GAUSSIAN CONTROL BARRIER FUNCTIONS : A GAUSSIAN PROCESS BASED APPROACH TO SAFETY FOR ROBOTS

A Dissertation
Presented to
The Academic Faculty

By

Mouhyemen Ahmed Khan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering
College of Engineering

Georgia Institute of Technology

December 2022

GAUSSIAN CONTROL BARRIER FUNCTIONS : A GAUSSIAN PROCESS BASED APPROACH TO SAFETY FOR ROBOTS

Thesis committee:

Dr. Abhijit Chatterjee, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. David Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
Department of Electrical Engineering and Computer Science
University of California, Irvine

Dr. Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Tatsuya Ibuki
Graduate School of Science and Technology
Meiji University

Date approved: December 10, 2022

إِقْرَأْ بِاسْمِ رَبِك الَذي خَلَقَ

"Read in the Name of your Lord who created."

خَلَقَ الْإِنْسَانَ مِنْ عَلَقٍ

"Created human from a clot of blood."

إِقْرَأْ وَرَبُك الْأَكْرَمُ

"Read! For your Lord is the Most Generous."

- Qur'an, Chapter 96, Verses 1-3

For Mum, Dad, Farhaan, and Faheem.

ACKNOWLEDGMENTS

Research funding support: This research was supported by the US National Science Foundation under Grants S&AS:1723997 and CCF 2128419.

In the writing of my thesis, I left this section for the very end. And not without good reason. I sincerely think that at a personal level, this section carries the most weight and impact for me. And I hope that I do not do any injustice here in acknowledging the people that played such an important role in my dissertation.

In 2008, I watched Iron Man in the movie theater and I remember vividly that I had then decided that I will study robotics. The movie not only played a critical role in jump-starting the Marvel Cinematic Universe, but certainly my academic life also. Ambitious yet naive, I decided I will make Iron Man one day. Little did I know back then that it would be a long 14 year journey since then, before I can confidently admit that I still do not know *completely* how to make Iron Man, and probably never will. But the ambitious drive of it has taken me through a very interesting journey over the last decade and a half.

I will be acknowledging below several key individuals chronologically and try my best to share how they impacted me. The order here by no means reflects the impact or contribution but simply the ease of narration.

Dr. Beena Ahmed: I began my undergraduate degree in Electrical and Computer Engineering at Texas A&M University, Qatar in the year 2010. That is when I first met Dr. Beena Ahmed who was my professor for a freshman-level engineering course. The final project in the course involved a pick and place robotics challenge and I could not have been more excited about it. I was thrilled to begin my robotics education through coursework and projects. That very semester was the first time I ever built a robot (using LEGO Mindstorms Kit) and programmed the robot for a pick and place challenge. Although, it was beyond anything I had ever done, I still was not satisfied. So I approached Dr. Beena and shared with her my interest in wanting to learn more about robotics. She encouraged me to

participate in robotics competitions at a collegiate or university level. I then represented my university at two international competitions, one for a sumo robot competition in Ankara, Turkey in 2013, and the second for an aerial obstacle course competition in Vienna, Austria in 2014. Finally, for my undergraduate thesis, my team and I worked on a tele-operated robotic hand controlled wirelessly using a smart globe with Dr. Beena as our undergraduate thesis advisor. We won first place for the thesis project in 2014. I knew I still had more to learn.

My undergraduate years were filled with countless days and nights of building, programming, and reading on various kinds of robots, how to make a robot move, what is the role of an actuator, what kinds of sensors are needed, so on and so forth. However, I would often hit a dead end very quickly as soon as I would land upon a research paper. I simply did not know how to read academic papers describing the control design or kinematics/-dynamics of a robot. The mathematical symbols were difficult to decipher and translate to practice.

Dr. Swaroop Darbha: This is where I met one of the most intellectually gifted and respected individuals in my life. Till this very day, I feel humbled and intimidated by Dr. Swaroop. I met him in the spring of 2014. He was a visiting professor to the Qatar campus from the main campus of Texas A&M, College Station, Texas. My good friend at the time told me about a robotics professor visiting the Qatar campus for the semester. I quickly ran to Dr. Swaroop's office and introduced myself and asked if he could offer a robotics course. Texas A&M University, Qatar opened a course where only 3 students had enrolled in it; myself and my 2 other good friends at the time. Dr. Swaroop was my first formal robotics professor and introduced me to Lagrangian and Eulerian representation, rotation matrices, dynamics and kinematics, Lyapunov analysis, LaSalle's invariance principle, and many other topics in modern control theory. I must admit that it was very difficult for me to grasp and internalize those concepts at the time. But Dr. Swaroop was also very patient with me. I remember thinking several times if these topics truly help in robotics. I wanted

to build robots and program them to move from one place to another. But little did my naive understanding of the topics convince me that these topics will be fundamental in my graduate studies down the road. After studying under Dr. Swaroop I was finally able to read some of the academic papers for the first time.

Dr. Khaled Harras: I graduated from my bachelor's in May, 2014 and was looking for robotics opportunities at the time in Qatar. A friend of mine, Sidra Alam, told me that a professor at Carnegie Mellon University, Qatar is looking for an outreach tutor in computer science where robotics workshops will be taught to school students. I was running my own robotics institute at the time in Education City with a similar goal of sharing robotics education in Qatar and helping people interested in the topic. So I make an appointment and head over to meet Dr. Khaled Harras. He interviewed me and asked why I was interested in the outreach program. As the conversation went further, he noticed that my true interest lied in furthering my robotics knowledge and getting deeper into the fundamentals. He looked at my resume and asked if I had formally worked on any robotics research. Khaled then told me that I needed a mentor who can help me in pursuing research. This was the starting point of research for me with Khaled for the next 2 years. I learned immensely under him and he was strict but fair with me. I can write on my research work experience with Khaled for many pages, but I will keep it short by saying that he taught me how to write *good* research papers, how to investigate a research problem formally, how to strengthen an argument using quantitative metrics, how to take ownership of a research project. I remember very well when I finally got admitted to Georgia-Tech's PhD program, he told me, "When you are at your lowest, and everything becomes very difficult, know that you are on the right track.", and emphasized several times not to give up.

Dr. Abhijit Chatterjee: I finally come to Georgia-Tech, Atlanta, and begin my PhD journey. This is where I meet my dear advisor, Dr. Abhijit Chatterjee (Chatt), who took me under his wing at a critical time in my PhD studies. I was at the verge of quitting my PhD because of the harsh difficulties I was facing at the time, from lack of funding

guarantees to lack of proper conduct by certain faculty members, all of which were having an immense stress on my mental well being. I will never forget the attitude that Chatt showed me. He was warm, welcoming, and above all, very respectful. I immediately felt a calming presence entering his office and talking to him. This happened in November, 2018. Since January, 2019 until today, and hopefully for many more years to come, my time with Chatt has been nothing short of ease, comfort, and intellectually engaging. Chatt played a very instrumental role in shaping my thesis and my mentality as well in overcoming the difficulties during my PhD. I have told all my friends and colleagues that my advisor is like an uncle to me. He is very fatherly in his conduct and always went out of his way to take care of me. I have immense respect for Chatt and I cannot thank him enough for everything he has done.

Mentors: In my second semester as a PhD student, I met the most important person in my graduate studies, Munzir Zafar, another senior PhD student at the time nearing his graduation. Little did I know how instrumental he will be in laying the groundwork for my thesis. He was the teaching assistant for a course I was taking at the time. As part of the final project in the course, he offered several topics that interested students could work in. I needed guidance both academically and research wise since I was very new to the PhD program. That started the most important learning phase in my PhD. Munzir guided me through the project and asked me to continue working with him over the summer in 2018. He said to me, "I cannot offer you anything for working with me. But I can teach you things". Munzir taught me the theory of Gaussian processes and control barrier functions (both the topics that ultimately became the crux of my dissertation). He was not only my research and academic mentor, but also spiritual friend and an older brother. I owe a lot to Munzir for the path he put me on.

The same semester I met yet another individual, a fellow PhD student, Vishal Murali. We both were taking a course together and Vishal sat next to me one fine day. I offered him almonds that I was having at the time. He was shy and nervous; classic introvert. We began

talking after and I was bewildered by his depth in mathematical concepts. He reminded me of Dr. Swaroop at every instance. I always ran to him to discuss, learn, and question many mathematical concepts. He showed me the wonders of math and how it applies to robotics. Vishal was always patient and kind with me. Without Vishal, I would not have been able to complete this dissertation.

Friends: During my graduate studies, I met many wonderful people and was lucky to befriend them. The days in Bhaishizhou during Georgia-tech, Shenzhen was fantastic thanks to Aditya, Ambuz, and Sidd. In particular, Aditya, was a very dear and close friend of mine since and taught me many wonderful things in friendship. Among the most wonderful human beings that I have had the good fortune of knowing are Nicolas Shu and Akash Patel. Nicolas has been such a force of good and strength for me over the years that I cannot pen down properly. I truly think that a strong source of light during my PhD journey is Nicolas. The same holds true for Akash. Akash helped me and lifted me up during many difficult times. He worked on my experiments for countless hours and provided support that I can never payback. A very important person I met in 2020 is Nevena, my love. Without her continued support and strength, I would not have been able to come this far alone. This journey can be very difficult, and Nevena held my hands (literally and figuratively) through it all. Additionally, I would also like to note many others by name who made the journey really fun and exciting. Thank you Bogdan, Chris, Eric, Sandy, and Soliman!

Family: Finally, I come to my family. My most important supports and pillars were and are my parents and my brothers. No words can do justice to what they have sacrificed and done for me. This entire thesis is dedicated to them.

I want to thank Almighty Allah for His strength and light, and showering me with good fortune from His treasures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In recent years, the need for safety of autonomous and intelligent robots has increased. Today, as robots are being increasingly deployed in closer proximity to humans, there is an exigency for safety since human lives may be at risk, e.g., self-driving vehicles or surgical robots. The objective of this thesis is to present a safety framework for dynamical systems that leverages tools from control theory and machine learning. More formally, the thesis presents a data-driven framework within a Bayesian optimization setting for designing safety function candidates which ensure properties of forward invariance. The potential benefits of the results presented in this thesis are expected to help applications such as safe exploration, collision avoidance problems, manipulation tasks, and planning.

We utilize Gaussian processes (GP) to place a prior on the desired safety function candidate, which is to be utilized as a control barrier function (CBF). The resultant formulation is called *Gaussian CBFs* and they reside in a reproducing kernel Hilbert space. A key concept behind Gaussian CBFs is the incorporation of both safety belief as well as safety uncertainty, which former barrier function formulations did not consider. This is achieved by exploiting robust posterior estimates from a GP where the posterior mean and variance serve as surrogates for the safety belief and uncertainty respectively. We synthesize safe controllers by framing a convex optimization problem where the kernel-based representation of GPs allows computing the derivatives in closed-form analytically.

Finally, in addition to the theoretical and algorithmic frameworks in this thesis, we rigorously test our methods in hardware on a quadrotor platform. The platform used is a Crazyflie 2.1 which is a versatile palm-sized quadrotor. We provide our insights and present detailed discussions on the hardware implementations which will be useful for large-scale deployment of the techniques presented in this dissertation.

CHAPTER 1
INTRODUCTION

With the rise of autonomous systems, assuring their safety is of paramount importance. For instance, an autonomous drone should not crash during its mission, or a self-driving vehicle should not collide with other vehicles. Safety-critical robotic systems are increasing rapidly in today's data-driven technology due to more domains seeking intelligent robots such as medicine, agriculture, warehouse, disaster response, and defense [1, 2, 3, 4]. How does one guarantee safety for such diverse set of operations and robotic systems? The primary concern related to safety-critical systems, both intuitively and formally, is to do with the consequences of failure. Then the next logical thing to ponder over is how can we specify constraints so that failures can be avoided. And it is not clear always how to tackle this. Formulating constraints for these applications to ensure safety is a challenging task and requires no small consideration on the designer's part. Accordingly, many safety-critical robots need to be dependable, since failure to do so could endanger human life and lead to substantial economic loss [5, 6]. Modern robotic systems are increasingly working in uncertain environments, hence, robots need to sense and determine safe navigable zones in real-time. The single greatest obstacle to widespread utilization of intelligent systems in the world today is to do with the safety guarantees of these systems.

This dissertation focuses on addressing how sensed data can be effectively exploited to ensure that the robot operates in a safe manner. Sensed data heres refers to data collected from a myriad of sensors such as thermal, ground-positioning system (GPS), camera, telemeters, or LiDARs. The goal is to model and formulate safety objectives for robotic systems using data-driven techniques. We leverage key ideas from control theory and Bayesian statistical inference to design safety objectives while computationally being efficient in synthesizing safe controllers. To this end, we use Gaussian processes (GPs)

to place priors with certain smoothness properties to construct control barrier functions (CBFs), a Lyapunov based method, to enforce safety on the dynamical system of interest. By coupling convex optimization with the synthesized function, the approach becomes amenable to real-time implementation for practical systems. The tools developed in this thesis are deployed on a hardware quadrotor platform over numerous experimental studies.

1.1 Background Review

A popular approach to ensuring safety of dynamical systems leverages set theoretic ideas. To be more specific, the theory of controlled set invariance is employed where a system is defined to be safe if (a subset of) its states remain within a prescribed set [7]. This forms the basis of control barrier functions (CBFs) which have been successfully demonstrated on many safety-critical applications [8, 9, 10, 11]. The two most common certificate based methods for addressing system safety and stability are barrier certificates and Lyapunov certificates respectively. In this thesis, since we are interested in the safety of dynamical systems, we focus on CBFs. A comprehensive review for both methods can be found in [12, 13].

1.1.1 Control Barrier Functions

The notion of ensuring forward invariance for a dynamical system goes as back as 1940's where Nagumo's seminal work provided necessary and sufficient conditions for the system to remain inside the desired (safe) set [14]. Decades later, in the early 2000's, the topic forward invariant safe sets were formulated as barrier certificates in the context of nonlinear and hybrid systems [15, 16]. The authors presented barrier functions which ensured that the dynamical system's states never entered an unsafe set in the state space. Eventually, these functions were elegantly composed as a convex optimization problem in the form of quadratic programs (QPs) by [8]. This modern form of the barrier function is popularly called control barrier functions (CBFs) today and forms the basis of the safety function in

this thesis. CBFs are continuously differentiable functions where the function characterizes a set, in which the dynamical system should remain forward invariant. To look at it from another angle, the system's safety is encoded using these safety barrier certificates (or safe sets) with the aid of a continuously differentiable function.

The authors in [17] first demonstrated the convex optimization based synthesis for generating control actions on a driver-assist problem of adaptive cruise control. By virtue of a quadratic cost and linear constraint, the QP optimization problem resulted in real-time computational solution. Subsequently, the authors in [18] implemented CBFs on the adaptive cruise control problem using the QP framework on scaled-down model cars. Subsequent research with CBFs is spread over a wide range of applications, most notably bio-inspired robotic systems, swarm coordination, manipulation tasks, and automotive applications.

To extend the reach of CBFs to systems with higher relative degree, exponential CBFs were proposed in [19]. Exponential CBFs were then generalized into Higher-order CBFs by [20]. Since many dynamical systems often require higher degrees of control authority, exponential CBFs were critical in demonstrating safety performance for legged robotics [11, 21, 22, 23]. CBFs were also successfully demonstrated on dynamic gait stepping for both 2D and 3D settings in [24, 25]. For manipulation tasks, CBFs were also used for ensuring that the end-effector does not collide with the environment as well as for ensuring inter-link collisions [26, 27, 28]. Moving from single-agent systems to multi-agent systems, CBFs have shown its effectiveness in the context of collision avoidance between robots. The safety objective is to maintain a minimum safe distance between each agent by computing the distance between each agent. Then a QP is set up with multiple linear constraints with each constraint typically satisfying the safety inequality criteria between any two given robots. Such a framework has been shown in [9, 29, 10, 30, 31, 32]. For for the interested reader, we refer to [33] for the theoretical background of CBFs and its applications to mainly robotic settings.

