Learning-Based Optimal Control for Safe Quadrotor Navigation in Windy Environments

Johanna Probst

Literature Survey

Delft University of Technology

Delft Center for Systems and Control

Learning-Based Optimal Control for Safe Quadrotor Navigation in Windy Environments

Master of Science Literature Survey

Johanna Probst

September 8, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Table of Contents

Chapter 1

Introduction

1-1 Background

Unmanned aerial vehicles (UAVs), such as drones, are becoming increasingly important in our society. Tasks of UAVs include search and rescue, environment monitoring, security surveillance, transportation, and inspection [1]. Developing autonomous drones that are capable of executing complex missions poses various challenges for research. One of these challenges is safe navigation under real-world conditions where drones are exposed to external disturbances and other sources of uncertainty. Especially in complex, cluttered environments such as forests or dense urban areas, unforeseen disturbances can cause the drone to deviate significantly from its course and potentially collide with nearby objects. The inclusion of external disturbances for reliable trajectory generation and tracking is therefore an important aspect to consider for real-world applications.

1-2 Problem Definition

This project tackles the problem of safe navigation of UAVs under real-world disturbances. In particular, the focus lies on aerodynamic disturbances such as wind, which vary spatially in an area inhabited by trees or other densely spaced obstacles. A quadrotor, which is a special configuration of UAV with four rotors, is given a start and desired end position within this environment. The goal is to generate and follow a safe trajectory that takes wind disturbances into account by adding them to the system model. In doing so, the quadrotor should be able to find and execute an optimal course through the environment, maximizing the progress to the goal while avoiding crashes with any nearby obstacles. Meanwhile, physical limitations of the platform such as motor speeds should not be exceeded. The complexity of this problem is illustrated in Figure 1-1.

Navigating the quadrotor through the environment requires motion planning, to generate a feasible trajectory, and control, to track the given trajectory. After giving a general introduction to both motion planning and control, this study focuses on optimal control methods that provide a way to combine motion planning and control into one problem while taking into account the disturbance information and physical limitations.

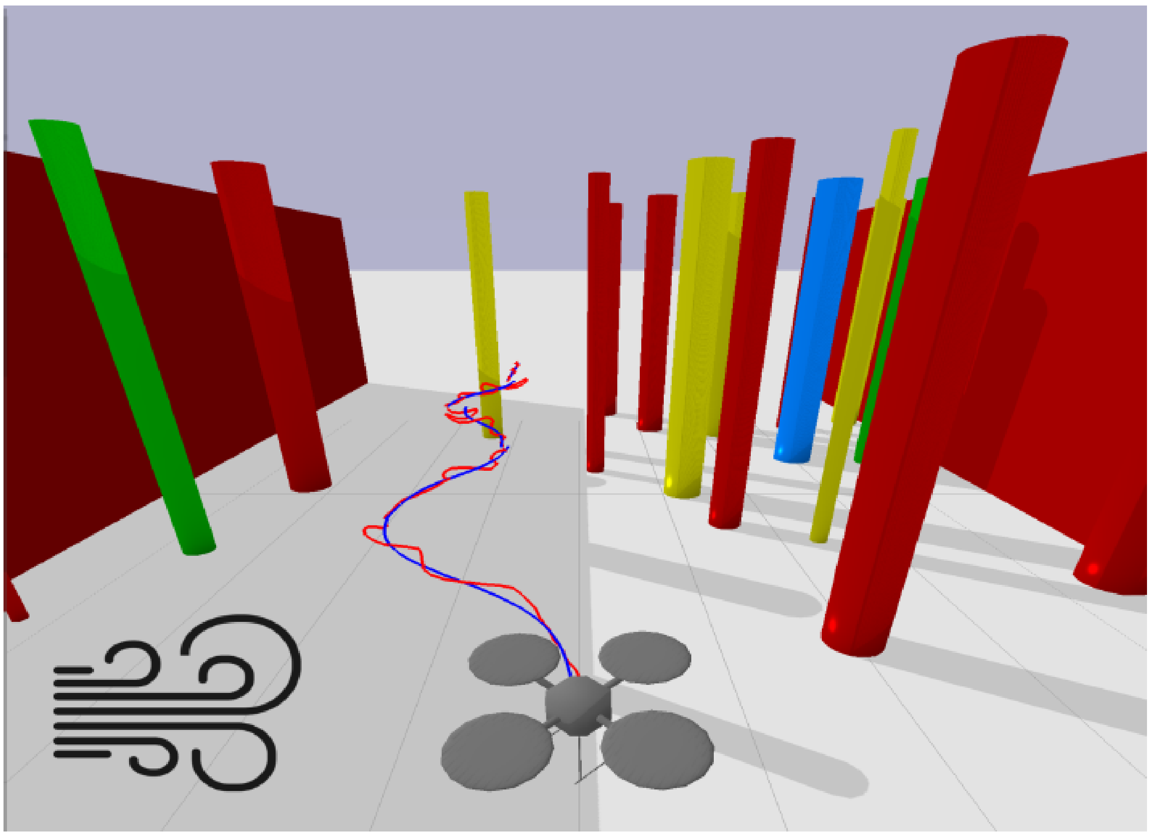


Figure 1-1: A quadrotor flying through a cluttered environment while being subject to wind disturbances [2].

Optimal control methods rely on an accurate system model to describe the behavior of the quadrotor. For modeling wind disturbances this survey goes beyond classical approaches of assuming constant or bounded disturbances and instead focuses on modern learning-based methods to improve the wind-disturbance model. Among the learning-based methods, there is a focus on Gaussian Process (GP) models, which not only learn the nominal disturbance but also provide an uncertain description of the disturbance estimate.

Using GPs for modeling the wind disturbances opens up further questions on how to train the model, how to choose training inputs and targets, how to deal with large amounts of data, and how to incorporate data that becomes available online during flight. This report presents an overview of GPs and collects related research to answer these questions.

The chosen methods will be implemented in simulation and on a real-world quadrotor platform, shown in Figure 1-2, to validate the approach.

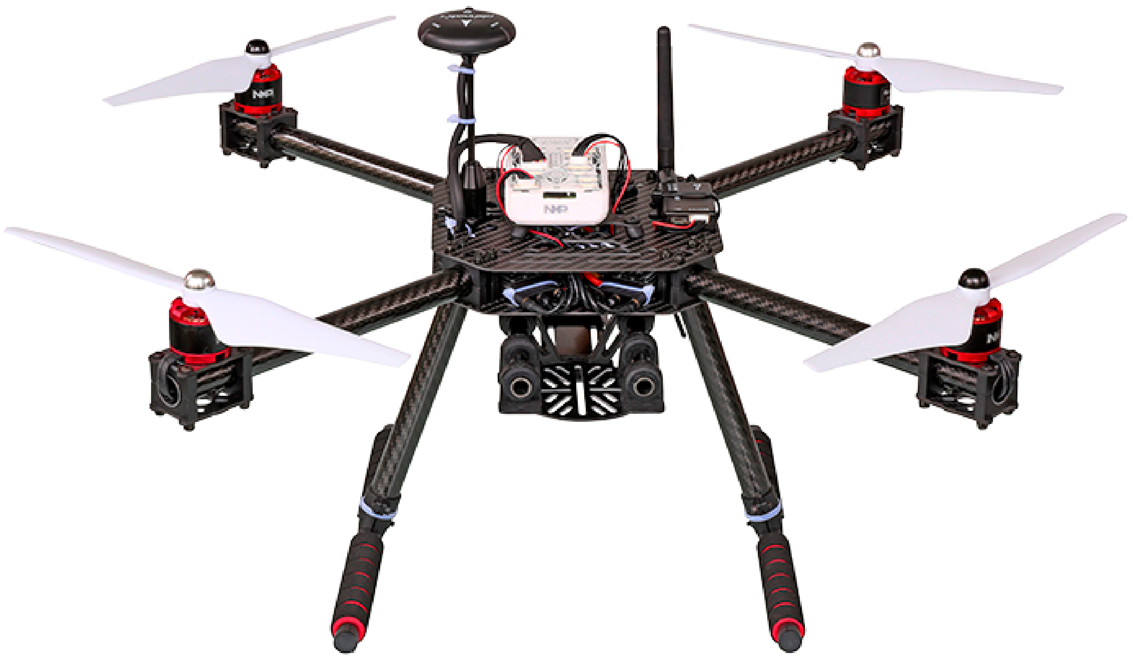


Figure 1-2: The quadrotor platform used for this research [3].

1-3 Report Outline

The report is structured as follows:

- Chapter 2 gives an overview of the quadrotor model and learning-based methods to describe the wind disturbances.
- Chapter 3 goes into the mathematical details of GPs and how to tackle practical issues that arise with GP dynamic models.
- Chapter 4 gives an overview of motion planning and control methods before going into the details of how to use optimal control methods such as model predictive control and incorporate the disturbance information.
- Chapter 5 gives a summary of the findings of this literature review, the problem definition, and a planning of the master thesis following up on this review.

Chapter 2

Quadrotor Model

For the design of the motion planner and controller, the behavior of the quadrotor to certain inputs has to be known to describe the evolution of the future quadrotor states. In this chapter, different methods to derive the quadrotor model are presented. First, the quadrotor equations of motion (EOM) are derived based on first-principle models. In this context, it is also discussed how an inner-loop attitude controller can be integrated into the dynamic model of the quadrotor and how the system dynamics can be linearized. Since the first-principle model is only an approximation of the system dynamics, methods for deriving or improving the quadrotor model-based on measurement data in a learning environment are presented below. In particular, the study discusses how model learning can be used to describe wind disturbances and compares it with classical approaches to wind disturbance estimation.

2-1 First Principle Model

First principle modeling techniques have been extensively studied in literature to derive a mathematical quadrotor model. They are based on simplifying assumptions of real-world dynamics.

2-1-1 Quadrotor Equations of Motion

Newton-Euler equations and Euler-Lagrange formalisms are used in most studies of the quadrotor model (e.g. in [4], [5] and [6]). Moreover, the quadrotor dynamics can also be represented based on quaternions as in [7] and [8]. These representations are often found for attitude control design and aggressive flight maneuvers when the Gimbal lock associated with the Newton-Euler representation becomes a problem [9]. However, since this work deals with the aspect of motion planning that takes place in the position control loop, the focus is on deriving the Euler representation of quadrotor dynamics.

Quadrotor configuration The quadrotor consists of a rigid body cross-frame and four fixed-pitch rotors at each end. Each rotor's speed can be varied individually to control the output

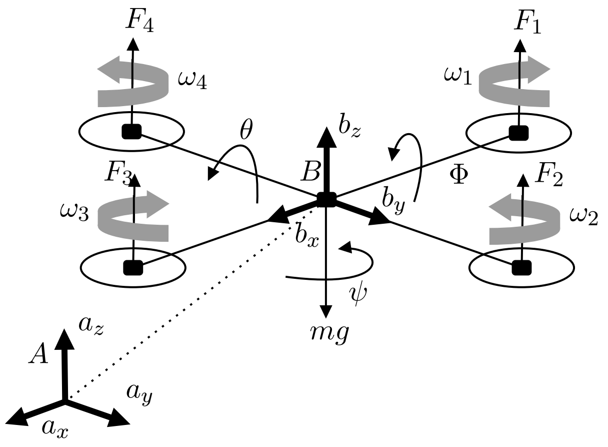


Figure 2-1: Quadrotor configuration with world frame A and body frame B.

states of the quadrotor. The output states are the Cartesian position states x, y and z, and the three Euler angles roll, pitch, and yaw. Having four inputs and six outputs, the quadrotor is an under-actuated system. The quadrotor is treated as a rigid body with mass m and inertia $I \in \mathbb{R}^{3\times 3}$. Two reference frames are defined as shown in Figure **??** to describe the quadrotor motion: The position of the quadrotor $\mathbf{p} = [x, y, z]^T \in \mathbb{R}^3$, and velocity $\mathbf{v} = [v_x, v_y, v_z] \in \mathbb{R}^3$ are expressed in the world frame $\{A\}$. The roll, pitch, and yaw angles of the quadrotor are denoted by ϕ, θ, and ψ, respectively. They specify the orientation of the quadrotor body frame $\{B\}$.

Rotation matrix Different Euler angle representations can be used to describe the rotation from the world frame to the body frame of the quadrotor $\{B\}$. Using a Z - X - Y Euler angle configuration the rotation matrix is described as:

$$R = R_{B\to A} = R_Z(\psi)R_X(\phi)R_Y(\theta) \tag{2-1a}$$

$$= \begin{bmatrix} c\psi c\theta & c\psi s\phi s\theta - c\phi s\psi & s\psi s\phi + c\psi c\phi s\theta \\ c\theta s\psi & c\phi c\psi + s\psi s\phi s\theta & c\phi s\theta s\psi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\phi c\theta \end{bmatrix}. \tag{2-1b}$$

Newton-Euler equations The longitudinal dynamics are then described through Newton's EOM as:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{2-2a}$$

$$m\dot{\mathbf{v}} = -mg\mathbf{a}_3 + R\mathbf{T} \tag{2-2b}$$

where $\mathbf{a}_3 = [0, 0, 1]^T \in \{A\}$ is the unit vector in the z axis of the world frame, and $\mathbf{T} = [0, 0, T]^T \in \{B\}$ is the generated thrust. Expanding (2-2b) results in:

$$m\dot{v_x} = T(\sin\phi \sin\psi + \cos\phi \sin\theta \cos\psi) \tag{2-3a}$$

$$m\dot{v_y} = T(-\sin\phi \cos\psi + \cos\phi \sin\theta \sin\psi) \tag{2-3b}$$

$$m\dot{v_z} = T(\cos\phi \cos\theta) - mg. \tag{2-3c}$$

The rotational dynamics are described by Euler's EOM:

$$\dot{R} = R\Omega_\times \tag{2-4a}$$

$$I\dot{\Omega} = -\Omega \times I\Omega + \tau. \tag{2-4b}$$

where $\Omega = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T \in \{B\}$ is the angular velocity of the quadrotor with respect to $\{A\}$, and $\tau \in \{B\}$ denotes the moments applied to the quadrotor. $\Omega_\times$denotes the skew-symmetric matrix:

$$\Omega_\times = \begin{bmatrix} 0 & -\dot{\psi} & \dot{\theta} \\ \dot{\psi} & 0 & -\dot{\phi} \\ -\dot{\theta} & \dot{\phi} & 0 \end{bmatrix}. \tag{2-5}$$

Since torques τ be measured directly during flight, they cannot serve as model inputs. Instead, they can be expressed in terms of the motor speeds ω_i. This results in the system description [10]:

$$\ddot{\phi} = \frac{(I_{yy} - I_{zz})\,\dot{\theta}\dot{\psi}}{I_{xx}} + \frac{U_1}{I_{xx}} \tag{2-6a}$$

$$\ddot{\theta} = \frac{(I_{zz} - I_{xx})\,\dot{\phi}\dot{\psi}}{I_{yy}} + \frac{U_2}{I_{yy}} \tag{2-6b}$$

$$\ddot{\psi} = \frac{(I_{xx} - I_{yy})\,\dot{\phi}\dot{\theta}}{I_{zz}} + \frac{U_3}{I_{zz}} \tag{2-6c}$$

with control inputs

$$U_1 = db\left(\omega_4^2 - \omega_2^2\right) \tag{2-7a}$$

$$U_2 = db\left(\omega_1^2 - \omega_3^2\right) \tag{2-7b}$$

$$U_3 = k\left(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2\right). \tag{2-7c}$$

The thrust coefficient is b, k is the aerodynamic drag coefficient and d is the moment arm of the propellers. Similarly, the torque T can be expressed in terms of the rotor speeds:

$$T = \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} \omega_i^2. \tag{2-8}$$

Cascaded control approach In practice, the control is often split such that a low-level attitude controller is present as an inner-loop, while a model-based trajectory tracking controller is running as an outer-loop [11]. The inner-loop attitude controller is stabilizing the system, which can avoid numerical problems in optimization software and allows for a lower sampling rate of the position controller [12]. This also introduces a separation layer to keep critical code running despite any failure in the high-level computer [13].

Attitude model To achieve accurate trajectory tracking, the high-level position controller must consider the inner loop system dynamics. These dynamics can either be calculated if the control structure is known or can be approximated by a simple linear model of the form:

$$\dot{\phi} = \frac{1}{\tau_\phi}\left(k_\phi \phi_d - \phi\right) \tag{2-9a}$$

$$\dot{\theta} = \frac{1}{\tau_\theta}\left(k_\theta \theta_d - \theta\right) \tag{2-9b}$$

$$\dot{\psi} = \dot{\psi}_d \tag{2-9c}$$

$$T = T_d \tag{2-9d}$$

with system inputs $\mathbf{u} = [\phi_d, \theta_d, \psi_d, T_d]$. The coefficients of the first-order model can be found using system identification. While most research uses first-order models (e.g. [5], [14] and [15]), it is also possible to consider higher order models, leading to a higher approximation accuracy at the expense of a more complex model.

Aerodynamic effects on the quadrotor can be included in the mathematical quadrotor model. The two main effects when flying are blade flapping and induced drag [16]. One can account for these effects by adding an external drag force proportional to the system velocity:

$$\dot{m}\mathbf{v} = -mg\mathbf{a}_3 + R\mathbf{T} - D\mathbf{v} \tag{2-10}$$

where $D = \operatorname{diag}(k_D, k_D, k_D)$ and k_D is the drag coefficient.

