^T \in \mathbb{R}^4$, where ϕ_c, θ_c, T_c, $\dot{\psi}_c$ are the commanded roll angle, pitch angle, mass normalized thrust (positive in upward body frame direction) and yaw rate, respectively. Additionally, the model assumes a first order model for the roll and yaw rate, no time delay for the commanded inputs and a linear relationship between the velocity and the air resistance. The

[1]https://www.parrot.com/assets/s3fs-public/2022-01/whitepaperanafiai.pdf

[2]https://developer.dji.com/onboard-sdk/documentation/introduction/homepage.html

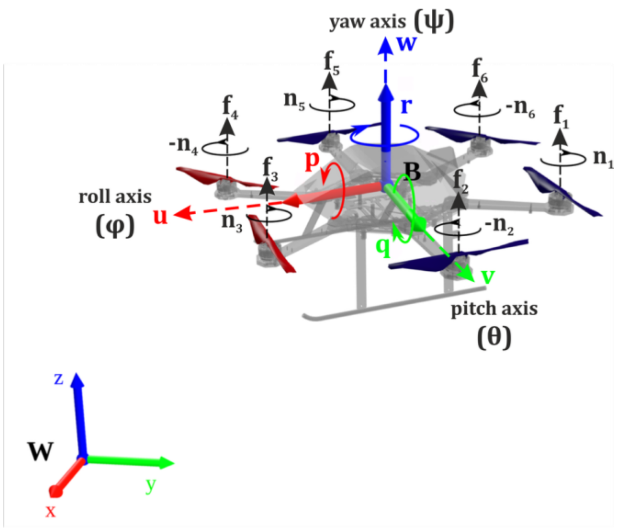


Figure 4.1: The coordinate system used in this model of a multicopter with global inertial frame w and body fixed frame b. [3]

dynamic equations are given as follows:

$$
\begin{aligned}
\dot{\mathbf{p}}(t) &= \mathbf{v}(t) \\
\dot{\mathbf{v}}(t) &= \mathbf{R}(\psi,\theta,\phi)\begin{pmatrix}0\\0\\T_c\end{pmatrix} + \begin{pmatrix}0\\0\\-g\end{pmatrix} - \begin{pmatrix}A_x & 0 & 0\\ 0 & A_y & 0\\ 0 & 0 & A_z\end{pmatrix}\mathbf{v}(t) \\
\dot{\phi}(t) &= \frac{1}{\tau_\phi}\left(K_\phi\phi_c(t) - \phi(t)\right) \\
\dot{\theta}(t) &= \frac{1}{\tau_\theta}\left(K_\theta\theta_c(t) - \theta(t)\right) \\
\dot{\psi}(t) &= \dot{\psi}_c(t)
\end{aligned}
\tag{4.9}
$$

The state $\mathbf{x} \in \mathbb{R}^9$ was defined in Section 4.2. $\mathbf{R}(\psi,\theta,\phi)$ is the rotation matrix from the body frame (u,v,w) of reference to the global frame of reference (x,y,z). g is the gravitational acceleration. T is the mass normalized thrust, A_x, A_y and A_z indicate the mass normalized drag coefficients τ_ϕ, K_ϕ and τ_θ, K_θ are the time constant and gain of inner-loop behavior for roll angle and pitch angle, respectively.
It can be noted that the system is control-affine.
Note that this is the continuous version of the dynamics equations, which can be made discrete by using a multiple shooting approach.

[3]https://www.autonomousrobotslab.com/multirotor-dynamics.html

5 Model Predictive Control

This section covers both deterministic and probabilistic model predictive control (MPC) structures for online trajectory planning. MPC's offer the advantage that they optimize over a time horizon, instead of a single timestep (such as a PD controller). Additionally, constraints can be included explicitly. The goal is to minimize a cost function (usually consisting of a stage state cost, a terminal state cost, an input cost and sometimes an input smoothness cost) given a UAV model and a reference state. The problem is usually constrained in input (actuation limits), position (collision constraints) and sometimes in state (safety limits). The chapter is divided into three sections: Deterministic MPC's, Probabilistic MPC's and a discussion on the provided literature.

5.1. Deterministic Model Predictive Control

Figure 5.1 shows the control block diagram of such a deterministic model predictive controller. In this figure, $\mathbf{x}_{ref}$ is the reference state, $\mathbf{x}$ is the state, which is assumed fully known, $\mathbf{u}^*$ is the optimum control input and $\mathbf{y}$ is the sensor output of the drone.

The authors of [14] implemented such a nonlinear MPC (NMPC). The authors proposed a classification scheme to distinguish between trajectory types (static, linear, projectile motion) to feed the MPC the future positions of obstacles as part of the optimization problem. Obstacles are modeled as spheres. The resulting problem was implemented using the PANOC solver using a separate laptop and sending the control inputs to the drone.
The NMPC scheme was tested experimentally using a motion detection system and compared to an artificial potential field method and an NMPC scheme without an object classifier (assuming all objects are static). As expected, the drone can avoid collisions more efficiently if the solver has information about the future position. In the case of a bouncing ball as obstacle, only the NMPC scheme with trajectory classifier avoided a collision consistently.

In [21] tries to solve a similar problem, using the same solver. It is assumed that the future positions

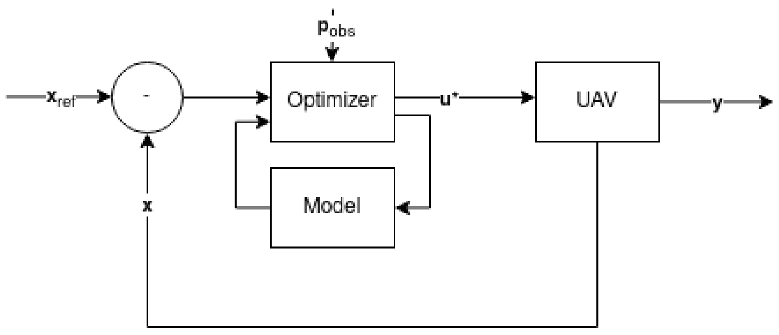


Figure 5.1: Block diagram of MPC assuming full state knowledge.