In all of the aforementioned works, CBFs are used in a constructive or parametric man-

ner which means that they are typically hand-designed functions. Depending on the application and problem statement, CBFs need to be carefully designed. This can have limitations since it is not always practical to hand craft safety objective functions especially for unstructured and environmentally unpredictable settings. We next look at the literature for data-driven CBF design methods.

1.1.2 Learning based Barrier Functions

Data-driven techniques to generate CBFs are very actively pursued today given the huge success of CBFs for ensuring safety of modern robotic systems. Expert demonstrations involving state and control trajectories were used in [34, 35] to generate CBFs. However, having access to expert demonstrations, especially when deploying a system online in an unseen environment, can become a practical limitation. While the previous works empirically verify their findings, they do not provide hardware experimental validation of their methods. Safety certificates in the form of neural certificates, which use neural networks, were constructed for achieving safe control [36, 37, 38, 39]. These neural certificates provide formal proofs of correctness to their learning-based controllers from data. However, all these studies have been confined to simulation experiments and are limited to offline training which limits their applicability in many practical online settings.

Episodic learning was used to update the controller for robotic systems while carrying out safe constrained control using CBFs [40, 41]. Both the papers show hardware results, however, the goal was to learn the system model uncertainty in an episodic fashion, as opposed to constructing a CBF from data. A similar framework uses Gaussian processes (GPs) for expanding an initial conservative safe set to a desired safe set through adaptive sampling and learning the unmodeled dynamics of a quadrotor system in simulation [42]. More precisely, the data was used to learn the unmodeled dynamics and based on the confidence region of the learned model, the parametric constructed CBF was altered to give rise a more expansive safe set. A second-order cone program was formulated in [43], with GPs

used for modeling the control input and dynamic model uncertainty learned in an episodic manner. A particular type of kernel was developed to satisfy affineness properties in order to appear as a linear constraint in the second-order cone program. However, the study was limited to offline training which limits its applicability in many practical online settings.

Using a sums-of-squares (SoS) approach, permissive barrier certificates were proposed in [44] to simultaneously guarantee the stability and safety of the dynamical system. An iterative search algorithm is used to search for the maximum volume barrier region which is proved to be strictly larger than safe regions using Lyapunov sublevel set based methods. The research investigation was verified and shown as a simulation study. For systems with polynomial dynamics, an optimization routine can be set up as a convex semi-definite problem using sum-of-squares (SoS) technique to search for a valid safety certificate [45, 46]. However, SoS methods also scale poorly with high dimensions, and are limited to polynomial system dynamics.

A supervised learning approach was shown in [47] using support vector machines (SVMs) to characterize CBFs using sensor measurements both in an offline and online manner. A dataset is first carefully constructed based on distance ranged measurements from a sensor which is shown to be provably safe. Next, a hard margin classifier is then used to synthesize the safety boundary. SVM classifiers are a very popular non-parametric kernel based approach in the machine learning domain, however, weights for the SVM classifier need to be carefully determined to work. Additionally, the study was confined to 2D simulation results. Another similar work to [47] was based on the idea of using signed distance fields (SDFs) with replay memory to construct CBFs fully from data [48]. SDFs are implicit surface representations, indicating whether a given datapoint $\mathbf{x}$ in the metric space for a set Ω ,characterizing the SDF, belongs to the set Ω or not. SDFs have been shown to be advantageous as a natural representation for both navigation and planning [49]. The authors in [48] used a multi-layer perceptron to train and characterize the CBF along with a replay memory buffer for being more amenable to online learning. However, this study

was also limited to 2D data sensing as well as conducted only in simulation.

Barrier functions were also combined with reinforcement learning (RL) strategies to guide the exploration policy to be safe. Safe RL is a growing field of research investigation primarily because exploration is a critical component for learning a meaningful policy. Consequently, the policy may require the system to visit unsafe states, which is particularly expensive since the cost of failure is high for safety-critical systems. As a result, in the literature, there have been efforts to combine CBFs with existing RL strategies [50, 51, 52]. An end-to-end safe RL policy was obtained by utilizing CBFs to serve as safe guides in [50] where knowledge of the CBF along a prior on the dynamics model is assumed. Differentiable robust CBFs compatible with standard RL policy gradients is presented in [52]. A practical limitation of RL based policies is their training time which in the majority of safety-critical applications can be very limiting.

1.2 Thesis Outline

We provide an outline of the chapters in this thesis.

- In Chapter 2, given a parametric CBF, we augment uncertainty associated with sensed objects around the robot to the given CBF. Since data from the sensors are used to map or model the environment around the robot, quantification of uncertainty associated with the sensed objects aids in the construction of conservative and arbitrary safe zones for the robot to navigate in. To this end, we use the Gaussian process posterior variance to quantify the *uncertainty in the synthesis of the safety uncertainty component resulting from the sensed data*. This uncertainty is then augmented to the given CBF resulting in a variance-based CBF, also called the safety function candidate. By adopting a kernel based approach, we are able to compute the Lie derivatives in closed-form which act as constraints for a convex optimization problem to generate safe control inputs. We showcase our approach in hardware on a quadrotor system by expanding an initial conservative safe set to a final desired safe set online [53].

- Chapter 3 constitutes the main contribution of this thesis. In the previous chapter, we introduced safety uncertainty arising from sensed data to an existing parametric CBF. We now place a Gaussian process prior on the desired safety function candidate and draw a CBF from a Gaussian process. In other words, the *CBF is a Gaussian process*. We term this non-parametric formulation as Gaussian control barrier functions (Gaussian CBFs). We work with noisy safety samples (utilized as outputs or observations to a GP) to derive our Gaussian CBF by exploiting the posterior mean and variance from a Gaussian process. These safety samples represent the sensed data. Additionally, we also investigate the case when the query input to the GP (test point) is noisy. Since we use sensed data now to synthesize the CBF, the CBF can take any arbitrary shape and is not limited to convex shapes only. Moreover, the presented approach is robust when it comes to noisy inputs and outputs to the GP. We demonstrate our approach over three test cases in hardware. In the first test case, we generate arbitrary safe sets randomly and perform constrained control for a quadrotor. In the second test case, we synthesize online the safe set and perform a collision avoidance study. In the final test case, we juxtapose Gaussian CBF against standard regular CBF in the presence of noisy position state for a quadrotor platform [54].

- In Chapter 4, we extend our previous methodology of Gaussian CBFs to not only construct safe sets but also safe implicit surfaces. Inspired by the research work in implicit surface representations, we provide a unified approach to constructing safe implicit surfaces using Gaussian CBFs. The safety aspect here comes from Gaussian CBFs enforcing safe control actions based on the Lie derivatives which are used to satisfy the constraints in the convex optimization framework. Since Gaussian CBFs are GPs, a key bottleneck with GPs is its cubic complexity. Hence, we derive sparse Gaussian CBFs in this chapter we reduce the computational complexity. We first discuss a simulation study where a 7-DOF robotic manipulator is tasked with avoiding the Stanford Bunny, a 3D volumetric object, whose boundary is characterized as a

Gaussian CBF with and without sparsity. By using point cloud and surface normals, the safety surface is constructed using Gaussian CBFs. In the second test case, we perform safe teleoperation and safe autonomous navigation using a quadrotor in the presence of a chair.

- In Chapter 5, we present a semi-parametric framework to learning-based control using Gaussian processes to learn the residual dynamics of a dynamical system. We utilize multiple sparse models to divide the observed training inputs and outputs into sparse regional models. By optimizing each model, we preserve the richness and uniqueness of each regional sparse model. We term this formulation as *Multi-sparse Gaussian process* and use it a learning-based control problem. We assume a parametric dynamical model for the system based on physics first principles, and learn the residual unmodeled dynamics using multi-sparse GPs. The resultant formulation is verified against regular GP, sparse GP, and local GP. We verify our approach both in simulation and hardware [55].

CHAPTER 2

SAFETY UNCERTAINTY IN CONTROL BARRIER FUNCTIONS

Today we are witnessing a high demand and increase in the deployment of many intelligent systems. Some notable examples of such systems range from self-driving vehicular units, whether it be cars or trucks, to smart consumer products such as indoor vacuuming or mopping robots. Many more of these systems are increasingly deployed in search and rescue, health care, transportation, and surveillance applications to name some [2, 3]. In all these applications, ensuring safety of the operating system is imperative. A natural question to ask is, *"how does one ensure safety for such a system?"*. It is easy to see that this question is fairly broad and can be tackled from many angles based on the definition of safety, criteria for safety satisfiability, and more. We look towards control theoretic methods and adopt level-set based approaches to tackle this issue.

Control theoretic methods play a critical role and provide a rigorous mathematical framework for addressing safety of a dynamical system. To this end, control barrier functions (CBFs) are a class of functions, inspired from Lyapunov level-set methods, to address the safety requirement of a dynamical system [8]. CBFs were successfully demonstrated on safety-critical applications such as adaptive cruise control, quadrotors, and bipedal robots [8, 56, 57, 11]. A dynamical system is defined to be safe if a subset of its states remains within a prescribed set, also called the *safe set or barrier certificates*, for all future time. Barrier certificates [15, 16] are characterized with the aid of a candidate scalar valued smooth function which ultimately forms the CBF. To incorporate safety using CBFs, we require two items: $1)$ a nominal model of the system dynamics and $2)$ a safety function candidate characterizing the safe set. An inner product is taken between the nominal dynamics and the safety function candidate to serve as the constraint in a convex optimization program resulting in *the safety of the system*. Traditionally, the primary focus has been

on learning residual dynamics and accounting for uncertainties in the dynamical model. Therefore, the effect on *system safety* comes from learning and modeling the residual dynamics. However, there has been limited effort into the synthesis of safety function candidates from sensed data while accounting for uncertainty in the safety function synthesis. Traditionally, CBFs have always been designed as deterministic functions providing a numerical scalar value. *How sure are we of this numerical scalar quantity?*

In this chapter, we introduce a notion of *safety uncertainty* to an existing CBF. Safety uncertainty refers to the uncertainty in the estimation or design of a given CBF when estimated from sensed data. For instance, consider a CBF that is designed for an obstacle avoidance scenario. Here, state estimates and range measurements from the sensors are used to update the value of the CBF. Both the state estimation and measurement process will invariably have noise or uncertainty. Naturally, one would like to encode this uncertainty in the computation of the CBF especially for safety-critical applications where being conservative based on the uncertainty could be useful. By augmenting the Gaussian process (GP) posterior variance to a given CBF, we introduce safety uncertainty to a given CBF. We demonstrate our approach on a problem of safe set expansion in a real robotic experiment using a quadrotor. The experiment video can be seen at: `https://youtu.be/9qvOf1UpRPw`.

2.1 Background Preliminaries

Here, we introduce and present the mathematical preliminaries upon which our methodology is based. We first review the CBF framework for achieving safety constrained control. In this thesis, we limit our scope to affine dynamical systems only. CBFs used in the context of non-affine systems, see [58]. We then go over positive semidefinite kernels and discuss GP posterior variance which will be used for incorporating the uncertainty in the CBF formulation.

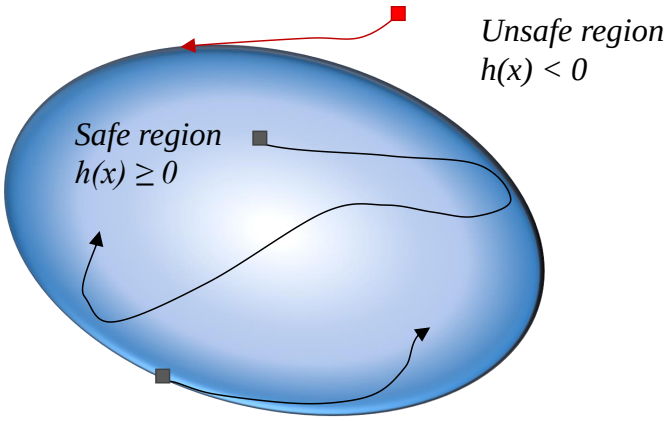

Figure 2.1: The safe set $\mathcal{S}$ is the superlevel set of $h(\mathbf{x})$. The curves are the trajectory of the dynamical system, while the diamonds represent the initial states. Inside the safe set, the system moves freely and can even approach the boundary. Outside the safe region, it will asymptotically converge to the safe set.

2.1.1 Control Barrier Function

Control Barrier Functions ensure that the system's safety requirements are addressed through the forward invariance property. Consider a general control affine system given by,

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) + g(\mathbf{x}(t))\mathbf{u}(t), \tag{2.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state and $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input. The drift vector field, $f : \mathbb{R}^n \to \mathbb{R}^n$, and the control matrix field, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$, are assumed to be locally Lipschitz continuous. Let the safety for (2.1) be encoded as the superlevel set $\mathcal{S}$ of a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$.

$$\mathcal{S} = \{\mathbf{x}(t) \in \mathbb{R}^n \mid h(\mathbf{x}(t)) \geq 0\}. \tag{2.2}$$

Definition 1 (Control Barrier Function [8])**.** *The function $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is defined as a control barrier function (CBF), if there exists an extended class-κ function $alpha$ ($\alpha(0) = 0$*

and strictly increasing) such that for any $\mathbf{x} \in \mathcal{S}$*,*

$$\sup_{u \in \mathcal{U}} \Big\{ L_f h(\mathbf{x}) + L_g h(\mathbf{x}) \mathbf{u} + \alpha(h(\mathbf{x})) \Big\} \geq 0, \tag{2.3}$$

where $L_f h(\mathbf{x}) = \frac{\partial h}{\partial \mathbf{x}} f(\mathbf{x})$ and $L_g h(\mathbf{x}) = \frac{\partial h}{\partial \mathbf{x}} g(\mathbf{x})$ are the Lie derivatives of $h(\mathbf{x})$ along $f(\mathbf{x})$ and $g(\mathbf{x})$ respectively. In Figure 2.1, we see a pictorial representation of a safe region whose superlevel set is defined by a candidate smooth $h(\mathbf{x})$. Inside the safe region, the dynamical system state is allowed to freely move. It can even approach the boundary of the safe region, but it will not go outside. If the system initially begins outside the safe region, then it will asymptotically approach the boundary of the safe region. For proofs of forward invariance inside the safe set and asymptotic convergence outside the safe set, see [8]. The theorem below establishes the forward invariance for the (subset of) system states for which the CBF is defined.

Theorem 1 (Safety Condition [8])**.** *Given a system* (2.1)*, with safe set* $\mathcal{S} \subset \mathbb{R}^n$ *defined by* (2.2)*, and a continuously differentiable CBF* $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ *given by* (2.3)*, any Lipschitz continuous controller* $\mathbf{u} \in \mathbb{R}^m$*, chosen from* $\mathrm{K}_{\mathrm{cbf}} = \{\mathbf{u} \in \mathbb{R}^m \mid L_f h(\mathbf{x}) + L_g h(\mathbf{x}) \mathbf{u} + \alpha(h(\mathbf{x})) \geq 0\}$ *for any* $\mathbf{x} \in \mathbb{R}^n$*, renders the set* $\mathcal{S}$ *forward invariant for* (2.1)*.*

We can rewrite the set of valid control inputs, $\mathrm{K}_{\mathrm{cbf}}$ as follows,

$$\mathrm{K}_{\mathrm{cbf}} = \{\mathbf{u} \in \mathbb{R}^m \mid L_f^{\rho} h(\mathbf{x}) + L_g L_f^{\rho-1} h(\mathbf{x}) \mathbf{u} + \alpha(h(\mathbf{x})) \geq 0\}, \;\; \rho = 1$$

As can be seen from the expression above, CBFs are limited to systems with relative degree, $\rho = 1$, where $\rho \in \mathbb{N}$. CBFs have been extended to deal with systems having higher relative degree ($\rho > 1$), namely the Higher-Order CBFs [20], and a special case of Higher-Order CBFs called the Exponential CBFs [19]. For the purposes of this thesis, Exponential CBFs suffice to deal with higher relative degree systems.