Derived quadrotor Model The resulting quadrotor model is summarized below.

$$\begin{aligned}
\dot{\mathbf{p}} &= \mathbf{v} && \text{(2-11a)}\\
m\dot{v}_x &= T_d\left(sin(\psi)sin(\phi) + cos(\phi)sin(\theta)cos(\psi)\right) - k_D v_x && \text{(2-11b)}\\
m\dot{v}_y &= T_d\left(-sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi)\right) - k_D v_y && \text{(2-11c)}\\
m\dot{v}_z &= T_d\left(cos(\phi)cos(\theta)\right) - mg - k_D v_z && \text{(2-11d)}\\
\dot{\phi} &= \frac{1}{\tau_\phi}(k_\phi \phi_d - \phi) && \text{(2-11e)}\\
\dot{\theta} &= \frac{1}{\tau_\theta}(k_\theta \theta_d - \theta) && \text{(2-11f)}\\
\dot{\psi} &= \dot{\psi}_d && \text{(2-11g)}
\end{aligned}$$

Linear quadrotor model Assuming small attitude angles, the quadrotor dynamics model can be linearized around its hovering condition with $\dot{v}_z = 0$. Furthermore, the vehicle heading is aligned with the inertial frame x-axis, such that the yaw angle $\psi = 0$. Defining the state vector $\mathbf{x} = \left[\mathbf{p}^T, \mathbf{v}^T, \phi, \theta, \psi\right]^T$ and the input vector $\mathbf{u} = \left[\phi_d, \theta_d, \dot{\psi}_d, T_d\right]^T$ the linearized system can be written as:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{v_x}\\ \dot{v_y}\\ \dot{v_z}\\ \dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & -k_D* & 0 & 0 & 0 & g & 0\\
0 & 0 & 0 & 0 & -k_D* & 0 & -g & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -k_D* & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_\phi} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_\theta} & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_\psi}
\end{bmatrix} \begin{bmatrix}x\\ y\\ z\\ v_x\\ v_y\\ v_z\\ \phi\\ \theta\\ \psi\end{bmatrix} + \begin{bmatrix}
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 1\\
\frac{k_\phi}{\tau_\phi} & 0 & 0 & 0\\
0 & \frac{k_\theta}{\tau_\theta} & 0 & 0\\
0 & 0 & \frac{k_\psi}{\tau_\psi} & 0
\end{bmatrix} \begin{bmatrix}\phi_d\\ \theta_d\\ \psi_d\\ T_d\end{bmatrix} \tag{2-12}$$

with the mass-normalized drag coefficient $k_D* = \frac{k_D}{m}$ [17].

2-2 Learning-based Model

Traditional first-principle models only provide an approximate model of the system. Since the controller relies on an accurate model of the system, its performance increases the more exact the system model is. Traditional modeling approaches often render the subsequent control design infeasible when trying to incorporate more complex or nonlinear model dynamics. Learning-based modeling techniques can be used to gradually improve control performance.

2-2-1 Learning the Quadrotor Dynamics

Learning-based modeling techniques can be used to learn the underlying quadrotor dynamics. The environment in which learning takes place can vary widely.

Learning the complete model One approach is to learn the entire quadrotor dynamics from data. Using available machine learning techniques, the quadrotor dynamics are described by a GP [18] or a Neural Network [19], for example. However, this method is very computationally expensive and does not exploit the available information from the first-principle model.

Learning the residual dynamics The more intuitive approach is to use the quadrotor model as an initial approximation and improve it over time as more data becomes available. In the latter case, the system model is often described by a nominal model f_n plus an additional learned model f_1 containing parametric uncertainty θ and a disturbance term $\mathbf{w}$:

$$f(\mathbf{x}, \mathbf{u}, \theta, \mathbf{w}) = f_\text{n}(\mathbf{x}, \mathbf{u}) + f_1(\mathbf{x}, \mathbf{u}, \theta, \mathbf{w}). \tag{2-13}$$

It is assumed that the true dynamics of the system lie within the considered predictive dynamics $f(\mathbf{x}, \mathbf{u}, \theta, \mathbf{w})$. While one might only adjust the nominal values of the model parameters and disturbances, it is common to use a robust or stochastic paradigm as shown in Figure 2-2 to describe the uncertainty of the learned model [20].

Learning-based MPC The idea of combining model learning and control was first presented in the context of learning-based model predictive control (LBMPC) by the authors in [21]. The LBMPC framework considers two linear models. A nominal model and a learned model:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k + \mathcal{O}\left(\mathbf{x}_k, \mathbf{u}_k, k\right) \tag{2-14}$$

where $\mathcal{O}\left(\mathbf{x}_k, \mathbf{u}_k, k\right)$ is called the oracle. The oracle can be any type of statistical learning tool as long as it guarantees that the learned model stays within uncertain bounds $\mathcal{O}\left(\mathbf{x}_k, \mathbf{u}_k, k\right) \in \overline{\mathcal{W}}$. The bounds are set in advance. The nominal model provides theoretic results on the control design while the learned model improves system performance. A practical implementation of the LBMPC method was demonstrated on a quadrotor [15]. Learning was implemented using an extended Kalman filter (EKF) that provides corrections to some entries of the system matrices A and B. Experiments show that the method can help to improve performance while being robust even during incorrect learning [15].

Robust parametric models Although the LBMPC method improves performance for correct learning of the model, a drawback is that the model uncertainty is fixed beforehand. To overcome that restriction, the model uncertainty can be learned online. A prominent method for learning a robust representation of the model uncertainty is set-membership identification [22], [23]. It uses a state-space formulation:

$$\mathbf{x}_{k+1} = A(\theta)\mathbf{x}_k + B(\theta)\mathbf{u}_k + \mathbf{w}_k. \tag{2-15}$$

The system matrices are functions of uncertain parameters θ, that belong to a bounded set Θ. Starting from an initial set of parameters Θ_0, the parameter set will be refined online using incoming state and input measurements. A practical implementation of set-membership identification together with model predictive control on a quadrotor is presented by the authors in [24]. Using set-membership identification, the drone is able to adapt parameters in uncertain wind conditions, while satisfying the constraints [24].

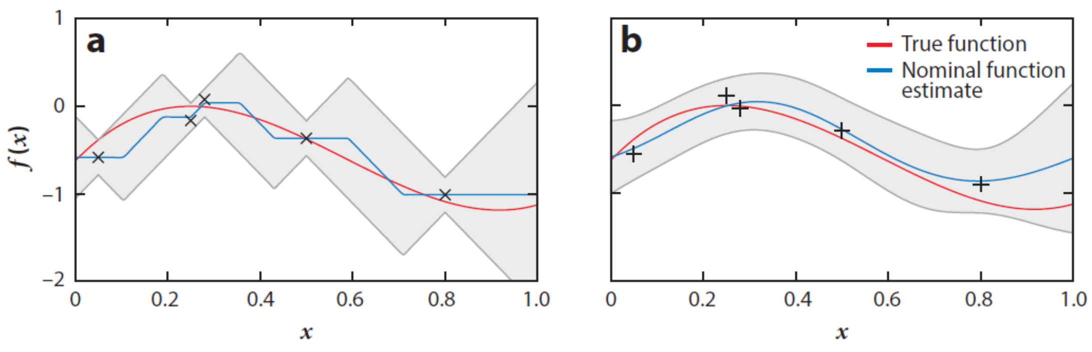


Figure 2-2: Comparison of robust (a) and stochastic (b) estimation techniques [20].

Stochastic non-parametric models Since robust models have to include all uncertainties into a hard bound, they can be conservative. Stochastic models offer a possibility to overcome this drawback by learning a probability distribution of the dynamics instead. Here, Gaussian process regression (GPR) [25] is a commonly used technique. GPs learn a joint distribution of collected data samples and use it to predict the mean and uncertainty at unseen data points. Due to their flexible and non-parametric nature, GPs are particularly suitable to identify residual dynamics with little prior process knowledge available [20]. Using the representation in (2-13), the GP fits the error between the nominal model and the measured states:

$$f_1(\mathbf{x}, \mathbf{u}) \sim \mathcal{N}(\mu(\mathbf{x}, \mathbf{u}), \Sigma(\mathbf{x}, \mathbf{u})). \tag{2-16}$$

The mean $\mu(\mathbf{x}, \mathbf{u})$ is the nominal function value of the residual dynamics and the covariance $\Sigma(\mathbf{x}, \mathbf{u})$ describes its uncertainty. Modeling the residual dynamics with a GP together with model-based control methods has proven to improve the controller performance for application to race cars [26] and a robotic manipulator [27]. Stochastic methods have proven themselves in practice, but they have the disadvantage that it is difficult to give theoretical guarantees [20].

Gaussian process quadrotor model In the context of quadrotor control, GPs have been used to describe the nonlinear dynamics and aerodynamic effects during aggressive flight maneuvers [8] and to guarantee robust obstacle avoidance [28]. To model aerodynamic effects during extreme quadrotor flight, the authors in [8] train three independent GPs that map the body frame velocities to body frame acceleration disturbances:

$$\boldsymbol{a}_{ek} = \mu\left(\boldsymbol{v}_k\right) = \begin{bmatrix} \mu_{v_x}\left(v_{x_k}\right) \\ \mu_{v_y}\left(v_{y_k}\right) \\ \mu_{v_z}\left(v_{z_k}\right) \end{bmatrix}. \tag{2-17}$$

The authors in [28] use GPs to learn the model parameters instead of a black-box residual model. Furthermore, the authors in [29] and [30] use GPs to model residual dynamics and uncertainties. However, both works bound the uncertainty by converting the stochastic formulation back to a robust formulation. While this provides theoretical guarantees, it makes the control design more conservative.

2-3 Wind Disturbances

In many practical applications, the quadrotor is subject to external wind disturbances that are not taken into account in the nominal model. When using a model-based controller or motion planner, this can cause the quadrotor to deviate from the reference path. Especially when avoiding obstacles, a position error can lead to a crash. Learning-based modeling techniques can be used not only to improve the system dynamics, but also to model external uncertainties. While this is a side effect of some of the research presented above, the following section discusses research that focuses specifically on modeling and learning external disturbances such as wind. Before doing so, there is a brief overview of classical approaches to incorporating wind disturbances into the model or controller.

2-3-1 Classic Approach

Wind disturbances can be included into the controller design in several ways. The simplest way to consider wind disturbances is to add a constant disturbance force f_{ext} to the nominal model:

$$m\dot{\mathbf{v}} = -mg\mathbf{a}_3 + R\mathbf{T} + f_{\text{ext}}. \tag{2-18}$$

In most cases, however, the wind field may vary in time and space, which must be considered in the controller design. While some designs reject the wind disturbance as soon as it is observed, others derive explicit wind models.

Disturbance estimation and rejection There are several control algorithms that observe the disturbance online and reject it by calculating a compensating control input. If the drone is equipped with airspeed sensors such as Pitot tubes, a local estimate of the wind can be determined directly [31]. In most cases, however, these sensors are not present on board the quadrotor. Instead, one could use the model of the quadrotor to estimate the wind based on the difference between desired and measured velocity or acceleration. This idea is used for a nonlinear inversion-based position controller resulting in a significant improvement of disturbance rejection in [32]. The authors in [33] use an observer design to estimate the external force vectors. However, this method only provides a deterministic estimate of the disturbance. More information about the disturbance can be obtained using stochastic estimators such as the extended or unscented Kalman filter [34]. In addition to the nominal disturbance force or nominal disturbance torque, these estimators also provide an uncertainty of the estimate that can be incorporated into the controller design.

Disturbance modeling The approaches presented above can only reject the disturbance once it has been observed. It is advantageous in motion planning and predictive control to know the evolution of the disturbance over future states. One way to accomplish this is to have a comprehensive aerodynamic model to describe wind disturbances [35]. A drawback is that determining the model parameters requires extensive experimentation, and incorporating the complex model into a predictive controller increases the computational cost. In addition, robust methods have been developed [36] that assume a constant disturbance bound for all future states, and guarantee performance for all disturbances within this bound. Since the disturbance bound has to be chosen beforehand, this results in conservative control designs and reduced system performance.

2-3-2 Learning Wind Disturbances

Learning-based modeling approaches are promising because they combine the available system data and a predictive model to describe the external disturbances. As more data becomes available during quadrotor flight, the wind disturbance model will be updated.

Learned wind disturbance model Most learning methods for improving the system model focus on the residual model dynamics and uncertainties. Few works address modeling of external disturbances such as wind. Related work on other platforms concerns the modeling of external environmental disturbances for a mobile robot [37]. The residual dynamics are modeled by a GP, which is trained based on the current and previous state vector. By modelling external disturbances caused by uneven terrain, the mobile robot is able to decrease path tracking errors as it passes the terrain multiple times [37]. The authors in [38] use meta-learning with a deep neural network to model the residual dynamics associated with wind disturbances on a quadrotor platform. Preliminary tests in a wind tunnel show that the learned model has a lower prediction error than the nominal model. Furthermore GPs are used to model wind uncertainties in [39]. Three independent GPs are trained based on the observed disturbances. Therefore, the approach can only respond to disturbances if they have been observed. Moreover, the authors in [40] use the Gaussian process model in (2-17) to model not only aerodynamic effects but also strong wind. The work in [41] creates a wind map based on the quadrotor position fusing Inertial Measurement Unit (IMU) and airflow measurements. Depending on whether the wind varies only in space or also over time, both works have the potential to similarly model wind disturbances in this application.

2-4 Discussion

An accurate system model is required for model-based trajectory planning and tracking with a motion planner and controller. In the case of outdoor quadrotor flight, it is also important to consider external wind disturbances. In this section, the basic quadrotor model was presented, which is based on first-principle modeling. However, this model is only an approximation of the system dynamics, especially when using a linear model. To further improve the system dynamics, learning-based methods can be used to improve the system model. Stochastic and robust paradigms were considered. Since robust methods need a hard bound on the uncertainty, they can be more conservative. On the other hand, they can provide safety guarantees, whereas stochastic methods only provide guarantees up to a certain risk level. GPs are most commonly used in the context of robotic applications and quadrotor flight and offer great potential to improve the system dynamics.

While most learning-based techniques focus on the residual dynamics, we specifically want to model external disturbances to predict their behavior in the future. In comparison to classic approaches, learning-based techniques combine the system data which is used to estimate and reject disturbances with a system model that can predict the disturbance. However, little research exists for modeling external disturbances with learning-based methods. GPs have the potential to model the external wind disturbances as they do not rely on a parametric model of the wind, which is difficult to obtain. They also provide less conservative disturbance limits. Since the probability distribution of the wind is known, this information can be incorporated into the motion planner and controller to make it more robust. In the following, Gaussian processes and its challenges in the context of model learning and prediction are explained in more detail.

Chapter 3

Gaussian Processes

Gaussian processes are data-driven machine learning models widely used in regression and classification tasks. Due to their ability to model nonlinear functions without any prior knowledge, GPs have gained much interest in recent years in the context of control. Since GPs are non-parametric, they offer great flexibility. Moreover, unlike other machine learning methods, GPs are able to describe the uncertainty in their prediction, which can be accounted for in the controller.

This chapter of the report discusses Gaussian process regression and its mathematical background. It then focuses specifically on GPs for learning a dynamic model and the challenges associated with them. One of these challenges is the computational complexity of learning and predicting with large data sets, which can be reduced using sparse GPs. In addition, ways to transfer the uncertainty into the system state and to predict the GP multiple steps into the future are presented. In the last section of this chapter, methods for online learning of GPs are discussed.

3-1 Gaussian Process Regression

In supervised learning, regression is concerned with the prediction of continuous quantities. The goal is to obtain an approximation of a nonlinear function $f(\mathbf{x})$ of the input vector $\mathbf{x}$. Assuming a GP prior of the nonlinear function $f(\mathbf{x})$, GP regression uses a Bayesian approach to determine a predictive distribution of the function value $f(\mathbf{x}_*)$ at unseen test locations $\mathbf{x}_*$ [25].

3-1-1 Mathematical Background

As the math behind GPs is based on a Bayesian approach, Bayesian inference is explained first before going into the mathematical details of training a GP and making predictions.

Bayesian inference Gaussian process regression is based on Bayesian inference, which is quickly explained with a linear model:

$$y = f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + \varepsilon \tag{3-1}$$

with unknown parameter $\mathbf{w}$ and noise ε. The Bayesian approach works by specifying a prior $p(\mathbf{w})$ on the parameter $\mathbf{w}$ and shifting probabilities based on the observed data. Using Bayes' rule:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \quad p(\mathbf{w} \mid X, \mathbf{y}) = \frac{p(\mathbf{y} \mid X, \mathbf{w}) p(\mathbf{w})}{p(\mathbf{y} \mid X)} \tag{3-2}$$

one obtains the posterior distribution of the dataset $\mathcal{D} = \{(X, \mathbf{y})\}$. To make predictions at unseen inputs $\mathbf{x}_*$, the predictive distribution is calculated based on the posterior distribution according to:

$$p\left(f(\mathbf{x}_*) \mid \mathbf{x}_*, X, \mathbf{y}\right) = \int p\left(f(\mathbf{x}_*) \mid \mathbf{x}_*, \mathbf{w}\right) p(\mathbf{w} \mid X, \mathbf{y}) d\mathbf{w}. \tag{3-3}$$

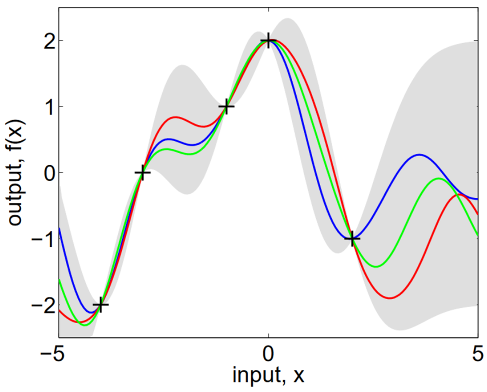


Figure 3-1: Gaussian process posterior distribution conditioned on five, noise free observations. The grey area represents the mean ± two times the standard deviation. In colour are three random samples drawn from the posterior distribution [25].