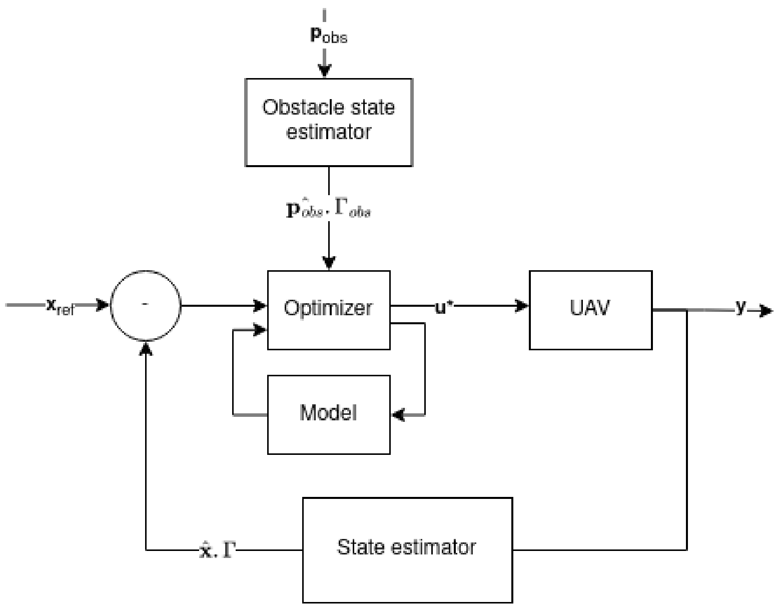


Figure 5.2: Block diagram of MPC assuming a state estimator.

of obstacles are known, removing the need for a trajectory classifier. The proposed methodology can accommodate obstacles of arbitrary geometry. Additionally, the NMPC controls the thrust coefficient, rather than the thrust (using a thrust estimator). This effectively means the NMPC is robust to mass changes.
The resulting control structure was tested experimentally using a motion detection system. The first experiment conducted was to fly between two waypoints, while avoiding obstacles. The drone managed to avoid the obstacles consistently. Secondly the thrust coefficient was tested by a position hold task up and until the battery is empty. This worked for as long as the battery could produce enough power to hover. Note the total thrust of a quadcopter decreases with decreasing battery level.

The authors of [22] propose a similar framework used for cinematography. The main difference with the previous paper is that the quadcopter now also considers a gimbal pitch and yaw angle, and the authors force the quadcopter to follow a virtual rails (input), in analogy to physical camera cranes. The solver used was the commercial FORCES Pro software.
Experiments were conducted using a motion system and a laptop to solve the optimization problem to film multiple challenging movie shots with multiple drones. Since the authors use a sequential planning approach, it is shown that the computational cost grows linearly with the number of drones. Additionally, a qualitative experiment is conducted where an expert gives feedback on the cinematographic capabilities of the drone. The feedback includes that using this method can simplify creating certain shots, especially in dynamic environments.

5.2. Probabilistic Model Predictive Control

This section covers probabilistic model predictive controllers. The problem is like the deterministic case, where a cost function will be minimized given a model and constraints. The difference is that the position of the vehicle and the obstacles are unknown, only the probability distributions are known, which are the result from a state estimator. A block diagram of such a framework is shown in Figure 5.2. In this figure, $\mathbf{x}_{ref}$ is the reference state, $\hat{\mathbf{x}}$ is the mean of the state, Γ is the covariance matrix of the states, $\mathbf{u}^*$ is the optimum control input and $\mathbf{y}$ is the sensor output of the drone. This scenario is deemed more realistic since the state is not fully known due to sensor noise.

In [23] a MPC framework is combined with a potential field like cost function. Potential fields do not provide guarantees and are thus combined with hard constrained in this paper to ensure no collisions.

Specifically, both the potential field term and the hard constraint are dependent on the uncertainty in position of both the vehicle and the obstacle. Note that this uncertainty is increasing with time. The state estimator used is an Extended Kalman Filter. The solver used was generated by ACADO toolkit [24].
The resulting framework was tested experimentally using two UAVs and a motion system with injected noise. The first experimental test was to try to crash a manually flown UAV into a position holding drone. The second experimental task was for two UAVs to follow crossing reference trajectories. In both experiments the minimum distance hard constraint was not violated.

The authors of [5] treat the uncertainty explicitly by implementing a chance constrained NMPC (CCNMPC). The obstacles were modeled as ellipsoids. For the state estimation an unscented Kalman filter (UFK) was used for vehicles and a linear Kalman filter (LFK) for obstacles. The solver used is the commercial FORCES Pro software.
The authors have tested the resulting framework experimentally using a motion system with injected noise. Firstly, a position swap experiment between two quadcopters shows a significant improvement over both [23], which framework is described in the previous paragraph, and a deterministic NMPC used for cinematography [22], which framework is described in Section 5.1. Secondly, the framework was tested in a close environment with two humans and two quadcopters. The minimum distance requirement was not violated. Lastly, a comparison was made between three different multirobot planning settings (distributed, decentralized, and centralized) for a position swap experiment: In the case of the distributed setting the minimum distance requirement was violated. Since the distributed setting assumed a constant velocity model, this caused a mismatch with reality giving rise to a collision (given the used thresholds). The other two settings were successful in this experiment.

In [25] the authors implement a similar CCNMPC framework but propose a conservative solution to save computational cost. Rather than integrating the overlap of probability distributions a bound is put on the probability distributions where overlap is not allowed. The authors have also implemented a sample average approximation for comparison, which can be seen as a method to integrate the probability distributions.
The resulting framework was tested in simulation for planar UAV's passing through a gap. The conservative solution offers a computational improvement (factor of 50) over the sample average approximation implementation for a decrease in cost function of 4 percent. Additionally, the authors produce empirical evidence using simulation that the probabilistic constraints are better suited than robust control constraints in case there is a model of the stochastic disturbance.

Lastly, the authors of [26] propose an NMPC framework for motion planning. However, this time the authors assume a 3D point cloud obtained from a lidar. The authors note three main contributions: First, the ability to reveal underlying planes from a point cloud using a subspace clustering method. Second, the ability to incorporate these planes in an NMPC. Lastly, by considering uncertainty in the localization using the Shannon's entropy to construct the weight matrix for the waypoint error.
Three simulation experiments were conducted. In the first experiment the plane segmentation is compared to three other clustering techniques. The proposed technique achieves comparable results while reducing the computational cost by an order of magnitude. In the second experiment the micro aerial vehicle (MAV) was instructed to fly in a corridor environment, namely the "*house_maze*" world available in the VoxBlox repository[1]. Noise was injected into the system. The MAV managed to keep the minimum distance needed for several velocities. The authors compared the results to a potential field strategy that did not reach the final goal. In the third experiment the MAV was instructed to hold a waypoint in a confined space with injected state noise. It was shown that the MPC with adaptive weights has less oscillations and the non-adaptive weights variant has more oscillations and sometimes crashes into one the walls.

