Robust Vision-Based Planning for Quadrotor UAV Using Funnel Libraries

Ali Ekin Gurgen, '21

Submitted to the
Department of Mechanical and Aerospace Engineering
Princeton University
in partial fulfillment of the requirements of
Undergraduate Independent Work.

Final Report

April 28, 2021

Anirudha Majumdar
Clarence Rowley
MAE 442
48 pages
File Copy

This thesis represents my own work in accordance with University regulations.

Abstract

End-to-end vision-based solutions to robot control and planning problems have been the subject of significant research in recent years as deep neural networks started to give promising results. However, due to their lack of interpretability, providing rigorous guarantees on safety and performance of vision-based approaches is a very challenging yet crucial problem, especially when the system is subject to unknown disturbances that are not present in the training environments. Our work combines ideas from model-based reachability analysis and deep reinforcement learning to design a vision-based motion planning policy that holds PAC-Bayes generalization guarantees for planning in novel environments with unknown disturbances. The method presented in this thesis exploits the model knowledge to perform the offline computation of over-approximated reachable sets (funnels) around the motion primitive trajectories. Then, we use deep reinforcement learning to synthesize a vision-based motion planning policy that utilizes the precomputed funnel library. Finally, the PAC-Bayes framework is utilized to achieve strong generalization bounds on the average cost of the policies in novel environments. We implement this framework on a computer simulation of an autonomous quadrotor UAV navigating through obstacle-dense environments and achieve a final PAC-Bayes Bound of 20.4%. We evaluate the posterior policy distribution to validate that the PAC-Bayes generalization bound holds in novel environments with unknown external disturbances.

Acknowledgements

I would like to thank Professor Anirudha Majumdar for his continuous support and mentorship throughout the project as he provided me with the necessary computational resources and the background knowledge for conducting this research. I would like to thank him for instilling a love for robotics in me through his excellent teaching and exciting research. The dynamics and robotics courses I have taken from him have become significant factors in shaping my current career path, and the opportunity to interact with his research lab has taught me invaluable career and life lessons. I would also like to thank the members of the IRoM Lab for listening to my preliminary project presentation earlier in the year and providing valuable advice.

I would like to express my deepest appreciation towards IRoM Lab's postdoctoral research associate Sushant Veer for formulating this research topic and continuously advising me on theory and implementation throughout the project. Sushant is one of the kindest, smartest, and most helpful people I have met during my undergraduate years at Princeton. He has been an excellent mentor and a great role model. I am very grateful to Sushant for his priceless advice and support on my career and life decisions. It has been a great pleasure to work with him.

I would like to thank Jo Ann Kropilak-Love for attending to my questions regarding the project's logistical issues and her overall guidance throughout my time in the MAE department.

I would like to thank my dear friends Erce, Begum, and Elif for always bringing me so much joy and making Princeton feel like home. I am very grateful for their collective support and always being with me when I needed to share my celebrations and frustrations. I would like to thank my dear friend Utkan whose friendship has been the most valuable asset I have acquired. I am extremely grateful for his unconditional love and support since our high school years at Robert College.

Finally, I would like to thank my parents who made me the person I am today. Their unconditional love and support throughout my life have given me the strength to chase my dreams. I am also very grateful to have my precious cat Zeze who has provided me so much emotional support throughout all the difficult times in 2020. I am dedicating this thesis to my little sister Zeynep who will start school this year. Zeynep, you are the reason I keep pushing. I love you so much.

To my lovely sister Zeynep.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Providing end-to-end learning-based solutions to robot planning and control problems has been the subject of significant research as deep neural networks enabled a range of emerging applications. In comparison to the classical approaches, a learning-based approach forgoes the explicit geometric representations of the environment and does not rely on heavy hardware components. Most of these end-to-end learning-based approaches leverage the ability to *learn* from the statistical regularities in the visual data and are computationally less expensive once they have been trained.

While the improvements in computer vision and deep learning demonstrated various promising vision-based planning and control methods in research-driven robotic applications, very few purely vision-based approaches have been successfully transferred to industrial applications that are *safety-critical*, i.e. applications that need to hold provable robustness guarantees under certain safety constraints [50]. Due to the ambiguous nature of many perception problems, vision-based systems today are far from being perfect [52]. Furthermore, the lack of interpretability that exists in many deep-learning-based approaches creates an additional layer of complexity for providing safety and performance guarantees. A fundamental problem that is inherent in any learning-based approach is that the method's performance highly depends on the distribution that the training environments are drawn from. Therefore, it is very challenging to provide any formal safety and performance guarantees when this method is deployed in novel environments (i.e., environments that are unseen during the training) or environments with unknown disturbances.

To present a more concrete example of this challenge, imagine an unmanned aerial vehicle (UAV) trying to navigate through an obstacle-dense environment while only relying on its RGB-D sensor and a trained neural-network-based motion planner.

Two crucial questions naturally arise here:

1. How can we provide performance guarantees for this planner in novel environments (i.e., environments that are drawn from the same training distribution but are not encountered during training)?

2. How can we guarantee this robot's safety in the presence of unknown disturbances that have not been seen during training?

These challenges can be considered under the broader topic of *out-of-distribution generalization*. In learning theory, generalization of a learned model to test data that is statistically different than the training data is known as out-of-distribution generalization. One possibly naive attempt to address these challenges could be introducing some uncertainties to the training environments. For instance, one can introduce some wind force as an external disturbance to the quadrotor example, where we draw the magnitude and direction of the wind from some training distribution D. Then the planner's performance would become highly dependent on these training environments and still provide no out-of-distribution guarantees. Out-of-distribution generalization is currently an open problem and can be a crucial limiting factor for the deployment of learning-based robotic systems in safety-critical applications.

1.1 Contributions

In this thesis, we address these challenges by combining ideas from reachability analysis, control theory, computer vision, and machine learning to develop a robust vision-based motion planning policy that can provide rigorous safety and performance guarantees in novel environments with unknown disturbances. To our knowledge, the results in this thesis constitute the first attempt to provide safety and performance guarantees on generalization performance for vision-based planning in the presence of unknown disturbances.

The approach presented in this thesis combines model-based control and reachability analysis with a vision-based learning framework to achieve out-of-distribution generalization guarantees that hold under unknown disturbances. In particular, we first design a finite library of desired trajectories, called *motion primitives*, and implement a controller for the undertaken dynamical system. Then our next step is to perform an offline computation of over-approximated reachable sets, known as *funnels*, around the motion primitives to create a library of funnels. A funnel can be

described as a set of states that the dynamical system is guaranteed to stay inside during the execution of the motion primitive as long as the uncertainties are bounded and the initial state of the system is within the funnel's initial set of states. Finally, we train a neural-network-based motion planner with *PAC-Bayes generalization guarantees* that utilizes the precomputed funnel library to avoid collision in the presence of bounded unknown disturbances. We perform training in the absence of disturbances and use deep reinforcement learning to train the planner to choose the motion primitive whose funnel does not collide with any obstacles. Since the dynamical system is guaranteed to stay within the bounds of the precomputed funnel under the given uncertainty constraints, we achieve a robust vision-based planner with safety and generalization guarantees.

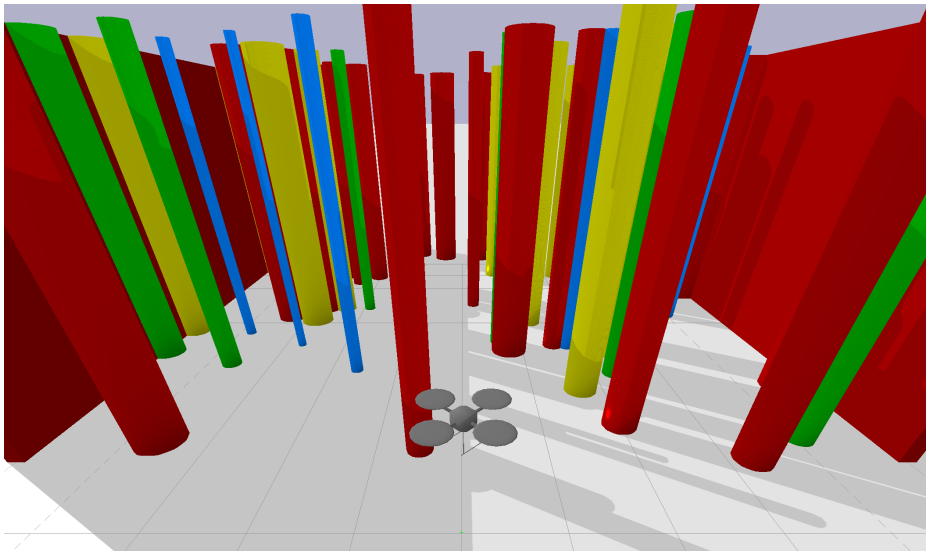


Figure 1.1: Simulation of a UAV navigating through an obstacle-dense environment.

We demonstrate and validate our method through a simulation experiment of an autonomous quadrotor navigating through obstacle-dense environments (see Figure 1.1). The quadrotor UAV uses an onboard RGB-D camera as its only sensor modality and has no prior knowledge of the environment. The robot's task is to move in the nominal direction (i.e., initial heading direction) while avoiding obstacles until it reaches the end of the room. The framework presented in this thesis can be applied to different linear and nonlinear dynamical systems (e.g., autonomous ground vehicle navigation, robotic arm manipulation, etc.) as well as to different task specifications (e.g., reaching a goal region, following checkpoints along a path, etc.).