Definition 2 (Exponential Control Barrier Function [19])**.** *A* ρ*-times continuously differ-*

entiable function $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, is defined as an exponential control barrier function (ECBF), with relative degree $\rho \in \mathbb{N}$, if there exists $\mathcal{K} \in \mathbb{R}^\rho$ such that for any $\mathbf{x} \in \mathbb{R}^n$,

$$\sup_{\mathbf{u} \in \mathbb{R}^m} L_f^\rho h(\mathbf{x}) + L_g L_f^{\rho\,-1} h(\mathbf{x})\mathbf{u} + \mathcal{K}^\top \mathcal{H} \geq 0,$$

where $\mathcal{H} = [h(\mathbf{x}), L_f h(\mathbf{x}), ..., L_f^{\rho\,-1} h(\mathbf{x})]^\top \in \mathbb{R}^\rho$ is the Lie derivative vector for $h(\mathbf{x})$, and $\mathcal{K} = [k_0, k_1, ..., k_{\rho-1}]^\top \in \mathbb{R}^\rho$ is the coefficient gain vector for $\mathcal{H}$. $\mathcal{K}$ can be determined using linear control methods such as pole placement. We refer the reader to [19] for proofs of ECBF forward invariance.

***Remark* 1.** *We see that CBFs do not account for any uncertainty. Assuming a CBF to be deterministic in every instance can lead to safety failures, for instance, when sensor measurements are noisy. This is one of the motivations behind purposing a notion of uncertainty in the design of CBFs.*

2.1.2 Positive Definite Kernels

Kernels furnish a notion of similarity between pairs of input points, $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$. However, any arbitrary function of input pairs will not constitute a valid kernel. To be a valid kernel, it should satisfy positive semidefiniteness, see [59].

Definition 3 (Positive semidefinite kernel)**.** *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a nonempty set. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel [59], if the matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, with entries, $[\mathbf{K}]_{(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j)$, is positive semidefinite ($\mathbf{x}^\top \mathbf{K} \mathbf{x} \geq 0,\ \forall \mathbf{x} \in \mathbb{R}^n$), for a finite set $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n$, $i = \{1, \ldots, N\}$.*

A stationary kernel is a function of $|\mathbf{x}_i - \mathbf{x}_j|$ and is invariant to translations in the input space. A popular choice of the kernel function is the squared exponential (SE) kernel, also called the Gaussian kernel or radial basis kernel, which is given as follows,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{L}^{-2} (\mathbf{x}_i - \mathbf{x}_j)}{2}\right) + \delta_{ij}\sigma_\omega^2, \tag{2.4}$$

where δ_{ij} is the Kronecker delta, $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The characteristic length scale is, $\mathbf{l} \in \mathbb{R}^n$, with $\mathbf{L} = \text{diag}(\mathbf{l}) \in \mathbb{R}^{n \times n}$. The signal scale and measurement noise are given by $\sigma_f^2 \in \mathbb{R}$ and $\sigma_\omega^2 \in \mathbb{R}$ respectively. Together, these free parameters constitute the SE kernel's hyperparameters, $\Theta = [\mathbf{L}, \sigma_f^2, \sigma_\omega^2] \in \mathbb{R}^{n \times n} \times \mathbb{R} \times \mathbb{R}$.

For stationary kernels, checking its mean square continuity implies checking for continuity at $k(0, 0)$. The mean-square differentiability (or smoothness property) of a stationary kernel process is determined around $\mathbf{x} = 0$ [59]. Note that different kernels have different hyperparameters. Intuitively, choosing a kernel means that we are modeling the underlying class of functions to be well represented by the kernel. Hence, the choice of the kernel is very problem dependent.

The SE kernel (or Gaussian kernel) is infinitely differentiable, and hence, it is infinitely mean-square differentiable. This allows for a flexible parameterization of safety uncertainty into the function $h(\mathbf{x})$. Although, not all kernels are differentiable, we highlight that there are a broad class of kernels that are differentiable and exhibit good modeling characteristics in many engineering domains, see [60].

2.2 Problem Statement

Consider a CBF $h(\mathbf{x})$ which encodes safety given by an initial safe set $\mathcal{S}_o$ as in (2.2). *Our objective is to incorporate safety uncertainty in the synthesis of* $h(\mathbf{x})$ *from online observations of the system states and ensure that* (2.1) *remains safe.* This leads to the following candidate function,

$$h_s(\mathbf{x}(t)) := \underbrace{h(\mathbf{x}(t))}_{\text{desired safety}} - \underbrace{h_{\text{u}}(\mathbf{x}(t); \Theta)}_{\text{safety uncertainty}}. \tag{2.5}$$

Assumption 1. *We assume observability of the states* $\mathbf{x} \in \mathcal{X}$ *at time* t *for the system* (2.1)*.*

The system's overall safety is given by $h_s(\mathbf{x})$ which incorporates a desired safety component given by $h(\mathbf{x})$ and an uncertainty component given by $h_u(\mathbf{x})$. Intuitively, if the

safety uncertainty decreases, then the overall safety increases and approaches the desired safety. Moreover, the uncertainty component has hyperparameters Θ that can alter the relative notion of safety uncertainty, see Section 2.1.2.

Problem 1. *Given N measurements of the system state $\mathbf{x}$, a prior CBF $h(\mathbf{x})$, synthesize $h_s(\mathbf{x})$ with safety uncertainty using a positive semidefinite kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ by conditioning on $\mathbf{X}_N = \{\mathbf{x}_i\}$, $i \in \{1, \ldots, N\}$.*

Problem 2. *Given the synthesized $h_s(\mathbf{x})$ constructed from a CBF $h(\mathbf{x})$ and a positive semidefinite kernel $k(\mathbf{x}_i, \mathbf{x}_j)$, synthesize a controller that ensures* (2.1) *remains safe.*

Note that incorporating uncertainty in the formulation of a CBF is particularly challenging. This is because, time derivatives are computed on $h(\mathbf{x})$ in order to satisfy the forward invariance properties for CBFs. It is an ill-posed problem to compute the time derivative of an unknown entity, i.e. the system's safety uncertainty, without making prior assumptions. For alleviating such an issue, a kernel representation is used for capturing the safety uncertainty.

2.3 Proposed Methodology

In this section, we present our proposed approach for incorporating a notion of safety uncertainty into the formulation of CBFs. We are interested in determining the uncertainty of the system safety by observing the system state, $\mathbf{x} \in \mathcal{X}$. Given N state data points, with input vectors $\mathbf{x} \in \mathbb{R}^n$, we create the input matrix: $\mathbf{X}_N = \{\mathbf{x}_i\}_{i=1}^N$. The Gaussian process (GP) framework allows computing the posterior variance for an arbitrary query point $\mathbf{x}_* \in \mathbb{R}^n$, by conditioning on previous observations. The posterior predictive variance $\sigma^2 \in \mathbb{R}$ is then given by [59]:

$$\sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^\top \, \overline{\mathbf{K}}^{\,-1} \mathbf{k}(\mathbf{x}_*), \tag{2.6}$$

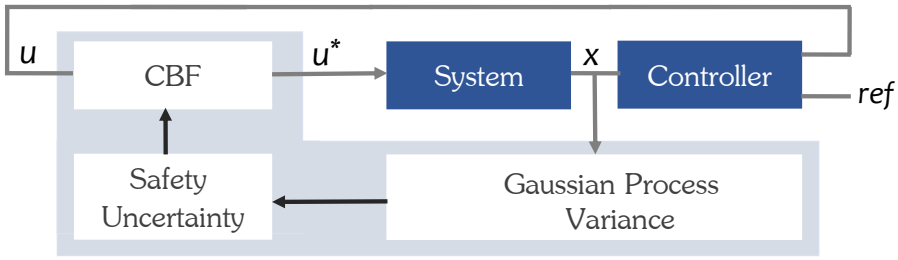

Figure 2.2: CBF augmented with safety uncertainty using the GP posterior variance based on past measurement data.

where $\mathbf{k}(\mathbf{x}_*) = \left[k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_N, \mathbf{x}_*)\right]^\top \in \mathbb{R}^N$ is the covariance vector between $\mathbf{x}_N$ and $\mathbf{x}_*$, $\overline{\mathbf{K}} \in \mathbb{R}^{N \times N}$, with entries $\left[\overline{\mathbf{K}}\right]_{(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j),\ i, j \in \{1, \ldots, N\}$, is the covariance matrix between pairs of input points, and $k(\mathbf{x}_*, \mathbf{x}_*) \in \mathbb{R}$ is the prior covariance. Given a continuously differentiable function $h(\mathbf{x})$, and a positive semidefinite kernel $k(\mathbf{x}_i, \mathbf{x}_j)$, $i, j = \{1, \ldots, N\}$, where N is the number of data points, we propose the following variance-based CBF $h_s(\mathbf{x})$ that incorporates safety uncertainty *online* using (3.2) conditioned on past measurements,

$$\begin{aligned} h_s(\mathbf{x}) &= h(\mathbf{x}) - \sigma^2(\mathbf{x}) \\ &= \underbrace{h(\mathbf{x})}_{\text{desired safety}} - \Big(\underbrace{k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \overline{\mathbf{K}}^{-1} \mathbf{k}(\mathbf{x})}_{\text{safety uncertainty}} \Big). \end{aligned} \tag{2.7}$$

Note that the expression above is a candidate for a variance-based CBF. The main idea is to use the posterior variance as a metric for the uncertainty in the safety estimation.

***Remark* 2.** *The variance based CBF construction above has an analytical form for the safety uncertainty which allows computing the Lie derivatives of $h_s(\mathbf{x})$ in closed-form. Additionally, a data-driven paradigm is formulated since past measurement data is incorporated to determine safety uncertainty. This allows the possibility of constructing conservative safe sets due to the uncertainty quantified using the posterior variance.*

2.3.1 Lie Derivatives of Variance based CBFs

The proposed CBF framework includes the uncertainty in the state space for safety in addition to the desired safety function. Hence, we must also compute the Lie derivatives of the uncertainty component. First, we take the partial derivative of $h_s(\mathbf{x})$ with respect to $\mathbf{x}$ at a query point $\mathbf{x}_*$,

$$\begin{aligned}\left.\frac{\partial h_s(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} &= \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} - \left.\frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} \\ &= \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} - \left(-2\mathbf{k}(\mathbf{x}_*)^\top \overline{\mathbf{K}}^{-1} \left.\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} \right) \\ &= \left.\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} + 2\mathbf{k}(\mathbf{x}_*)^\top \overline{\mathbf{K}}^{-1} \left.\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*}. \end{aligned} \tag{2.8}$$

The kernel derivative in (2.8) is given by,

$$\left.\frac{\partial \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} = (\mathbf{x}_{(i)} - \mathbf{x}_*)^\top \, k(\mathbf{x}_{(i)}, \mathbf{x}_*)\mathbf{L}^{-2}, \tag{2.9}$$

where $\mathbf{k}_{(i)}$ is the i^{th} element of $\mathbf{k}(\mathbf{x})$, and (2.9) is the i^{th} row of $\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{N \times n}$. Now, we can compute the Lie derivatives of $h_s(\mathbf{x})$ by taking its time derivative as follows,

$$\begin{aligned}\dot{h}_s(\mathbf{x}) &= \frac{\partial h_s(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}) + \frac{\partial h_s(\mathbf{x})}{\partial \mathbf{x}} g(\mathbf{x})\mathbf{u} \\ &= L_f h_s(\mathbf{x}) + L_g h_s(\mathbf{x})\mathbf{u}, \end{aligned} \tag{2.10}$$

where (2.8) is used in the Lie derivatives, $L_f h_s(\mathbf{x}) = \frac{\partial h_s}{\partial \mathbf{x}} f(\mathbf{x})$ and $L_g h_s(\mathbf{x}) = \frac{\partial h_s}{\partial \mathbf{x}} g(\mathbf{x})$. In prior literature, the derivative predictions of GP posterior mean and variance are exploited [61, 62]. Note that the derivative prediction of GP posterior variance represents the variance on the derivative prediction of the GP posterior mean. Here, we take the partial derivative of $\sigma^2(\mathbf{x})$ with respect to the state $\mathbf{x}$ which is different from the derivative prediction of GP posterior variance. The GP posterior variance and its partial derivative are shown in

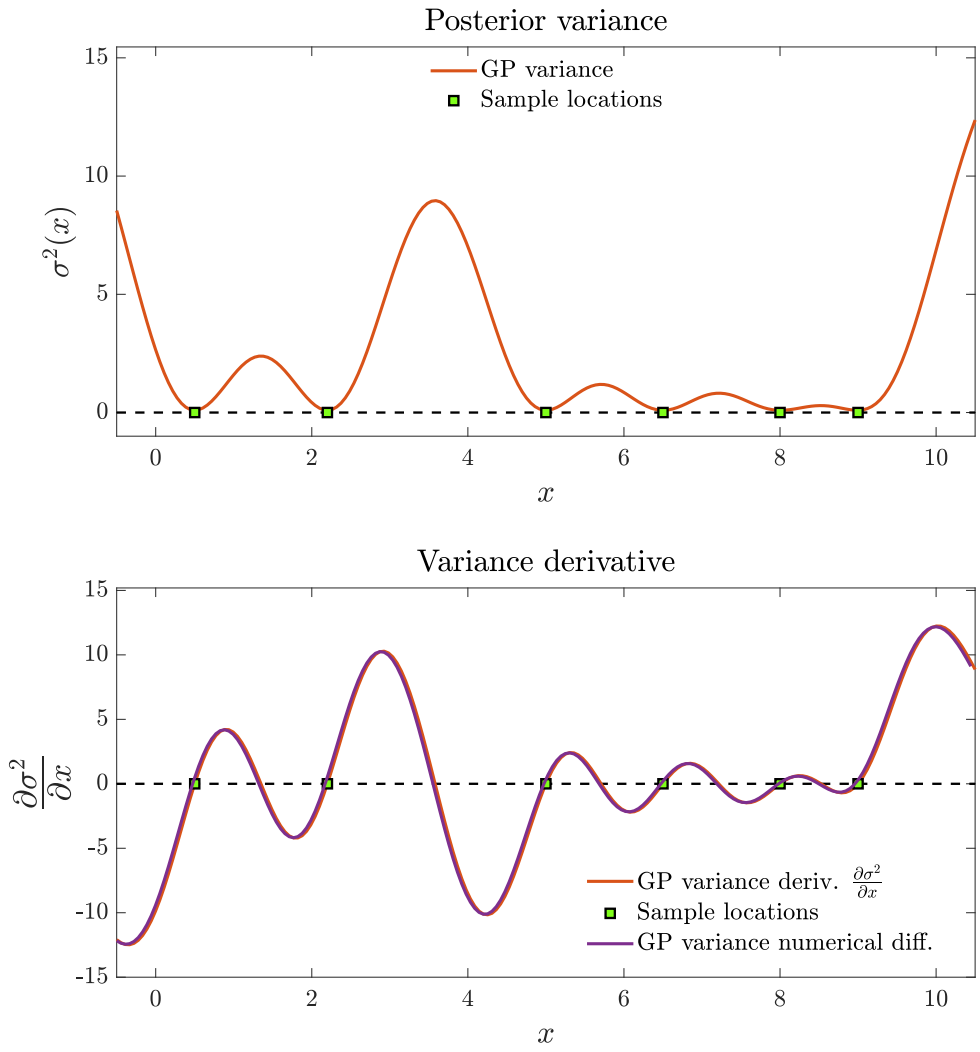

Figure 2.3: The GP posterior variance (top) and its derivative (bottom) is plotted using 5 input samples. The posterior variance is small at sample location inputs, as expected. The analytical form of variance derivative is compared with numerical differentiation.

Figure 2.3. At the sample location inputs, we see that the variance is small (or zero). This makes sense since the variance will be high at the locations where we do not have data samples. The posterior variance derivative derived above is verified with the numerical differentiation using finite differences.

2.3.2 Online Safety Control

We now synthesize the controller that will ensure the state of the system (2.1) will be forward invariant in a prescribed safe set. CBFs can be used in conjunction with a convex optimization program, in particular a quadratic program (QP), to derive a computationally efficient method to enforce safety for the control-affine systems of the form (2.1).

Assumption 2. *We assume the existence of a nominal control input* $\mathbf{u}_{\text{nom}}$ *that drives the state* $\mathbf{x}$ *of system* (2.1) *to a desired state* $\mathbf{x}_{\text{des}}$.