[1]https://github.com/ethz-asl/voxblox

5.3. A Discussion on Model Predictive Control

The main challenge in using an NMPC for all these papers is computational cost. All the methods rely on a powerful onboard computer or a separate laptop, while all using specialized solver. The solvers will be discussed in more detail in Chapter 8. The horizon is between $1-2\,[s]$ for all papers, solving the problem at approximately $10-20\,[Hz]$. While these numbers are hard to compare for different hardware, they do give the reader an indication of an order of magnitude.

While probabilistic MPC's were more robust to noise, they are also more computationally expensive than their deterministic counterparts, due to evaluating collision probabilities and (potentially) handling chance constraints. While sampling-based evaluation strategies can achieve exact results asymptotically, they are computationally intensive. For a full overview of computing waypoint collision probabilities and how to combine them for a trajectory, the reader is referred to [27, p. 8-14].

6 Control Barrier Functions

Another technique that can be used for motion planning is using control barrier functions. As described in Section 4.1, control barrier functions are used to construct a constraint to ensure safety. Safety is defined as being forward invariant, i.e., if the vehicle starts in the safe set, it will stay in it. This constraint is also called a safety barrier certificate (SBC) in literature. These terms will be used interchangeably. The chapter is divided into three sections: Deterministic control barrier functions, probabilistic control barrier functions and a discussion on the provided literature.

6.1. Deterministic Control Barrier Functions

In [15], the authors provide the readers with both theory, which forms the basis for the background section Section 4.1 and four applications, one of which will be highlighted here. This is the application of dynamic balancing of a Segway. The SBC is related to the tipping angle. Secondly there are constraints on the forward velocity and the angular rate. The authors perform a reachability analysis to combine these constraints in one safety barrier certificate to guarantee feasibility. The experiment compares a PD controller with and without the safety filter. The PD controller could be made unstable with an external disturbance, while the PD controller + safety filter managed to stay upright under the same disturbance. A video of this experiment can be found here: `https://youtu.be/RYXcGTo8Chg`.

In [28] the authors compare three SBCs: A nominal certificate, a relaxed certificate (with extra optimized parameters) and a feasible certificate (with a backup strategy), as can be seen in Table 6.1. This was applied to a multirobot system. The authors also designed a mechanism to tackle deadlocks. The resulting framework results in a quadratic programming problem since the model used was linear. The solver used was the MATLAB quadprog solver.

Table 6.1: Comparison Across Three Decentralized Safety Barrier Certificates [28]

Type of certificate	Guaranteed safety	Guaranteed feasibility	Admissible control space
Nominal	✓	×	Standard
Relaxed	✓	×	Enlarged
Feasible	✓	✓	Shrunken

The authors compare the three different versions of the decentralized safety barrier certificates. From a go-to-goal task experiment, it was concluded that the relaxed SBCs are the least invasive to the nominal controller of the three. The total time of intervention of a simulated position swap experiment was $2.9[s]$ for the relaxed SBC versus $4.5[s]$ and $5.4[s]$ for the nominal and relaxed SBC, respectively. The resulting framework has also been experimentally tested using unmanned ground vehicles (UGV) in a two-dimensional setting for a position swap. A video of this can be seen here: `https://www.youtube.com/watch?v=-WUkzik1_VQ`.

In [29] the authors again implement a minimal invasive safety filter and a backup controller to prove guaranteed feasibility. The resulting framework was tested in simulation using a 17-dimensional model for two quadrotors with the task of avoiding each other in a decentralized way (no communication). The solver used was a non-public implementation of the OSQP solver [30]. The safety barrier function is related to the distance between the quadrotors and the back-up policy is to level the drone with zero velocity. The minimum distance constraint was not violated. The code can be found at: `https://github.com/DrewSingletary/uav_sim_ros`.

In [31] the authors take another approach by constructing a CLF-CBF to control a planar quadrotor for obstacle and collision avoidance. The CLF constraint is used as a soft constraint to optimize stability and the CBF to guarantee safety. The resulting problem is solved pointwise in time as a sequential quadratic programming problem. The main contribution of the authors is to augment the control barrier function from a distance function (used by the previous three papers) with an orientation term, to make the constraint explicitly dependent on the moment control input. The optimal control problem was solved using an unspecified interior point method.
The resulting controller with augmented CBF was tested in simulation and compared to a controller without CBF in simulation with extra constraints on the maximum force and moment. The former managed to successfully reach a goal pose without collisions, while the latter did not. Unfortunately, the authors did not compare the resulting controller to a controller with unaugmented CBF. Because of that, it is hard to conclude what exactly the contribution of this extra term is.

6.2. Probabilistic Control Barrier Functions

In [32] the term probabilistic SBC (PrSBC) is introduced. Rather than enforcing a safety barrier certificate, there is a constraint on the chance that the safety barrier certificate is violated considering a mean and variance on the state. The PrSBC is defined as a convex set. Additionally, it is only constructed for single integrator dynamics.
The authors conduct two simulation experiments with injected noise: Firstly, the authors compare their PrSBC safety filter to a conventional SBC safety filter in a multirobot two-dimensional position swap task, with dynamic obstacles. The PrSBC controller managed to perform the task consistently successfully, while the conventional SBC controller did not. Secondly, the authors tested their PrSBC safety filter in a multi UAV simulator, with the task of writing letters in the air while avoiding dynamic obstacles. Also in this experiment a minimum separation distance was kept consistently.