1.2 Outline

Chapter 2 discusses the prior work relevant to vision-based planning and robust motion planning and also provides some background information about motion primitives, funnel libraries, and deep-learning-based robot planning and control methods. Chapter 3 defines the nonlinear quadrotor dynamics used throughout this work. Chapter 4 discusses the generation of motion primitives, shows the design and implementation of a nonlinear controller for the quadrotor UAV, and describes the offline reachability analysis that is performed to compute funnels. Chapter 5 describes the vision-based learning framework that utilizes the funnel library and discusses the training procedure in detail. Chapter 6 presents and discusses the results of this robust vision-based planning framework's implementation on the quadrotor UAV and finally concludes the thesis with a discussion of the challenges and future work.

Chapter 2

Relevant Work and Background

The navigation of mobile robots through obstacle-dense environments is one of the most fundamental problems in robotics. A classical approach to tackling the navigation problem involves solving three main subtasks: localization, mapping, and motion planning. Localization is the problem of estimating the robot's pose relative to a given map of the environment. Probabilistic localization algorithms that are variants of the basic Bayes filter can be used to perform mobile robot localization [45]. Mapping, on the other hand, is the problem of constructing a representation of the surrounding environment relative to the position of the robot. With the assumption that the robot's pose is known, algorithms such as Occupancy Grid Mapping can be used to generate maps from sensor measurements [46]. When these two subtasks are performed concurrently, we arrive at the Simultaneous Localization and Mapping, commonly abbreviated as SLAM, which has become the standard solution to the mapping and localization problems in many robotic applications [47]. Motion planning is the task of determining a viable path from point A to point B in the configuration space. Obstacle avoidance is the core problem of motion planning and various discrete (e.g., Bellman-Ford, A*) and continuous (e.g., rapidly-exploring random trees, probabilistic roadmap method) path planning algorithms are commonly used to perform this task.

2.1 Vision-based Planning

In the last decades, the use of computer vision in robot planning and control has become a very powerful method. Vision can efficiently encode rich information about the geometric features of an environment, and it provides an energy-efficient and

lightweight sensor solution to the robot navigation problem.

Classical approaches that incorporate computer vision into robot planning involve using RGB-D images to perform state estimation and mapping, which are later used for classical motion planning algorithms. With increasing computational power, vision-based motion planning has shifted towards a deep learning-based approach using neural networks to produce *end-to-end* solutions to the navigation problem. Some of the emerging vision-based planning methods can be considered in three categories: (1) self-supervised learning-based approaches; (2) imitation learning-based approaches; (3) deep reinforcement learning-based approaches [49].

2.1.1 Self-supervised learning-based approaches

In general, self-supervised learning can be described as an autonomous form of supervised learning where the machine does not require any human input for data labeling. In vision-based planning applications, self-supervised learning-based approaches often aim to exploit visual data to learn low-dimensional embeddings of the robot's high-dimensional dynamics and try to perform planning in the lower-dimensional latent space [23, 24, 19, 51].

2.1.2 Imitation learning-based approaches

In imitation learning, the machine is trained to perform a task by observing an expert's demonstrations. For example, the steering angle of a human driving a car can act as the training signal of a self-driving car [9]. Similarly, imitation learning has been an effective method used in many manipulation tasks and humanoid applications [39, 10]. In recent work, imitation learning has been used to train deep neural networks that provide sampling-based [18] or complete [38] feasible motion plans for robot planning.

2.1.3 Deep reinforcement learning-based approaches

In reinforcement learning (RL), the agent learns a policy to maximize a cumulative reward by trial and error. Deep reinforcement learning combines RL with deep learning to learn complex robotic skills from raw visual input by uncovering the lower-dimensional latent space representations that are relevant to the planning task [42, 43, 16, 14, 25].

2.2 Planning with Motion Primitives

A common approach in robot control and planning is to utilize motion primitives that consist of precomputed trajectories that can be sequentially composed to generate a rich class of motions that are feasible for the system's dynamic constraints [49]. These trajectory libraries have been used in diverse robotic applications such as ground vehicle navigation [44], aerial vehicle navigation [49], humanoid control [30], and grasping [6]. In motion primitive-based approaches to the robot navigation problem, usually each primitive represents a single step in the robot's trajectory.

Although simple motion primitive libraries can greatly simplify the undertaken control problem and easily generate feasible collision-free paths even in obstacle-dense environments, they cannot provide provable performance guarantees under unknown disturbances.

2.3 Robust Motion Planning

Ensuring the safety of a robotic system in deployment is a challenging task, especially in the presence of uncertainties. Robust motion planning addresses the challenges related to uncertainties in the system dynamics and the environment. For linear dynamical systems with Gaussian uncertainty, one can perform robust motion planning by keeping track of a *belief state* distribution over the state space [3]. However, these approaches become computationally demanding for non-linear dynamics and non-Gaussian uncertainties. In this thesis, we consider *nonlinear system dynamics* and *bounded unknown disturbances*.

Reachability analysis has been a prevalent method in robotics for performing safety verification of linear and nonlinear dynamical systems. In general, reachability analysis aims to compute the set of states that a dynamical system can enter during a given time frame. A computed reachable set of states can then be used for the safety verification of the system. However, reachability analysis of nonlinear systems is a very challenging problem since nonlinearity makes it computationally very difficult to track the evolution of all possible sets of states in time. In fact, it is not yet possible to compute exact reachable sets for nonlinear continuous systems [2]. Current approaches for approximating reachable sets can be considered in three categories [28]:

1. Taylor model-based approaches: These approaches approximate nonlinear dynamics using Taylor expansions [3, 2]. Some of the available tools adopting

these approaches are Flow* [11] and JuliaReach [8]. In this thesis, we take a similar approach and utilize the JuliaReach toolbox.

2. Hybridization-based approaches: These approaches reduce nonlinear systems to affine systems with uncertain inputs by rewriting them as piecewise linear dynamics [5].

3. Simulation-based approaches: These approaches perform reachability analysis by simulating the nonlinear dynamical system multiple times with different initial conditions, parameters, and disturbances [22]. Although these approaches can efficiently conclude that a system is unsafe, they cannot provide provable safety guarantees.

A common approach in robust planning is to perform reachability analysis around precomputed motion primitives to generate libraries of safe sets. Early work on robust planning with motion primitives focused on generating "tubes" of specified radii around open-loop trajectories such that all points in the tube correspond to collision-free configurations [21]. Later work on this topic has utilized tools from control theory, optimization, and dynamic programming to compute tubes of reachable sets that are guaranteed to be collision-free [41, 35].

A similar approach has utilized Lyapunov functions and sums-of-squares programming to minimize the size of the reachable sets around motion primitives to compute outer approximations of safe reachable sets called *funnels*. This thesis draws inspiration from previous studies that use funnels to guarantee robustness under bounded unknown disturbances in real-time planning for a quadrotor UAV [3]. A pictorial depiction of robust planning under uncertainty with funnels is given below in Figure 2.1. Since the funnels are an outer approximation of the reachable sets, the quadrotor is guaranteed to stay inside the funnel as long as the uncertainties are bounded and the initial state of the quadrotor is within the funnel's initial set of states (inlet). Notice that Figure 2.1 depicts a two-dimensional quadrotor UAV navigation example, while funnels can be computed in higher dimensions to include other state variables as well. In this thesis, we consider the full three-dimensional, nonlinear quadrotor dynamics and compute funnels for all 12 state variables.

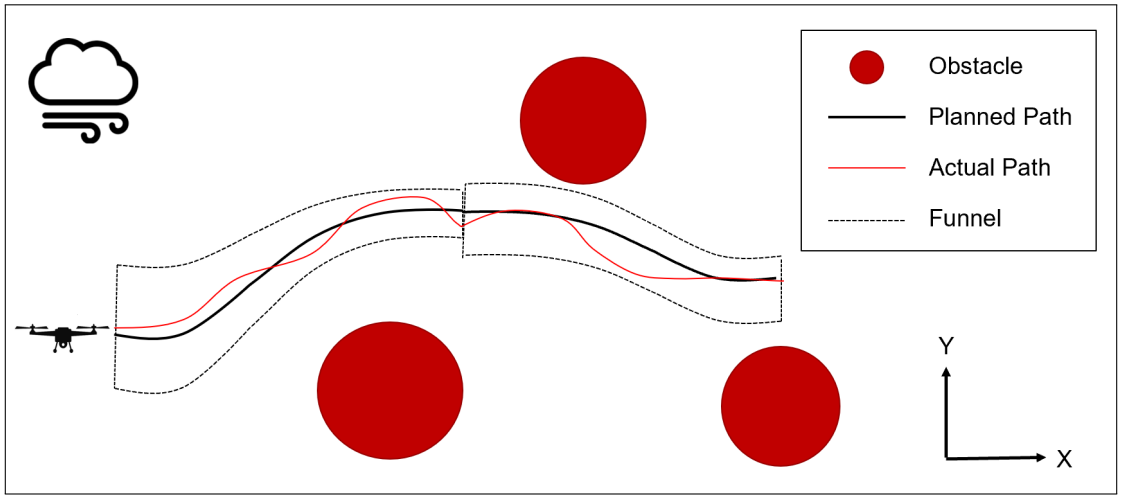


Figure 2.1: Motion planning for a quadrotor UAV using funnels in a two-dimensional navigation task. UAV's goal is to move in the nominal X-direction while avoiding obstacles in the environment. Bounded uncertainties are present in the form of wind gusts and cause the actual path of the UAV to deviate from the planned trajectories.