Gaussian process prior The ideas of Bayesian inference for the linear model (3-1) can be transferred to Gaussian process regression. In GPR, the prior is specified by a GP:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right). \tag{3-4}$$

More specifically, a GP is defined as a collection of random variables with the property that the joint distribution of any finite subset is also Gaussian [25]. Prior knowledge can be incorporated into the GP through the selection of its mean and covariance functions:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{3-5a}$$
$$k\left(\mathbf{x}, \mathbf{x}'\right) = \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f\left(\mathbf{x}'\right) - m\left(\mathbf{x}'\right)\right)\right]. \tag{3-5b}$$

Dataset The dataset $\mathcal{D}$ of n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$, is generated according to:

$$y_i = f\left(\mathbf{x}_i\right) + \varepsilon_i \tag{3-6}$$

where $f : \mathbb{R}^D \rightarrow \mathbb{R}$ and $\varepsilon_i \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ is Gaussian measurement noise with zero mean and variance σ_ε [42]. Noise in the data can be added to the covariance function. The prior on the measured function values is defined as $p\left(\mathbf{y} \mid X\right) = \mathcal{N}\left(0, K(X,X) + \sigma_\varepsilon^2 I\right)$. Here, $K(X,X)$ denotes the $n \times n$ matrix of the covariances evaluated at pairs of the input points. From the GP prior, the available system data and the desired test locations, one can write the joint distribution of the observed values $\mathbf{y}$ and function values $\mathbf{f}_*$ as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) + \sigma_\varepsilon^2 I & K\left(X, X_*\right) \\ K\left(X_*, X\right) & K\left(X_*, X_*\right) \end{bmatrix}\right). \tag{3-7}$$

Inference Conditioning the joint Gaussian prior distribution on the observations, one can derive the posterior distribution:

$$p\left(\mathbf{f}_* \mid X, \mathbf{y}, X_*\right) \sim \mathcal{N}\left(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*\right), \text{ where} \tag{3-8a}$$

$$\boldsymbol{\mu}_* = K\left(X_*, X\right)\left[K(X,X) + \sigma_\varepsilon^2 I\right]^{-1} \mathbf{y}, \tag{3-8b}$$

$$\boldsymbol{\Sigma}_* = K\left(X_*, X_*\right) - K\left(X_*, X\right)\left[K(X,X) + \sigma_\varepsilon^2 I\right]^{-1} K\left(X, X_*\right). \tag{3-8c}$$

In the case that there is only one test point x_* the equations reduce to:

$$\mu_* = \mathbf{k}_*^\top \left(K + \sigma_\varepsilon^2 I\right)^{-1} \mathbf{y} \tag{3-8d}$$

$$\Sigma_* = k\left(x_*, x_*\right) - \mathbf{k}_*^\top \left(K + \sigma_\varepsilon^2 I\right)^{-1} \mathbf{k}_* \tag{3-8e}$$

with $\mathbf{k}_* = K(x_*, X)$ and $\mathbf{k}_*^T = K(X, x_*)$. A GP posterior distribution for noise-free data is shown in Figure 3-1.

Model selection The mean and covariance function of the GP prior must be chosen to describe the underlying model and form a critical part of the prediction process [43]. The mean of the GP prior is typically chosen to be zero, if no other information is available. The covariance is specified by a kernel function. The choice of kernel or covariance function expresses the similarity between function values. It specifies certain properties of the GP such as smoothness or periodicity [43]. The most commonly used kernel is the squared exponential kernel:

$$k_{\text{SE}}\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \left(\mathbf{x}_d - \mathbf{x}'_d\right)^2 \boldsymbol{l}_d^{-2}\right). \tag{3-9}$$

Other kernel choices are constant, linear, polynomial or neural network kernels [25]. It is furthermore possible to construct kernels based on existing ones. The sum of two kernels:

$$k_{\text{sum}}\left(\mathbf{x}, \mathbf{x}'\right) = k_1\left(\mathbf{x}, \mathbf{x}'\right) + k_2\left(\mathbf{x}, \mathbf{x}'\right), \tag{3-10}$$

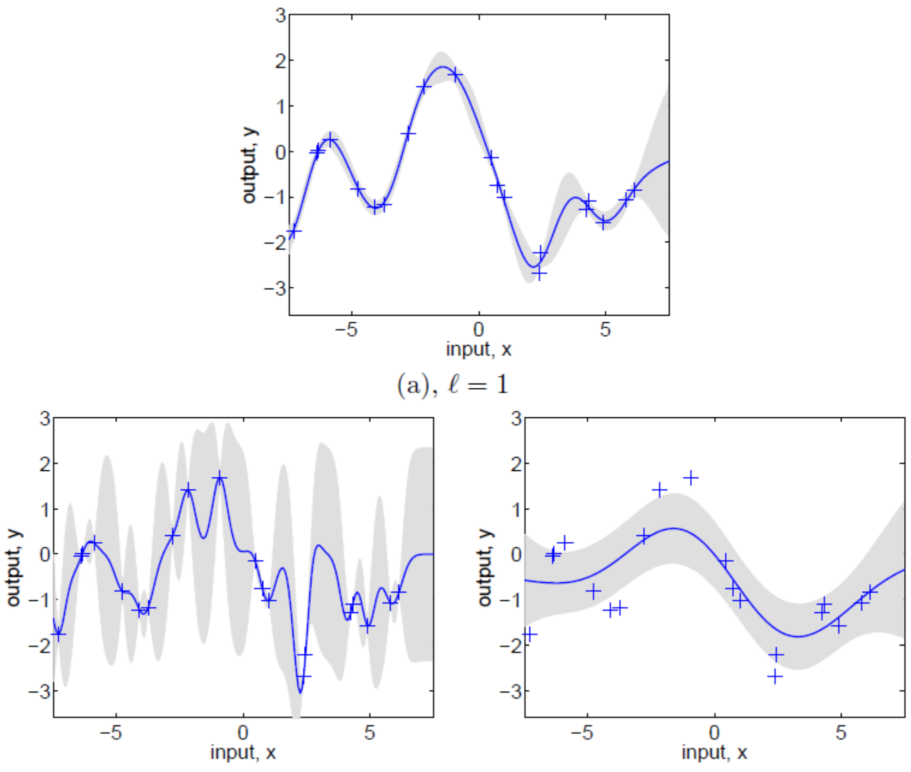
and the product of two kernels:

$$k_{\text{prod}}\left(\mathbf{x}, \mathbf{x}'\right) = k_1\left(\mathbf{x}, \mathbf{x}'\right) k_2\left(\mathbf{x}, \mathbf{x}'\right) \tag{3-11}$$

form a valid kernel.

Hyperparameters The kernel function has free parameters which define the properties of the GP. The squared exponential function, for example, has a total of $D+1$ parameters.

The horizontal length scale for each input dimension l_d and the output variance ${\sigma_f}^2$. The horizontal length scale determines how correlated neighboring inputs are. A smaller length scale allows for more variations between nearby points. With a larger length scale, the GP is smoother. This effect is shown in Figure 3-2. The output variance ${\sigma_f}^2$ determines the average distance of the function away from its mean. In addition, the noise covariance ${\sigma_\varepsilon}^2$ can be considered as a hyperparameter of the kernel. The hyperparameters do not have to be determined beforehand but are trained as part of the GP prior using the measured data $\mathcal{D} = \{(X, \mathbf{y})\}$.



(a), $\ell = 1$

(b), $\ell = 0.3$ (c), $\ell = 3$

Figure 3-2: Influence of different length scales on the GP prediction [25].

Hyperparameter training The hyperparameters can be found by maximizing the marginal likelihood (or evidence) of the prior, which is conditioned on the hyperparameters $\boldsymbol{\theta}$ through the kernel function $K_\theta = K(\mathbf{x}, \mathbf{x})(\boldsymbol{\theta})$:

$$\mathcal{L} = \log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^\top \left(K_\theta + \sigma_\varepsilon^2 I\right)^{-1} \mathbf{y} - \frac{1}{2}\log\left|K_\theta + \sigma_\varepsilon^2 I\right| - \frac{n}{2}\log(2\pi). \tag{3-12}$$

The hyperparameter vector:

$$\hat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}} \log p(\mathbf{y} \mid X, \boldsymbol{\theta}) \tag{3-13}$$

is the maximum likelihood estimate of the hyperparameters, which trades off model complexity versus the data fit [42]. Maximizing the likelihood is a non-convex, nonlinear optimization problem which can be hard to solve depending on the available data [42]. Alternatives are cross-validation, which is computationally expensive, or joint marginalization based on Monte-Carlo methods [25].

Computational complexity Training the GP with a training set of size n requires $\mathcal{O}\left(n^3\right)$ per gradient step due to the inversion of the Kernel matrix $K(X, X)$ which makes training very expensive [25]. To make predictions, the inversion of the kernel matrix demands most operations and results in a computational complexity of $\mathcal{O}\left(n^3\right)$ per prediction. Cholesky factorization can help reduce the computational complexity of the matrix inversion to $\mathcal{O}\left(n^3/6\right)$. Sparse GPs can further reduce the complexity.

3-1-2 Sparse Gaussian Processes

Sparse GPs reduce the complexity of training the GP model and making predictions. Instead of using all n training inputs, a smaller training set of size m, also referred to as inducing inputs or active test set is used to form the covariance matrix $K(X, X)$. The complexity of inverting the covariance matrix is thereby reduced to $\mathcal{O}\left(nm^2\right)$ - a significant speed up.

Choosing inducing inputs Using fewer inputs to train and predict GPs also reduces the accuracy and expressiveness of the GP. This raises the problem of how to select the active test set. If it is selected greedily from the available data, it becomes difficult to include all the necessary information. Instead, there are several methods to optimally select the inducing inputs and contain as much information as possible in the covariance matrix. Two popular methods are the fully independent training conditional (FITC) [44], and variational free energy (VFE) [45].

Pseudo inputs Both methods detach from the idea of choosing the active set from available input data and choose pseudo data points instead. The location of the pseudo inputs is optimized together with the the system hyperparameters. The FITC approximation can be viewed as changing the GP prior:

$$f(x) \sim \mathcal{GP}\left(0, Q_{NN} - \operatorname{diag}\left[K_{NN} - Q_{NN}\right]\right) \tag{3-14}$$

with $Q_{NN} = K_{NM} K_M^{-1} K_{MN}$ and full covariance matrix K_{NN} [46]. VFE, on the other hand, defines the inducing inputs as variational parameters which are selected by minimizing the Kullback-Leibler divergence between the variational distribution and the exact posterior GP [45].

A unifying description of sparse GP As both methods train the inducing inputs as part of the hyperparameters, the marginal likelihood changes. It can be summarized as:

$$-\mathcal{L} = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log\left|Q_{NN} + G\right| + \frac{1}{2} y^T \left(Q_{NN} + G\right)^{-1} y + \frac{1}{2\sigma_n^2}\operatorname{tr}(T) \tag{3-15}$$

where $G_{FITC} = \operatorname{diag}\left[K_{NN} - Q_{NN}\right] + \sigma_\varepsilon^2 I$, and $G_{VFE} = \sigma_n^2 I$. Also, $T_{VFE} = K_{NN} - Q_{NN}$, while $T_{FITC} = 0$.

Posterior distribution Given the inducing inputs, the posterior distribution of the GP can be found using Bayes' rule:

$$\mu_* = \mathbf{k}_*^\top Q_M^{-1} K_{MN} \left(\mathbf{\Lambda} + \sigma_\varepsilon^2 I\right)^{-1} \mathbf{y} \tag{3-16a}$$

$$\Sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top \left(K_M^{-1} - Q_M^{-1}\right) \mathbf{k}_* + \sigma_\varepsilon^2. \tag{3-16b}$$

K_M denotes the covariance matrix of the m inducing points and K_{MN} denotes the covariance matrix between the original dataset of size n and the inducing points. $\mathbf{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda}), \lambda_n = K_{NN} - \mathbf{k}_n^\top K_M^{-1}\mathbf{k}_n$ and $Q_M = K_M + K_{MN}\left(\mathbf{\Lambda} + \sigma^2 I\right)^{-1} K_{NM}$ [46].

Practical use of sparse GP FITC and VFE are frequently used in practice. A comparison of the two methods was presented in [47]. FITC may tend to overestimate the marginal likelihood and underestimate the noise variance, and thus may not be able to recover the true posterior value [47]. On the other hand, VFE is susceptible to local optima and might be more difficult to optimize than FITC. The superiority of VFE over FITC is concluded in [47].

VFE was further developed by the authors in [48], where two novel variational bounds for performing sparse GP regression are presented. By introducing additional variational parameters, it is possible to optimize the marginal likelihood in a stochastic or distributed fashion, making it more computationally scalable. The method has been implemented in GPyTorch [49] which is a GP library implemented using PyTorch used for sparse GP inference.

3-2 Gaussian Process Dynamics Model

After discussing the mathematical background of GPs, this section of the chapter addresses challenges that arise when modeling external wind disturbances with a GP. Modeling the disturbance as a GP results in a system model of the following form:

$$f(\mathbf{x}, \mathbf{u}) = f_n(\mathbf{x}, \mathbf{u}) + f_{\mathrm{GP}}(\mathbf{a}) \tag{3-17}$$

where $f_{\mathrm{GP}}(\mathbf{a}) \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{a}), \mathbf{\Sigma}(\mathbf{a}))$ is the GP model capturing the wind dynamics with mean $\boldsymbol{\mu}_{\mathrm{GP}}$ and covariance $\mathbf{\Sigma}_{\mathrm{GP}}$. The feature vector $\mathbf{a}$ is selected from the state vector $\mathbf{x}$. Since the perturbation acts on the system in multiple directions, but the GP can only describe one-dimensional outputs, it is discussed how GPs can be used for multivariate predictions. It is also explained how the probabilistic disturbance can be propagated into the system state and across a sequence of states. This is important information for the predictive controller or motion planner.

3-2-1 Multivariate Predictions

Classic GPs, as described in section 3-1, are only capable of modelling nonlinear dynamics with one output. If the output $\mathbf{y} \in \mathbb{R}^T$ is a multivariate target, one can either train each output independently or include the correlation between different outputs in the GP model.

Independent outputs Training the outputs independently, T GP models are trained using the same training inputs $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, but different training targets. This assumes that the function values for different outputs are computationally independent, given the input $\mathbf{x}$ [42]. The predictive distribution is given by:

$$\boldsymbol{\mu}_* = [\mu_{1*}, \mu_{2*}, \ldots, \mu_{T*}]^\top \tag{3-18a}$$

$$\mathbf{\Sigma}_* = \mathrm{diag}\left(\begin{bmatrix} \Sigma_{1*}^2 & \ldots & \Sigma_{T*}^2 \end{bmatrix}\right) \tag{3-18b}$$

Multi-output GPs If the outputs are correlated, modelling them individually results in loss of information. To model the correlation between different outputs, one can use multi-output Gaussian processes (MOGP) [50]. The GP prior is defined as:

$$\mathbf{f}(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, \mathcal{K}_M\left(\mathbf{x}, \mathbf{x}'\right)\right) \tag{3-19}$$

with multiple outputs $\mathbf{f} = \{f_1, \ldots, f_T\}^\top$. $\mathcal{K}_M\left(\mathbf{x}, \mathbf{x}'\right) \in R^{T \times T}$ denotes the multi-output covariance matrix:

$$\mathcal{K}_M\left(\mathbf{x}, \mathbf{x}'\right) = \begin{bmatrix} k_{11}\left(\mathbf{x}, \mathbf{x}'\right) & \cdots & k_{1T}\left(\mathbf{x}, \mathbf{x}'\right) \\ \vdots & \ddots & \vdots \\ k_{T1}\left(\mathbf{x}, \mathbf{x}'\right) & \cdots & k_{TT}\left(\mathbf{x}, \mathbf{x}'\right) \end{bmatrix} \tag{3-20}$$

that captures the similarity between outputs. The posterior distribution and the prediction are based on the Bayes' method. Choosing the right multi-output covariance matrix is important for the performance of the GP model. An overview of different MOGPs can be found in [50].

3-2-2 Predicting Uncertain Trajectories

The GP model provides the mean and uncertainty of the disturbance as a function of the system state. To use this information in a predictive setting, the uncertainty has to be propagated into the next system state and over multiple future states. This poses two challenges: firstly, GP prediction of uncertainty in the case of an uncertain state input, and secondly, propagation of the uncertain disturbance through the possibly nonlinear system dynamics. Both of these challenges can be addressed with approximate inference.

Approximate Inference

Computing the exact posterior distribution of the GP in Equation (3-3) at a probabilistic test input is computationally intractable. Instead, the posterior distribution of the GP model is approximated as a Gaussian distribution with mean μ and variance Σ. These values can be obtained by approximate inference. The GP posterior is approximated either by linearization, similar to the extended Kalman filter, by an unscented transformation, as used in the unscented Kalman filter, or by the moment matching technique. The three methods provide more or less accurate approximations. Linearization is the least accurate method, while the moment-matching technique provides the most accurate approximation, but with a higher computational cost [51].