In [27] the SBC method of [28] (described in the previous section) is extended from the deterministic case to the probabilistic case. This results in chance constrained SBCs (CC-SBC). CC-CBCs differ from PrSBCs in the sense that they are a non-convex set and hold for general nonlinear control affine dynamical systems (not only single integrator systems). Like PrSBCs, they define a chance that the safety barrier certificate is violated considering a mean and variance on the state. The CC-SBCs is approximated with quadratic constraints resulting in a quadratically constrained quadratic program (QCQP). The barrier certificates are used as a safety filter with the goal of minimally changing the nominal controller's input, while ensuring safety. The resulting framework has been tested in simulation and compared to it's deterministic counterpart for a position swap experiment with injected noise. The CC-SBC's framework outperforms the SBC's framework. While the probabilistic variant manages to consistently retain the minimum required distance (over multiple runs), the deterministic variant does not manage to do this.

To the authors' best knowledge these are the only two examples of probabilistic control barrier function in literature.

6.3. A Discussion on Control Barrier Functions

In most of the above papers, the control barrier function h was a function on the distance between the vehicle and another vehicle or obstacle. In these cases, the control barrier function handles collision avoidance. Additionally, most of the papers implement the control barrier certificates as a safety filter. Note that these kind of problems are quadratic programming problems since there is no nonlinear model. CBFs are used both in deterministic and probabilistic contexts. Probabilistic CBFs were shown to be more robust to noise, however limited literature is available on this.

The main challenge with control barrier functions is feasibility of the optimized problem (i.e., existence of

a solution) given a limited control space. While satisfying the SBC implies safety, the counterpart does not hold. I.e., there is no guarantee that the SBC is feasible when it is still possible to stay in the safe set. This was the motivation for creating back-up controllers. This implies that control barrier functions must be constructed so that such a back-up controller can guarantee safety. E.g., the minimum distance between two vehicles is based on the maximum deceleration. The dependence of the choice of γ in this feasibility problem is visualized in Figure 7.1 for a discrete control barrier function with γ as a constant.
This problem can also be mitigated by defining a viable safe set. I.e., a safe set in which it is always possible to go to another point in the safe set giving the control input space and SBC. This process is described in [33]. Computing this safe set comes down to solving the Hamilton-Jacobi partial differential equation, which scales exponentially. Solving the Hamilton-Jacobi partial differential equation for a full drone model (with nine states and four control inputs) is currently not possible in a reasonable amount of time.
This feasibility problem undermines the theoretical background on the safety guarantees of CBFs.

7 Combining Model Predictive Control with Control Barrier Functions

In this chapter a combination of MPC and CBF is discussed. This chapter is divided into three sections: A section describing the motivation, a section of the work published and a discussion on a combination of the two. Unfortunately, the available literature is limited to only distance control barrier function, effectively replacing the collision constraints.

7.1. Motivation

Control barrier functions can guarantee safety (forward invariance) if used correctly. Control barrier functions are used as a constraint, without providing a measure on the quality of a state. Therefore, control barrier functions are always combined with another cost function/ controller to fulfill control purposes. Previously mentioned strategies included the use of CBFs as safety filters that minimally invade a nominal control, while guaranteeing safety and the combination with a control Lyapunov function (CLF-CBF). Note that both strategies optimize point wise in time and can thus be seen as a greedy approach. MPC's offer the advantage of optimizing over a time horizon, as described in the chapter introduction of Chapter 5. This chapter explores the concept of combining these two advantages.

7.2. Literature on MPC combined with CBF

In [34] the authors combine an MPC and a CLF-CBF in a multi-rate controller. The MPC is used as a high-level planner at a low frequency and the CLF-CBF is used as a low-level tracker at a high frequency. The authors prove the resulting closed-loop system is forward set invariant, thus safe. The CLF-CBF is designed with the full model, while the MPC uses a simplified model.
The resulting framework is benchmarked against linear and non-linear MPC's in a simulated Segway experiment. The authors show that in the case where the rod angle is constrained to a certain angle, the multi-rate controller is successful at the task, while both the linear and nonlinear MPC's fail.

In [7] the authors describe how to use continuous CBF constraints in a non-affine NMPC framework. Since the continuous control barrier function ensures safety over an infinite timeline, it is possible to decrease the horizon, thus saving computational power. The paper's main contribution is a description of how exponential control barrier functions can be constructed for systems with a relative degree higher than one for non-affine systems, namely using pole placement on a new dynamical system.
The resulting framework was validated using high fidelity vehicle dynamics and traffic/sensor models. The task is to avoid an obstacle having torque and acceleration as input (relative degree 2). The experiment shows that a conventional NMPC could not avoid the obstacle, while the NMPC combined with the CBF constraint could avoid it.

In [6] three frameworks are compared: DCLF-DCBF, discrete control barrier functions combined with a discrete Lyapunov controller, DMPC-DCBF, discrete model predictive control combined with discrete control barrier functions and MPC-DC, discrete conventional MPC. In both MPC cases the cost function

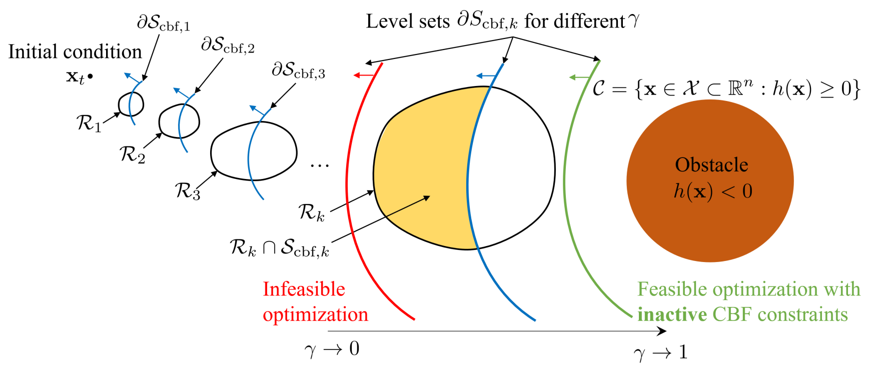

$\partial\mathcal{S}_{\text{cbf},1}$
Initial condition
$\mathbf{x}_t$
$\partial\mathcal{S}_{\text{cbf},2}$
$\partial\mathcal{S}_{\text{cbf},3}$
Level sets $\partial\mathcal{S}_{\text{cbf},k}$ for different γ
$\mathcal{C} = \{\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n : h(\mathbf{x}) \geq 0\}$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_3$
...
Obstacle
$h(\mathbf{x}) < 0$
$\mathcal{R}_k$
$\mathcal{R}_k \cap \mathcal{S}_{\text{cbf},k}$
Infeasible optimization
Feasible optimization with **inactive** CBF constraints
$\gamma \to 0$
$\gamma \to 1$