2.4 Generalization Guarantees

Machine learning-based methods strive to achieve *generalization* that is beyond the training data, assuming that the test and training data both come from the same underlying distribution. In learning theory, Probably Approximately Correct (PAC) learning is a framework for learning a function (also known as a hypothesis) that has a low generalization error with a high probability. PAC-Bayes takes a Bayesian approach by assuming a prior over the hypothesis class and aims to learn a posterior distribution from the training data [29]. In recent years, the PAC-Bayes framework has been successfully used in providing generalization bounds for deep neural network-based supervised learning approaches [13, 40]. This thesis builds on previous work that used the PAC-Bayes framework with deep reinforcement learning to provide provable generalization guarantees for learned vision-based motion planning policies [49].

2.5 Quadrotor UAV Navigation

In recent years, unmanned aerial vehicles (UAVs), especially quadrotors, have been the subject of significant research in academia and industry. Due to their agility, small size, autonomy, and low cost, autonomous navigation of quadrotor UAVs became an ideal method for exploration [17, 7, 15] and search & rescue [34] applications in

confined, obstacle-dense environments [53]. Studies have focused on state estimation [32], motion control [26], and trajectory planning [33] problems of the quadrotor navigation task [37]. The next chapter gives a more detailed overview of the quadrotor UAV and describes the dynamical model used in this thesis.

Chapter 3

Quadrotor Dynamics

Quadrotor (also known as a quadcopter) is an aircraft with four rotors on the extremities of its frame and an electric board in the middle. Due to its high maneuverability, low cost, and vertical takeoff ability, the quadrotor has emerged as a popular platform for robotic research. In this chapter, we present an overview of the three-dimensional, non-linear rigid body dynamics of the quadrotor and explain the mathematical model that is used in the rest of this thesis. Refer to [31] and [33] for a more detailed version of this overview.

3.1 Motor Model

We can control the quadrotor's position and attitude by changing the four rotor thrusts. Each rotor has an angular speed ω_i and produces a force F_i and a moment M_i:

$$F_i = k_f \omega_i^2 \tag{3.1.1}$$

$$M_i = k_m \omega_i^2 \tag{3.1.2}$$

Although we can express the control inputs using rotor thrusts F_1, F_2, F_3, and F_4, it is more convenient to use a control input vector $\boldsymbol{u}$ where u_1 is the total thrust ($\sum_{i=1}^{4} F_i$) and u_2, u_3, u_4 are the body moments. Defining L to be the distance between

the center of the quadrotor and the rotors, we have:

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \underbrace{\begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & k_f L & 0 & -k_f L \\ -k_f L & 0 & k_f L & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix}}_{\Gamma} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3.1.3}$$

Then, one can design and operate a controller using only $\boldsymbol{u}$ and compute the required rotor thrusts using the inverse of the matrix Γ.

3.2 Inertial Properties

Important inertial properties of the quadrotor model are the total mass of the quadrotor m and the rigid body moment of inertia matrix I. It is assumed that the body frame is aligned with the quadrotor's body principal axes of inertia such that the inertia matrix is diagonal:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.2.1}$$

In practice, the three moments of inertia I_x, I_y, I_z can be estimated from an accurate CAD model of the quadrotor. Properties of the quadrotor model used in this thesis are given in Table 3.1 below.

Table 3.1: Inertial properties of the quadrotor.

m	$0.65\ kg$
I_{xx}	$7.5 \times 10^{-3}\ \frac{kg}{m^2}$
I_{yy}	$7.5 \times 10^{-3}\ \frac{kg}{m^2}$
I_{zz}	$1.3 \times 10^{-2}\ \frac{kg}{m^2}$

3.3 Dynamical Model

The three-dimensional quadrotor dynamical model has 6 degrees of freedom: 3 positions (x, y, z) and 3 orientations (ϕ, θ, ψ).

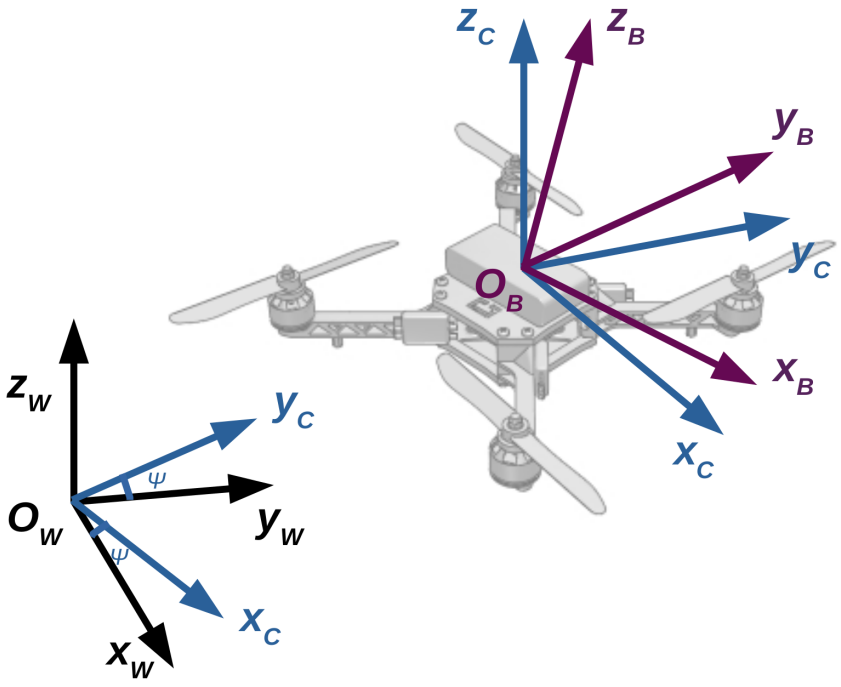


Figure 3.1: Quadrotor model reference frames using Z-X-Y Euler angle convention. Quadrotor figure reference: [20]

One can conveniently describe the kinematics of the quadrotor using additional reference frames and Euler angle rotations. The world frame $\mathcal{W}$ is defined by x_W, y_W, and z_W as shown in Figure 3.1. One can introduce an intermediate frame $\mathcal{C}$ (defined by x_C, y_C, and z_C) by rotating the world frame $\mathcal{W}$ by an angle ψ about the z_W axis. Finally, the body frame $\mathcal{B}$ of the quadrotor is defined by rotating $\mathcal{C}$ by an angle ϕ about the x axis and by an angle θ about the y axis. The origin of the resulting body frame $\mathcal{B}$ is attached to the center of mass of the quadrotor with z_B perpendicular to the plane of rotors and pointing vertically up. $\mathcal{B}$ is defined by the basis $\{x_B, y_B, z_B\}$. This is known as the $Z - X - Y$ Euler angle convention and produces the following rotation matrix ${}^{\mathcal{W}}R_{\mathcal{B}}$ from $\mathcal{W}$ to $\mathcal{B}$.

$$ {}^{\mathcal{W}}R_{\mathcal{B}} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi s\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{3.3.1} $$

The angular velocity of the quadrotor in the body frame, ω_{BW}, is given by p, q, and r, which are related to the derivatives of the roll, pitch and yaw angles (ϕ, θ, ψ) by Equation 3.3.3.

$$\omega_{BW} = p\boldsymbol{x}_B + q\boldsymbol{y}_B + r\boldsymbol{z}_B \tag{3.3.2}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.3.3}$$

If we denote the position vector of the quadrotor's center of mass by $\boldsymbol{r}$ and assume that the only forces acting on the system are gravity (in the $-z_W$ direction) and forces from the rotors, we arrive at the following equation governing the acceleration of the quadrotor's center of mass:

$$m\ddot{\boldsymbol{r}} = -mg\boldsymbol{z}_W + u_1\boldsymbol{z}_B \tag{3.3.4}$$

Euler's equations provide us the following equation for determining the angular acceleration of the quadrotor:

$$\dot{\omega}_{BW} = I^{-1}\left[-\omega_{BW} \times I\omega_{BW} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right] \tag{3.3.5}$$

where I is the moment of inertia matrix given in Equation 3.2.1. Finally, the state of the system can be given by the position and velocity of the center of mass as well as the orientation and the angular velocity. The 12-dimensional state vector given in Equation 3.3.6 and the representation of the nonlinear, continuous quadrotor dynamics given in Equation 3.3.7 will be used throughout this thesis.

$$\boldsymbol{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & \dot{x} & \dot{y} & \dot{z} & p & q & r \end{bmatrix}^T \tag{3.3.6}$$

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3.3.7}$$

Chapter 4

Computing Funnels

This chapter starts with a discussion of the trajectory generation problem for the quadrotor's motion primitive library. Then, we show how to utilize *differential flatness* of the quadrotor dynamics to generate the full trajectory states from the *flat outputs*. After that, we explain the implementation of a nonlinear controller for the quadrotor UAV that can successfully track the computed trajectories. Finally, we address the funnel computation problem and explain the offline reachability analysis procedure that is performed in this thesis.