PILCO Linearization and the moment matching technique were discussed by Deisenroth el al. in the context of PILCO (probabilistic inference for learning control), which was first implemented for model-based reinforcement learning [51] and later extended to model predictive control [52]. In their research, the entire dynamics are modeled by a GP but the mathematical relations can be similarly extended for learning only part of the system dynamics.

Linearization Using linearization [51], the predicted mean of the GP posterior is obtained by evaluating the GP at the mean of the input distribution $\mathbf{a}$ with mean $\boldsymbol{\mu}_\mathbf{a}$ and covariance $\boldsymbol{\Sigma}_\mathbf{a}$:

$$\boldsymbol{\mu}_{\mathrm{GP}}(\mathbf{a}) = \mathbb{E}\left[f_{\mathrm{GP}}(\boldsymbol{\mu}_\mathbf{a})(\mathbf{a})\right]. \tag{3-21}$$

To compute the covariance matrix, the posterior GP function is linearized around the mean of the input distribution. Then the uncertainty can be mapped through the linear model, leading to the predictive covariance:

$$\mathbf{\Sigma}_{\text{GP}}(\mathbf{a}) = V\mathbf{\Sigma}_{\mathbf{a}}(\mathbf{a})V^{\top} + \Sigma_{\varepsilon} \tag{3-22a}$$

$$V = \frac{\partial \boldsymbol{\mu}_{\text{GP}}(\mathbf{a})}{\partial \boldsymbol{\mu}_{\mathbf{a}}(\mathbf{a})}. \tag{3-22b}$$

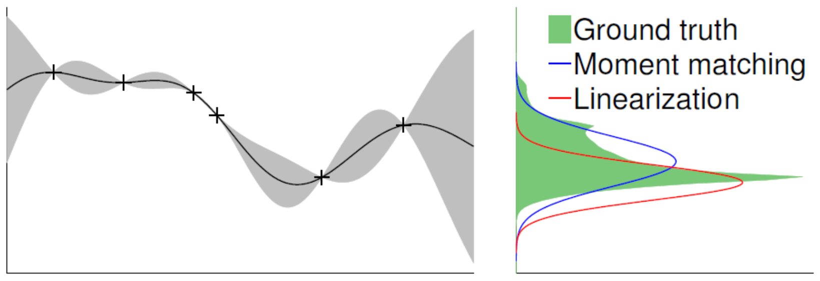


Figure 3-3: Approximation of the posterior GP distribution at a probabilistic test input by a Gaussian distribution using linearization versus using the moment matching technique [51].

Moment Matching Moment matching is a more accurate but also more complex approach to approximate the posterior distribution. It was initially introduced by [53]. The moment matching technique fits a probabilistic distribution to a more complex distribution such that some moments of the two functions coincide. In the case of fitting the exact posterior distribution in Equation (3-3), also shown in Figure 3-3, it is approximated by a Gaussian distribution that matches in the first two moments: the mean μ and covariance Σ. In [42] the mean and covariance are derived for multivariate predictions with a squared exponential kernel.

Sigma Point Transformation Another method for propagating uncertainty is the sigma point transformation [54], which is also used in the unscented Kalman filter. It can capture high-order information of the distribution and has a similar cost to the linearization method. In this method, a set of points (sigma points) are chosen such that their mean and covariance are equal to the input distribution. The nonlinear function is applied to each point, resulting in a could of transformed points. The posterior distribution is approximated by the mean and covariance of the transformed point cloud. The sigma points are deterministically chosen to have certain properties. An algorithm for selecting the sigma points and computing the mean and covariance of the posterior distribution is discussed in [54]. This method was used in the context of GP models by [37]. The state evolution for an example model propagating uncertainty is shown in Figure 3-4.

Propagating Uncertainty

Given the mean and covariance of the GP, these must be propagated through the system model to obtain the uncertain state description. For an autonomous linear model with disturbance:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + \mathbf{w}_k \tag{3-23}$$

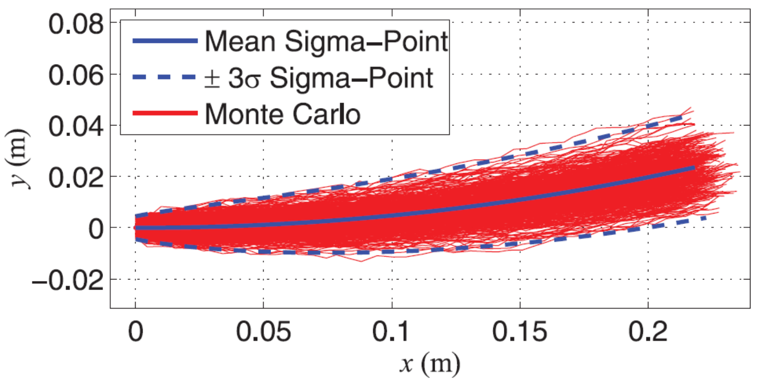


Figure 3-4: Illustration of propagating uncertainty using a Sigma-Point transformation with a 3-σ confidence interval compared to a sampling-based Monte Carlo method [37].

the uncertainty propagation is a linear transformation. If $\mathbf{w}_k$ is Gaussian distributed with mean $\boldsymbol{\mu}_{\mathbf{w}_k}$ and uncertainty $\boldsymbol{\Sigma}_{\mathbf{w}_k}$ the mean and uncertainty of the state can be expressed as follows:

$$\boldsymbol{\mu}(\mathbf{x})_{k+1} = \boldsymbol{\mu}(\mathbf{x})_k + \boldsymbol{\mu}_{\mathbf{w}_k} \tag{3-24a}$$

$$\boldsymbol{\Sigma}(\mathbf{x})_{k+1} = A\boldsymbol{\Sigma}(\mathbf{x})_k A^T + \boldsymbol{\Sigma}_{\mathbf{w}_k} + A\boldsymbol{\Sigma}_{\mathbf{w}_k} + \boldsymbol{\Sigma}_{\mathbf{w}_k} A^T. \tag{3-24b}$$

For nonlinear models, however, the exact mean and uncertainty cannot be easily derived. Here, too, approximate inference is used. The authors in [55] discuss propagation for their dynamics model for both linearization and moment matching. However, for a dynamical model as in (3-17), these derivations become even more complex. Often researchers using a similar model ignore the uncertainty of the current state input and propagate only the mean of the state and the uncertainty of the disturbance using the approximate inference, (e.g. [37] [26] [27]). Not taking into account the uncertainty of the system state simplifies the GP prediction to the standard case with a deterministic input. While this simplifies the calculations, it is a further approximation to the actual model, making the predictions less accurate.

3-3 Online Learning with Gaussian Processes

Previously, it was assumed that all data is available a priori to train the GP. However, when learning a dynamic model, more data becomes available during flight that can be used to improve the GP over time. While training of the GP can be done after each iteration, as in [27] or [8], one can improve the performance online by incorporating new data. In the following section, several online learning methods which are implemented for GP models are introduced.

Sparse online Gaussian Processes In its simplest form, online learning can be implemented by adding new data points to the data set as they become available. However, this would quickly make the prediction computationally expensive, even with sparse GPs. Instead, one must manage experiences by discarding old data as new data becomes available.

The authors in [20] simply discard the oldest data point. A more targeted selection of the data points is performed by the authors in [26]. Data points are replaced based on a distance measure describing the coverage of the input data points. One can also manage experiences in bins [37]. The oldest data point is discarded if new data gets added to the bin. Moreover, the authors in [20] choose the inducing points of the GP based on the trajectory, which are therefore updated online as the robot moves along the trajectory.

Online learning of hyperparameters By updating only the dataset online, the hyperparameters are fixed beforehand. There also exist methods to update the hyperparameters together with the incoming data based on Bayesian inference. This includes, for example, recursive GPs [56] or incremental learning [57]. However, these methods have rarely been implemented in a learning-based control environment, most likely due to the higher computational cost. Furthermore, in many cases, some data is available beforehand which can be used to train the hyperparameters. However, if the underlying model is more complex and evolves over time, (e.g. in [58] [39]), updating the hyperparameters online is desired.

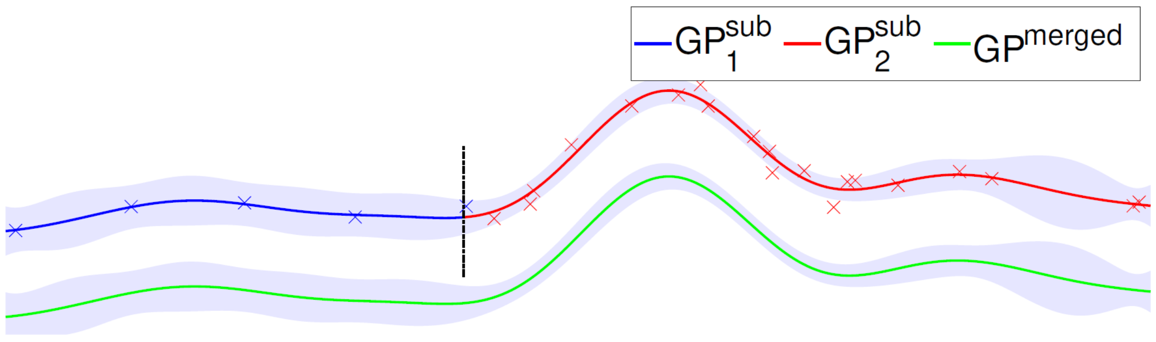


Figure 3-5: LSTM feature of the Gaussian process. $\mathrm{GP}_1^{\mathrm{sub}}$ is the Gaussian process trained based on long-term memory and $\mathrm{GP}_2^{\mathrm{sub}}$ is trained with the last 200 samples [39].

LSTM One issue that arises with updating the GP online is the forgetting of data, particularly if the hyperparameters are learned online. Research in this field considers long-short term memory (LSTM) models, where two GPs are trained. One GP is trained offline or online with data dating back some time. The other GP is trained with more recent data. A dual GP method to learn the entire quadrotor dynamics is used in [58]. LSTM models were furthermore used to capture wind uncertainty based on previously recorded uncertainty [39]. An example LSTM GP model is shown in Figure 3-5.

Meta-learning Another way of learning the dynamics model online is meta-learning. In meta-learning, a big part of the model is trained offline and only a few parameters are learned online to adapt to changing dynamics or new information from incoming data. This method has been presented for learning wind disturbances with a neural network in [38]. A meta-learning method for GPs is presented in [59] in the context of MPC. The GP model is expressed by a linear combination of basis functions:

$$f(\mathbf{x}) = \sum_{i=1}^{E} \phi_i(\mathbf{x})\alpha_i^m \tag{3-25}$$

where $\boldsymbol{\alpha}^m = [\alpha_1^m, \ldots, \alpha_E^m]^T \sim \mathcal{N}(\mu_\alpha^m, \Sigma_\alpha^m)$, with mean and variance computed via Bayesian linear regression. The basis functions $\phi_i(\mathbf{x})$ are trained offline, whereas the linear parameters $\alpha^m = [\alpha_1^m, \dots, \alpha_E^m]$ are adapted online to taylor the GP model to the specific situation. This method has been implemented in simulation for a miniature race car adapting to different kinds of tasks [59].

Extend the GP with online methods A method that is not directly related to online learning of a GP, but is still worth mentioning, is to augment an offline trained GP with a simpler online learning method to adapt to changing conditions. This idea is similar to LSTM models, where the short-term GP is replaced by another disturbance model or estimator. An example of a GP model together with a Kalman filter to estimate the remaining disturbances has been implemented on a robot manipulator by [27].

3-4 Discussion

GPs are a powerful machine learning tool for learning a nonlinear function from available data and making predictions at unknown locations based on Bayesian inference. Since GPs are based on a Gaussian distribution, they can provide not only the mean prediction but also the uncertainty of the prediction like no other machine learning method.

The ability of the GP to describe any nonlinear function with little prior knowledge makes it a promising approach for modelling system dynamics whose structure is unknown. This makes it particularly suitable for modelling wind uncertainty. The process of model selection involves choosing an appropriate mean and kernel function. Since using all available data leads to large computation times during training and prediction, this is not practical for real-time motion planning and control. Sparse GPs, such as FITC and VFE, reduce the computational complexity.

To model a system dynamics with more than one output, as is the case with a wind disturbance acting in three directions, one needs to train a GP model that can make multivariate predictions. For this purpose, a separate GP model can be trained for each direction, provided there is no correlation. If there is correlation, it can be captured by GPs with multiple outputs, but this leads to more complex training and prediction. Furthermore, there are several approximation methods to propagate the uncertainty through the system model. In general, it holds that the more accurate the approximation, the more expensive the calculation.

The last section of this chapter discussed how the data collected during the flight can be added to the model to improve it online. The simplest way is to add a new data point while deleting an old one. There are several methods to manage the data flow. Learning hyperparameters online is also possible, but involves a lot of computational effort and risks forgetting the old dynamics. LSTM models offer a way to prevent this. However, this form of online learning is only useful if the dynamics change over time (e.g. in the case of strongly fluctuating wind conditions). Models that change only slightly over time can also be captured with meta-learning methods or by extending GP with other online learning methods. For the problem considered in this research, wind conditions will be assumed to change only slightly over time, which is why a simple method to manage data flow can be chosen.

The next chapter discusses how the GP dynamics model can be integrated into the motion planner and controller, and in particular how uncertainty can be incorporated into the description.

Chapter 4

Quadrotor Motion Planning and Control

In this chapter the problem of navigating the quadrotor in a cluttered environment is addressed. The problem of finding a feasible trajectory given an obstacle map is called *motion planning*. An introduction to the most important motion planning algorithms and important terms and mathematical formulations is given. To track the trajectory, a *controller* has to be in place. This chapter also gives an overview of the general control structure used for quadrotor control and commonly used controllers. The focus is on the position controller which is responsible for tacking the trajectory. Furthermore, emphasis is put on model-based controllers to integrate the derived model discussed in previous chapters into the motion planner and the controller. This also provides the opportunity to combine trajectory generation and tracking into a single optimal control formulation based on the system model.

Adding a GP to the model results in an uncertain, probabilistic formulation of the quadrotor states. For this reason, the second half of this chapter discusses methods for incorporating this uncertainty into the motion planning and control algorithm. Before going into the mathematical details of extending the deterministic framework with robust or stochastic formulations, an overview of the state of the art in motion planning and control under uncertainty is given.

4-1 State of the Art

This section provides an overview of the state of the art in motion planning and control methods for quadrotors. The planner and controller need to work together to safely navigate the quadrotor through a given, cluttered environment. In the conventional case, the complete control structure can be divided into four steps [6] as shown in Figure 4-1. These are:

1. The motion planner, that sends the desired trajectory to the position controller.
2. The position controller that gives Euler angles as a reference to the attitude controller.

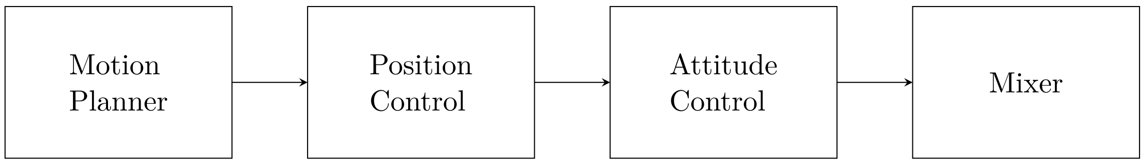


Figure 4-1: Commonly used control structure of the motion planner and controller.

3. The attitude controller that controls the orientation of the robot on its three Euler angular velocities and the desired thrust.
4. The mixer, that translates the angular velocities and thrust command into individual rotor commands.

In this overview, the focus is on the first two steps. In the following, a general introduction to motion planning is given before algorithms that are commonly implemented on quadrotors are explained. In addition, various control algorithms for quadrotor control are presented.

4-1-1 Motion Planning

The goal of the motion planner is to find a valid motion sequence of the quadrotor from its start position to the goal position, without colliding with obstacles in the environment. The configuration of the robot q is described in the configuration space $\mathcal{C}$, which contains all possible configurations of the robot. The configuration space is divided into free space $\mathcal{C}_{\text{free}}$, where the robot does not contact any obstacle, and the obstacle space $\mathcal{C}_{\text{obs}}$, the set of configurations where the robot collides with an obstacle:

$$q \in \mathcal{C} \subseteq \mathbb{R}^n \tag{4-1a}$$

$$\mathcal{C} = \mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}} \; . \tag{4-1b}$$

Global versus local planners Motion planners can be divided into *global* planners and *local* planners. Global planners plan the entire motion sequence from start to end position using a map of the environment. Local planners update waypoints as the quadrotor moves, taking into account dynamic obstacles and vehicle constraints.

Path versus trajectory planners A distinction is also made between *path* planners and *trajectory* planners. Path planners ignore the dynamics of the quadrotor and other differential constraints. They focus primarily on the translations and rotations required to move the robot. However, if the robot is subject to dynamics:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{4-2}$$

with mechanical constraints on the state space $\mathbf{x} \in \mathcal{X}$ and input space $\mathbf{u} \in \mathcal{U}$, these must be considered in the motion planning problem. This is called the trajectory planning problem, where the trajectory describes the evolution of vehicle configurations over time. Typically, the motion planner is split into a front-end path-finding problem and a back-end trajectory optimizer. First, some common path planners are presented before introducing trajectory optimizers that have been successfully implemented on real quadrotor platforms.