Figure 7.1: The reachable set R_k and the safe set $S_{cbf,k}$ as a function of timestep k for initial condition x_t. $\delta S_{cbf,k}$ are the level sets of the safe set S. The union of R_k and $S_{cbf,k}$ are feasible solution to the MPC-CBF problem. The feasibility is shown for different γ. [6]

is a Lyapunov potential function (which is minimized). The writers hypothesize that DMPC-DCBF can outperform MPC-DC. This is because MPC-DC will only act in the context of obstacle avoidance if the distance norm is small. This is not the case for a CBF constraint since it is a constraint on one or multiple derivatives. The conversion from continuous to discrete control barrier functions is described by [35].
The authors note that if a time horizon of $N = 1$ is used for the MPC-CBF, the resulting framework is like the CLF-CBF framework. The difference is that the CLF constraint in the CLF-CBF framework is in the cost function of the MPC-CBF framework (with $N = 1$).
Additionally, the authors note that if $\gamma = 1$ is used for the MPC-CBF, the safety barrier certificate is like the distance constraint in the MPC framework. The difference is that the safety barrier certificate for $\gamma = 1$ is the distance constraint, but for the next horizon time step, while for the MPC it is of the current time step. The effect of the choice of γ is nicely illustrated in Figure 7.1.
Both analyses have also been confirmed in simulated car obstacle avoidance experiment, where MPC-CBF with $N = 1$ provides similar output to CLF-CBF and MPC-CBF with $\gamma = 1$ provides similar output to MPC-DC. Additionally, it is shown in a second race-car simulation that MPC-CBF outperforms both other frameworks, confirming the hypothesis of the writers. The solver used was ipopt.
However, the authors do not consider that their relative degree is higher than one (again a constraint on position with acceleration and torque as input, thus $r = 2$). According to [7] they would need a second derivative and first derivative on the control barrier function to guarantee safety, which is not used in their implementation. Only a discrete variant of the first derivative is considered.

7.3. A Discussion on MPC-CBF

While [6] provided a discussion on the theoretical consequences of combining MPC with CBF, it is not validated in a realistic setting on hardware. MPC combined with CBF is a novel research area with many unknowns. Specifically, the unknowns are as follows:
Firstly, all three papers published on a combination of MPC and CBF assume a perfectly known state. An opportunity for future work is to combine CC-SBCs, as described in [27] with an MPC. This would involve the use of a state estimator and would change the resulting framework from safety guarantees to chance constrained safety guarantees.
Secondly, all three papers test their framework only in simulations. Since computational cost is the number one challenge in model predictive control and control barrier functions can be seen as a way to artificially expand the horizon, it would be interesting to see the performance in a realistic setting. It would be extra interesting to see this in combination with the previous point.
Lastly, in [36] the authors combine a control Lyapunov function with NMPC. Using a CLF term in the cost function as a terminal condition is used by multiple authors before, e.g. [37]. However, the main contribution of this paper is to experiment with several ways to include the CLF in the NMPC. The author deems it interesting to also see such a study being performed on (N)MPC combined with CBF.

8 Numerical Optimization

This section aims to provide an overview of numerical optimization for NMPC purposes. First, a problem formulation is given. After that literature on numerical solvers is discussed. Lastly, a comparison is made between the discussed numerical solvers.

8.1. Problem Formulation

The following finite-horizon problem is considered:

$$\underset{\mathbf{u}\in U,\mathbf{x}\in X}{\text{argmin}} \sum_{n=0}^{N-1} \ell_n\left(x_n, u_n\right) + \ell_N\left(x_N\right) \tag{8.1}$$

$$\text{s.t. } \mathbf{x}_0 = \bar{\mathbf{x}} \tag{8.2}$$

$$\mathbf{x}_{n+1} = f\left(\mathbf{x}_n, \mathbf{u}_n\right) \tag{8.3}$$

$$z_i\left(\mathbf{x}_n, \mathbf{u}_n\right) \leq 0, i = 1, 2, 3..., Z \tag{8.4}$$

Where ℓ_n and ℓ_N are the stage cost and terminal cost, respectively. z is a state and or input constraint and $\bar{\mathbf{x}}$ is the known initial condition for state $\mathbf{x}$. Furthermore, it is assumed that ℓ_n, ℓ_N, f and z are smooth, possibly non-convex functions.
Ideally the goal is to find the global optimum considering the constraints, however finding a local optimum is more realistic given the real-time application.

8.2. Literature on Numerical Optimization

This section contains an overview of literature on numerical optimization. This section will be split into two subsections, first order methods and second order methods. First order methods only use the first derivative, second order methods use both the first and the second derivative. Note that the algorithms will not be discussed elaborately, rather their advantages and disadvantages will be laid out. Additionally, their relative speed is compared.

8.2.1. First Order Methods

First order methods utilize the gradient to find the search direction in each iteration. Since there is no need to compute the Hessian, iterations are relatively cheap. However, this does come at the cost of slower convergence rates. Despite slower convergence rates, first order methods have proved to be useful in practice for solving NMPC problems [38].
The authors of [39] propose an algorithm, PANOC, used for solving optimal control problems for NMPC. PANOC proposes an alternative for sequential quadratic programming (SQP). SQP requires iterative procedures, which can take long if the problem is ill-conditioned. Ill-conditioning is customary in NMPC problems due to nonlinear dynamics and the horizon length. The authors propose a line-search algorithm with Forward-Backward iterations with Newton-Type steps. The resulting algorithm is completely matrix-free and thus has low memory requirements. This enables embedded applications. The authors compared

their algorithm to forward-backward splitting, MATLABS FMINCON function (SQP)[1], and IPOPT [40] (an interior point method) in terms of computational speed. The PANOC algorithm outperforms the algorithms for the given problem. An open-source implementation of PANOC is available in the c++ library opEn[2]. opEn relies on generated code, thus no extra compiler optimization can be performed, and flexibility is limited.

8.2.2. Second Order Methods

Second order methods use not only the gradient, but also the Hessian is needed for each iteration. This makes iterations more expensive, but also improves the convergence rate [38]. Two second order methods are discussed: Sequential quadratic programming and interior point methods.