4.1 Motion Primitives

As mentioned earlier in Chapter 2, the approach presented in this thesis utilizes a library of funnels and the underlying motion primitives to successfully navigate the quadrotor. The first step of computing funnels is to generate the underlying nominal trajectories $\boldsymbol{x}_T : [0, T] \rightarrow \mathbb{R}^n$. Since the quadrotor dynamics are differentially flat, one can express the nominal trajectory states and the control inputs as algebraic functions of four flat output states x, y, z, and ψ. Therefore, any smooth trajectory $\boldsymbol{\sigma}(t)$ in the flat output space with reasonably bounded derivatives can be followed by the quadrotor [33]. Then the full trajectory states $\boldsymbol{x}_T(t)$ and the nominal control inputs $\boldsymbol{u}_T(t)$ can be reconstructed from $\boldsymbol{\sigma}(t)$ and its derivatives as shown in Equation 4.1.2.

$$\boldsymbol{\sigma}(t) = [x(t), y(t), z(t), \psi(t)]^T \text{ for } t \in [0, T] \tag{4.1.1}$$

$$(\boldsymbol{x}_T, \boldsymbol{u}_T) = \Phi(\sigma, \dot{\sigma}, \ddot{\sigma}, \dddot{\sigma}, \ddddot{\sigma}) \tag{4.1.2}$$

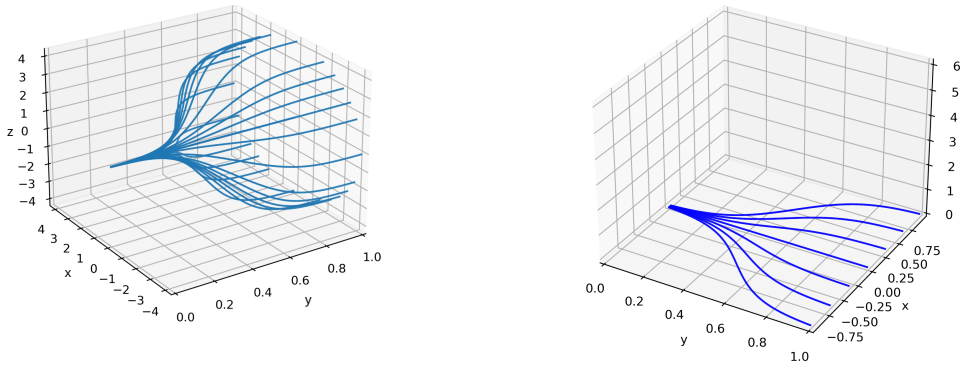
We assume that the robot moves in the nominal forward direction y with constant velocity v_0. We also assume that $\psi(t) = 0$ for $t \in [0, T]$ in all trajectories (i.e., yaw angle does not change and the quadrotor always faces the forward direction). Then, we generate smooth, sigmoid-like trajectories in $\mathbb{R}^3$ using the following equations:

$$x(t) = x_0 + \frac{a}{1 + e^{b(t-c)}} \tag{4.1.3}$$

$$y(t) = y_0 + v_0 \times t \tag{4.1.4}$$

$$z(t) = z_0 + \frac{a}{1 + e^{b(t-c)}} \tag{4.1.5}$$

where parameters a, b, c can be varied to change the shape of the trajectories in the three-dimensional space. Initially, we have generated 25 motion primitive trajectories where both y and z positions were changing while moving in the nominal x-direction. The library of motion primitives with 25 trajectories is shown in Figure 4.1a.



(a) Motion primitives with 25 trajectories. (b) Motion primitives with 7 trajectories.

Figure 4.1: Motion primitive libraries plotted in $\mathbb{R}^3$.

These trajectories provide a rich class of motions in the three-dimensional space that can be very useful in certain environments. However, since our environment (as shown in Figure 1.1) only consists of cylindrical obstacles, we have decided that variation in the z-direction was not crucial for obstacle avoidance. Therefore, we have decided to assume $z(t) = z_0$ for $t \in [0, T]$ in all motion primitives and generated a new library with 7 trajectories, as shown in Figure 4.1b. Figure 4.2 shows the motion primitive trajectories in the PyBullet simulation of the quadrotor.

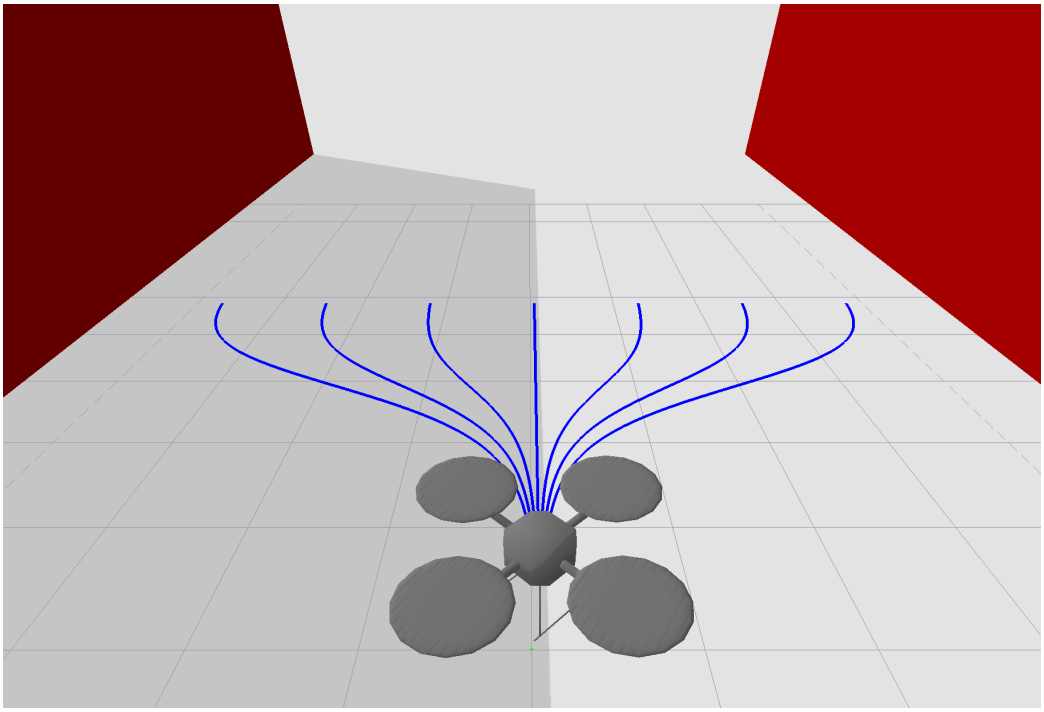


Figure 4.2: Motion primitive trajectories in the PyBullet simulation.

4.2 Controller

After generating the motion primitives and reconstructing the full trajectory states and nominal control inputs, the next step in our approach is to design and implement a controller that enables the robot to track a generated trajectory $\boldsymbol{x}_T(t)$. In this thesis, we implement the nonlinear tracking controller presented in [33]. First, we define the errors on position, velocity, orientation, and angular velocity as

$$\boldsymbol{e}_p = \boldsymbol{r} - \boldsymbol{r}_T \tag{4.2.1}$$

$$\boldsymbol{e}_v = \dot{\boldsymbol{r}} - \dot{\boldsymbol{r}}_T \tag{4.2.2}$$

$$\boldsymbol{e}_R = \frac{1}{2}(R_{des}^T \, {}^{\mathcal{W}}R_{\mathcal{B}} - {}^{\mathcal{W}}R_{\mathcal{B}}^T R_{des})^\vee \tag{4.2.3}$$

$$\boldsymbol{e}_\omega = {}^{B}[\omega_{BW}] - {}^{B}[\omega_{BW,T}] \tag{4.2.4}$$

where R_{des} is the desired rotation matrix that transforms z_W to the desired total thrust vector $z_{B,des}$ and $\vee$ is the *vee* map operation that maps a skew-symmetric

matrix to a vector. Having defined the errors, the control inputs are computed using the control law given below.

$$u_1 = -K_p \boldsymbol{e}_p - K_v \boldsymbol{e}_v + mg\boldsymbol{z}_W + m\ddot{\boldsymbol{r}}_T \tag{4.2.5}$$

$$\begin{bmatrix} u_2 & u_3 & u_4 \end{bmatrix}^T = -K_R \boldsymbol{e}_R - K_\omega \boldsymbol{e}_\omega \tag{4.2.6}$$

where K_p, K_v, K_R, and K_ω are positive definite gain matrices where the parameter for each state variable can be individually tuned for better tracking. These control parameters were manually tuned during the reachability analysis computation to give tighter funnels. Another approach, that is computationally very demanding. would be running a gradient-based optimization algorithm to optimize the selection of control parameters such that the volumes of the resulting reachable sets are minimized.

4.3 Reachability Analysis

This section describes the methodology used for performing the offline reachability analysis to compute the quadrotor's funnel library.