The given nominal control input $\mathbf{u}_{\text{nom}}(t) \in \mathbb{R}^m$ is designed as the feedback policy for the system (2.1). However, this control policy may not confine the system inside the safe set at all times. An online QP rectifies $\mathbf{u}_{\text{nom}}$ whose constraints are given by the Lie derivatives in (2.10) [8].

Variance-based CBF-QP: *Control Input Rectification*

$$\mathbf{u}^* = \underset{\mathbf{u} \in \mathbb{R}^m}{\arg\min} \frac{1}{2}\left\|\mathbf{u} - \mathbf{u}_{\text{nom}}\right\|^2 \text{ s.t.} \tag{2.11}$$
$$L_f h_s(\mathbf{x}) + L_g h_s(\mathbf{x})\mathbf{u} + \alpha(h_s(\mathbf{x})) \geq 0,$$

---

where $\mathbf{u}^*$ is the rectified control input. By rectifying the control policy, the system is guaranteed to remain forward invariant for the safe set $\mathcal{S}$.

2.4 2D Safe Set Expansion using Exploratory Samples

Here, we extend the framework above to allow expansion of an initial safe set $\mathcal{S}_o$ to a desired final set $\mathcal{S}_f$. We denote the boundary of $\mathcal{S}_o$ as $\partial\mathcal{S}_o = \{\mathbf{x} \in \mathbb{R}^n \mid h_s(\mathbf{x}) = 0\}$. Since there can be uncountably infinite points in $\partial\mathcal{S}_o$, we make the following assumption,

Assumption 3. *Given* $h_s(\mathbf{x})$ *encoding superlevel safe set* $\mathcal{S}_o$ *with* $\partial\mathcal{S}_o$ *denoting its boundary, there exists* $\mathbf{x}_b \in \mathbb{R}^n$ *such that* $h_s(\mathbf{x}_b) = 0$.

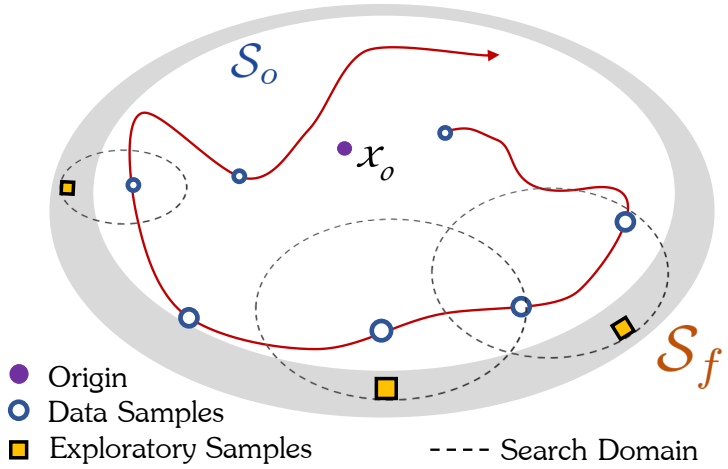

Figure 2.4: Expansion of initial safe set $\mathcal{S}_o$ to expanded safe set $\mathcal{S}_f$. Exploratory samples are located within a proximal search space (black dashed).

It is reasonable to assume knowledge of $\mathbf{x}_b$. This is because a candidate CBF is designed such that $\mathbf{x}_b$ represents the state with the least safety. A query state $\mathbf{x}_* \in \mathbb{R}^n$ is sampled if,

$$\left\|\mathbf{x}_{(i)} - \mathbf{x}_*\right\| \geq \tau, \qquad i = \{1, \ldots, N\}, \tag{2.12}$$

where $\tau \in \mathbb{R}$ denotes the distance between any two samples. This avoids dense sampling of the states resulting in computational tractability. Moreover, samples too close together may result in an ill-conditioned covariance matrix [59].

We assume there is an external (safety) observer, such as a high-level vision or semantic segmentation planner, detecting safety candidates outside the current safety limit. As the system is moving, a local safety map is computed to determine safety candidates with minimal safety exceeding the initial safety limit $\mathbf{x}_b$, see Figure 2.4. As more data is collected, the overall safety increases. Consequently, the safety margin increases. Hence, a solution $\mathbf{x}_*$ may exist for $|h_s(\mathbf{x}_*)| < \epsilon$ with $\mathbf{x}_*$ exceeding $\mathbf{x}_b$ as the system moves closer to $\mathbf{x}_b$. As there can be multiple $\mathbf{x}_*$ satisfying $|h_s(\mathbf{x}_*)| < \epsilon$, we choose the $\mathbf{x}_*$ furthest away from $\mathbf{x}_b$.

We compose a set E of exploratory samples given by,

$$E = \{\mathbf{x}_* \in \mathbb{R}^n \mid \left\|\mathbf{x}_* - \mathbf{x}_o\right\| - \left\|\mathbf{x}_b - \mathbf{x}_o\right\| > 0 \} \text{ such that,} \tag{2.13}$$

$$|h_s(\mathbf{x}_*)| < \epsilon,\ \mathbf{x}_* \in [\mathbf{x}_* - \delta\tau, \mathbf{x}_* + \delta\tau] \tag{2.14}$$

where $\mathbf{x}_o \in \mathbb{R}^n$ is the origin, $\delta \in \mathbb{R}$ is a scaling parameter for the domain search space, and $\epsilon \in \mathbb{R}$ is minimum safety requirement. If the cardinality of set E is greater than a given number of required exploratory samples, i.e. $|E| > n_{\text{explore}}$, then the safe set can be expanded by setting $\mathbf{x}_b \leftarrow \max\,[\,E\,]$, where $\max[\,]$ returns the exploratory sample furthest away from the origin $\mathbf{x}_o$.

Note that a less conservative expansion could involve applying the $\max$ operator to E. The cardinality of E will be non-zero only if exploration happens at the boundary of the safe set. As more samples are collected, the safety uncertainty encoded by $h_u(\mathbf{x})$ decreases, thereby increasing the overall safety of $h_s(\mathbf{x})$. Inclusion of safety uncertainty along with safe expansion of the initial set is summarized in Algorithm 1.

Algorithm 1 Variance-based CBF Synthesis with Expansion in 2D

procedure EXPAND($\mathcal{S}_o(\mathbf{x}_b), \mathcal{S}_f(\mathbf{x}_{\text{des}}), n_{\text{explore}}$)
  while $\|\mathbf{x}_b\| \leq \|\mathbf{x}_{\text{des}}\|$ **do** ▷ Final safe set reached
    SAMPLE input points $\mathbf{X}_N \leftarrow \mathbf{x}_*$ using (3.6)
    LOCATE safety candidate $\mathbf{x}_*$ using (2.14)
    COMPOSE exploratory set E using (2.13)
    if $|E| > n_{\text{explore}}$ **then**
      $\mathbf{x}_b \leftarrow \max[\,E\,]$
    SYNTHESIZE $h_s(\mathbf{x}; \mathbf{x}_b, \mathbf{X}_N)$ using (3.8)
    RECTIFY $\mathbf{u}_{\text{nom}}$ using (2.8)-(2.11)
  return $\mathbf{x}_b$

2.5 Application Test Case: Safe Set Expansion using 3D Quadrotor

In this section, we demonstrate the efficacy of our method on a quadrotor example in hardware. This is an interesting and challenging problem due to the safety-critical nature of

quadrotors. Quadrotors are dynamically unstable, hence if any uncertainty is not mitigated, the system may become unstable or potentially crash. Our objective is to constrain the quadrotor in the position space and allow safe expansion of the barrier certificates in position space. Through our approach, a safe data-driven paradigm can be employed on a quadrotor for safe exploration.

First, we review the quadrotor dynamics followed by safety rectification on a quadrotor using the variance augmented CBF discussed in Section 2.3. CBFs have been typically applied on a quadrotor platform without incorporating safety uncertainty in its design. Using the variance based CBF and the incremental expansion of the set using Algorithm 1, the quadrotor is able to safely explore and expand the initial conservative safe set.

2.5.1 Quadrotor Dynamics

We consider the position dynamics and attitude kinematics of a quadrotor model evolving in a coordinate-free framework. This framework uses a geometric representation for its attitude given by a rotation matrix $\mathbf{R}$ on $SO(3) := \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I}_3,\ \det(\mathbf{R}) = 1\}$. $\mathbf{R}$ represents the rotation from the body-frame to the inertial-frame. The origin of the quadrotor's body-frame is given by the quadrotor's center of mass, denoted by $\mathbf{r} \in \mathbb{R}^3$. A quadrotor is an underactuated system since it has 6 DOF, due to its configuration space being $SE(3)$, but 4 control inputs; thrust $F \in \mathbb{R}$ and moments $\mathbf{M} \in \mathbb{R}^3$. The equations of motion are:

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{2.15}$$

$$m\dot{\mathbf{v}} = -mg\mathbf{e}_3 + F\mathbf{R}\mathbf{e}_3, \tag{2.16}$$

$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\Omega}^\times, \tag{2.17}$$

$$\mathbf{J}\dot{\boldsymbol{\Omega}} = \mathbf{M} - (\boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega}), \tag{2.18}$$

where $\mathbf{v} \in \mathbb{R}^3$ is the velocity in the inertial frame, $m \in \mathbb{R}_{>0}$ is the quadrotor mass, $g \in \mathbb{R}$ is gravity, $\mathbf{e}_3 = [0,\ 0,\ 1]^\top$, $\mathbf{\Omega} \in \mathbb{R}^3$ is the body-frame angular velocity, $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the inertia matrix, and $(\cdot)^\times : \mathbb{R}^3 \to so(3)$ is the skew-symmetric operator, such that $\forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^3, \mathbf{x}^\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$

2.5.2 Setpoint Generation for Quadrotor

For achieving safety constrained control of the quadrotor, we employ the strategy of designing safe setpoints. We send setpoints to the quadrotor, a Crazyflie 2.1 platform, in the form of desired thrust, F_{des}, and desired roll, pitch, yaw angles, $\eta = [\phi_{\text{des}}\ \theta_{\text{des}}\ \psi_{\text{des}}]^\top \in \mathbb{R}^3$. Crazyflie is equipped with a fast response low-level onboard controller that can directly track these setpoint commands. More details on the hardware experimental setup are covered in Section 2.6.1.

Given a desired trajectory, $\mathbf{r}_{\text{des}} \in \mathbb{R}^3$, that is twice differentiable, a second-order integrator model can be setup,

$$\underbrace{\begin{bmatrix}\dot{\mathbf{r}}\\ \ddot{\mathbf{r}}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{r}\\ \dot{\mathbf{r}}\end{bmatrix}}_{f(\mathbf{x})} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}}_{g(\mathbf{x})}\mathbf{u}, \tag{2.19}$$

where $\mathbf{r}$, $\dot{\mathbf{r}}$, $\ddot{\mathbf{r}}$ are the quadrotor's position, velocity, and acceleration respectively, $\mathbf{x} = [\mathbf{r}^\top, \dot{\mathbf{r}}^\top]^\top \in \mathbb{R}^6$, and $\mathbf{u} = \ddot{\mathbf{r}}_{\text{des}} \in \mathbb{R}^3$. We assume the desired yaw to be zero and make small angle approximations to invert the quadrotor's nonlinear dynamics [63], [64]. The relation between the second-order integrator model control input and the quadrotor's desired setpoints are given by,

$$u_{1,\text{nom}} = g(\theta_{\text{des}} \cos\psi + \phi_{\text{des}} \sin\psi), \tag{2.20}$$

$$u_{2,\text{nom}} = g(\theta_{\text{des}} \sin\psi - \phi_{\text{des}} \cos\psi), \tag{2.21}$$

$$u_{3,\text{nom}} = \frac{F_{\text{des}}}{m} - g, \tag{2.22}$$

where the desired Euler angles, $\theta_{\text{des}},\ \phi_{\text{des}}$, and desired thrust, F_{des}, are coming from the human teleoperator. The nominal control inputs $\mathbf{u}_{\text{nom}} = [u_{1,\text{nom}},\ u_{2,\text{nom}},\ u_{3,\text{nom}}]^\top \in \mathbb{R}^3$ are then rectified with the synthesized Gaussian CBF to generate rectified control inputs $\mathbf{u}_{\text{rect}}$. These rectified control inputs are then inverted to get rectified setpoints for the quadrotor using,

$$\phi_{\text{rect}} = \frac{(u_{1,\text{rect}} \sin\psi - u_{2,\text{rect}} \cos\psi)}{g} \tag{2.23}$$

$$\theta_{\text{rect}} = \frac{(u_{1,\text{rect}} \cos\psi + u_{2,\text{rect}} \sin\psi)}{g} \tag{2.24}$$

$$F_{\text{rect}} = m(u_{3,\text{rect}} + g). \tag{2.25}$$

Next, we discuss the safety rectification of $\mathbf{u}$ to compute $\mathbf{u}_{\text{rect}}$ using the CBF in (2.5).

2.5.3 Quadrotor Online Control Rectification

Our objective is to constrain the quadrotor in the limited position space and allow safe expansion by incorporating safety uncertainty in the formulation. Given a variance augmented CBF expressed in the position space, then the relative degree for the system (2.19) is $\rho = 2$. The associated Lie derivatives for the variance based CBF are given by,

$$L_f h_s(\mathbf{x}) = L_f h(\mathbf{x}) - \left(\nabla\sigma^2(\mathbf{x})\right)^\top f(\mathbf{x}),$$

$$L_f^2 h_s(\mathbf{x}) = L_f^2 h(\mathbf{x}) - f(\mathbf{x})^\top \mathbf{H}_{\sigma^2}(\mathbf{x}) f(\mathbf{x}) - \left(\nabla\sigma^2(\mathbf{x})\right)^\top \cdot \nabla f(\mathbf{x}) \cdot f(\mathbf{x}),$$

$$L_g L_f h_s(\mathbf{x}) = L_g L_f h(\mathbf{x}) - f(\mathbf{x})^\top \mathbf{H}_{\sigma^2}(\mathbf{x}) g(\mathbf{x}) - \left(\nabla\sigma^2(\mathbf{x})\right)^\top \cdot \nabla f(\mathbf{x}) \cdot g(\mathbf{x}),$$

where $\nabla\sigma^2(\mathbf{x}) = \frac{\partial\sigma^2(\mathbf{x})^\top}{\partial\mathbf{x}}$ is the gradient of GP variance (as seen in (2.8)) and $\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}$ is the Jacobian of $f(\mathbf{x})$. $\mathbf{H}_{\sigma^2}(\mathbf{x})$ is the Hessian of GP variance given by,

$$\mathbf{H}_{\sigma^2}(\mathbf{x}) = -2\nabla\mathbf{k}(\mathbf{x})\overline{\mathbf{K}}^{-1}\nabla\mathbf{k}(\mathbf{x})^\top - 2\frac{\partial}{\partial\mathbf{x}}\left(\sum_i^N b_i \frac{\partial\mathbf{k}_i(\mathbf{x})}{\partial\mathbf{x}}\right),$$

where $\nabla \mathbf{k}(\mathbf{x}) = \frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}^\top \in \mathbb{R}^{n \times N}$, b_i is the i^{th} element of $\mathbf{k}(\mathbf{x}_*)^\top \overline{\mathbf{K}}^{-1} \in \mathbb{R}^{1 \times N}$, and $\frac{\partial \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}}$ is given by (2.9). Given the nominal control input $\mathbf{u}_{\text{nom}} \in \mathbb{R}^3$ in (2.19), the QP below rectifies $\mathbf{u}_{\text{nom}}$ into $\mathbf{u}_{\text{rect}} \in \mathbb{R}^3$,

---

Variance based CBF-QP: *Quadrotor Input Rectification*

$$\begin{aligned} \mathbf{u}_{\text{rect}} = \underset{\mathbf{u} \,\in\, \mathbb{R}^3}{\arg\min} \frac{1}{2} \big\| \mathbf{u} - \mathbf{u}_{\text{nom}} \big\|^2 \ \text{ s.t.} \\ L_f^2 h_s(\mathbf{x}) + L_g L_f h_s(\mathbf{x}) \mathbf{u} + \mathcal{K}^\top \mathcal{H} \geq 0, \end{aligned} \tag{2.26}$$

---

where $\mathcal{K} = [k_1 \ k_2]^\top \in \mathbb{R}^2$, and $\mathcal{H} = [L_f h_s(\mathbf{x}) \ h_s(\mathbf{x})]^\top \in \mathbb{R}^2$. The QP solution $\mathbf{u}_{\text{rect}}$ is then used to compute the rectified setpoints using (2.23)-(2.25). Finally, the rectified setpoints are then sent to the Crazyflie 2.1.