Path Finding

With the definitions in Equation (4-1), one can describe the path planning problem as finding a continuous path $\tau : [0,1] \to \mathcal{C}_{\text{free}}$, such that $\tau(0) = q_{\text{init}}$ and $\tau(1) = q_{\text{goal}}$ where $q_{\text{init}}, q_{\text{goal}} \in \mathcal{C}_{\text{free}}$. This idea is shown for a simple scenario in Figure 4-2. The path planning problem usually ignores dynamics and other differential constraints and focuses primarily on the translations and rotations required to move the robot. Common path planners are search-based methods, sampling-based methods and potential field methods [60].

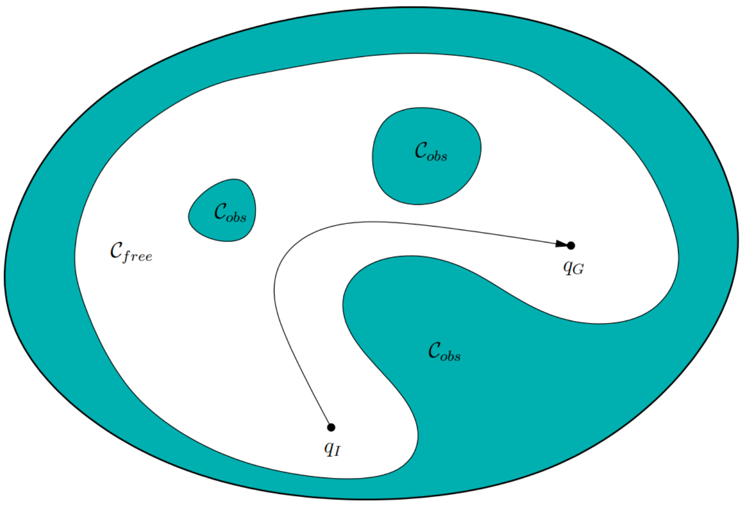


Figure 4-2: The path finding problem [61].

Search-based methods discretize the configuration space into smaller convex polygons or discrete states, which are then connected by a graph, turning the path finding problem into a graph search. Typical graph search methods are Dijkstra's [62] or the A* [63] algorithm. There are various extensions of these methods, e.g. to adapt to changing environments or to accept suboptimal solutions under time pressure [64].

Sampling-based methods solve problems in continuous space without a graph representation. Probabilistic Road-Map (PRM) [65] and Rapidly-Exploring Random Tree (RRT) [66] are two representative methods. A feasible path is generated by randomly sampling from the free space and connecting it to its nearby configurations (PRM) or expanding the tree (RRT). Sampling-based methods are much faster than search-based methods and are therefore preferred for large configuration spaces but lead to suboptimal solutions. Extensions of these algorithms to generate optimal paths are PRM*, RRT* or RRT*-Smart [60].

Potential field methods assign an artificial potential field to every point in the configuration space. The goal configuration has the lowest potential, while the initial configuration has the highest potential. Furthermore, the obstacle space gets assigned a high potential. The robot then plans from the maximum potential to the minimum potential.

Trajectory Planning

In contrast to the path planner, the trajectory planner also assigns a time law to the geometric path, which takes into account the kinodynamic constraints of the quadrotor. Usually, the path planner precedes the trajectory planner, which then time-parameterizes the solutions of the path planner [64]. Some trajectory planners also solve both problems simultaneously. The existing approaches for quadrotor trajectory planning can mainly be divided into polynomial approaches, discrete-time search methods, sampling-based approaches and optimization methods [67].

Polynomial trajectory planners are one common category of trajectory planning algorithm. A pioneering work is the minimum-snap trajectory generation algorithm proposed by the authors in [68]. The trajectories are represented by smooth piecewise polynomials making use of the differential flatness property of the quadrotor. The minimum snap trajectory can be generated by solving a quadratic programming (QP) problem. However, polynomial control inputs are smooth and cannot fully exploit the potential of the actuator [69].

Search- and sampling-based methods Discretizing time and the state space, search-based methods can also be applied to the trajectory planning problem. The authors in [70] compute motion primitives offline using a time-optimal LQR control policy and a search graph is expanded online. Sampling-based kinodynamic methods [71] create a tree of states by selecting a random state at each iteration, integrating it forward using random inputs for a random time. A hierarchical, sampling-based trajectory generation method for minimum-time quadrotor flight in cluttered environments is presented by the authors in [67].

Optimization-based trajectories By introducing a formulation of progress along the trajectory that allows simultaneous optimization of the time allocation and the trajectory itself, time-optimal trajectories can be generated. State and input constraints are enforced. Several formulations have been discussed in literature and range from the use of point-mass models to simplified quadrotor models and full-state models [69]. However, optimization-based are mostly unusable if additional non-convex constraints are introduced to prevent collisions [67].

4-1-2 Trajectory Tracking Control

The desired trajectory generated by the motion planner must be followed by a suitable controller. This section presents different types of controllers used in quadrotor control.

Control Architecture

The quadrotor controller is usually divided into two parts. The two controllers are arranged in a cascaded structure, with the outer loop responsible for position control and the inner loop responsible for attitude control. The cascaded structure is shown in Figure 4-3. The position controller has gained attention from the community in recent years due to its role in enabling the quadrotor to track the desired trajectories [6]. The attitude controller controls the angular velocity and thrust needed to keep the quadrotor in the air. It is usually already embedded in the motor control board such as the Pixhawk [72] and stabilizes the quadrotor. When research focuses on the attitude controller, it is common practice to approximate the dynamics of the attitude controller by a linear model, as discussed earlier in section 2-1-1.

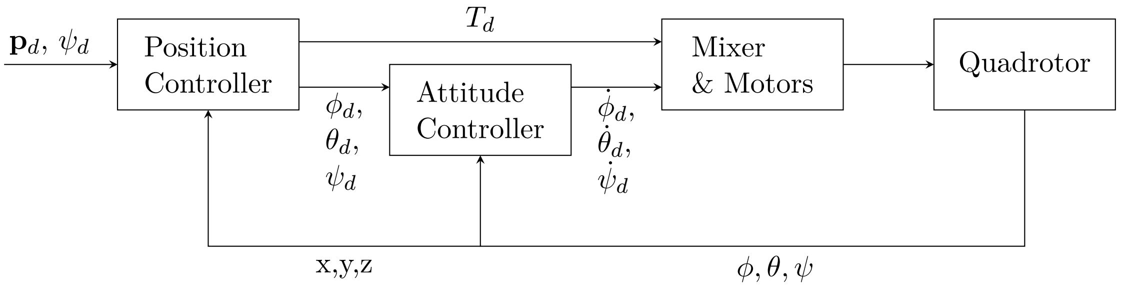


Figure 4-3: Cascaded control architecture of a common quadrotor controller.

Control Algorithms

Due to the increased research interest in quadrotors in the last decades, numerous control techniques have been implemented for the quadrotor, ranging from linear to nonlinear control methods, model-based versus model-free controllers, robust, adaptive, and intelligent controllers. There are several studies on quadrotor control (e.g.. [6], [73], [74], [75], [76]). Below is a brief summary of key points from these reports.

Linear controllers are PID, LQR and H_∞ control. For PID control, the P, I, and D gains are set manually and are therefore independent of the system model. LQR, on the other hand, is a model-based controller design in which the optimal feedback gains are calculated based on the matrices of the linear system using dynamic programming. It assumes complete state information. It can be extended to LQG control, where unknown states are estimated using a Kalman filter. H_∞ control provides a robust feedback controller that can handle model uncertainties and disturbances. Since these controllers are based on a linear system model, they only work near the hovering state.

Nonlinear controllers overcome the limitations of linear controllers, allowing the drone to operate over its entire flight envelope. A common nonlinear control approach is feedback linearization. The system is converted to a linear system by adding state feedback. Then the controller can be designed using linear control methods. However, feedback linearization depends on an accurate system model and therefore cannot handle system uncertainties or disturbances[75]. Other nonlinear control approaches include backstepping and sliding mode control.

Model predictive control (MPC) is a more advanced control method for constrained multiple input, multiple output systems. Based on the system model, an OCP is solved to obtain the input command. Control methods exist for both linear and nonlinear flight dynamics. By solving an OCP based on the current system state, it is possible to track a reference trajectory while satisfying the state and input constraints. Nominal MPC relies on an accurate representation of the system model, which is often difficult to derive analytically. Robust and stochastic MPC provide ways to incorporate the uncertainty of the system into the MPC formulation.

Robust control The quadrotor may be subject to model uncertainties and external disturbances such as wind gusts that affect the performance of the controller. While other control methods may fail in the case of an inaccurate system model, robust control schemes are able to maintain a certain performance despite some uncertainty in the system. The uncertainty is given as a hard bound. While some controllers are inherently robust, such as the H_∞ controller, other controllers can be augmented with a robust compensator to achieve robustness.

Adaptive control In contrast to the robust controller, which deals with general system uncertainties, adaptive controllers deal with parametric uncertainty by automatically adjusting system model parameters such as mass, inertia, or aerodynamic coefficients. An example of adaptive control is adaptive model reference control (MRAC). Adaptive and robust control methods can also be combined by adjusting the parameters and parameter limits simultaneously so that the controller can handle almost all types of uncertainties.

Intelligent control design incorporates computational intelligence. Intelligent controllers include neural network controllers, fuzzy logic controllers and reinforcement learning. Due to their intelligent nature, intelligent control methods are able to accurately model unknown dynamic behavior and adapt to changes in the system model, which increases the robustness of the controller [9].

Comparison of Control Algorithms

There are a number of different control algorithms that have been presented for quadrotors. One can distinguish between linear and non-linear controllers and between model-based and model-free controllers. Robust, adaptive and intelligent control algorithms are extensions of these categories of controllers.

Linear controllers are applicable where the quadrotor does not deviate significantly from the hovering state. This assumption typically applies to position control and reference tracking when the quadrotor does not use the entire flight envelope [5]. For extreme flight manoeuvres, e.g. racing, or when controlling the attitude of the quadrotor, non-linear control methods are typically used [5].

When a model of the quadrotor dynamics is available, model-based methods provide a way to incorporate the system dynamics into the controller design and facilitate tuning of the control parameters. Among model-based methods, MPC is a popular method for quadrotor control because it also allows the inclusion of dynamic constraints. It also provides a degree of robustness by recalculating the control law at each time [5]. When information about the system uncertainty is available, robustness can be further improved by robust or stochastic MPC methods [9]. The combination of MPC with a learning model, as presented in section 2-2, has already been investigated in the context of learning-based MPC [20]. Here, the MPC controller is extended to be both intelligent, by incorporating the GP model, and adaptive, when the GP model is updated online.

Moreover, using an MPC controller offers the possibility to combine both trajectory planning and control.

4-1-3 Combining Motion Planning and Control

So far, motion planning and trajectory tracking were considered as two separate problems. First, the desired trajectory is generated by the motion planner and then this trajectory is tracked by the controller. With optimisation-based methods, it is possible to combine both trajectory planning and trajectory tracking in one optimization problem. Given a path by the planner, the optimization problem solves for the optimal system states and system inputs. The key to solving this problem is to use an MPC algorithm that is modified to simultaneously solve the trajectory generation problem. One popular method to achieve this is Model Predictive Contouring Control (MPCC) [77]. The following section first describes the mathematical details of general MPC before explaining the modifications for MPCC.

4-2 Model Predictive Control

MPC [78] solves an optimal control problem (OCP) on-line at each sampling instant taking the current state of the plant as the initial state. Based on the system model, it predicts the future evolution of the states over a finite horizon. With the known system evolution, it optimizes for a sequence of control inputs, given a certain cost. After obtaining the optimal control sequence, only the first control input is applied to the system. At the subsequent time-step, the OCP is solved with the new current state. This procedure is also known as the *moving horizon principle* and shown in Figure 4-4. The formulation of the OCP allows to directly impose constraints on the system, which is one of the major advantages of MPC over other control methods.

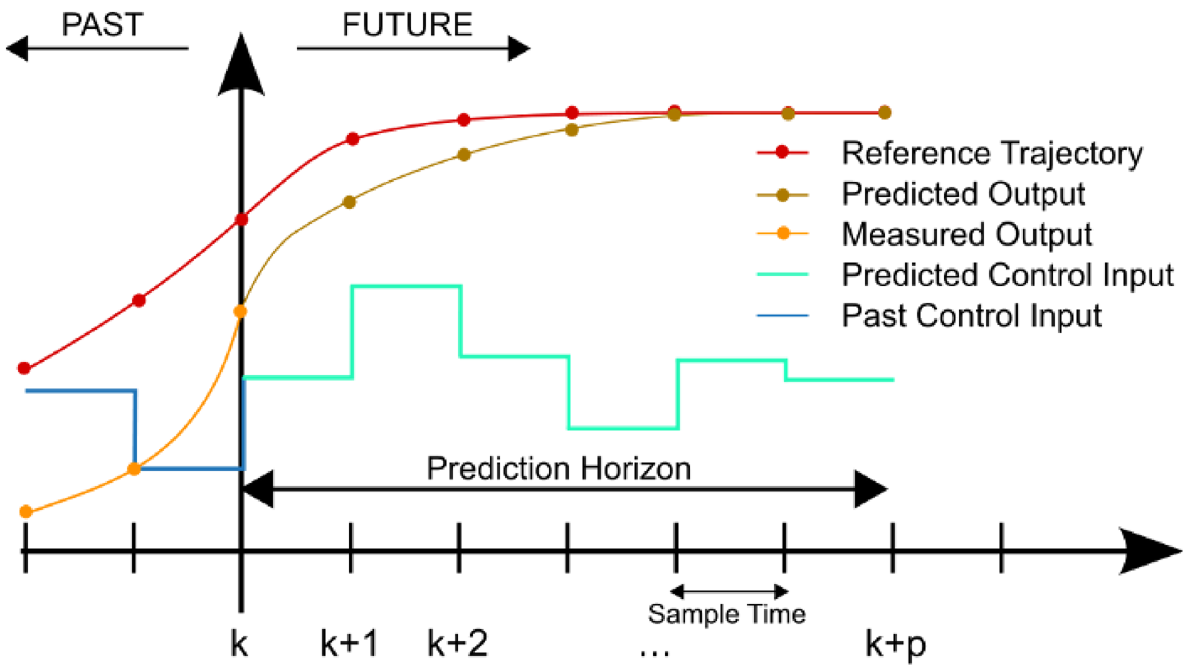


Figure 4-4: MPC scheme [79].

4-2-1 Standard Model Predictive Control

In standard MPC, the desired reference trajectory is given and the cost is formulated so that the quadrotor follows this trajectory as well as possible, taking into account the system dynamics and constraints.

System dynamics and constraints Consider the discrete-time nonlinear system dynamics of the quadrotor which are described by a model of the form:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{4-3a}$$
$$\mathbf{y}_k = h(\mathbf{x}_k) \tag{4-3b}$$

where $\mathbf{x}(t) \in \mathbb{R}^n, \mathbf{u}(t) \in \mathbb{R}^m, \mathbf{y}(t) \in \mathbb{R}^p$. If the system is subject to constraints, these can be formulated as:

$$\mathbf{u} \in \mathbb{U} \tag{4-4a}$$
$$\mathbf{x} \in \mathbb{X} \tag{4-4b}$$
$$(\mathbf{x}, \mathbf{u}) \in \mathbb{Z}. \tag{4-4c}$$

Cost function The control objective is expressed by the cost function:

$$V(\mathbf{x}, \mathbf{u}) = \sum_{i=0}^{N-1} \ell(\mathbf{x}_i, \mathbf{u}_i) + V_f(\mathbf{x}_N) \tag{4-5}$$

with stage cost $\ell(\mathbf{x}, \mathbf{u})$ and final cost V_f. The stage cost usually aims to steer the system to an equilibrium state with minimal control effort $\mathbf{u}$. The final cost is imposed on the final system state for theoretical guarantees. A terminal constraint:

$$\mathbf{x}_{k+N} \in X_f \subset \mathbb{X} \tag{4-6}$$

is sometimes added to the MPC formulation for the same reason of making theoretical guarantees. Standard MPC typically minimizes a quadratic penalty on the error between the predicted states and inputs and a given reference trajectory. The given reference trajectory is dynamically feasible. The objective is expressed by the cost:

$$\ell(\mathbf{x}_i, \mathbf{u}_i) = \|\Delta\mathbf{x}_i\|^2_{\mathbf{Q}} + \|\Delta\mathbf{u}_i\|^2_{\mathbf{R}} \tag{4-7}$$

where $\Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i_{ref}}$ and $\Delta\mathbf{u}_i = \mathbf{u}_i - \mathbf{u}_{i_{ref}}$.

Theoretical guarantees The two main theoretical properties that the MPC controller should guarantee are recursive feasibility and stability [80]. Recursive feasibility ensures that the OCP can be solved again at the next time instant after applying the first control input. Then, the problem is feasible for all future time instants. The stability analysis is based on Lyapunov stability theory.

MPC formulation In summary, the OCP takes the following form:

$$\min_{\mathbf{u}_{0:N-1}} \quad \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + V_f(\mathbf{x}_N) \tag{4-8a}$$
$$\text{s.t.} \quad \mathbf{x}_0 = \mathbf{x}(0) \tag{4-8b}$$
$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \tag{4-8c}$$
$$\mathbf{u} \in \mathbb{U} \tag{4-8d}$$
$$\mathbf{x} \in \mathbb{X} \tag{4-8e}$$
$$(\mathbf{x}, \mathbf{u}) \in \mathbb{Z} \tag{4-8f}$$
$$\mathbf{x}_{k+N} \in X_f \subset \mathbb{X} \tag{4-8g}$$

Linear MPC For a linear system:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k \tag{4-9}$$

$$\mathbf{y}_k = C\mathbf{x}_k + D\mathbf{u}_k \tag{4-10}$$

with a quadratic cost function and convex constraints, the OCP is convex and can be solved efficiently with available optimization tools. Theoretical guarantees exist for the stability of the control problem for a certain choice of terminal cost and terminal constraint set [78].