4.3.1 Introducing Uncertainties

The main challenge addressed in our approach is to successfully perform robust vision-based planning when the system is subject to bounded unknown uncertainties. These uncertainties can exist in terms of external disturbances or model uncertainties. Suppose that the system is subject to an uncertainty term $w(t) \in \mathbb{R}^d$ and the dynamics of the system becomes:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \tag{4.3.1}$$

In the case of the quadrotor, $w(t)$ can be thought of as an external disturbance such as a wind force in the environment. We modeled this external force by adding bounded uncertainty terms to the linear accelerations during the reachability analysis. Similarly, we modeled the torque caused by the external disturbances by adding bounded uncertainty terms to the body moments in the control input. The uncertainty models along with their respective bounds are given below in Equations 4.3.2 and 4.3.3.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \ddot{x}_{nominal} \\ \ddot{y}_{nominal} \\ \ddot{z}_{nominal} \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \quad \text{with } w_x, w_y, w_z \in [-0.1, 0.1] \; \frac{m}{s^2} \tag{4.3.2}$$

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} u_{2,nominal} \\ u_{3,nominal} \\ u_{4,nominal} \end{bmatrix} + \begin{bmatrix} w_{u_2} \\ w_{u_3} \\ w_{u_4} \end{bmatrix} \quad \text{with } w_{u_2}, w_{u_3}, w_{u_4} \in [-0.025, 0.025] \; N \cdot m \tag{4.3.3}$$

Uncertainties in the system cause the robot to deviate from its nominal trajectory. The feedback controller proposed in Section 4.2 corrects for these deviations. Figure 4.3 shows an example in $\mathbb{R}^3$ where the robot starts with initial offsets in its x and z positions as well as in its orientation. The implemented feedback controller successfully corrects for the deviations and tracks the nominal trajectory in all 12-dimensional state space (only x, y, z tracking shown in the figure). Similarly, Figure 4.4 shows an example from the computer simulation where the robot is subject to external disturbances in the environment. The blue line represents the nominal motion primitive trajectory and the red line is the actual path of the robot using the controller presented in Section 4.2.

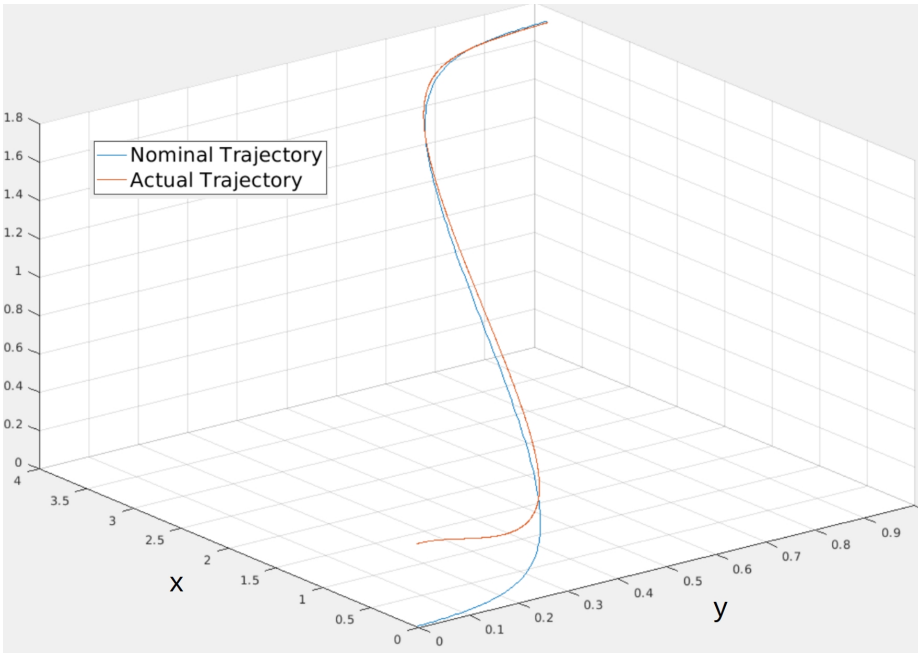


Figure 4.3: Controller corrects for deviations around the nominal trajectory.

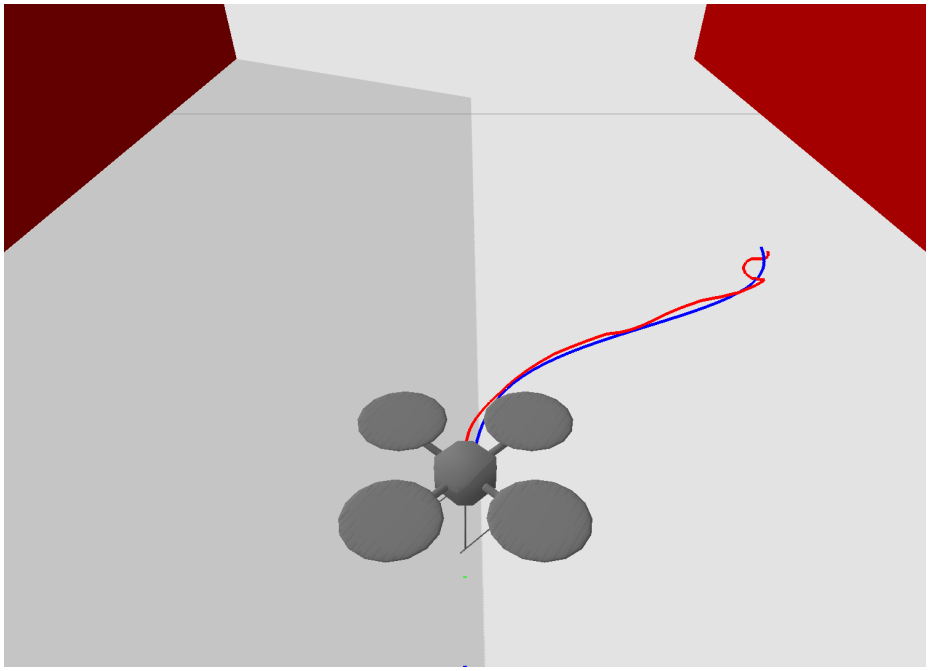


Figure 4.4: Simulation example showing the controller correcting for deviations around the nominal trajectory when the robot is subject to external disturbances in the environment. The blue line represents the nominal motion primitive trajectory and the red line is the actual path of the robot.

4.3.2 Problem Formulation

Having introduced a model of the uncertainties to the system, the next step is to compute the funnels. Consider the dynamical system:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t)) \tag{4.3.4}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state of the system at time t, $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the control input, and $\boldsymbol{w}(t) \in \mathbb{R}^d$ is the bounded unknown uncertainties in the system. Given a nominal trajectory $\boldsymbol{x}_T : [0, T] \to \mathbb{R}^n$, the corresponding nominal control input $\boldsymbol{u}_T : [0, T] \to \mathbb{R}^m$, and a set of initial states $X_0 \subset \mathbb{R}^n$, our goal is to compute a tight over-approximation of the reachable set of states $F(t) \subset \mathbb{R}^n$ that the system may evolve to at time $t \in [0, T]$. These over-approximations of the reachable sets will form the funnel library that will be used during online vision-based planning. Section 2 summarizes different optimization methods that have been used in previous work to perform such a reachability computation. In this thesis, we utilized a Julia package called JuliaReach.

4.3.3 JuliaReach

JuliaReach is an open-source library written in the Julia programming language that can compute rigorous approximations of the set of states reachable by a dynamical system [8]. JuliaReach implements various reachability analysis algorithms and can solve systems of ODEs for both continuous and hybrid dynamical systems.

In order to compute funnels using JuliaReach, we defined the quadrotor dynamics with uncertainties, imported the algebraic equations for the motion primitive trajectories, and implemented the nonlinear feedback controller in Julia. After defining the set of initial states $X_0 \subset \mathbb{R}^n$, we utilized JuliaReach's TMJets algorithm to solve the reachability problem that is formulated in 4.3.2. TMJets is a Taylor model-based algorithm that uses Taylor polynomials with guaranteed error bounds to approximate the nonlinear dynamics. We used Taylor model approximations of order 3 since higher order approximations became computationally challenging. The computation time for each funnel was approximately 5 minutes, running on a 4.5 GHz laptop with 16 GB memory and 12 cores.

Finally, we used hyperrectangles to over-approximate the computed reachable set of states. A hyperrectangle is the Cartesian product of one-dimensional intervals. Therefore, at any given time, the funnel is defined by a one-dimensional interval for each of its 12 state variables. Evaluation of all 12 state variables in time for a computed funnel is shown in Figure 4.5. Figure 4.6 shows the representations of the funnels in $\mathbb{R}^3$ for all 7 motion primitives in the library.

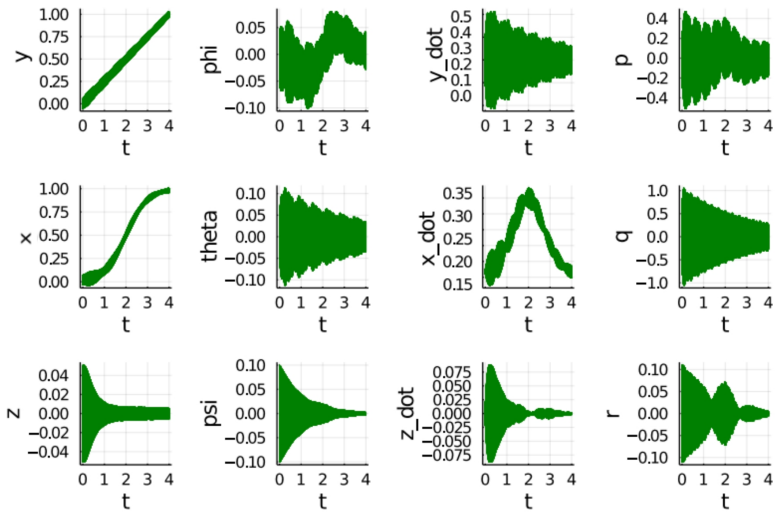


Figure 4.5: Evaluation of all 12 state variables in time for Funnel 0.

Notice that the time horizon of each funnel is 4 seconds and with a constant forward velocity v_0 of 0.25 m/s, the robot travels 1 meter in the nominal y-direction.