Sequential Quadratic Programming
Sequential Quadratic Programming (SQP) methods linearize the nonlinear constraints resulting in a quadratic programming method (QP). These QP's are solved consecutively. In every QP the active constraints are treated as equality constraints. The solution to the QP determines which constraints are active in the next QP. This solving process can be made efficiently by exploiting the structure of the optimal control problem (OCP).
An advantage of SQP methods over interior point methods, which are discussed in the next subsection, is that they can be warm started. I.e., the previous solution, which is readily available in MPC problems, can be used as an initial guess [41].
In [42] the authors propose a SQP algorithm: qpOASES. qpOASES is a dense solver and is implemented in the acados software package [43]. The acados software package for optimal control and estimation problems. Acados is the successor of the ACADO toolkit. Acados is faster, produces exactly the same results and does not rely on pre-generated code as compared to ACADO toolkit. It can be noted that states are only dependent on the previous states and the input. Using this property is called condensing. qpOASES is a dense solver which uses condensing by solving smaller QP problems.
Another active set method is qpDUNES [44]. qpDUNES exploits the block-banded structure of OCP's [41]. qpDUNES is also implemented in the acados software package. The authors compare their algorithm to among others FORCES [45], the predecessor of FORCES Pro, which is discussed in the next subsection on interior point methods. The active set method outperforms the FORCES code generation tool with an order of magnitude in terms of computational speed. In [41] qpDUNES is shown to outperform qpOASES in terms of computational speed. Additionally, the authors show HPMPC, the predecessor of HPIPM, an interior point method discussed in the next subsection outperforms qpDUNES and qpOASES in terms of computational speed for the same benchmark.

Interior Point
In interior point (IP) methods a barrier function is used to implement constraints. Usually, a logarithmic barrier function is used to enforce a variable to be larger than zero. If the weight of this barrier term is small, the solution to the modified problem approaches the original solution. Note that a barrier function is a function that goes to infinity at the boundaries, not to be confused with a control barrier function.
An example of such an implementation is in ipopt. Ipopt is an open-source (Eclipse Public License) software library[3], which implements an interior point line search filter method [40]. The authors compare their algorithm against LOQO[46] and KNITRO[4], two previously often used interior point methods. Ipopt shows favorable performance as compared to the other two.
Another alternative is FORCES Pro[5]. FORCES Pro is commercial software that generates c code for optimized NMPC embedded problems. The paid license states that disclosing computational performance is not allowed. It is mentioned here since it is used in MPC literature. E.g., in [27] and [22] (both discussed earlier).
In [47] the authors propose another interior point method for optimal control problems: HPIPM. HPIPM uses partial condensing, a combination of condensing (as in qpOASES) and exploiting the block-banded

[1]https://www.mathworks.com/help/optim/ug/fmincon.html
[2]https://alphaville.github.io/optimization-engine/
[3]https://coin-or.github.io/Ipopt/
[4]https://tomopt.com/docs/knitro/tomlab_knitro002.php
[5]https://www.embotech.com/products/forcespro/overview/

structure (as in qpDUNES). HPIPM is one of the solvers implemented in the acados software package[43]. The authors compare the performance to qpOASES (also implemented in acados), which it outperforms in terms of computational speed.

8.3. A Discussion on Numerical Solvers

Previously one first order method (PANOC), and five second order methods, of which two SQP methods (qpOASES, qpDUNES) and three IP methods (ipopt, FORCES Pro, HPIPM) were discussed. This section aims at advising the reader which solver to use.

Ipopt was shown to be outperformed multiple times, thus this solver is not advised. FORCES Pro has a paid license, which can either be a disadvantage or an advantage since it comes with support. HPIPM was shown to be the fastest second order solver in a benchmark test. HPIPM, qpOASES and qpDUNES are all implemented in the open-source library acados, and thus it is easy to switch between these solvers. PANOC, implemented in the open-source library opEn, can also be a satisfactory solution, since it is matrix-free and thus has low memory requirements, making it a solid choice for embedded applications. Moreover, PANOC is a first order method, thus no second derivatives are needed. While there is no direct speed comparison between PANOC and HPIPM, both are shown to be about two orders of magnitude faster than ipopt [43], [39].
In the end the choice of solver depends on a trade-off between flexibility, memory usage and speed. PANOC and HPIPM score about equal in terms of speed. PANOC has lower memory requirements, being completely matrix free. HPIPM, implemented in acados, scores higher on flexibility, since it does not rely on pre-generated code and is implemented in a modular software package as compared to PANOC implemented in opEn.

9 Conclusion

The aim of this review was to provide the reader with an overview of how model predictive control can be used for trajectory planning. That is, to answer the research question: *How can model predictive control be used for real time trajectory planning under the constraints of not colliding and staying within predefined state limits?* This research question was divided in three sub-questions:

- What are the considerations in using a probabilistic method for MPC trajectory planning?
- Can forward-invariance of a safe set be guaranteed?
- How can a numerical solver be selected for the resulting OCP?

These three sub-questions are answered in order, followed by an answer to the research question and a paragraph on future work.

Uncertainties in the state estimation can be handled by using chance constraints. Since the state of the vehicle is usually unknown, a state estimator is used. The output of this state estimator is a probability distribution. This probability distribution can be used to handle chance constraints. There are examples of this both for collision constraints as well as control barrier functions. Probabilistic methods are computationally more intensive than their deterministic counterpart, but also more robust to noise. Therefore, the main considerations are the sensor quality and the computational power available.

Control barrier functions can be used to guarantee safety, where safety is defined as forward invariance. However, there is no guarantee there is always a feasible solution given a limited control space. This feasibility problem can be solved by designing a back-up controller that always produces a control input that ensures safety. An example of this is an emergency brake maneuver. Note that this does imply that a compatible control barrier function is needed, i.e., a distance that takes the maximum deceleration of the vehicle into account. For state constraints an emergency maneuver would be to hover. Therefore, given that a back-up controller is allowed and control barrier functions are constructed in a smart way, safety can be guaranteed. This can also be done with regular hard constraints, however there are indications that control barrier functions offer improved performance.