Nonlinear MPC For nonlinear systems, the MPC problem is usually non-convex. However, many solvers such as FORCES PRO [81] exist, to efficiently solve nonlinear MPC (NMPC) problems. Moreover, NMPC poses challenges for stability guarantees in finding the terminal cost and terminal constraint set [82].

4-2-2 Model Predictive Contouring Control

In contrast to standard MPC, which tracks a desired trajectory, MPCC [77] solves the problem of tracking a desired path. The path is tracked in an optimal way by maximizing the progress along the path or tracking a given velocity while minimizing the the distance to the path. This objective is included in the cost function of the OCP. At the same time, static and dynamic obstacles can be considered by adding constraints to the OCP.

State of the art MPCC has been applied for static and dynamic obstacle avoidance with ground vehicles [83] and autonomous ground robots [84]. As it solves a time-optimal problem, it has also been applied to racing with both ground vehicles [85] and quadrotors [69]. The main mathematical concepts of MPCC are discussed below.

Reference path The input to the MPCC controller is a reference path, which can be as simple as a straight line from the starting position to the goal position, a sequence of waypoints or a smooth polynomial path. In addition, any of the above path planning methods can be used to find the path. The path itself does not have to be feasible yet.

Progress along the path The length of the reference path or the progress along the path is described by a variable θ. It is assumed that the desired path is parameterised by $\mathbf{p}(\theta)$. For a given longitudinal vehicle speed $\mathbf{v}_k$ at time step k, the approximate progress along the reference path can be described by:

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \mathbf{v}_k \Delta t. \tag{4-11}$$

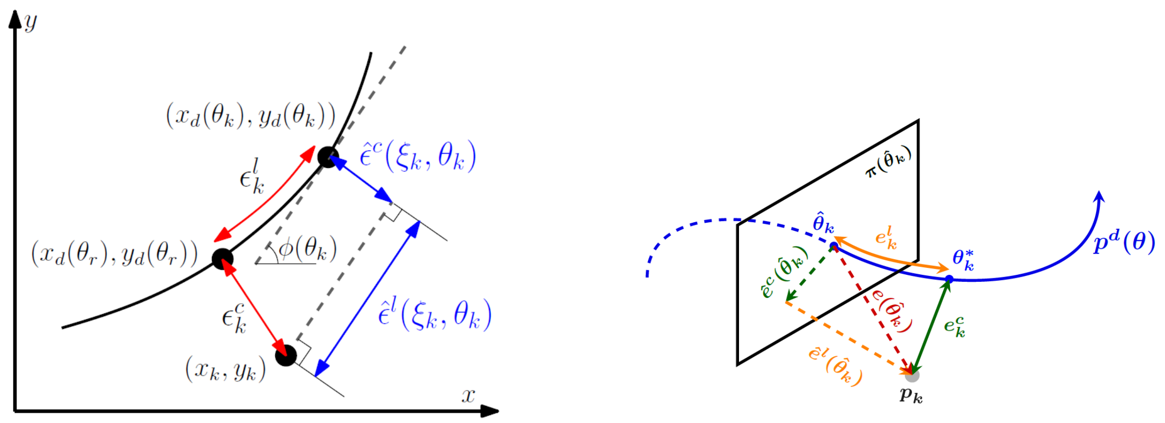
Depending on the goal, different cost functions can be chosen to achieve the desired progress on the path. To maximize the progress [69], one can simply add a cost function:

$$V_{\text{progress}}(\mathbf{x}_k, \mathbf{u}_k, \theta) = -\rho \mathbf{v}_k. \tag{4-12}$$

If the goal is to track a desired speed [84], the cost function can be expressed as follows:

$$V_{\text{speed}}(\mathbf{x}_k, \mathbf{u}_k, \theta) = \rho \left(\mathbf{v}_{\text{ref}} - \mathbf{v}_k\right)^2 \tag{4-13}$$

where in each case, ρ is a tuning parameter of the OCP.



y
x
$(x_d(\theta_k), y_d(\theta_k))$
$(x_d(\theta_r), y_d(\theta_r))$
(x_k, y_k)
ϵ_k^l
ϵ_k^c
$\hat{\epsilon}^c(\xi_k, \theta_k)$
$\hat{\epsilon}^l(\xi_k, \theta_k)$
$\phi(\theta_k)$
$\pi(\hat{\theta}_k)$
$\hat{\theta}_k$
e_k^l
θ_k^*
$p^d(\theta)$
$\tilde{e}^c(\hat{\theta}_k)$
$e(\hat{\theta}_k)$
e_k^c
$\tilde{e}^l(\hat{\theta}_k)$
p_k

(a) 2D [77]. **(b)** 3D [69].

Figure 4-5: Contour and lag error and their approximations in 2D and 3D.

Contour error The contour error describes the distance between the position $\mathbf{p}_k$ at time k and the desired position on the reference path $\mathbf{p}(\theta_k)$. Finding the exact contour error:

$$e_k^c = \min_{\theta} \|\mathbf{p}_k - \mathbf{p}(\theta)\|_2 \tag{4-14}$$

involves embedding an optimization which is not well suited. Instead, the contour error is approximated by a longitudinal error (lag) and lateral error (contouring) normal to the path [77]. Depending on the application and whether the system is defined in 2D or 3D, these errors are calculated differently, as shown in Figure 4-5. Examples can be found in the works mentioned in this section. The cost of the contouring error is defined as:

$$V_{\text{contour}}(\mathbf{x}_k, \mathbf{u}_k, \theta) = \mathbf{e}_k^T Q \mathbf{e}_k \tag{4-15}$$

with:

$$\mathbf{e}_k = \begin{bmatrix} \tilde{e}^l(\mathbf{x}_k, \theta_k) \\ \tilde{e}^c(\mathbf{x}_k, \theta_k) \end{bmatrix}. \tag{4-16}$$

The weight Q is another tuning parameter. Generally, there is a trade-off between progress along the path and the accuracy of tracking the desired path, which is determined by the tuning parameters.

Objective function The baseline method, that includes the contour and progress objective can be further modified and customized to the application by adding additional cost terms on the system input [83] or guaranteeing an extra safety margin [84], for example. The overall cost function is then given as:

$$V_{\text{MPCC}}(\mathbf{x}_k, \mathbf{u}_k, \theta) = V_{\text{contour}}(\mathbf{x}_k, \mathbf{u}_k, \theta) + V_{\text{progress}}(\mathbf{x}_k, \mathbf{u}_k, \theta) \tag{4-17a}$$

$$V(\mathbf{x}_k, \mathbf{u}_k, \theta) = V_{\text{MPCC}}(\mathbf{x}_k, \mathbf{u}_k, \theta) + V_{\text{custom}}(\mathbf{x}_k, \mathbf{u}_k, \theta). \tag{4-17b}$$

Obstacle avoidance constraints To guarantee that the generated trajectory is feasible, given the obstacle space, obstacle avoidance constraints have to be enforced such that:

$$\mathcal{B}(\mathbf{x}_k) \cap \mathcal{O} = \emptyset \tag{4-18}$$

where $\mathcal{B}(\mathbf{x}_k)$ represents the occupancy volume of the quadrotor. Obstacles can be both static and dynamic. To guarantee collision avoidance of the quadrotor with a straight wall, one can define a linear constraint of the form:

$$\mathbf{a}^T \mathbf{x}_k \leq b. \tag{4-19}$$

The obstacle-free space can also be defined by a convex polytope. In this case the constraints have the form:

$$\bigcap_{l=1}^{o_n} \mathbf{a}_l^T \mathbf{x}_k \leq b_l \tag{4-20}$$

with o_n intercepting hyperplanes. In addition, one can define spherical or ellipsoidal obstacle constraints that lead to quadratic constraints of the form:

$$(\mathbf{x}_k)^T R \mathbf{x}_k \leq 1. \tag{4-21}$$

MPCC formulation Including the system dynamics and additional constraints on the system states and inputs, the complete MPCC problem has the following form:

$$\begin{aligned}
\min_{\mathbf{x}_{0:N}, \mathbf{u}_{0:N-1}, \theta_{0:N-1}} \quad & \sum_{k=0}^{N-1} V(\boldsymbol{x}_k, \boldsymbol{u}_k, \theta_k) + V(\boldsymbol{x}_N, \theta_N) && \text{(4-22a)}\\
\text{s.t.} \quad & \mathbf{x}_0 = x(0), \quad \theta_0 = \theta(0) && \text{(4-22b)}\\
& \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad \theta_{k+1} = \theta_k + \mathbf{v}_k \Delta t && \text{(4-22c)}\\
& \mathcal{B}(\mathbf{x}_k) \cap \left(\mathcal{O}^{\text{static}} \cup \mathcal{O}_k^{\text{dyn}}\right) = \emptyset && \text{(4-22d)}\\
& \mathbf{u} \in \mathbb{U}, \mathbf{x} \in \mathbb{X}, (\mathbf{x}, \mathbf{u}) \in \mathbb{Z}. && \text{(4-22e)}
\end{aligned}$$

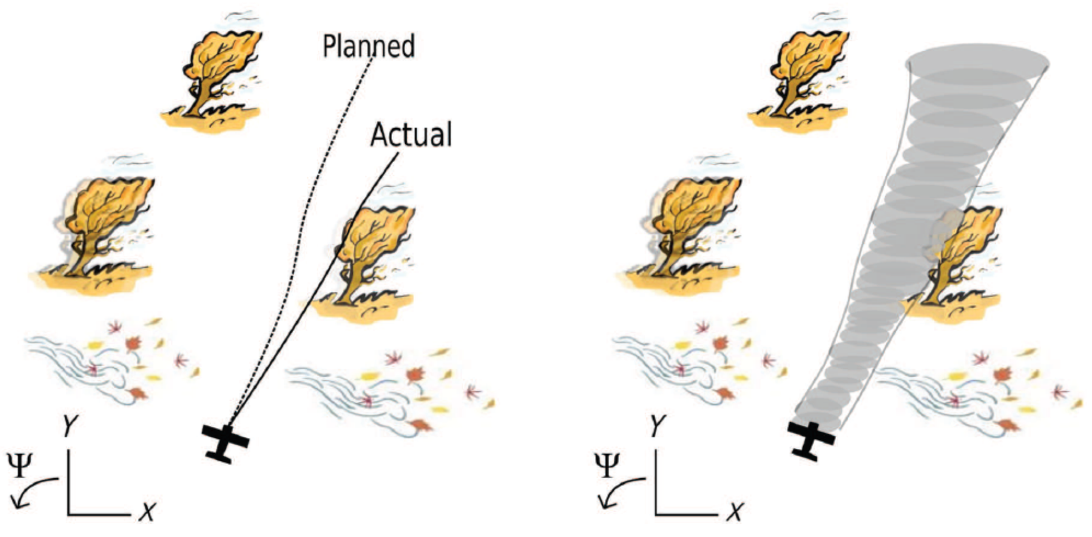
4-3 Motion Planning and Control Under Uncertainty

So far, motion planning and control have been considered for deterministic models of the quadrotor. However, if system uncertainties are present, they must be taken into account to ensure safe navigation. This is especially important in cluttered environments where small deviations from the reference path can cause the quadrotor to crash. In recent years, several algorithms have been presented that address the problem of safe motion planning despite uncertainties.

4-3-1 State of the Art

Robust motion planning algorithms have been active areas of research in the past few years and can be categorized into three general classes: Sampling-based methods, reachability-based methods and nonlinear MPC methods [86].

Sampling-based methods Robust sampling-based methods build on the sampling-based methods described in section 4-1-1. Safety is guaranteed by checking collision using robust or stochastic formulations [86]. However, most attempts to include uncertainty into sampling-based methods are rather heuristic [87].



(a) A plane deviating from its nominal planned trajectory due to a heavy cross-wind.

(b) The "funnel" of possible trajectories.

Figure 4-6: Safety funnel for a plane navigating in a forest with heavy cross-wind [88].

Reachability-based methods The use of reachability theory is another way to introduce robustness into the motion planner. A reachable set is defined as all possible states that the quadrotor can take, given information about its uncertainty. The authors in [88] precompute a library of safety funnels around trajectories, where the safety funnels are outer approximations of the reachable set given a certain feedback controller. The funnel library is pieced together at runtime to generate feasible and safe trajectories. The idea of using safety funnels for safe navigation in a forest is depicted in Figure 4-6. Furthermore, Hamilton-Jacobi (HJ) reachability analysis was developed for robust offline trajectory planning in fully known environments and independent of the motion planner, called FasTrack [89]. FasTrack computes the tracking error as a function of control inputs and generates a feedback controller to compensate for it. A similar method based on HJ-reachability analysis for real-world quadrotor flight was shown by the authors in [90]. The work presented by the authors in [86] uses forward reachable sets (FRS) that bound the tracking error and are computed offline. During online planning, the FRS are used to map obstacles to parameterized trajectories so that a safe trajectory can be selected. Furthermore, FRS were computed to consider aerodynamic effects [91]. The authors in [92] use FRS to compute outer bounding ellipsoids that approximate the position of the quadrotor plus its uncertainty, which are then used in an NMPC controller (see Figure 4-7). However, the reachability-based methods rely on a robust or bounded representation of the uncertainty. Furthermore, most of the methods presented above calculate the uncertain boundaries offline.

Nonlinear MPC Both restrictions can be overcome with NMPC that can deal with set-bounded and probabilistic uncertainties and plans trajectories online. To deal with uncertainties in MPC, the general MPC formulation in (4-8) is extended. Depending on the representation of uncertainty, robust and stochastic reformulations exist. Robust MPC formulations

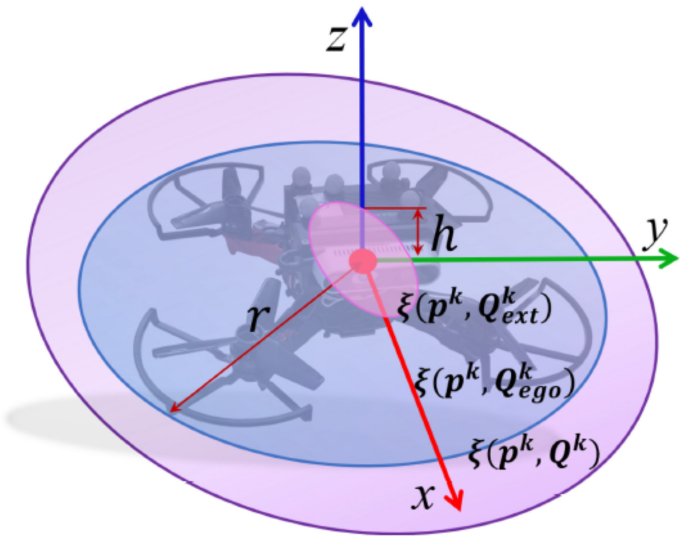


Figure 4-7: Outer bounding ellipsoid of the quadrotor derived from the space taken by the quadrotor plus the position uncertainty [92].

were used in the context of trajectory generation and tracking by the authors in [93] and [94]. Another application of robust MPC and planning is the robust collision avoidance problem in [13]. Additionally, the authors in [95] build on the idea of robust MPC and use control contraction metrics to design a feedback control law and tube. Stochastic MPC formulations for collision and obstacle avoidance were initially presented by the authors in [96], where polytopic chance constraints were reformulated into deterministic constraints. This work was extended with quadratic obstacle avoidance constraints for dynamic collision avoidance [14]. The ideas from robust and stochastic MPC can be combined by using a feedback controller and probabilistic reachable sets to reformulate chance constraints [97]. However, this work is limited to linear systems.

Given the deterministic MPCC formulation for motion planning and tracking and the extensions of the nonlinear MPC algorithms for motion planning and control under uncertainty, there is the potential to combine both ideas. By extending MPCC to a robust or stochastic formulation, it is possible to achieve simultaneous trajectory planning and tracking while ensuring safety when avoiding obstacles. The following section deals with the mathematical details of MPC under uncertainty to explore how this idea can be implemented.

4-3-2 Model Predictive Control Under Uncertainty

Nominal MPC relies on an accurate representation of the system model, which is often hard to derive analytically. Robust and stochastic MPC provide ways to include system uncertainty into the MPC formulation.

Robust MPC

In robust MPC [98] [99], the uncertainty of the system model is taken into account in the MPC formulation. This guarantees robust constraint handling, stability and performance. The nominal model is additionally described with an uncertainty $\mathbf{w}_k$:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \tag{4-23}$$

which is assumed to be bounded by a compact set $\mathbf{w}_k \in \mathbb{W}$.

Solving robust MPC There are several solutions to incorporate uncertainty into the OCP. One approach is to minimize the cost function for the worst-case uncertainty resulting in a min-max OCP [100]. Another way is to describe the uncertainty of the system by a tube containing all possible trajectories of the uncertain system. With the tube description, the constraints can be tightened accordingly so that the fulfilment of the constraints is guaranteed for all disturbance sequences. Combined with a linear feedback controller, this approach, called tube-based MPC, is the most prominent in literature [99] [78].