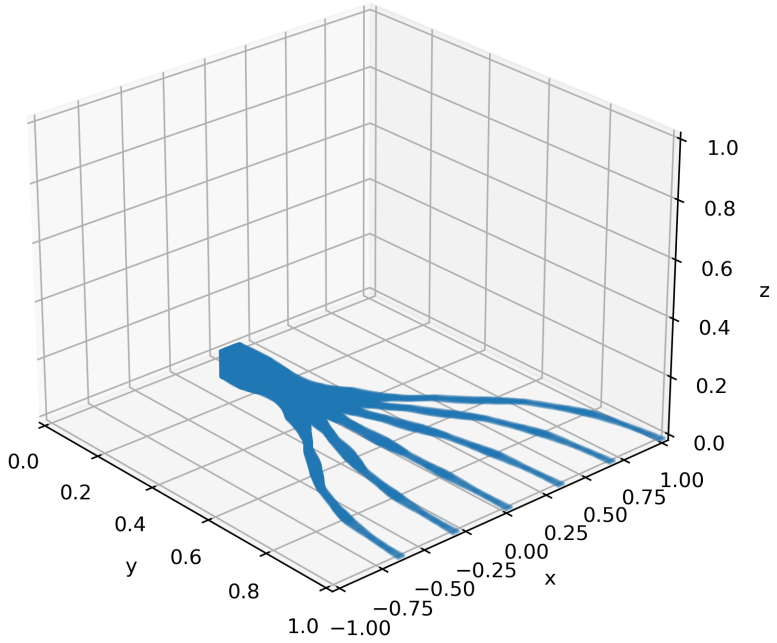


Figure 4.6: Funnel library plotted in $\mathbb{R}^3$.

4.3.4 Composability of Funnels

An issue that we have to address at this point is the sequential composability of funnels which is required to ensure that funnels can be sequentially added one after another to form longer motion plans. A funnel F_1 can be followed by another funnel F_2 if the entire inlet set of states of F_2 are contained within the outlet set of states of F_1. Majumdar et al. [3] define sequential composability as:

Definition 1 (Sequential Composability) *An ordered pair (F_1, F_2) of funnels $F_1 : [0, T_1] \to \mathbb{R}^n$ and $F_2 : [0, T_2] \to \mathbb{R}^n$ is sequentially composable if $F_1(T_1) \subset F_2(0)$.*

One way to address this issue during online motion planning is to construct a directed graph $G(F)$ that encodes the sequential composability information of the funnels in the library [3]. In the case of our quadrotor funnel library, we have seen that all 7 funnels are sequentially composable.

Chapter 5

Training with Funnels

This chapter describes the deep reinforcement learning and the PAC-Bayes optimization frameworks used to train a vision-based planning policy for the Quadrotor UAV that utilizes the computed funnel library.

5.1 PAC-Bayes Generalization Bounds

This thesis builds on the previous work that extends the PAC-Bayes Control framework [1] to vision-based planning with motion primitives [49]. Our work takes a slightly different approach than [49] by utilizing the precomputed funnels during training instead of the underlying motion primitives.

Assume that there is an underlying unknown distribution $\mathcal{D}$ from which a sample of N i.i.d. environments $S \coloneqq \{E_1, E_2, ..., E_N\}$ are drawn from to be used for training. Further, assume that there is a cost $C(\pi; E)$ associated with deploying a policy π on a particular environment E. In the context of robot navigation in obstacle-dense environments, we can assign a cost of 1 for colliding with an obstacle and a cost of 0 for successfully navigating to the goal region. The aim of the PAC-Bayes learning framework is to learn a distribution P over the space of policies that minimizes the expected cost across novel environments that are also drawn from $\mathcal{D}$:

$$C^* \coloneqq \min_{P \in \mathcal{P}} \mathop{\mathbb{E}}_{E \sim \mathcal{D}} \mathop{\mathbb{E}}_{\pi \sim P} [C(\pi; E)] \tag{5.1.1}$$

For a particular choice of posterior P, we define the *empirical cost* as the expected cost across the training environments in S:

$$C_S(P) \coloneqq \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{w \sim P} [C(\pi_w; E)] \tag{5.1.2}$$

By minimizing $C_S(P)$, we aim to achieve a PAC-Bayes generalization bound $C_{PAC}(P)$ on the true cost $C_{\mathcal{D}}(P)$ for the unknown distribution $\mathcal{D}$ of the environments. Let P_0 represent a *prior* distribution over the control policies, and let $R(P, P_0)$ be a regularization term, PAC-Bayes generalization theorem states that:

$$C_{\mathcal{D}}(P) \leq C_{PAC}(P, P_0) \coloneqq C_S(P) + \sqrt{R(P, P_0)} \tag{5.1.3}$$

where $R(P, P_0)$ is defined as:

$$R(P, P_0) \coloneqq \frac{KL(P||P_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N} \tag{5.1.4}$$

where $\delta \in (0, 1)$ and KL is the Kullback-Leibler divergence that measures the difference between two probability distributions. The upper bound $C_{PAC}(P, P_0)$ on $C_{\mathcal{D}}(P)$ holds with a probability $1 - \delta$, and allows us to quantify the generalization guarantee of a policy distribution P for an unknown environment distribution $\mathcal{D}$. [49] explains the PAC-Bayes generalization bounds for vision-based planning with further details and proofs.

5.2 Training

Our training methodology leverages the theory and implementation developed in [49]. Our goal is to train vision-based planning policies for the Quadrotor UAV that hold PAC-Bayes generalization guarantees on novel environments. As mentioned earlier, the first step of the PAC-Bayes framework is to train a prior policy distribution P_0. We use a deep reinforcement learning-based approach to train the prior.

5.2.1 Training a Prior

Our control policy π takes a 50x50 depth map image, scores each of the funnels in the library and applies the underlying motion primitive of the funnel with the highest score. The policy processes the depth map in the deep neural network architecture given in Figure 5.1. In addition to this policy network, the depth map is processed through a depth filter that is fixed and does not have any learning parameters. This

filter divides the depth map into 7 regions corresponding to the 7 funnels in the library and scores the funnels based on the computed mean of depth values in each region as shown in Figure 5.2.

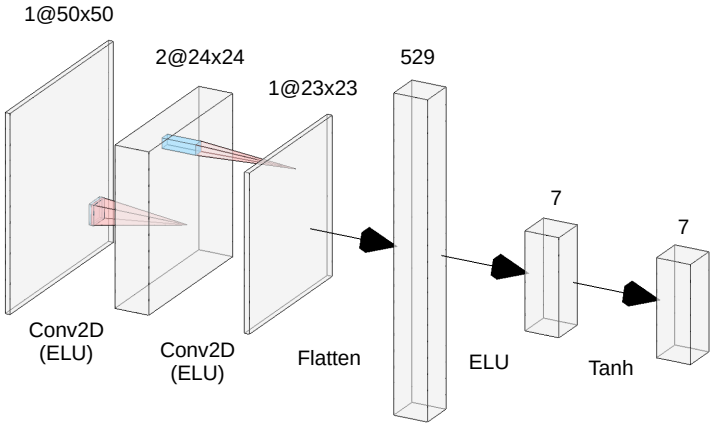


Figure 5.1: Neural network architecture of the policy network. Visualization tool reference: [27]

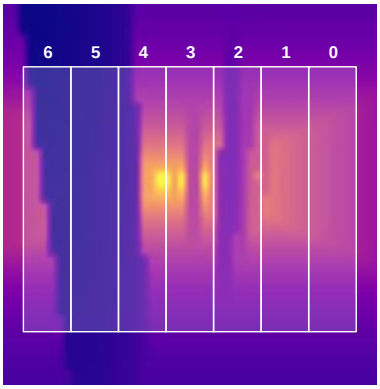


Figure 5.2: Visualization of the depth filter processing the depth map to score the 7 funnels. Note that the 7 funnels have been numbered from 0 to 6 and a color map has been applied to the depth map for visualization purposes.

The addition of this depth filter increases the policy's performance and enables more efficient training for the policy network. Funnel scores from both the policy network and the depth filter are summed to achieve a final score vector as shown in Figure 5.3. The policy applies the underlying motion primitive of the funnel with the highest score.