Several numerical solvers are compared to solve the nonlinear optimal control problem. The considerations to select these solvers are computational speed, flexibility, and the size of the available memory. Two solvers stood out based on these criteria. The first solver algorithm is PANOC, which is implemented in the open-source code generator opEn. PANOC is a first order solver (only requiring the Jacobian matrix), which is completely matrix-free. Therefore, it has low memory requirements making it ideal for embedded applications. The second solver is the HPIPM solver in the acados package. Acados is a modular software package that implements multiple second order solvers (also requiring the Hessian matrix): qpOASES, qpDUNES and HPIPM. Acados is more flexible since it does not rely on generated code. Switching between the implemented solvers is simple and code maintainability is high. Both software packages exploit the structure of the OCP and are used multiple times throughout the literature on MPC's for trajectory planning. In terms of computational speed the performance is similar can be conluded from an indrect comparison.

Coming back to the main research question: Model predictive control can be used for online trajectory planning in a probabilistic and deterministic framework. Probabilistic frameworks are more robust to noise, but also require more computational power. Safety can be guaranteed with control barrier functions, if used correctly. While literature on using CBFs as MPC constraints is sparse, there are clear signs that using control barrier functions can increase performance over using conventional hard constraints. The main challenge for MPCs for online trajectory planning is computational cost, therefore a specialized numerical solver is needed. Two excellent solvers, used multiple times throughout literature, include the first order PANOC implementation of opEn, which relies on generated code and the second order HPIPM solver of acados, which is more flexible since it does not rely on generated code. Both are open-source software libraries and score similar in terms of computational speed.

Answering these research questions also raised new questions which form the basis for future research. Especially in the combination with MPC and CBF there are still a lot of unknowns. Specifically, the author deems research in the following topics interesting:

- The effect of selecting a certain γ (not necessarily a constant) in the control barrier function.
- The effect of using a control barrier function as constraint on all states, instead of only on the position state for collision avoidance.
- The feasibility of a combination between CBF and MPC on hardware in a real-time application.
- The effect on the feasibility problem on the usability of control barrier functions in the context of MPC.
- The combination of probabilistic control barrier functions with MPC altogether.
- The stage in which to apply the control barrier function constraint. I.e., on which timestep to apply the constraint.
- The feasibility of applying a combination of MPC and CBF for trajectory planning on a quadcopter.

Note that the first two topics are not specific to a combination between MPC and CBF, but also hold for CBF's in general. However, they are still included since they can be studied in the context of MPC and different contexts. These items are summarized in the following research question: *What are the main considerations when combining MPC with CBF for trajectory planning?* This research question forms the basis of a master thesis that is to be done directly after this literature report.

References

[1] Javier Irizarry et al. “Usability Assessment of Drone Technology as Safety Inspection Tools”. In: *Electronic Journal of Information Technology in Construction* 17 (Sept. 2012), pp. 194–212.

[2] Taha Elmokadem et al. “Towards Fully Autonomous UAVs: A Survey”. In: *Sensors* 21.18 (2021). DOI: 10.3390/s21186223. URL: https://www.mdpi.com/1424-8220/21/18/6223.

[3] Avraiem Iskander et al. “Minimum Snap Trajectory Tracking for a Quadrotor UAV using Nonlinear Model Predictive Control”. In: *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*. 2020, pp. 344–349. DOI: 10.1109/NILES50944.2020.9257897.

[4] Helen Oleynikova et al. “Safe Local Exploration for Replanning in Cluttered Unknown Environments for Micro-Aerial Vehicles”. In: *CoRR* abs/1710.00604 (2017). arXiv: 1710.00604. URL: http://arxiv.org/abs/1710.00604.

[5] Hai Zhu et al. “Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 776–783. DOI: 10.1109/LRA.2019.2893494.

[6] Jun Zeng et al. “Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function”. In: *2021 American Control Conference (ACC)*. 2021, pp. 3882–3889. DOI: 10.23919/ACC50511.2021.9483029.

[7] Tong Duy Son et al. “Safety-Critical Control for Non-affine Nonlinear Systems with Application on Autonomous Vehicle”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 7623–7628. DOI: 10.1109/CDC40024.2019.9029446.

[8] Ruben Grandia et al. “Multi-layered safety for legged robots via control barrier functions and model predictive control”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 8352–8358.

[9] Ming Li et al. *Moving Obstacle Collision Avoidance via Chance-Constrained MPC with CBF*. 2023. arXiv: 2304.01639 [eess.SY].

[10] Ioannis Mademlis et al. “A multiple-UAV architecture for Autonomous Media Production”. In: *Multimedia Tools and Applications* (2022). DOI: 10.1007/s11042-022-13319-8.

[11] Pranith Ruwanpathirana et al. “Unmanned Aerial Vehicles (UAV) in Precision Agriculture: Applications, challenges, and Future Perspectives”. In: 7 (June 2022), p. 36.

[12] Alessia Vacca et al. “Drones: military weapons, surveillance or mapping tools for environmental monitoring? The need for legal framework is required”. In: *Transportation Research Procedia* 25 (2017). World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016, pp. 51–62. DOI: https://doi.org/10.1016/j.trpro.2017.05.209. URL: https://www.sciencedirect.com/science/article/pii/S2352146517305100.

[13] Jürgen Scherer et al. “An Autonomous Multi-UAV System for Search and Rescue”. In: DroNet ’15. Florence, Italy: Association for Computing Machinery, 2015, pp. 33–38. DOI: 10.1145/2750675.2750683. URL: https://doi.org/10.1145/2750675.2750683.

[14] Bjorn Lindqvist et al. “Nonlinear MPC for collision avoidance and control of uavs with dynamic obstacles”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6001–6008. DOI: 10.1109/lra.2020.3010730.

[15] Aaron D. Ames et al. “Control Barrier Functions: Theory and Applications”. In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3420–3431. DOI: 10.23919/ECC.2019.8796030.

[16] Peter Giesl et al. “Review on computational methods for Lyapunov functions”. In: *Discrete and Continuous Dynamical Systems - B* 20.8 (2015), pp. 2291–2331. DOI: 10.3934/dcdsb.2015.20.2291. URL: /article/id/ee7e4111-9a13-4e44-95c8-14c5557278fc.

[17] Wei Xiao et al. “Control Barrier Functions for Systems with High Relative Degree”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 474–479. DOI: 10.1109/CDC40024.2019.9029455.