2.6 Hardware Experimental Verification

In this section, we discuss the experimental setup and results on a hardware quadrotor. The objective of the experiment is to expand an initial lateral safe set along x and y online by collecting position and velocity state measurements of the quadrotor. Two individual variance based CBFs are used as follows,

$$\mathbf{h}_s(\mathbf{x}) = \begin{bmatrix} h_x(\mathbf{x}) \\ h_y(\mathbf{x}) \end{bmatrix}, \tag{2.27}$$

where $\mathbf{h}_s(\mathbf{x}) = [h_x(\mathbf{x}) \ h_y(\mathbf{x})]^\top \in \mathbb{R}^2$ are two separate variance based CBFs. Each CBF is designed to take the following form,

$$h_x(\mathbf{x}) = (a_x - b_x)\big(1 - \sigma^2(b_x)\big) + b_x - x - \sigma^2(\mathbf{x}), \tag{2.28}$$

$$h_y(\mathbf{x}) = (a_y - b_y)\big(1 - \sigma^2(b_y)\big) + b_y - y - \sigma^2(\mathbf{x}), \tag{2.29}$$

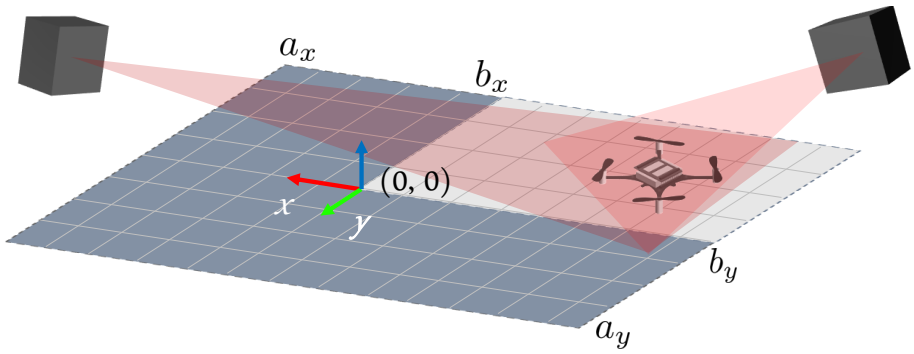

Figure 2.5: Experimental setup for expanding initial safe set independently along x and y. State estimation is done using an external lighthouse system.

where the initial safety limit along x and y positions is given by $\mathbf{b} = [b_x \; b_y]^\top \in \mathbb{R}^2$, $\mathbf{a} = [a_x \; a_y]^\top \in \mathbb{R}^2$ is the final safety limit, x and y are the x, y components of $\mathbf{r}$ position state, $\mathbf{x} = [\, x \;\; y \;\; \dot{x} \;\; \dot{y} \,]^\top \in \mathbb{R}^4$, and $\sigma^2(\cdot) \in \mathbb{R}$.

The safety uncertainty captures the variance in lateral position, velocity, and at the initial safety limit. Since the CBF should constrain the quadrotor only in the lateral position space, we use x, y components only and do not use the altitude position z. For the safety uncertainty, both the position and velocity states are used. This is because the quadrotor may visit previously sampled positions with differing velocities. This introduces uncertainty in previously visited sites, thus changing the safe set. The experimental video link is: `https://youtu.be/9qvOf1UpRPw`.

2.6.1 Experiment Setup

The Crazyflie 2.1 is used as the hardware quadrotor. State estimation is performed onboard with the help of an external low-cost lighthouse positioning system [65]. Setpoint commands are computed remotely on a ground station equipped with an Intel i7-9800X at 4.4GHz processor and 16 GB RAM. The `crazyflie_ros` API is used to communicate for interprocess communication, subscribing to pose information, and publishing setpoints over the Crazyradio PA USB dongle [66]. Figure 2.5 illustrates the experimental setup.

Data measurements composed of (x, y) positions and velocities are collected at 100Hz

with a data capacity set to 100 samples. The input rectification routine in (2.11))is run on a parallel thread at 50Hz where solving the QP takes under 5ms. The barrier expansion algorithm is run at 25Hz. Finding safe candidates using (2.14) takes under 5ms for each dimension. Setpoint commands are sent to the Crazyflie at 100Hz.

The parameters used in the experiment are $\tau = 0.1\text{m}$ for the sampling distance, a scaling parameter of $\delta = 2$ for the safety domain search space, and $(b_x, b_y) = (0, 0)$ as the initial safety limit, and $(a_x, a_y) = (0.5, 0.5)$ as the final safety limits. The quadrotor's initial location is at $(x_0, y_0, z_0) = (-0.45, -0.45, 0)$. The minimum safety requirement for detecting new safety candidates is $\epsilon = 0.01$, while the number of exploratory samples $n^x_{\text{explore}} = \lceil b_y/(\delta\tau) \rceil$ along x direction and $n^y_{\text{explore}} = \lceil b_x/(\delta\tau) \rceil$ along y direction. As the barrier expands along one direction, more exploratory samples are required in the other direction. The SE kernel's hyperparameters are set as $\Theta = [\text{diag}(0.2, 0.2), 1, 0.01]$. Since no hyperparameter tuning is required in our experimental test case, we preset the hyperparameters.

2.6.2 Experimental Results

The quadrotor initially moves along the positive x direction. To expand the boundary along x, exploration needs to take place in a vertical pattern. As seen in Figure 2.6, as samples are collected along the trajectory, there is reduced uncertainty. From the two local safety maps constructed online using (2.14), along each axis x and y, we see that the exploratory samples are found only for x local map. This is expected since the quadrotor is near the barrier limit for $h_x(\mathbf{x})$. Only after a minimum number of exploratory samples are found, the barrier now expands to the furthest safety candidate's location. The exploratory samples are selected such that $|h_x(\mathbf{x}_*)| < 0.01$. Hence, the safe set expands to a value which is at least minimally safe.

The system then is directed to expand along the positive y axis. Exploration now needs to happen in a horizontal flight to expand the barrier limit b_y. In Figure 2.7, we notice that

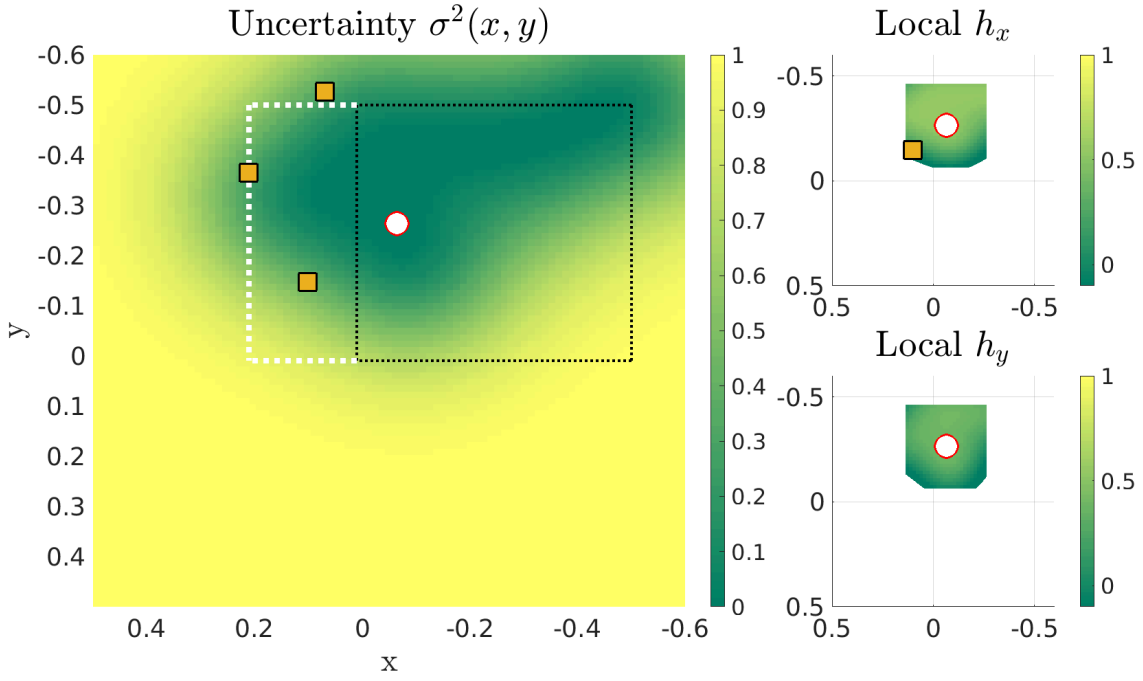

Figure 2.6: Uncertainty with initial safe set (black dashed), exploratory samples along x axis (orange square), quadrotor's location (white disk), and barrier expansion (white dotted). Local safety maps are constructed online using (2.14) for both x and y axes. Exploratory samples are found only for local $h_x(\mathbf{x})$ since the quadrotor is near the vertical border.

the safe set is longer along the x direction. Therefore, the number of exploratory samples required for expanding along y has increased since $n^y_{\text{explore}} = \lceil b_x/(\delta\tau) \rceil$. Although the two safety maps are constructed, we see that solutions are found only for $h_y(\mathbf{x})$ as opposed to $h_x(\mathbf{x})$, since the quadrotor is now near b_y. Analogous to the previous setting, the barrier now expands at the furthest minimally safe location.

As the quadrotor continues to fly and collect more samples, uncertainty reduces and as a result the overall safety increases. The framework continues until the desired safety set is achieved. In Figure 2.8, the superlevel zero set is shown for both the Gaussian CBFs, h_x and h_y. The system expanded the initial safe set to the desired final safe set using only 46 samples. Our algorithmic framework resynthesizes both h_x and h_y online ensuring that expansion happens only if exploratory samples are found which are minimally safe.

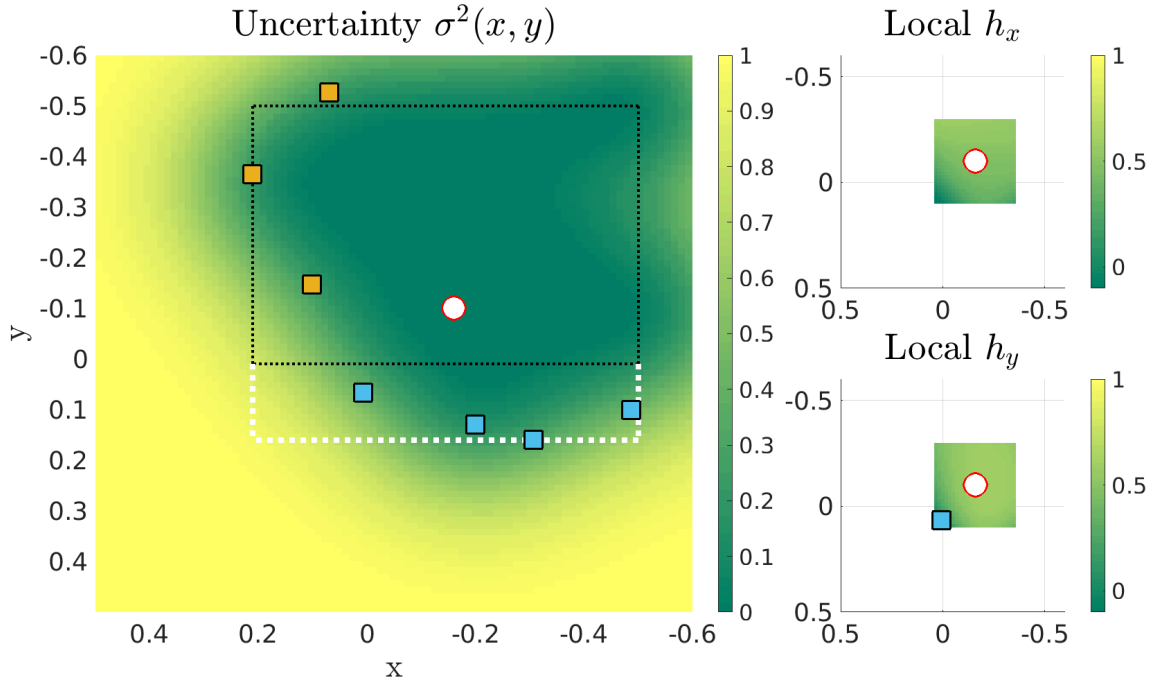

Figure 2.7: Uncertainty with initial safe set (black dashed), exploratory samples along y axis (blue square), quadrotor's location (white disk), and barrier expansion (white dotted). In this instance, more exploratory samples are required to expand along y axis due to the initial expansion along x axis.

2.7 Concluding Remarks

In this chapter we presented a framework for incorporating uncertainty in the synthesis of a given CBF. This uncertainty is parameterized using the Gaussian process variance. By collecting state measurements, safety uncertainty decreases and the overall safety increases, thus approaching the final desired safety. To expand the safe set, local safety maps are computed online at the present location to determine exploratory samples with minimal safety exceeding the current safety limit. We assume an external safety observer can detect these minimally safe candidates, e.g. high-level vision or semantic segmentation planner. Since the new candidate locations are minimally safe, expansion of the safe set happens always in a safe manner. This allows a safe exploratory process for the system. We successfully demonstrated our approach in a robotic experiment by expanding an initial safe set along x and y axes independently without risking any expensive system failures.

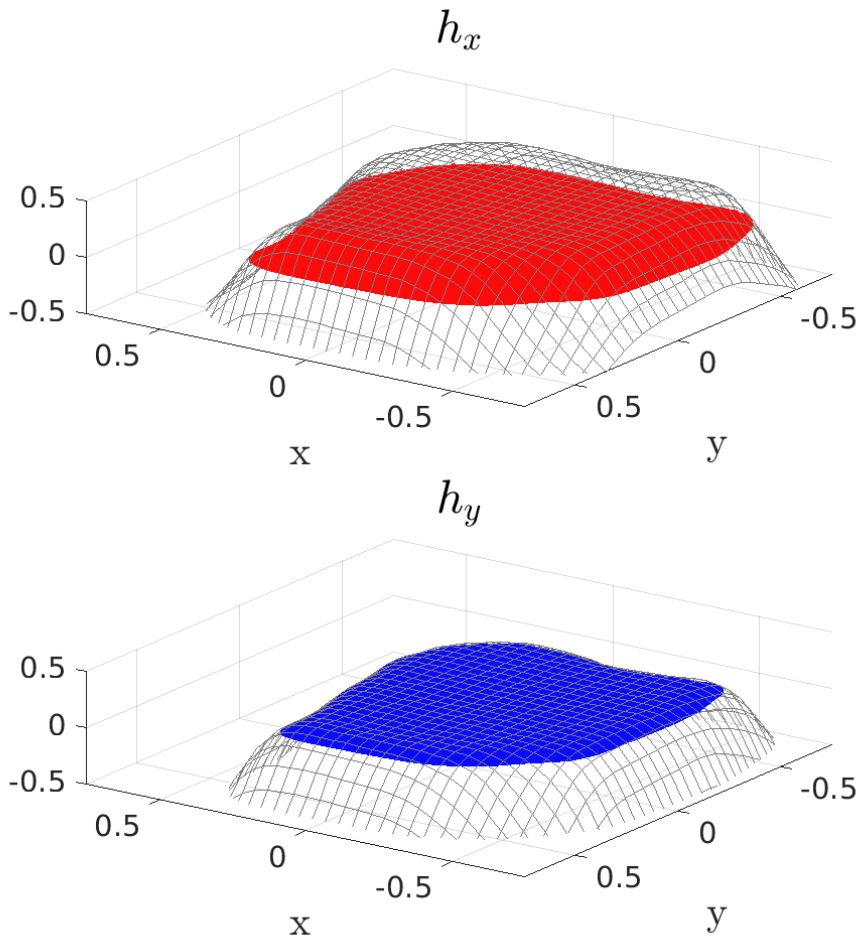

Figure 2.8: Final safe sets for $h_x(\mathbf{x})$ (red contour) and $h_y(\mathbf{x})$ (blue contour) where only the positions (x, y) are used to create the surface plot (using $\dot{x} = 0$, and $\dot{y} = 0$).