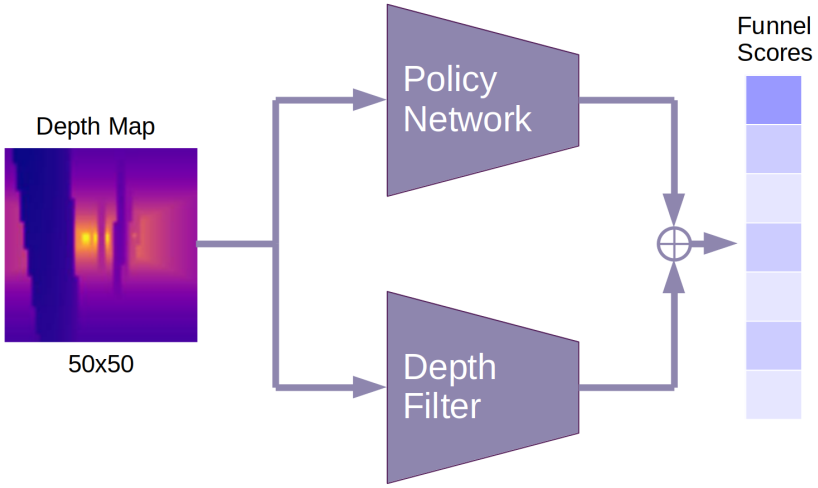
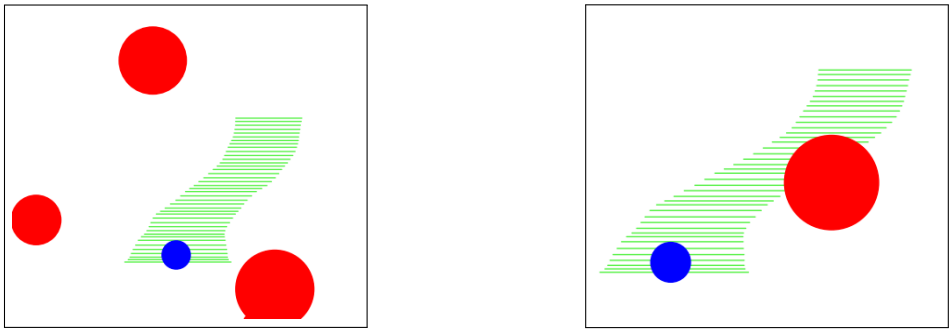


Figure 5.3: Visualization of the policy's funnel scoring procedure.

We adopt a reinforcement learning-based training approach called Evolutionary Strategies (ES). The cost associated with each rollout was determined by $1 - \frac{t}{T}$ where t is the time at which the robot collides with an obstacle and T is the total time horizon. In other words, a cost value of 1 corresponds to colliding at $t = 0$ and a cost value of 0 corresponds to successfully reaching the goal region. Since our work aims to train a *robust* vision-based planner that can operate under uncertainties, the collision criterion during the training was the *funnel colliding with an obstacle* rather than the robot colliding with an obstacle. Therefore, we maintain a geometric representation of obstacle locations and radii throughout the procedure to be able to check if an applied funnel collided with any obstacles. Figure 5.4 shows two-dimensional representations of a case where the funnel collides with an obstacle and a case where there is no collision.

This simulation-based deep reinforcement learning training has been implemented using PyBullet [12] and PyTorch [36]. 500 environments were drawn from the unknown distribution $\mathcal{D}$ for training the prior P_0. The training time was approximately 62 hours running on a server with 6 NVIDIA RTX 2080 GPUs and 96 CPU threads of which only 4 GPUs and 32 threads were utilized. The loss curve of this training is given below in Figure 5.5.



(a) Funnel does not collide with any obstacles. (b) Funnel collides with an obstacle.

Figure 5.4: Two-dimensional visualization of planning with funnels. Red circles represent the obstacles in the environment, blue circles represent the robot's location and the green lines represent the twelve-dimensional funnel's projection onto the $x - y$ plane.

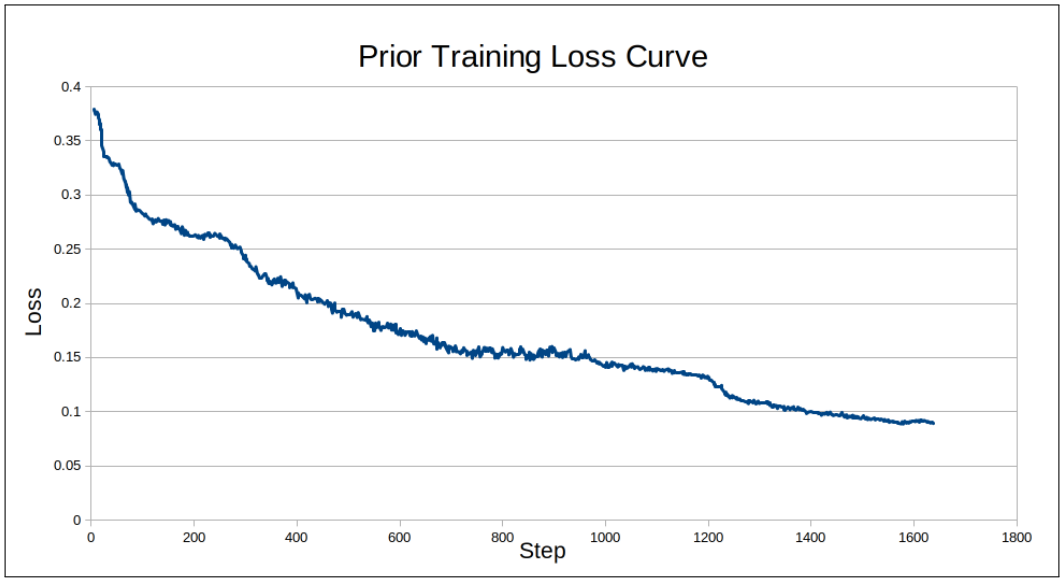


Figure 5.5: Loss curve of the deep reinforcement learning-based prior training. Notice that the loss starts at a value around 0.38, meaning that the depth filter alone can navigate the robot through about 62% of the obstacle course. It decreases to a final value of 0.09 (navigating through about 91% of the environment) as the policy network trains and improves the policy's performance.

5.2.2 PAC-Bayes Optimization

The next step in the training procedure is to perform PAC-Bayes optimization to achieve the generalization upper-bound explained in Equation 5.1.3. In order to do this, we draw 40 i.i.d. policies from the trained prior P_0 and compute the cost for each policy in 4000 new environments drawn i.i.d. from $\mathcal{D}$. This step yields a cost matrix

C where each column is a cost vector $C_i \in \mathbb{R}^{40}$ that holds the average cost of running policy π_i on environments $\{E_j\}_{j=1}^{4000}$. This computation took approximately 1 hour running with the same computational resources explained in the previous section.

After computing the cost matrix C, the next step is to run the PAC-Bayes Optimization algorithm (Algorithm 1) developed in [49] to achieve an optimized generalization upper bound for the policy distribution. We utilize the algorithm and the implementation presented in [49] which uses the MOSEK solver [4] to solve an efficiently solvable convex optimization program called *relative entropy program (REP)*. We set $\delta = 0.01$ to have PAC-Bayes bounds hold with a probability 0.99. This computation provides us a PAC-Bayes generalization bound of 20.4%, which can be verbally interpreted as: *on average the quadrotor will successfully navigate through 79.6% (100% - 20.4%) of the novel environments with a probability of 0.99.* This step was performed on the same computer server explained above and took a computation time in the order of a few seconds.

5.3 Evaluation

After successfully training a vision-based planner and achieving a PAC-Bayes bound on its generalization performance, the next step is to evaluate the planner's performance in novel environments. Since our planner claims to be a *robust* one, we also need to test our performance in novel environments where the robot is subject to external disturbances.

To evaluate the performance without external disturbances, we exhaustively simulate the final policy distribution on novel environments that have not been seen during the training. We draw 40 i.i.d. policies from the posterior policy distribution and test it on 4000 novel environments without disturbances. The estimated true cost of this evaluation was 15.9%, which means *on average the robot successfully navigates through 84.1% (100% - 15.9%) of the novel environments.* This value is well under the PAC-Bayes bound of 20.4%.

Finally, we add external disturbances to the robot's simulation and evaluate the posterior policy distribution's performance by testing 40 i.i.d. policies on 576 novel environments. Note that the disturbances have been added to the simulation through the dynamics of the robot as explained in Section 4.3.1. At every time step of the simulation, external disturbance forces and moments were randomly drawn from uniform distributions. Due to the computational inefficiencies experienced with the ODE solvers, we only test this case using 576 novel environments. The estimated true cost

of this evaluation was 19.0%, which means *on average the robot successfully navigates through 81.0% (100% - 19.0%) of the novel environments.* This value is still under the PAC-Bayes bound of 20.4%. These results are summarized below in Table 5.1:

Table 5.1: Quadrotor UAV robust vision-based planning evaluation results

External Disturbances	Number of Environments	PAC-Bayes Cost Bound	Estimated True Cost
No	4000	20.4%	15.9%
Yes	576	20.4%	19.0%

Chapter 6

Discussion and Conclusion

In this thesis, we have developed a robust vision-based motion planning policy that holds PAC-Bayes generalization guarantees for planning in novel environments with unknown disturbances. The method presented in this thesis exploits the model knowledge to perform the offline computation of over-approximated reachable sets (funnels) around the motion primitive trajectories. Then, the next step is to use deep reinforcement learning to synthesize a vision-based motion planning policy that utilizes the precomputed funnel library. Finally, PAC-Bayes optimization is performed on the trained policy distribution to achieve strong generalization bounds on the average cost of the policies in novel environments.

After implementing this framework on a computer simulation of an autonomous quadrotor UAV navigating through obstacle-dense environments, we have achieved a final PAC-Bayes bound of 20.4% on the true cost with a 99% probability. In other words, policies drawn from the posterior policy distribution are expected to successfully navigate through at least 79.6% of the environment on average. In order to validate this upper bound, we drew 40 i.i.d. policies from the posterior distribution and evaluated their performance on 4000 novel environments without introducing disturbances, and achieved a final true cost estimate of 15.9%, which is well under the PAC-Bayes generalization bound. This shows that the vision-based motion planner's PAC-Bayes bound holds for environments that have not been encountered during the training, and the next step is to evaluate its performance when unknown external disturbances are introduced to the novel environments.