Tube-based MPC For a linear model of the form:

$$\mathbf{x}_{k+1} \quad = A\mathbf{x}_k + B\mathbf{u}_k + \mathbf{w}_k \tag{4-24}$$

the outer bounding tubes are constructed based on the evolution of the error centered around the nominal trajectory and may grow very large. By introducing a stabilizing feedback controller:

$$\mathbf{u} = \bar{\mathbf{u}} + K(\mathbf{x} - \bar{\mathbf{x}}) \tag{4-25}$$

the error will be corrected at each time instant, such that the size of the tube can be contained. It is then possible to construct a disturbance invariant set $\mathbb{Z}$ that can be used to tighten the constraints as follows:

$$\mathbf{u}_k \in \bar{U} \triangleq \mathbb{U} \ominus KZ \tag{4-26a}$$
$$\mathbf{x}_k \in \overline{\mathbb{X}} \triangleq \mathbb{X} \ominus Z \tag{4-26b}$$
$$\mathbf{x}_N \in X_f \subset \mathbb{X} \ominus Z \tag{4-26c}$$

where the $\ominus$ denotes set subtraction. Constraint satisfaction of the nominal system for the tightened constraints guarantees constraint satisfaction for all possible disturbances. The cost function and state evolution still use the nominal description.

Theoretic guarantees Proofs of theoretic properties such as constraint satisfaction, stability and feasibility of the OCP for the tube-based formulation have been shown for linear systems [99]. Making certain assumptions, extensions to nonlinear systems are shown [101], but are still an ongoing field of research.

If the model uncertainty is given by a stochastic formulation or probability, the disturbance is no longer bounded. It can be overapproximated by sigma hulls to obtain a robust formulation again. However, making this approximation results in overconservative bounds [14], complicating to find feasible trajectories. In that case, a stochastic formulation of the OCP has advantages over a set-bounded approach.

Stochastic MPC

Stochastic MPC [102] methods take the distributional information of the state into account. As the disturbance is now of a probabilistic nature, the formulation of the cost function and constraints may change. Furthermore, because of the probabilistic nature of the disturbance $\mathbf{w}_k$, it is unbounded. Deterministic guarantees on stability and feasibility can not be extended to the stochastic case [20]. Proving theoretical guarantees for stochastic MPC is still ongoing research. Nevertheless, several practical implementations of stochastic MPC have been shown.

Probabilistic cost function As the predicted states are no longer deterministic, the cost function also becomes random. It can either be evaluated at the mean prediction:

$$V_i(\mathbf{x}, \mathbf{u}) = \ell(\mu_{\mathbf{x}_i}, \mathbf{u}_i) \tag{4-27}$$

or it is replaced by some statistical property which, for example, is the expected value of the cost [102]:

$$V_i(\mathbf{x}, \mathbf{u}) = \mathbb{E}\left[\ell(\mathbf{x}_i, \mathbf{u}_i)\right]. \tag{4-28}$$

Chance constraints As the disturbance becomes unbounded, it is also no longer possible to formulate hard constraints. Instead, the constraints are softened using a chance constrained formulation:

$$\Pr\left(\mathbf{x}_k \in \mathbb{X}\right) \geq 1 - \beta. \tag{4-29}$$

Joint versus individual chance constraints Inequality constraints of the form:

$$\Pr\left(h(\mathbf{x}, \xi) \geq 0\right) \geq 1 - \beta. \tag{4-30}$$

with uncertainty representation ξ are joint chance constraints since the probability has to be satisfied for the constraint vector. Joint chance constraints are hard to solve and therefore usually approximated by individual chance constraints:

$$\Pr\left(h_i(\mathbf{x}, \xi) \geq 0\right) \geq 1 - \beta^i. \tag{4-31}$$

which give an over-approximation of the joint constraints [102]. While the individual constraints are more conservative, they give a tractable convex expression for the optimization problem.

Solving stochastic MPC The two main classes to solve the stochastic MPC problem in practice are analytic approximation methods and scenario based approximation methods [103]. In the first approach, the stochastic constraints are reformulated into deterministic constraints. For scenario based MPC, random samples of the noisy realizations are generated and the OCP is applied to the samples. A combination of both approaches is also possible. A scenario based optimization problem is solved to reformulate the stochastic constraints, which are then used in a deterministic MPC problem [102]. In the following, the focus will be on methods to reformulate the change constraints into deterministic constraints for the types of constraints introduced in section 4-2-2.

Reformulating chance constraints In general, chance constraints can be reformulated in two ways: 1) enlarging the quadrotor using its standard deviation sigma or 2) based on the actual collision probability. While the formulations for sigma bounds are easier to derive they result in overconservative approximations [14]. Finding reformulations based on the collision probability is therefore preferred in cluttered spaces.

Even though they lead to a more conservative approximation, in practice the quadrotor condition is often magnified by many times its standard deviation. A simple form of constraint, the probabilistic band

Sigma confidence bounds Even though they lead to a more conservative approximation, enlarging the quadrotor state by multiples of its standard deviation is often done in practice. A simple form of constraint, the probabilistic band:

$$\Pr\left(\mathbf{x}_i \leq \mathbf{x}_{max,i}\right) \geq p \tag{4-32a}$$

$$\Pr\left(\mathbf{x}_i \geq \mathbf{x}_{min,i}\right) \geq p \tag{4-32b}$$

can be reformulated given the mean and covariance:

$$\mu_{\mathbf{x},i} + \alpha\Sigma_{\mathbf{x},i} \leq \mathbf{x}_{max,i} \tag{4-33a}$$

$$\mu_{\mathbf{x},i} - \alpha\Sigma_{\mathbf{x},i} \geq \mathbf{x}_{min,i} \tag{4-33b}$$

with α determining the confidence level. The authors in [37], for example, use this formulation in a LBMPC setting to define constraints on the upper and lower tracking limits. The authors in [18] directly formulate constraints on the system states. A similar idea of using the standard deviation to define a confidence level was presented for obstacle avoidance of a quadrotor [28]. In this work, safety margins are defined by generating a standard deviation ellipsoids around the quadrotor position and enforcing a positive distance of the ellipsoid to the obstacle. Furthermore, the authors in [104] enlarge the uncertain position of agents by sigma confidence levels to guarantee robust collision avoidance.

Reformulation based on collision probability For polytopic state constraints where the uncertainty ξ is distributed according to a Gaussian distribution $\xi \sim \mathcal{N}(\mu, \Sigma)$ it holds that:

$$\Pr\left(\xi^T\mathbf{x} \leq b\right) = \Phi\left(\frac{b - \mu^T\mathbf{x}}{\sqrt{\mathbf{x}^T \sum \mathbf{x}}}\right) \tag{4-34}$$

where Φ is the cumulative distribution function (CDF) of the standard normal distribution. The chance constraint $Pr\left(\xi^T\mathbf{x} \leq b\right) \geq 1 - \delta$ can then be reformulated to:

$$b - \mu^T\mathbf{x} \geq \Phi^{-1}(1-\delta)\sqrt{\mathbf{x}^T \sum \mathbf{x}}. \tag{4-35}$$

This idea of using the (inverse) CDF can also be extended to more complex constraints. An approximate reformulation of polytopic chance constraints of the form in (4-20), was first introduced by the authors in [96] and was for example used in in the context of GP based MPC [27] [43]. Furthermore, the CDF can be used to reformulate spherical or ellipsoidal constraints [26] [14] based on the relation in 4-34. This was used in practice to tighten the track radius during autonomous racing [26] and for probabilistic collision avoidance between multiple robots and obstacles.

Using feedback to contain the uncertainty Similar to the tube based MPC approach, a stabilizing feedback controller can be introduced in the stochastic MPC formulation to prevent the spread of the uncertainty over the prediction horizon. A successful hardware demonstration using a feedback controller and GP learning-based MPC was shown on an autonomous race car [97]. Using a linear state feedback law, the authors use probabilistic reachable sets [105] to tighten the constraints. In particular, two types of constraints are considered: Half-space constraints and two-norm ball constraints. This shows the practical feasibility of using feedback in stochastic MPC to contain the uncertainty.

4-4 Discussion

The problem of navigating a quadrotor in a cluttered environment requires both a motion planner and a controller. This chapter covered background on motion planning and different types of motion planners and controllers that can be used to achieve the goal of safe navigation. Among the control methods that were presented, MPC is particularly attractive as it allows for the incorporation of the system model and constraints. It furthermore offers the opportunity to combine both the motion planner and controller by formulating OCP that simultaneously solves for an optimal trajectory given a reference path and the optimal control input. Here, MPCC has been discussed as a well-studied method. By combining optimal control methods with the GP model discussed in previous sections, the control formulation is extended to be both adaptive and intelligent.

Since the GP model leads to a stochastic system description, the second part of this chapter discussed ways to incorporate uncertainty into motion planning and control. Several researchers have found ways to incorporate robustness independently of the actual motion planner by using reachability-based methods. These methods rely on bounded uncertainties that the GP model does not provide. They can be approximated using sigma hulls, but this is a conservative approach. Optimal control methods, on the other hand, allow the inclusion of both stochastic and bounded uncertainties using concepts of robust and stochastic model predictive control. Moreover, the combination of robust tube-based MPC and a stochastic, chance-constrained formulation can contain uncertainty while maintaining probabilistic collision checking.

By combining the MPCC approach and the stochastic MPC formulations, a motion planner and controller can be implemented that is robust to system uncertainties caused by wind while finding the optimal path through the cluttered environment. Incorporating all the information into an OCP provides the opportunity to slow down the quadrotor in regions of greater uncertainty while optimising flight time when uncertainty is low.

Chapter 5

Conclusions

The survey is concluded with a summary of the findings, the research proposal derived from these findings and a planning of the research following up on this report.

5-1 Summary

This study addressed the problem of safely navigating a quadrotor in cluttered environments under wind disturbances and addressed three main research problems. These are: 1) determination of the system model, 2) extension of the system model with a GP to describe the wind disturbance, and 3) incorporation of the system model together with the uncertain description into a motion planning and control formulation.

Linear and nonlinear models were presented for the system model. When designing a position controller that does not demand extreme maneuvers from the quadrotor, a linear model is often sufficient. The inner-loop attitude controller is approximated by a linear, first-order model. Wind disturbances can be incorporated into the controller either by disturbance estimation from data or by direct modelling of the disturbances. Learning-based methods combine both ideas by using system data to derive a disturbance model. While learning-based methods have been more commonly used to model the residual dynamics of the system, extending this idea to model external disturbances, in this case wind, is a promising approach. Among the learning-based models, GPs were chosen because they can model the underlying dynamics with little prior knowledge and describe the uncertainty of the estimate in a probabilistic way.

For training the GP, a suitable mean and kernel must be chosen to model the wind disturbances well. A zero mean and a squared exponential kernel are most commonly used and can be a good starting point. If a lot of data is available for training and prediction, sparse GPs should be used. Various levels of approximation methods based on approximate inference have been presented for deriving an uncertain description of future states. In order to deal with the incoming in-flight data, different online learning methods were discussed. For a constant time wind field, one can manage experiences by discarding old data and adding new data points during runtime or selecting data points based on the given trajectory. Elaborate online learning methods should not be necessary.

Motion planning and control can be combined into one optimal control problem using MPCC. This requires a rough reference path, which does not have to be feasible. The OCP solution generates the optimal trajectory and control inputs for the quadrotor, taking into account obstacles and physical constraints, while managing progress along the path. To account for the uncertain description resulting from the Gaussian wind model, several robust motion planning methods have been discussed. Among these, stochastic MPC formulations are particularly promising as they can be used to extend the MPCC framework while accounting for the probabilistic nature of the uncertain description. To contain uncertainty, stochastic MPC formulations could be extended by a feedback controller.

5-2 Research proposal

To safely navigate a quadrotor in a cluttered outdoor environment, a motion planning and control method is implemented in one OCP. The OCP takes into account the quadrotor's environment, such as trees, and the wind disturbances acting on the quadrotor. Based on the sensor mechanisms on board of the drone, a GP is trained that describes the wind disturbances acting on the quadrotor depending on its position. In this way, a disturbance map is created, which is added to the nominal model. It is also assumed that an occupancy map with the positions of the obstacles is given a priori.

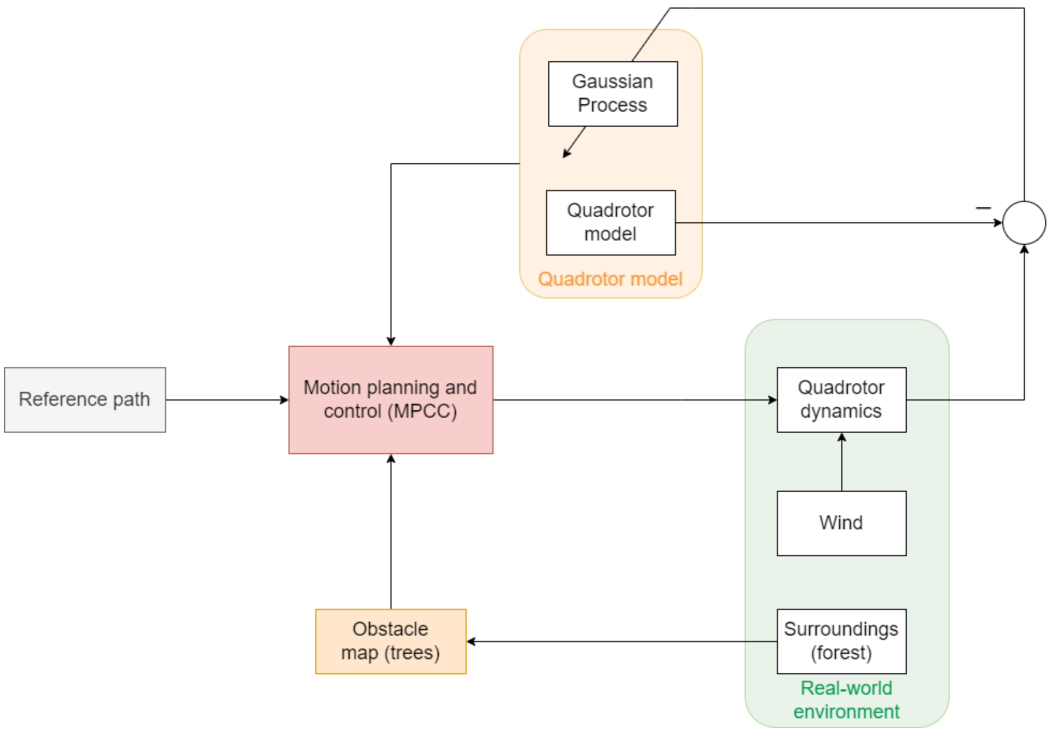


Figure 5-1: Overview of the learning-based motion planning and control method.

System Identification For the nominal model, the linear system description in (2-12) is used. In choosing this model, it is assumed that the quadrotor does not perform extreme manoeuvres to correct the wind disturbance. This also means that the attitude controller is given and only the position controller is integrated into the motion planner. To use this system model, system identification is performed in a first step to derive the attitude dynamics in (2-11e) and (2-11f), and a thrust model, which depends on the mass of the quadrotor.

Learning the wind disturbances While the occupancy map can be created based on on-board vision cameras, the quadrotor does not have any on-board sensors to directly measure the wind disturbance. Instead, the wind will be estimated using available state information as follows:

Starting from a nominal quadrotor model and assuming that there are no other disturbances acting on the quadrotor, the momentary wind is attributed to the difference between the predicted velocity $\hat{\mathbf{v}}_k$ and the measured velocity $\mathbf{v}_k$. With the measured error:

$$\mathbf{d} = \hat{\mathbf{v}}_k - \mathbf{v}_k \tag{5-1}$$

as the training objective and the quadrotor position $\mathbf{p}$ as the training input, a GP model is learned. Assuming that the quadrotor traverses the environment several times, the GP model is improved as more data becomes available. In this way, the disturbance map can be built and improved over time. Finally, the disturbance is added to the nominal model of the quadrotor:

$$f(\mathbf{x}, \mathbf{u}) = f_n(\mathbf{x}, \mathbf{u}) + B_d \mathbf{d}(\mathbf{p}), \tag{5-2}$$

where:

$$\mathbf{d}(\mathbf{p}) = \begin{bmatrix} d_{v_x}(\mathbf{p}) \sim \mathcal{N}\left(\mu^{d_{v_x}}(\mathbf{p}), \Sigma^{d_{v_x}}(\mathbf{p})\right) \\ d_{v_y}(\mathbf{p}) \sim \mathcal{N}\left(\mu^{d_{v_y}}(\mathbf{p}), \Sigma^{d_{v_y}}(\mathbf{p})\right) \\ d_{v_z}(\mathbf{p}) \sim \mathcal{N}\left(\mu^{d_{v_z}}(\mathbf{p}), \Sigma^{d_{v_z}}(\mathbf{p})\right) \end{bmatrix} \tag{5-3}$$

are three trained GPs and B_d is a selection matrix that maps the GP model to the velocity states. Since the disturbances are expressed by a mean and an uncertainty, the uncertainty is passed through the states, resulting in a probabilistic state description.

The main focus of this part of the research is to investigate how effectively the GPR can be used to create the wind disturbance map based on the available data. It will be investigated whether the three GPs can be trained individually or whether a correlation between the three wind directions needs to be included in the GP model. Furthermore, practical issues related to the reduction of computational complexity when using sparse GPs and the propagation of the uncertain system states are addressed.