CHAPTER 3
BAYESIAN APPROACH TO SAFETY USING GAUSSIAN CONTROL BARRIER FUNCTIONS

This chapter presents the main contribution and focus of the thesis. Inspired by the success of control barrier functions (CBFs) in addressing safety, and the rise of data-driven techniques for modeling functions, we propose a non-parametric approach for the online synthesis of CBFs using Gaussian Processes (GPs). As seen in Chapter 2, a dynamical system is defined to be safe if a subset of its states remains within the prescribed set, also called the *safe set*. CBFs achieve safety by designing a candidate function a priori. However, designing such a function can be challenging. Consider designing a CBF in a disaster recovery scenario where safe and navigable regions need to be determined. The decision boundary for safety here is unknown and cannot be designed a priori.

CBFs generally employ a parametric or constructive design approach, which means that they are hand-designed functions. Moreover, based on the design choice of the CBFs, it is not straight forward to handle arbitrary changes to the safe set in practice. In our approach, we work with *safety samples* as opposed to a hand-designed candidate function to construct the CBF online by assuming a flexible GP prior on these samples (see Figure 3.1). The resulting formulation is called *Gaussian Control Barrier Functions*. We already witnessed the augmentation of the GP posterior variance to a given candidate CBF in Chapter 2. Here, we fully synthesize the CBF as a GP. Gaussian CBFs have favorable properties such as analytical tractability and robust uncertainty estimation. This allows realizing the posterior with high safety guarantees while also computing associated partial derivatives analytically for safe control. Moreover, Gaussian CBFs can change the safe set arbitrarily based on sampled data, thus allowing non-convex safe sets.

In this chapter, we consider a fully non-parametric formulation for synthesizing the

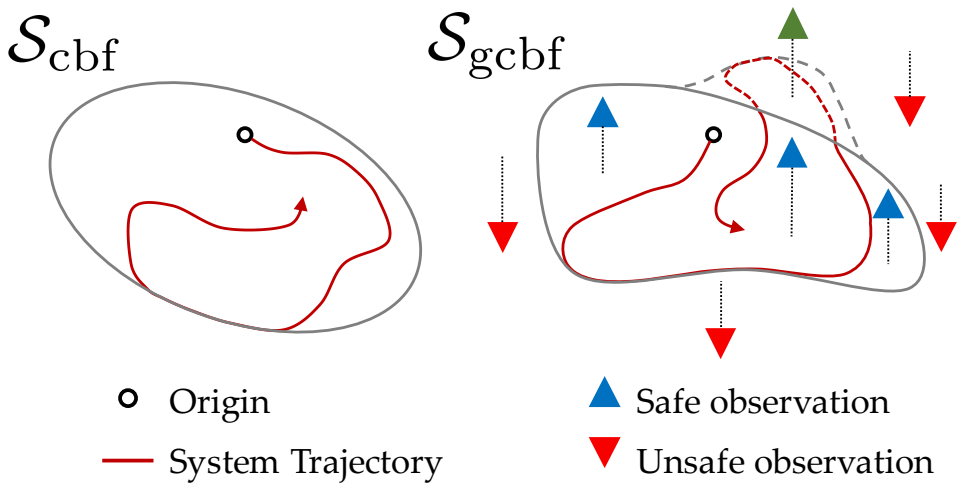

Figure 3.1: The Gaussian CBF uses a non-parametric design approach by relying on data to produce the safe sets. The 0-level contour sets are shown for a traditional CBF (left) and Gaussian CBF (right) with safe sets $\mathcal{S}_{\text{cbf}}$ and $\mathcal{S}_{\text{gcbf}}$ respectively. As a new observation is received (green upper triangle), the Gaussian CBF can change the safe set based on the data in a non-convex fashion.

safety function without requiring any parametric CBF candidate function. We first present the background preliminary on GP regression followed by the problem statement. We then go over our proposed methodology for Gaussian CBFs in detail. We present a novel approach for synthesizing CBFs in a data-driven non-parametric manner using GPs. This is achieved using safety samples as opposed to the prevailing use of CBFs which requires a function. We construct Gaussian CBFs to design safe sets based on the data. These sets are not confined to convex safe sets. GPs provide favorable properties such as analytical tractability and uncertainty estimation which are key enablers in finding closed-form safety function and associated Lie derivatives with high guarantees. We formulate Gaussian CBFs for safe control in the presence of noise for both the safety samples (observations to GPs) and the system states (inputs to the GPs). Finally, we validate Gaussian CBFs in hardware using a quadrotor for three case studies: (i) safe control for fixed but arbitrary safe sets, (ii) online obstacle avoidance with an evolving safe set, and (iii) juxtaposing Gaussian CBFs with regular CBFs for safe control in the presence of noisy system states. To the best of

our knowledge, we believe this is the first work that fully synthesizes a CBF in a non-parametric data-driven manner online using GPs and validate all findings in hardware. The experiment video link is: `https://youtu.be/HX6uokvCiGk`.

3.1 Background Preliminary

Here, we briefly discuss the mathematical background behind GPs, in particular GP regression. GPs are a popular choice in machine learning for nonparametric regression which rely on kernels. We take a function space viewpoint towards GPs where inference takes place directly in the function space.

3.1.1 Bayesian Modeling

Consider a simple 1-d regression problem, where we want to map an input x to an output $h(x)$. In Bayesian inference, to learn the underlying latent function $h(x)$, a prior is placed on the class of functions. This prior represents the belief over the kinds of functions we expect to see before observing any data. The specification of the prior function is based on the choice of the kernel or covariance function used satisfying certain properties, see Section 2.1.2. The kernel or covariance function induces the properties for the class of functions we are interested in learning. Suppose we are given a kernel function, for instance, the SE kernel (2.4), we can draw function priors using the following distribution,

$$h_{\text{prior}} \sim \mathcal{N}(0, \mathbf{K}(x_*, x_*)),$$

with each entry of $\mathbf{K}(x_*, x_*)$ given by $[\mathbf{K}]_{(i,j)} = k(x_i, x_j)$, where $i, j = \{1, \ldots, 100\}$. Priors drawn from the Gaussian distribution with 0 mean and SE kernel are shown in Figures 3.2 - 3.4. As the number of priors increases, we see a band of functions approaching 0 mean in Figure 3.4.

Now, suppose we are given $N = 6$ datapoints, we can update our prior belief using the

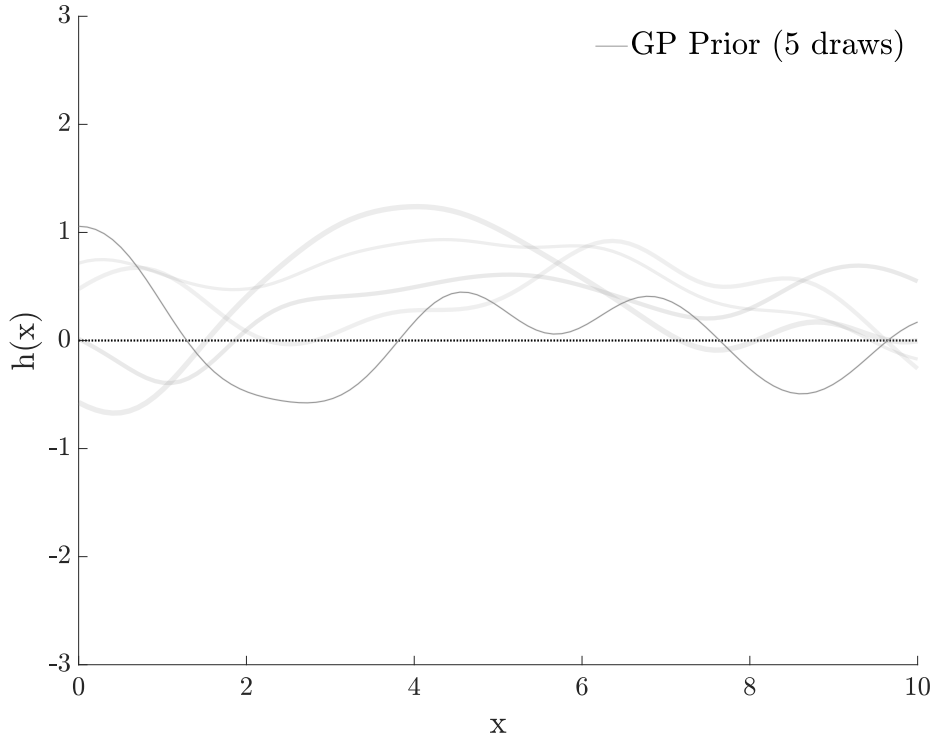

Figure 3.2: Drawing 5 functions using a GP prior using 100 equidistant test points.

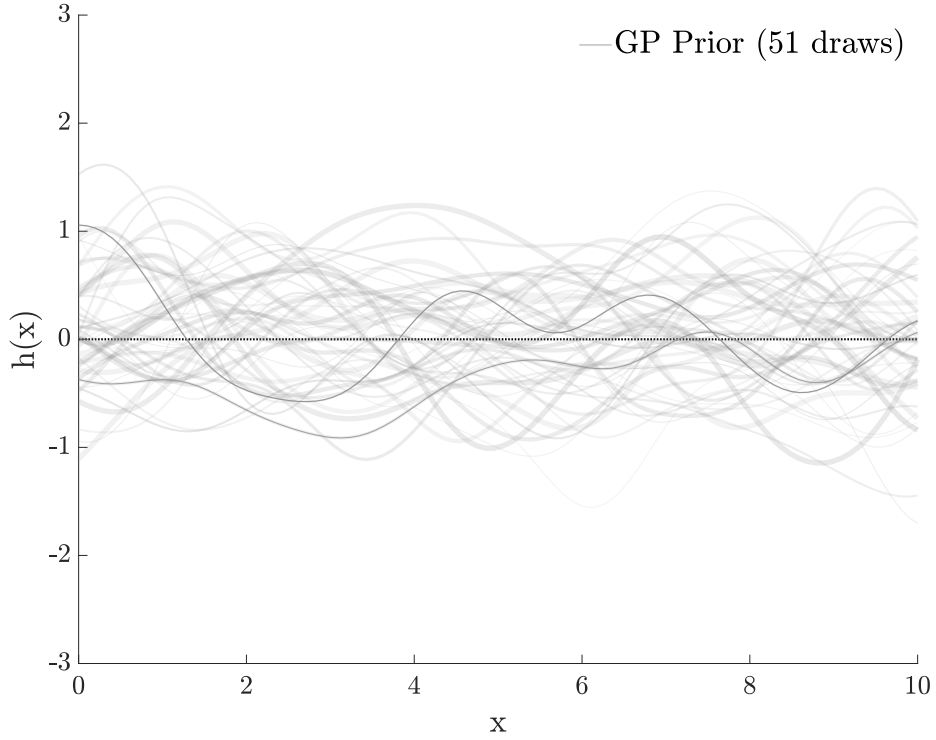

Figure 3.3: Drawing 51 functions using a GP prior using 100 equidistant test points.

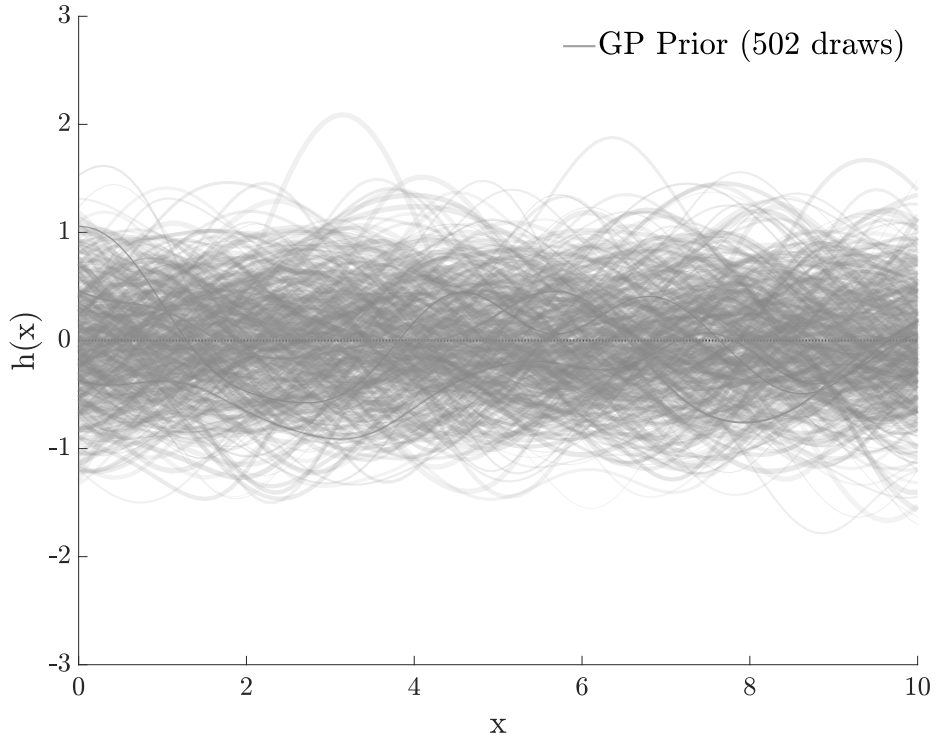

Figure 3.4: Drawing 502 functions using a GP prior using 100 equidistant test points.

help of data. In Figure 3.5, the 6 sample inputs and corresponding observations are shown. The combination of the data and the prior leads to the posterior distribution over functions. The posterior is derived by conditioning the prior on the 6 noise free observations given, which are shown in Figures 3.6 - 3.8. What is interesting to observe is that the posterior distribution over the functions passes through the points. If more datapoints were provided, the corresponding posterior mean would adjust itself to pass through the datapoints and the posterior uncertainty would reduce as a result around the datapoints. However, while being a perfectly valid way of doing inference, it is impractical due to computational limits. GPs provide an alternative way of computing function inference while providing closed-form analytical solutions.

3.1.2 Gaussian Process Regression

Definition 4 (Gaussian process (GP) [59])**.** A Gaussian process is a collection of random variables, any finite number of which has a joint Gaussian distribution.

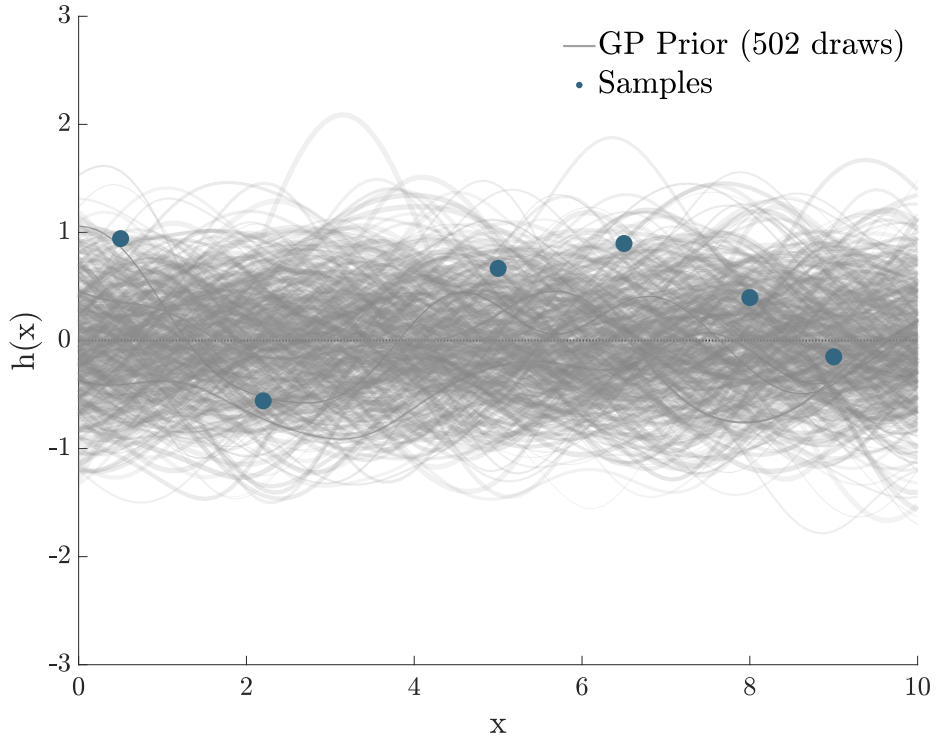

Figure 3.5: Noise-free data samples shown along with the priors drawn formerly.