The main challenge addressed in our work is the out-of-distribution generalization of a vision-based planner that has been trained in the absence of uncertainties. Since we train the motion planning policy using the funnel library (i.e., the collision criterion

is the funnel colliding with an obstacle), introducing external disturbances to the environment should not affect the planner's performance and the PAC-Bayes bound should hold. We evaluated the 40 i.i.d. policies on 576 novel environments and achieved a final true cost estimate of 19.0%. The final true cost estimate is below the PAC-Bayes bound, showing that we have successfully managed to design and implemented a robust vision-based motion planning policy that has certificates of performance that hold even in the presence of external disturbances.

6.1 Challenges and Extensions

One issue we have to address is the cost increase from 15.9% to 19.0% as we introduce external disturbances to the system. Since our method takes care of the disturbances by planning with funnels, one would expect to see no significant changes in the performance when external disturbances are present in the system. We believe that this may be due to several reasons:

6.1.1 Number of evaluation environments

As mentioned earlier, we could evaluate the system with disturbances on only 576 environments compared to the 4000 environments used for the zero-disturbance case. This was due to the inefficiencies in the custom ODE solver implementation. Therefore, it is very likely that 576 is not a large enough sample size to evaluate the final performance where both the PAC-Bayes bound and the zero-disturbance evaluation is computed over 4000 environments. The true cost estimate $C_{\mathcal{S}}(P)$ for a sampled set of N environments $S \coloneqq \{E_1, E_2, ..., E_N\}$ should approach the true cost of the policy's performance for the underlying unknown distribution $\mathcal{D}$ as $N \to \infty$. In other words:

$$\lim_{N\to\infty} C_{\mathcal{S}}(P) = C_{\mathcal{D}}(P) \tag{6.1.1}$$

In future work, we aim to evaluate this framework on more environments in the presence of external disturbances to have a more accurate comparison.

6.1.2 Discrete-time dynamics simulator

Similarly, another concern one might have with the current implementation is the imperfections in the discrete-time dynamics simulator. In order to speed up the

computation time on the final evaluation, we have increased the time step interval Δt of the discrete-time ODE solver implemented in the simulation. This could be a reason for the dynamical system to deviate further from the true trajectories in large disturbances and the computed funnels might not hold. In future work, we aim to achieve better implementation of the discrete-time dynamics simulator for the final evaluation of the system.

6.1.3 Camera rotations and translations

Since we have developed a *vision-based* planner that processes a depth map at every step and determines which funnel to apply, introducing external disturbances to the environment will cause rotations and translations in the robot's FPV camera after applying each funnel. This is because the final position and orientation of the robot will have slightly deviated from the final state of the underlying motion primitive in the presence of external disturbances. Since our motion planner is trained using depth map images gathered in the absence of external disturbances, the visual inputs at test time will be slightly different than the training data. These circumstances can decrease the vision-based planner's performance and cause PAC-Bayes bounds to not hold when the system is subject to disturbances. In future work, we aim to use the knowledge of the robot's position and orientation to perform the necessary transformations to the visual data before processing it in the neural network.

6.2 Future Work

In conclusion, our approach combines ideas from model-based reachability analysis and deep-learning-based planning to present a robust vision-based motion planner that can successfully operate under bounded unknown disturbances and generalize to novel environments with PAC-Bayes generalization guarantees. We have implemented this framework on a computer simulation of a quadrotor UAV navigating through an obstacle-dense environment. In future work, one can extend this framework to other robotic applications such as ground vehicle navigation or robotic arm manipulation.

Another exciting future direction would be to implement this approach on hardware where the dynamical system would be tested against real-world uncertainties. With accurate models of the dynamics and the external disturbances, we believe that our approach can provide successful results. Since our approach provides a quantified certificate of performance that would hold as long as the uncertainties are bounded

and the modeling is accurate, one can train a robotic system on simulation and easily determine if it is safe for deployment based on its PAC-Bayes bound. We aim to perform a hardware implementation either on a vision-based quadrotor UAV navigation task in the presence of wind gusts or a vision-based object manipulation task with a robotic arm (e.g., pushing an object to a goal region in the presence of obstacle and uncertainties).

Bibliography

[1] A. Farid A. Majumdar and A. Sonar. Pac-bayes control: Learning policies that provably generalize to novel environments. *arXiv preprint arXiv:1806.04225*, 2019.

[2] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. *47th IEEE Conference on Decision and Control*, pages 4042–4048, 2008.

[3] A.Majumdar and R.Tedrake. Funnel libraries for real-time robust feedback motion planning. *International Journal of Robotics Research*, 36:947–987, 2017.

[4] MOSEK ApS. *MOSEK Fusion API for Python 9.0.105*, 2019.

[5] S. Azuma, J. Imura, and T. Sugie. Lebesgue piecewise affine approximation of nonlinear systems. *Nonlinear Analysis: Hybrid Systems*, 4:92–102, 2010.

[6] D. Berenson, R. Diankov, and et al. Grasp planning in complex scenes. *International Conference on Humanoid Robots*, page 42–48, 2007.

[7] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon "next-best-view" planner for 3d exploration. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1462–1468, 2016.

[8] Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin, and Christian Schilling. Juliareach: a toolbox for set-based reachability. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 39–44, 2019.

[9] M. Bojarski, D. Del Testa, and et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[10] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, 17(2):44–54, 2010.

[11] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for nonlinear hybrid systems. *International Conference on Computer Aided Verification*, pages 258–263, 2013.

[12] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[13] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008, 2017*, 2017.

[14] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[15] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. *J. Field Robot*, 33:431–450, 2016.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[17] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. *Robotics Research*, pages 235–252, 2017.

[18] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning. *IEEE International Conference on Robotics and Automation (ICRA)*, page 7087–7094, 2018.

[19] B. Ichter and M. Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, 2019.

[20] Shakir Istiak. Quadcopter. `https://b2b.partcommunity.com/community/knowledge/en/detail/10011/Quadcopter`. [Online; accessed April 10, 2021].

[21] P. Jacobs and J. Canny. Robust motion planning for mobile robots. *Robotics and Automation*, page 2–7, 1990.

[22] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. *International Workshop on Hybrid Systems: Computation and Control*, pages 329–342, 2007.

[23] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data, 2017.

[24] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[25] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

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

[27] A. LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

[28] D. Li, S. Bak, and S. Bogomolov. Reachability analysis of nonlinear systems using hybridization and dynamics scaling. *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 265–282, 2020.

[29] R. Liao, R. Urtasun, and R. Zemel. A pac-bayesian approach to generalization bounds for graph neural networks. *arXiv preprint arXiv:2012.07690*, 2020.

[30] C. Liu and C.G. Atkeson. Standing balance control using a trajectory library. *IEEE/RSJ International Conference*, page 3031–3036, 2009.

[31] R. Mahony, V. Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, 19:20–32, 2012.

[32] A. Martinelli. Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *IEEE Transactions on Robotics*, 28:44–60, 2011.

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

[34] N. Michael, S. Shen, K. Mohta, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro. Collaborative mapping of an earthquake damaged building via ground and aerial robots. *Field and Service Robotics*, pages 33–47, 2014.

[35] J. Le Ny and G.J. Pappas. Sequential composition of robust controller specifications. *International Conference on Robotics and Automation (ICRA)*, 2012.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, and et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[37] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette. Vision-based minimum-time trajectory generation for a quadrotor uav. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6199–6206, 2017.

[38] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip. Motion planning networks. In *Proceedings of 2019 IEEE International Conference on Robotics and Automation*, page 2118–2124, 2019.

[39] R. Rahmatizadeh, P. Abolghasemi, A. Behal, and L. Bölöni. From virtual demonstration to real-world manipulation using lstm and mdn. *AAAI*, 2018.

[40] O. Rivasplata, V. M. Tankasali, and C. Szepesvari. Pac-bayes with backprop. *arXiv preprint arXiv:1908.07380*, 2019.

[41] T. Schouwenaars, B. Mettler, and et al. Robust motion planning using a maneuver automation with built-in uncertainties. *American Control Conference*, 3:2211–2216, 2003.

[42] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[44] P. Sermanet, M. Scoffier, and et al. Learning maneuver dictionaries for ground robot planning. *39th International Symposium on Robotics*, 2008.

[45] S. Thrun, D. Fox, and W. Burgard. *Probabilistic Robotics*, chapter Mobile Robot Localization. In [48], 2005.

[46] S. Thrun, D. Fox, and W. Burgard. *Probabilistic Robotics*, chapter Occupancy Grid Mapping. In [48], 2005.

[47] S. Thrun, D. Fox, and W. Burgard. *Probabilistic Robotics*, chapter Simultaneous Localization and Mapping. In [48], 2005.

[48] S. Thrun, D. Fox, and W. Burgard. *Probabilistic Robotics*. The MIT Press, 2005.

[49] S. Veer and A. Majumdar. Probably approximately correct vision-based planning using motion primitives. *arXiv preprint ArXiv:2002.12852*, 2020.

[50] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger. Probabilistic model predictive safety certification for learning-based control. *IEEE Transactions on Automatic Control*, 2021.

[51] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

[52] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh. Predicting failures of vision systems. In *Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2014.

[53] D. Zheng, H. Wang, Z. Xie, W. Chen, and X. Kong. Autonomous navigation of a quadrotor in unknown environments. *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1930–1935, 2017.

Appendix A

Code & Video

Our code is currently available at a private GitHub repository under the IRoM Lab. A video of the simulation with disturbances is at: https://youtu.be/g0nXfJvtSWQ.