Motion planning and control The second part of this research will focus on how to safely navigate the quadrotor through the environment given the obstacle positions and the probabilistic quadrotor model. For this purpose, a MPCC approach will be used, which enables motion planning in a model predictive control framework. Assuming a simple reference path and a given velocity, the optimal control formulation of the MPCC ensures obstacle avoidance while maintaining a reference velocity. Assuming a static map of obstacles, the main research question is how to extend the method with the learned map of wind disturbances. The probabilistic nature of the wind disturbance model leads to an uncertain state description, which is included in the MPCC framework. Inspired by work on robust and stochastic model predictive control, this is done by tightening the obstacle avoidance constraints based on an uncertain region around the quadrotor trajectory. Since the uncertainty information is probabilistic in nature, a chance constrained formulation is used to ensure safe obstacle avoidance. Furthermore, it is investigated whether the uncertainty can be contained by augmenting the MPCC controller with a linear feedback law, which is beneficial for the navigation of the quadrotor in a cluttered environment.

Research questions As a result of this research proposal, the following research questions should be answered:

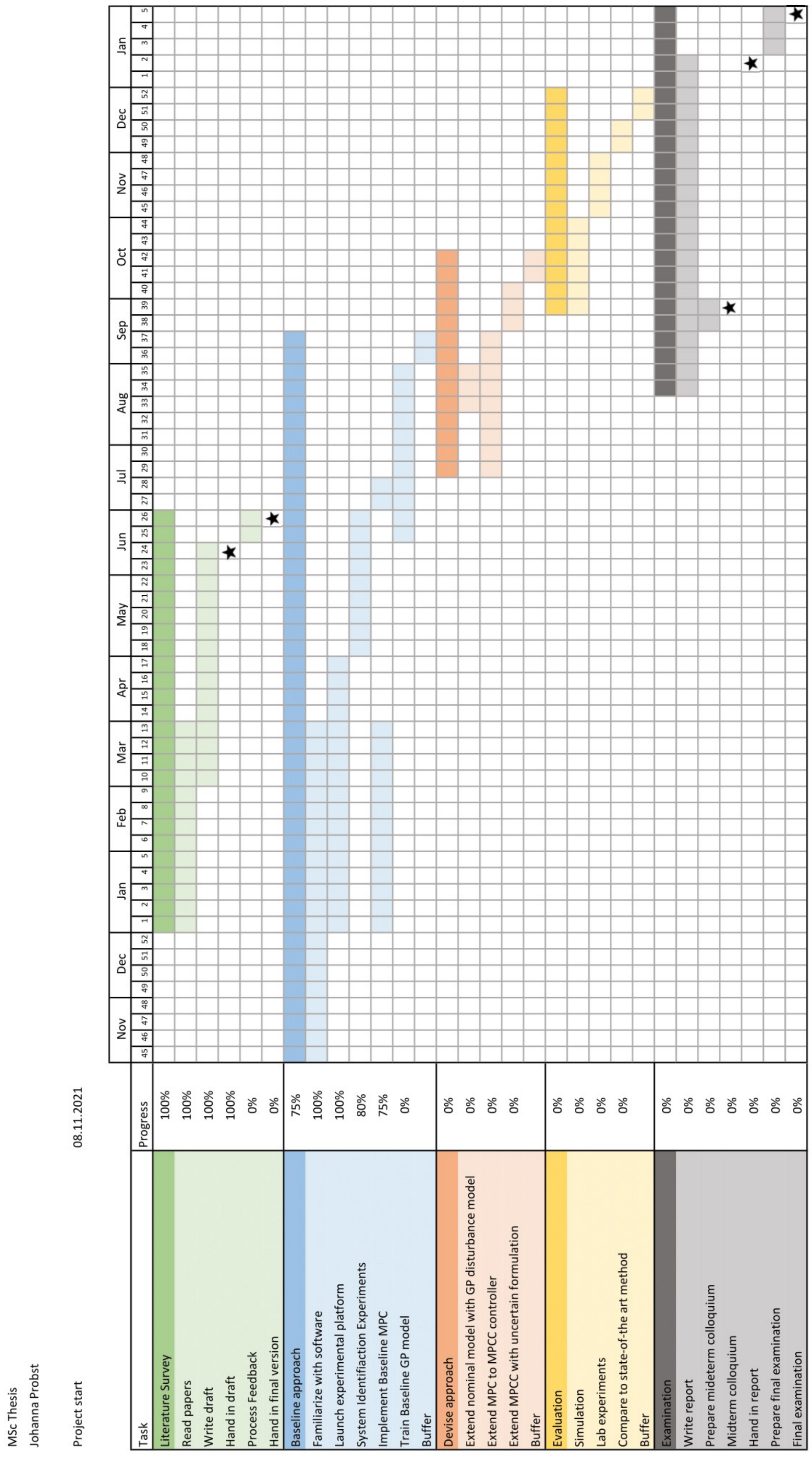
1. Are Gaussian processes a suitable method to model external wind disturbances using the quadrotor state information?
2. Is it possible to increase robustness of the MPCC controller by including the learned Gaussian process model into the MPCC formulation?

If time allows it, the research can be extended with the following question:

3. Can the learned Gaussian process model be update online to further improve the model accuracy in slightly changing environments?

An overview of the proposed method is shown in Figure 5-1. To demonstrate the research contribution of the proposed method, it is compared to a state-of-the art motion planner and controller that assumes a constant wind disturbance.

5-3 Planning



Task	Progress
Literature Survey	100%
Read papers	100%
Write draft	100%
Hand in draft	100%
Process Feedback	0%
Hand in final version	0%
Baseline approach	75%
Familiarize with software	100%
Launch experimental platform	100%
System Identifiaction Experiments	80%
Implement Baseline MPC	75%
Train Baseline GP model	0%
Buffer	
Devise approach	0%
Extend nominal model with GP disturbance model	0%
Extend MPC to MPCC controller	0%
Extend MPCC with uncertain formulation	0%
Buffer	
Evaluation	0%
Simulation	0%
Lab experiments	0%
Compare to state-of-the art method	0%
Buffer	
Examination	0%
Write report	0%
Prepare mideterm colloquium	0%
Midterm colloquium	0%
Hand in report	0%
Prepare final examination	0%
Final examination	0%

Figure 5-2: Planning of the MSc Thesis.

Bibliography

[1] G. Loianno and D. Scaramuzza, "Special issue on future challenges and opportunities in vision-based drone navigation," *Journal of Field Robotics*, vol. 37, no. 4, pp. 495–496, 2020.

[2] A. E. Gurgen, A. Majumdar, and S. Veer, "Learning provably robust motion planners using funnel libraries," *arXiv preprint arXiv:2111.08733*, 2021.

[3] HoverGames, "https://www.hovergames.com/."

[4] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," tech. rep., Epfl, 2007.

[5] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *2021 European Control Conference (ECC)*, pp. 1556–1563, IEEE, 2021.

[6] T. P. Nascimento and M. Saska, "Position and attitude control of multi-rotor aerial vehicles: A survey," *Annual Reviews in Control*, vol. 48, pp. 129–146, 2019.

[7] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control: Technical report," tech. rep., ETH Zurich, 2013.

[8] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[9] H. Mo and G. Farid, "Nonlinear and adaptive intelligent control techniques for quadrotor uav–a survey," *Asian Journal of Control*, vol. 21, no. 2, pp. 989–1008, 2019.

[10] K. Alexis, G. Nikolakopoulos, and A. Tzes, "On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances," *Asian Journal of Control*, vol. 16, no. 1, pp. 209–224, 2014.

[11] R. Amin, L. Aijun, and S. Shamshirband, "A review of quadrotor uav: control methodologies and performance evaluation," *International Journal of Automation and Control*, vol. 10, no. 2, pp. 87–103, 2016.

[12] T. Keviczky and G. J. Balas, “Receding horizon control of an f-16 aircraft: A comparative study,” *Control Engineering Practice*, vol. 14, no. 9, pp. 1023–1033, 2006.

[13] M. Kamel, M. Burri, and R. Siegwart, “Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[14] H. Zhu and J. Alonso-Mora, “Chance-constrained collision avoidance for mavs in dynamic environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.

[15] P. Bouffard, A. Aswani, and C. Tomlin, “Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results,” in *2012 IEEE International Conference on Robotics and Automation*, pp. 279–284, IEEE, 2012.

[16] R. Mahony, V. Kumar, and P. Corke, “Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor,” *IEEE Robotics and Automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[17] M. S. Kamel, T. Stastny, K. Alexis, and R. Siegwart, *Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System.* 05 2017.

[18] G. Cao, E. M. Lai, and F. Alam, “Gaussian process model predictive control of unmanned quadrotors,” in *2016 2nd International conference on control, automation and robotics (ICCAR)*, pp. 200–206, IEEE, 2016.

[19] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, “Learning quadrotor dynamics using neural network for flight control,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4653–4660, IEEE, 2016.

[20] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[21] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, “Provably safe and robust learning-based model predictive control,” *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[22] M. Lorenzen, M. Cannon, and F. Allgöwer, “Robust mpc with recursive model update,” *Automatica*, vol. 103, pp. 461–471, 2019.

[23] A. Didier, K. P. Wabersich, and M. N. Zeilinger, “Adaptive model predictive safety certification for learning-based control,” in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 809–815, IEEE, 2021.

[24] A. Didier, A. Parsi, J. Coulson, and R. S. Smith, “Robust adaptive model predictive control of quadrotors,” in *2021 European Control Conference (ECC)*, pp. 657–662, IEEE, 2021.

[25] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[26] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[27] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019.

[28] G. Garimella, M. Sheckells, and M. Kobilarov, "Robust obstacle avoidance for aerial platforms using adaptive model predictive control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5876–5882, IEEE, 2017.

[29] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with gaussian processes," in *2015 European Control Conference (ECC)*, pp. 2496–2501, IEEE, 2015.

[30] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE conference on decision and control (CDC)*, pp. 6059–6066, IEEE, 2018.

[31] N. Sydney, B. Smyth, and D. A. Paley, "Dynamic control of autonomous quadrotor flight in an estimated wind field," in *52nd IEEE Conference on Decision and Control*, pp. 3609–3616, IEEE, 2013.

[32] E. J. Smeur, G. C. de Croon, and Q. Chu, "Cascaded incremental nonlinear dynamic inversion for mav disturbance rejection," *Control Engineering Practice*, vol. 73, pp. 79–90, 2018.

[33] T. Tomić and S. Haddadin, "A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots," in *2014 IEEE/RSJ international conference on intelligent robots and systems*, pp. 4197–4204, IEEE, 2014.

[34] D. Hentzen, T. Stastny, R. Siegwart, and R. Brockers, "Disturbance estimation and rejection for high-precision multirotor position control," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2797–2804, IEEE, 2019.

[35] S. Waslander and C. Wang, "Wind disturbance estimation and rejection for quadrotor position control," in *AIAA Infotech@ Aerospace conference and AIAA unmanned... Unlimited conference*, p. 1983, 2009.

[36] C. T. Ton and W. Mackunis, "Robust attitude tracking control of a quadrotor helicopter in the presence of uncertainty," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 937–942, IEEE, 2012.

[37] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based nmpc enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

[38] M. O'Connell, G. Shi, X. Shi, and S.-J. Chung, "Meta-learning-based robust adaptive flight control under uncertain wind conditions," *arXiv preprint arXiv:2103.01932*, 2021.

[39] M. Mehndiratta and E. Kayacan, "Gaussian process-based learning control of aerial robots for precise visualization of geological outcrops," in *2020 European Control Conference (ECC)*, pp. 10–16, IEEE, 2020.

[40] Y. Wang, J. O'Keeffe, Q. Qian, and D. E. Boyle, "Kinojgm: A framework for efficient and accurate quadrotor trajectory generation and tracking in dynamic environments," *arXiv preprint arXiv:2202.12419*, 2022.

[41] A. Tagliabue and J. P. How, "Airflow-inertial odometry for resilient state estimation on multirotors," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5736–5743, IEEE, 2021.

[42] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*, vol. 9. KIT Scientific Publishing, 2010.

[43] M. Liu, G. Chowdhary, B. C. Da Silva, S.-Y. Liu, and J. P. How, "Gaussian processes for learning and control: A tutorial with examples," *IEEE Control Systems Magazine*, vol. 38, no. 5, pp. 53–86, 2018.

[44] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, 2005.

[45] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*, pp. 567–574, PMLR, 2009.

[46] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[47] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse gaussian process approximations," *Advances in neural information processing systems*, vol. 29, 2016.

[48] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian process classification," in *Artificial Intelligence and Statistics*, pp. 351–360, PMLR, 2015.

[49] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," in *Advances in Neural Information Processing Systems*, 2018.

[50] H. Liu, J. Cai, and Y.-S. Ong, "Remarks on multi-output gaussian process regression," *Knowledge-Based Systems*, vol. 144, pp. 102–121, 2018.

[51] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, Citeseer, 2011.

[52] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International conference on artificial intelligence and statistics*, pp. 1701–1710, PMLR, 2018.

[53] J. Quinonero-Candela, *Learning with uncertainty: Gaussian processes and relevance vector machines.* PhD thesis, Technical University of Denmark Lyngby, Denmark, 2004.

[54] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[55] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[56] M. F. Huber, "Recursive gaussian process: On-line regression and learning," *Pattern Recognition Letters*, vol. 45, pp. 85–91, 2014.

[57] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe, "Fixed-budget kernel recursive least-squares," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1882–1885, IEEE, 2010.

[58] Y. Liu and R. Tóth, "Learning based model predictive control for quadcopters with dual gaussian process," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 1515–1521, IEEE, 2021.

[59] E. Arcari, A. Carron, and M. N. Zeilinger, "Meta learning mpc using finite-dimensional gaussian process approximations," *arXiv preprint arXiv:2008.05984*, 2020.

[60] L. Quan, L. Han, B. Zhou, S. Shen, and F. Gao, "Survey of uav motion planning," *IET Cyber-systems and Robotics*, vol. 2, no. 1, pp. 14–21, 2020.

[61] S. M. LaValle, *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[62] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[63] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[64] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 65–100, 2010.

[65] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[66] S. LaValle and J. Kuffner, "Jj,"randomized kinodynamic planning,"," in *International Conference on Robotics and Automation*, vol. 1, pp. 473–479, 1999.

[67] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[68] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, pp. 2520–2525, IEEE, 2011.

[69] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for near-time-optimal quadrotor flight," *arXiv preprint arXiv:2108.13205*, 2021.

[70] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 2872–2879, IEEE, 2017.

[71] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[72] P. board, "Pixhawk," *Available from http://pixhawk.org*, 2021.

[73] A. Zulu and S. John, "A review of control algorithms for autonomous quadrotors," *arXiv preprint arXiv:1602.02622*, 2016.

[74] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, pp. 165–180, 2018.

[75] L. Li, L. Sun, and J. Jin, "Survey of advances in control algorithms of quadrotor unmanned aerial vehicle," in *2015 IEEE 16th international conference on communication technology (ICCT)*, pp. 107–111, IEEE, 2015.

[76] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[77] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 6137–6142, IEEE, 2010.

[78] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*, vol. 2. Nob Hill Publishing Madison, 2017.

[79] M. Behrendt, "Regelverhalten eines zeitdiskreten mpc-modells," 2009.

[80] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[81] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[82] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*, vol. 26. Birkhäuser, 2012.

[83] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 2017.

[84] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[85] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[86] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *The International Journal of Robotics Research*, vol. 39, no. 12, pp. 1419–1469, 2020.

[87] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, fast and slow: A framework for adaptive real-time safe trajectory planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 387–394, IEEE, 2018.

[88] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 947–982, 2017.

[89] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522, IEEE, 2017.

[90] H. Seo, D. Lee, C. Y. Son, C. J. Tomlin, and H. J. Kim, "Robust trajectory planning for a multirotor against disturbance based on hamilton-jacobi reachability analysis," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3150–3157, IEEE, 2019.

[91] S. Kim, D. Falanga, and D. Scaramuzza, "Computing the forward reachable set for a multirotor under first-order aerodynamic effects," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2934–2941, 2018.

[92] Y. Wu, Z. Ding, C. Xu, and F. Gao, "External forces resilient safe motion planning for quadrotor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8506–8513, 2021.

[93] A. Richards and J. P. How, "Robust variable horizon model predictive control for vehicle maneuvering," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 16, no. 7, pp. 333–351, 2006.

[94] S. Di Cairano and F. Borrelli, "Reference tracking with guaranteed error bound for constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2245–2250, 2015.

[95] S. Singh, B. Landry, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust feedback motion planning via contraction theory," *The International Journal of Robotics Research*, 2019.

[96] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1080–1094, 2011.

[97] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[98] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*, pp. 207–226, Springer, 1999.

[99] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[100] A. Bemporad, F. Borrelli, and M. Morari, "Min-max control of constrained uncertain discrete-time linear systems," *IEEE Transactions on automatic control*, vol. 48, no. 9, pp. 1600–1606, 2003.

[101] J. Köhler, M. A. Müller, and F. Allgöwer, "A novel constraint tightening approach for nonlinear robust model predictive control," in *2018 Annual American Control Conference (ACC)*, pp. 728–734, IEEE, 2018.

[102] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[103] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Transactions on automatic control*, vol. 51, no. 5, pp. 742–753, 2006.

[104] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "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)*, pp. 236–243, IEEE, 2017.

[105] L. Hewing and M. N. Zeilinger, "Stochastic model predictive control for linear systems using probabilistic reachable sets," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 5182–5188, IEEE, 2018.