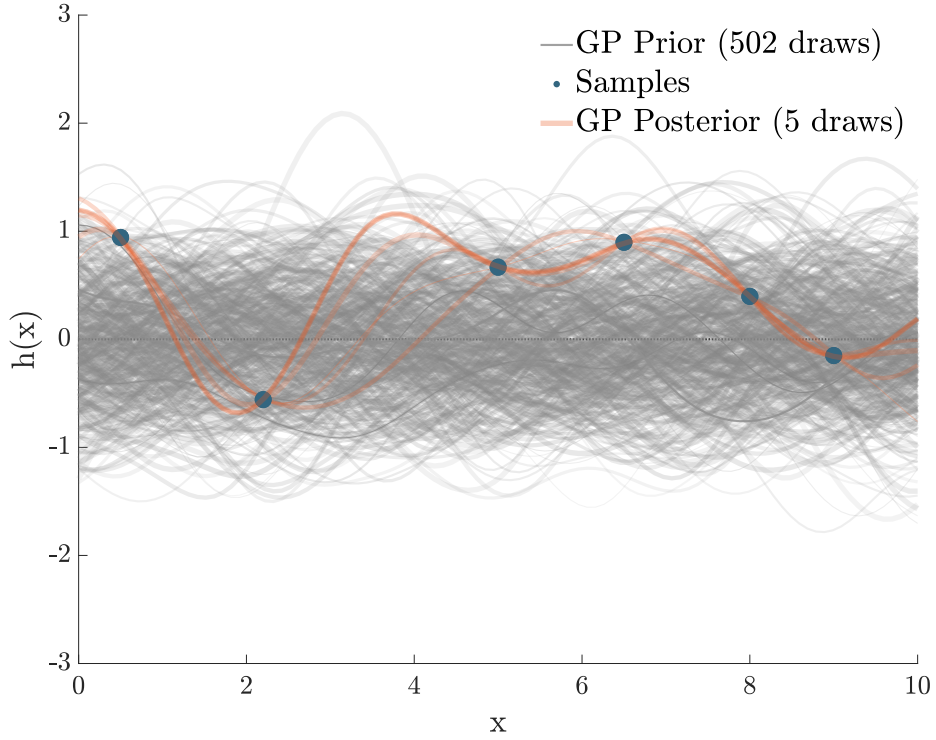

Figure 3.6: By conditioning the priors on the data, 5 posteriors are drawn.

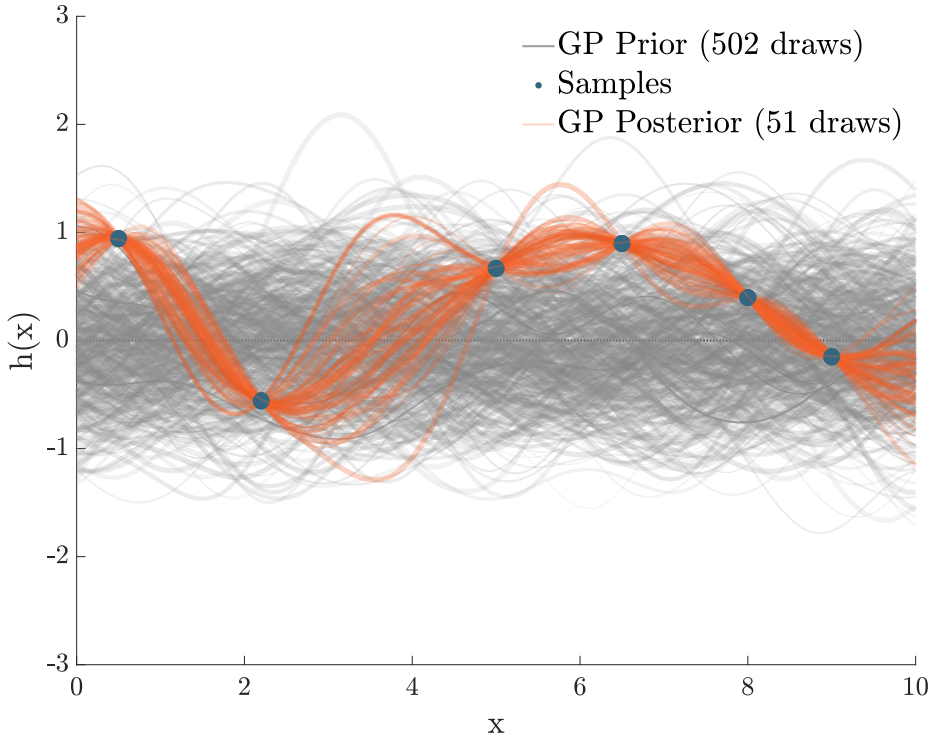

Figure 3.7: Updating the priors with the data, 51 posteriors are drawn.

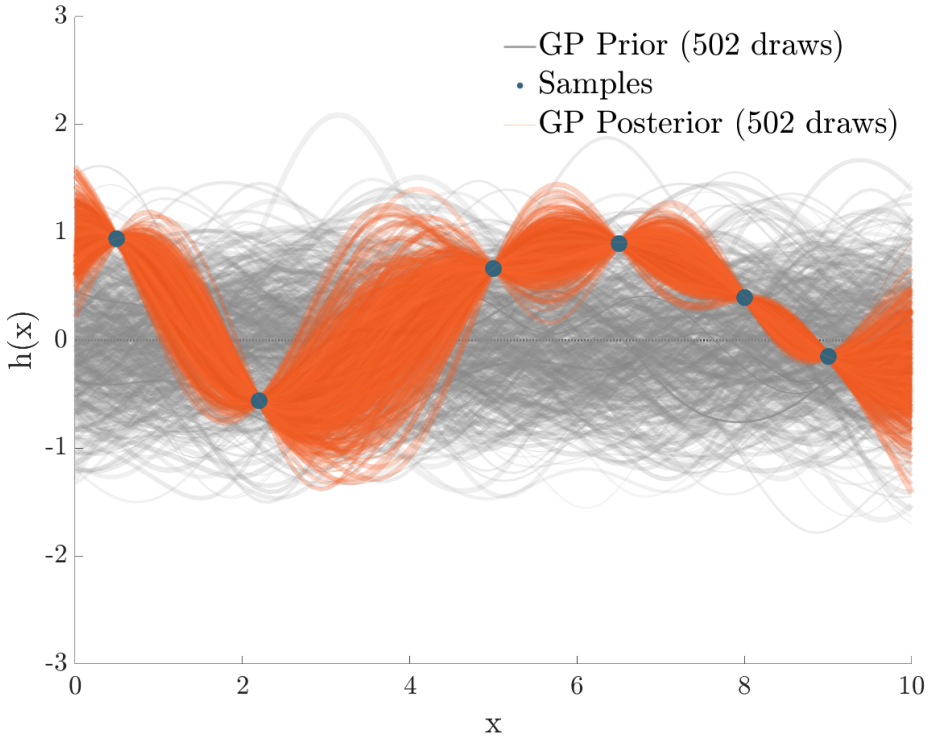

Figure 3.8: By conditioning the priors on the data, 502 posteriors are drawn.

A GP can be completely specified by its mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $h_{\text{gp}}(\mathbf{x})$ as follows,

$$h_{\text{gp}}(\mathbf{x}) \sim \mathcal{GP}\big(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big).$$

Given a set of N data points, with input vectors $\mathbf{x} \in \mathbb{R}^n$, and scalar targets $y \in \mathbb{R}$, we compose the dataset $\mathbf{D}_N = \{\mathbf{X}_N, \mathbf{y}_N\}$, where $\mathbf{X}_N = \{\mathbf{x}_i\}_{i=1}^N$ and $\mathbf{y}_N = \{y_i\}_{i=1}^N$. GPs can compute the posterior mean and variance for an arbitrary deterministic query point $\mathbf{x}_* \in \mathbb{R}^n$, by conditioning on previous measurements. We will later investigate how to handle the case when the query point $\mathbf{x}_*$ is noisy. The posterior mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}$ are given by [59],

$$\mu(\mathbf{x}_*) = \mathbf{k}(\mathbf{x}_*)^\top \, \overline{\mathbf{K}}^{-1} \mathbf{y}_N, \tag{3.1}$$

$$\sigma^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^\top \, \overline{\mathbf{K}}^{-1} \mathbf{k}(\mathbf{x}_*), \tag{3.2}$$

where $\mathbf{k}(\mathbf{x}_*) = \big[k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_N, \mathbf{x}_*)\big]^\top \in \mathbb{R}^N$ is the covariance vector between $\mathbf{X}_N$ and $\mathbf{x}_*$, $\overline{\mathbf{K}} \in \mathbb{R}^{N \times N}$, with entries $[\bar{k}]_{(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j),\ i, j \in \{1, \ldots, N\}$, is the covariance matrix between pairs of input points in $\mathbf{X}_N$, and $k(\mathbf{x}_*, \mathbf{x}_*) \in \mathbb{R}$ is the prior covariance.

Revisiting the problem above, where we are interested in doing inference in the function space, GPs can be used to derive the posterior mean and variance in closed-form. Using (3.1) and (3.2), the posterior mean and 95% confidence region is shown with pointwise mean plus and minus two times the standard deviation for each input value in Figure 3.9.

Using GP regression, we are interested in constructing a safety function for which we assume to have noisy scalar observations. These scalar observations will serve as the safety samples for our problem setting. An advantage of using GPs to synthesize CBFs is the use of kernel or covariance functions for function inference. A well chosen kernel function is differentiable allowing a flexible parameterization of the safety function and its Lie deriva-

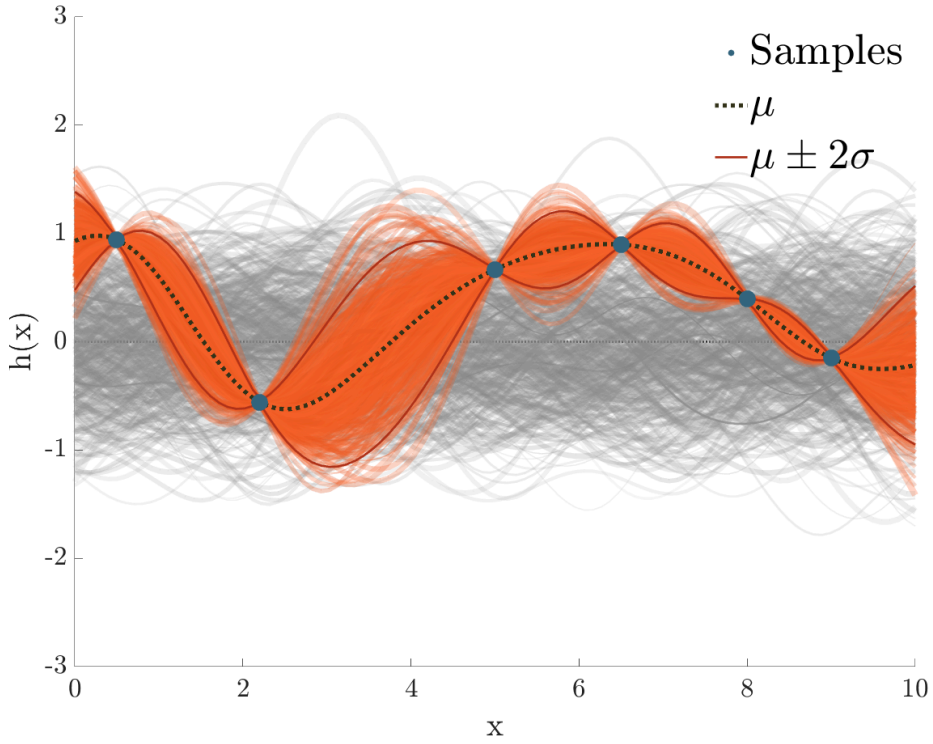

Figure 3.9: The mean and variance can be derived for the posterior distribution using GPs. The mean is shown in black dashed line with $\pm$ 2 standard deviation (bold maroon).

tives as we saw in Section 2.3.1. Although, there are different regression methods such as splines regression, ordinary-least squares regression, or logistic regression [67, 68, 69, 70, 71, 72, 73, 74], GP regression benefits from several advantages. Firstly, GPs can incorporate interpretable priors giving us flexibility in learning the model of interest. When it comes to splines, the order of the spline plays an important role in the regression. Additionally, for standard regression techniques, cross-validation of the data is required to prevent overfitting of the data and/or learning the hyperparameters of interest that accurately model the underlying data. For GP regression, hyperparameters can be directly learned from the data using automatic relevance determination [75, 76] or by maximizing the log-marginal likelihood of the latent underlying function we want to model (see Sections 4.1.2 and 5.2.2). Moreover, spline regression is a special case of GP regression as shown by the work in [77]. As we will see later in Section 4.3.2, we can also exploit sparsity with GPs to reduce data complexity without compromising the interpretability of the data.

3.2 Problem Statement

Consider a control affine system (2.1) is given, with access to its states $\mathbf{x}$, and scalar noisy observations y, that represents a metric for safety. The metric for safety cannot be generalized and therefore is very problem dependent. A distance sensor's readings for obstacle avoidance can be used as a metric for safety or a temperature sensor's readings for determining thermally acceptable regions to traverse. In a similar vein, a LiDAR scan creating $3\mathrm{D}$ point cloud information can be used to detect environmental hazards or a computer vision algorithm providing the decision boundary for safe regions of interest. In all these examples, we can easily sample from the data based on domain knowledge to construct a target metric for safety. *This provides us with the means to construct a valid safety certificate using the data as opposed to hand-designing a safety function which could be limited and requires manual effort along with good domain knowledge intuition.*

***Remark* 3.** *We assume there is a high-level planner or observer, e.g., sensors or computer vision algorithms, providing the necessary data observations. We acknowledge some feature engineering or data sampling may be involved which is very common in practice. These observations represent the safety sample candidates in our problem setting.*

Our objective is to *synthesize a safety function $h_{\mathrm{gp}}(\mathbf{x})$ in a non-parametric manner, from measurements of the system states and safety samples or observations, online and ensure that system (2.1) remains safe.* Data-based methods are ultimately approximations and hence, it is desirable to account for any uncertainty in the estimation of the safety function. This leads to the following candidate function,

$$\underbrace{h_{\mathrm{gp}}(\mathbf{x}(t))}_{\text{overall safety}} := \underbrace{h_{\mathrm{b}}(\mathbf{x}(t);\Theta)}_{\text{safety belief}} - \underbrace{h_{\mathrm{u}}(\mathbf{x}(t);\Theta)}_{\text{safety uncertainty}}. \tag{3.3}$$

The system's overall safety is given by $h_{\mathrm{gp}}(\mathbf{x})$ which has two components; a belief in safety given by $h_{\mathrm{b}}(\mathbf{x})$ and an associated uncertainty given by $h_{\mathrm{u}}(\mathbf{x})$. Intuitively, the safety

belief represents our best estimation of system safety and safety uncertainty represents the uncertainty in the estimation. Ideally, if there is no uncertainty, then the safety belief will perfectly match the overall final safety. Additionally, there are hyperparameters Θ that can alter the relative notion of safety belief and uncertainty.