[18] Xiangru Xu et al. “Robustness of Control Barrier Functions for Safety Critical Control**This work is partially supported by the National Science Foundation Grants 1239055, 1239037 and 1239085.” In: *IFAC-PapersOnLine* 48.27 (2015). Analysis and Design of Hybrid Systems ADHS, pp. 54–61. DOI: https://doi.org/10.1016/j.ifacol.2015.11.152. URL: https://www.sciencedirect.com/science/article/pii/S2405896315024106.

[19] Mina Samir Kamel et al. “Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System”. In: May 2017. DOI: 10.1007/978-3-319-54927-9_1.

[20] Lorenz Meier et al. “PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision”. In: *Autonomous Robots* 33 (Aug. 2012). DOI: 10.1007/s10514-012-9281-4.

[21] Elias Small et al. “Aerial navigation in obstructed environments with embedded nonlinear model predictive control”. In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3556–3563. DOI: 10.23919/ECC.2019.8796236.

[22] Tobias Nägeli et al. “Real-Time Planning for Automated Multi-View Drone Cinematography”. In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: 10.1145/3072959.3073712. URL: https://doi.org/10.1145/3072959.3073712.

[23] Mina Kamel et al. “Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 236–243. DOI: 10.1109/IROS.2017.8202163.

[24] Boris Houska et al. “ACADO toolkit—An open-source framework for automatic control and dynamic optimization”. In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312.

[25] Daniel Lyons et al. “Chance constrained model predictive control for multi-agent systems with coupling constraints”. In: *2012 American Control Conference (ACC)*. 2012, pp. 1223–1230. DOI: 10.1109/ACC.2012.6315153.

[26] Sina Sharif Mansouri et al. “A Unified NMPC Scheme for MAVs Navigation With 3D Collision Avoidance Under Position Uncertainty”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5740–5747. DOI: 10.1109/LRA.2020.3010485.

[27] H. Zhu. “Probabilistic Motion Planning for Multi-Robot Systems”. PhD thesis. Delft University of Technology, 2022. DOI: 10.4233/UUID:52038194-6CDF-4098-8723-852EB68DFD00. URL: http://resolver.tudelft.nl/uuid:52038194-6cdf-4098-8723-852eb68dfd00.

[28] Li Wang et al. “Safety Barrier Certificates for Collisions-Free Multirobot Systems”. In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674. DOI: 10.1109/TRO.2017.2659727.

[29] Yuxiao Chen et al. “Guaranteed Obstacle Avoidance for Multi-Robot Operations With Limited Actuation: A Control Barrier Function Approach”. In: *IEEE Control Systems Letters* 5.1 (2021), pp. 127–132. DOI: 10.1109/LCSYS.2020.3000748.

[30] Bartolomeo Stellato et al. “OSQP: An operator splitting solver for quadratic programs”. In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672.

[31] Guofan Wu et al. “Safety-critical control of a planar quadrotor”. In: *2016 American Control Conference (ACC)*. 2016, pp. 2252–2258. DOI: 10.1109/ACC.2016.7525253.

[32] Wenhao Luo et al. “Multi-Robot Collision Avoidance under Uncertainty with Probabilistic Safety Barrier Certificates”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 372–383. URL: https://proceedings.neurips.cc/paper/2020/file/03793ef7d06ffd63d34ade9d091f1ced-Paper.pdf.

[33] Thomas Gurriet et al. “Towards a Framework for Realizable Safety Critical Control through Active Set Invariance”. In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. 2018, pp. 98–106. DOI: 10.1109/ICCPS.2018.00018.

[34] Ugo Rosolia et al. “Multi-Rate Control Design Leveraging Control Barrier Functions and Model Predictive Control Policies”. In: *IEEE Control Systems Letters* 5.3 (2021), pp. 1007–1012. DOI: 10.1109/LCSYS.2020.3008326.

[35] Ayush Agrawal et al. “Discrete control barrier functions for safety-critical control of discrete systems with application to Bipedal Robot Navigation”. In: *Robotics: Science and Systems XIII* (2017). DOI: 10.15607/rss.2017.xiii.073.

[36] Ruben Grandia et al. “Nonlinear Model Predictive Control of Robotic Systems with Control Lyapunov Functions”. In: July 2020. DOI: 10.15607/RSS.2020.XVI.098.

[37] A. Jadbabaie et al. “Unconstrained receding-horizon control of nonlinear systems”. In: *IEEE Transactions on Automatic Control* 46.5 (2001), pp. 776–783. DOI: 10.1109/9.920800.

[38] Alberto Diaz Dorado. “Efficient Convex Quadratic Optimization Solver for Embedded MPC Applications”. PhD thesis. 2018. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-240421.

[39] Lorenzo Stella et al. “A simple and efficient algorithm for nonlinear model predictive control”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (2017), pp. 1939–1944.

[40] Andreas Wächter et al. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical Programming* 106 (2006), pp. 25–57.

[41] Dimitris Kouzoupis et al. “Recent advances in quadratic programming algorithms for nonlinear model predictive control”. In: *Vietnam Journal of Mathematics* 46.4 (Sept. 29, 2018), pp. 863–882. DOI: 10.1007/s10013-018-0311-1. URL: https://cdn.syscop.de/publications/Kouzoupis2018.pdf.

[42] Joachim Ferreau et al. “qpOASES: A parametric active-set algorithm for quadratic programming”. In: *Mathematical Programming Computation* 6 (Dec. 2014). DOI: 10.1007/s12532-014-0071-1.

[43] Robin Verschueren et al. “acados: a modular open-source framework for fast embedded optimal control”. In: (Oct. 2019).

[44] J. V. Frasch et al. “A parallel quadratic programming method for dynamic optimization problems”. In: *Mathematical Programming Computation* 7.3 (2015), pp. 289–329.

[45] Andrea Zanelli et al. “FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs”. In: *International Journal of Control* 93 (Apr. 2017), pp. 1–26. DOI: 10.1080/00207179.2017.1316017.

[46] Robert Vanderbei. “Loqo: An Interior Point Code For Quadratic Programming”. In: *Optimization Methods & Software* 11–12 (Mar. 2002). DOI: 10.1080/10556789908805759.

[47] Gianluca Frison et al. “HPIPM: a high-performance quadratic programming framework for model predictive control”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 6563–6569. DOI: https://doi.org/10.1016/j.ifacol.2020.12.073. URL: https://www.sciencedirect.com/science/article/pii/S2405896320303293.