Problem 3. *Given system* (2.1) *and online (noisy) measurements of the state* $\mathbf{x}_*$*, synthesize* $h_{\mathrm{gp}}(\mathbf{x})$ *with a safety belief and associated uncertainty, conditioned on past states and observations in the dataset given by:* $\mathbf{D}_N = \{\mathbf{X}_N, \mathbf{y}_N\}$*, where* $\mathbf{X}_N = \{\mathbf{x}_i\}_{i=1}^N$ *and* $\mathbf{y}_N = \{y_i\}_{i=1}^N$*, such that system (2.1) is safe.*

To ensure the system remains safe, we need to rectify a given nominal control input $\mathbf{u}_{\mathrm{nom}}$ to its rectified form $\mathbf{u}_{\mathrm{rect}}$ which is then applied to the system (2.1). This is done by making sure that the Lie derivatives of the corresponding candidate safety function satisfies the inequality described in (2.3).

Problem 4. *Given system* (2.1)*, synthesized* $h_{\mathrm{gp}}(\mathbf{x})$ *with safe set* $\mathcal{S}$*, and a nominal control input* $\mathbf{u}_{\mathrm{nom}}$*, design the rectified control input* $\mathbf{u}_{\mathrm{rect}}$ *such that system (2.1) is safe.*

Note that designing the control objective for a data-driven based CBF construction is particularly challenging. Unlike the variance based CBF discussed in Section 2.3, here a complete non-parametric approach is adopted where the data is fully exploited to construct the safe sets and the safety function hypothesis. Now, we additionally need to ensure that the Lie derivatives are computed for the posterior mean of the GP along with the posterior variance. Due to the use of kernels or covariance functions in GPs, we can tackle this requirement in a similar manner as done in Section 2.3.1.

3.3 Proposed Methodology

In this section, we present our proposed approach, where GPs are used for synthesizing the safety function. A key advantage of GPs over other models such as neural networks, radial basis functions or polynomial chaos, lies in its Bayesian non-parametric design. *By*

allowing a flexible prior over functions, GPs give a probabilistic work flow that gives robust posterior estimates in analytical form. This enables a flexible realization for our safety function as well as computing the associated Lie derivatives. The resulting architecture for our framework is shown in Figure 3.10.

3.3.1 Gaussian Control Barrier Function

A GP prior is placed on the desired candidate safety function, $h_{\text{gp}}(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$. Intuitively, this means that the CBF is a function drawn from a GP. By conditioning the prior belief on observed data, a posterior CBF will be derived. In Section 2.3, only the GP posterior variance was used to augment safety uncertainty for a given deterministic CBF. Here, the CBF is a stochastic process drawn from a GP. Note that the new formulation is a far more flexible realization *since the data is used for informing both the safety belief and associated uncertainty*. We operate under the following standard assumptions for GPs.

Assumption 4. *Each observation y_i is corrupted with Gaussian noise, $y_i \sim \mathcal{N}(p_i, \sigma_\omega)$, where p_i is the noise-free safety sample and σ_ω is the observation noise variance.*

The assumption above has practical implications since capturing the safety samples will not always be noise-free. This is a very realistic assumption, since in practice these safety samples can be captured from noisy sensory measurements.

Assumption 5. *Training input states $\mathbf{X}_N$ in the dataset $\mathbf{D}_N$ are noise-free.*

The input training data is considered to be noise-free, however, which is a common practice in machine learning. We first consider the case where the query point $\mathbf{x}_q$ is deterministic, i.e., noise-free. Later, we look at the case when the input query point is also noisy, see Section 3.3.4.

Assumption 6. *The safe set is nonempty with at least one datapoint, the initial state $\mathbf{x}(t_0)$ where t_0 is the initial time, with non-negative safety value $h_{\text{gp}}(\mathbf{x}(t_0)) \in \mathbb{R}_{\geq 0}$, to synthesize $h_{\text{gp}}(\mathbf{x})$.*

We assume that the system (2.1) begins in an initial compact safe set. Safety for h_{gp} is encoded as,

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n \mid h_{\text{gp}}(\mathbf{x}) \geq 0\}, \tag{3.4}$$

$$\partial\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^n \mid h_{\text{gp}}(\mathbf{x}) = 0\}. \tag{3.5}$$

We adopt a similar sampling strategy as done in Section 2.6.1, where a noise-free query state $\mathbf{x}_* \in \mathbb{R}^n$ with a noisy safety sample $y \in \mathbb{R}$ is sampled if,

$$\left\|\mathbf{x}_{(i)} - \mathbf{x}_*\right\| \geq \tau, \qquad i = \{1, \ldots, N\}, \tag{3.6}$$

where $\mathbf{x}_{(i)}$ are training input points, and $\tau \in \mathbb{R}$ is the sampling distance between any two input states. Given assumptions 4, and 5, 6, we now formally define a Gaussian CBF.

Definition 5 (Gaussian Control Barrier Function)**.** *A function $h_{\text{gp}}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is defined as a Gaussian CBF for* (2.1)*, if $h_{\text{gp}}(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}_i, \mathbf{x}_j))$ is a Gaussian process, with a smooth positive semidefinite kernel, $k(\mathbf{x}_i, \mathbf{x}_j) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, and if $\exists$ an extended class-κ function α such that for any $\mathbf{x} \in \mathbb{R}^n$,*

$$\sup_{\mathbf{u} \in \mathbb{R}^m} L_f h_{\text{gp}}(\mathbf{x}) + L_g h_{\text{gp}}(\mathbf{x})\mathbf{u} + \alpha(h_{\text{gp}}(\mathbf{x})) \geq 0. \tag{3.7}$$

***Remark* 4.** *The Gaussian CBF above has attractive properties. A GP prior is placed on the safety candidate function, giving rise to a non-parametric functionality. Thus, the data is used to fully realize the safety function a posteriori. As more data is collected, the overall safety encoded by $h_{\text{gp}}(\mathbf{x})$ changes. Moreover, it has an analytical form for both the safety belief and uncertainty. This enables computing Lie derivatives of $h_{\text{gp}}(\mathbf{x})$ in closed-form.*

The Lie derivatives in (3.7) require taking partial derivatives of h_{gp} with respect to $\mathbf{x}$ which we will discuss later in Section 3.3.2. We propose the Gaussian CBF $h_{\text{gp}}(\mathbf{x})$ that incorporates safety belief and uncertainty *online* using the GP posterior mean (3.1) and

variance (3.2) as follows[1],

$$
\begin{aligned}
h_{\mathrm{gp}}(\mathbf{x}) &:= \mu(\mathbf{x}) - \sigma^2(\mathbf{x}) \\
&= \underbrace{\mathbf{k}(\mathbf{x})^\top \overline{\mathbf{K}}^{-1} \mathbf{y}_N}_{\text{safety belief}} - \Big(\underbrace{k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \overline{\mathbf{K}}^{-1} \mathbf{k}(\mathbf{x})}_{\text{safety uncertainty}} \Big).
\end{aligned} \tag{3.8}
$$

The GP posterior mean represents the belief we have regarding safety whereas the GP posterior variance accounts for safety uncertainty. We require the following theorem to discuss forward invariance properties for the set $\mathcal{S}$ in (3.4)-(3.5).

Theorem 2 (Sample Path Differentiability [78])**.** *A Gaussian process with an isotropic correlation or kernel function that can be expressed in the Schoenberg representation [79], has M^{th}-order mean-square partial derivatives if M moments of the length-scale parameter, l, are finite.*

We first consider an unforced dynamical system given by $\dot{\mathbf{x}} = f(\mathbf{x})$, where $\mathbf{u}(t) = 0, \forall\, t \geq 0$. In this case, the Gaussian CBF will simply be considered as a Gaussian barrier function, since the control input does not appear.

Proposition 1. *Given a system $\dot{\mathbf{x}} = f(\mathbf{x})$ with a nonempty safe set $\mathcal{S}$ as defined by (3.4-3.5) for a Gaussian process h_{gp}, if h_{gp} is a Gaussian barrier function defined on the set $\mathcal{S}$, then $\mathcal{S}$ is forward invariant.*

Proof. First, we observe that h_{gp} uses an infinitely differentiable kernel. Hence, h_{gp} is also infinitely differentiable with respect to $\mathbf{x}$ due to Theorem 2 since the length-scale has infinitely many moments. Since h_{gp} is a Gaussian CBF and infinitely differentiable, then the inequality $L_f h_{\mathrm{gp}}(\mathbf{x}) \geq -\alpha(h_{\mathrm{gp}}(\mathbf{x}))$, is satisfied. Given Assumption (6), the set $\mathcal{S}$ is nonempty, for any $\mathbf{x} \in \partial\mathcal{S}$, $h_{\mathrm{gp}}(\mathbf{x}) = 0$ holds. As a result, $\alpha(h_{\mathrm{gp}}(\mathbf{x})) = 0$ which gives $L_f h_{\mathrm{gp}}(\mathbf{x}) \geq 0 \implies \dot{h}_{\mathrm{gp}} \geq 0$. By applying Nagumo's theorem [7], which states that for

[1]We can employ weights, w_μ and w_{σ^2}, to the posterior mean and variance respectively in order to adjust safety based on the application. For the sake of simplicity, we consider the weights to be unity in the problem statement.

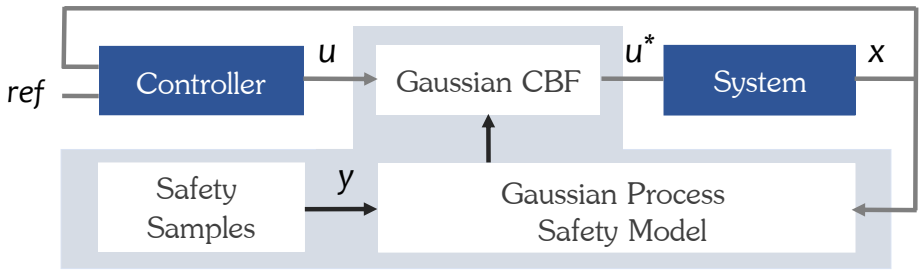

Figure 3.10: Gaussian CBF incorporates safety belief and uncertainty based on past measurement data.

any C^1 function h_{gp}, the condition $\dot{h}_{\text{gp}} \geq 0$ on $\partial\mathcal{S}$ is necessary and sufficient for the set $\mathcal{S}$ to be forward invariant, completes the proof. □

***Remark* 5.** *For the case when the kernel is only M times differentiable, and not infinitely differentiable, we require that $M > 2\rho$, where $\rho \in \mathbb{N}$ is the relative degree of the system. The proof above holds trivially for an M times differentiable kernel using Theorem 2.*

We are interested in ensuring forward invariance of $\mathcal{S}$ characterized by h_{gp} for the system defined by (2.1). The admissible control space for the Gaussian CBF is given by,

$$\text{K}_{\text{gcbf}} = \{\mathbf{u} \in \mathbb{R}^m \big| L_f h_{\text{gp}}(\mathbf{x}) + L_g h_{\text{gp}}(\mathbf{x})\mathbf{u} + \alpha(h_{\text{gp}}(\mathbf{x})) \geq 0\}. \tag{3.9}$$

Proposition 2. *Given a Gaussian CBF $h_{\text{gp}}(\mathbf{x}) : \mathcal{S} \to \mathbb{R}$ defined by (3.7), where $\mathcal{S}$ is nonempty (3.4), any Lipschitz continuous controller $\mathbf{u} \in \mathbb{R}^m$, that satisfies (3.9) for any $\mathbf{x} \in \mathbb{R}^n$, renders $\mathcal{S}$ forward invariant for the system (2.1).*

Proof. h_{gp} is a Gaussian process with an infinitely differentiable kernel. Using Theorem 2, h_{gp} is also infinitely differentiable and is, therefore, smooth. Since h_{gp} satisfies (3.9), we have $L_f h_{\text{gp}}(\mathbf{x}) + L_g h_{\text{gp}}(\mathbf{x})\mathbf{u} \geq -\alpha(h_{\text{gp}}(\mathbf{x}))$. Using Theorem 1, the proof is complete. □

3.3.2 Lie Derivatives of Gaussian CBF

The Gaussian CBF uses kernels for determining the safety belief and associated uncertainty in the state space. As stated earlier, we use the SE kernel (2.4) which is an infinitely dif-

ferentiable function. Later, we will see an example of using kernels from the Matérn class. Computing its Lie derivatives is necessary for rectifying the control input and ensuring forward invariance for the system in the safe set. First, we take the partial derivative of (3.8) with respect to $\mathbf{x}$ at a query point $\mathbf{x}_*$,

$$\begin{aligned}\left.\frac{\partial h_{\mathrm{gp}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} &= \left.\frac{\partial \mu(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} - \left.\frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} \\ &= \mathbf{y}_N^\top \overline{\mathbf{K}}^{-1} \left.\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} + 2\mathbf{k}(\mathbf{x}_*)^\top \overline{\mathbf{K}}^{-1} \left.\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*}. \end{aligned} \tag{3.10}$$

The kernel derivative in (3.10) is given by (2.9). Now, we can compute the Lie derivatives of $h_{\mathrm{gp}}(\mathbf{x})$ by taking its time derivative as follows,

$$\begin{aligned}\dot{h}_{\mathrm{gp}}(\mathbf{x}) &= \frac{\partial h_{\mathrm{gp}}(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}) + \frac{\partial h_{\mathrm{gp}}(\mathbf{x})}{\partial \mathbf{x}} g(\mathbf{x})\mathbf{u} \\ &= L_f h_{\mathrm{gp}}(\mathbf{x}) + L_g h_{\mathrm{gp}}(\mathbf{x})\mathbf{u}, \end{aligned} \tag{3.11}$$

where (3.10) is used in the Lie derivatives, $L_f h_{\mathrm{gp}}(\mathbf{x})$ and $L_g h_{\mathrm{gp}}(\mathbf{x})$.

3.3.3 Online Safety Control

Based on assumption 2, we are given a nominal control input $\mathbf{u}_{\mathrm{nom}} \in \mathbb{R}^m$ designed as the feedback policy for system (2.1). This control policy may not restrict the solution of system (2.1) inside the safe set. An online quadratic program (QP) rectifies $\mathbf{u}_{\mathrm{nom}}$ whose constraints are given by the Lie derivatives in (3.11) [8]. The QP optimization routine is set up as follows:

Gaussian CBF-QP: *Control Input modification*

$$\begin{aligned}\mathbf{u}_{\mathrm{rect}} = \underset{\mathbf{u} \in \mathbb{R}^m}{\arg\min} \frac{1}{2}\left\|\mathbf{u} - \mathbf{u}_{\mathrm{nom}}\right\|^2 \ \text{s.t.} \\ L_f h_{\mathrm{gp}}(\mathbf{x}) + L_g h_{\mathrm{gp}}(\mathbf{x})\mathbf{u} + \alpha(h_{\mathrm{gp}}(\mathbf{x})) \geq 0, \end{aligned} \tag{3.12}$$

---

where $\mathbf{u}_{\mathrm{rect}}$ is the rectified control input, and (3.8), (3.10), (3.11) are used in the QP con-

Algorithm 2 Gaussian CBF Synthesis & Safe Control

Input: GP PRIOR $h_{\text{gp}}(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$
SYSTEM (2.1)
NOMINAL INPUT $\mathbf{u}_{\text{nom}}$
Output: RECTIFIED INPUT $\mathbf{u}_{\text{rect}}$

procedure SAFECONTROL
  SAMPLE $\mathbf{X}_N \leftarrow \mathbf{x}_*$ & $\mathbf{y}_N \leftarrow y$ using (3.6)
  SYNTHESIZE $h_{\text{gp}}(\mathbf{X}_N, \mathbf{y}_N)$ using (3.8)
  COMPUTE $\frac{\partial h_{\text{gp}}(\mathbf{x})}{\partial \mathbf{x}}$ using (3.10)
  SETUP QP constraint using (3.11)
  RECTIFY $\mathbf{u}_{\text{nom}}$ using (3.19)
return $\mathbf{u}_{\text{rect}}$

straint. The QP constraint above ensures that the nominal control is followed as long as the safety condition is not violated, i.e., $h_{\text{gp}}(\mathbf{x}) \geq 0$. When approaching the boundary of the safe set, i.e., $h_{\text{gp}} \to 0$, the QP rectifies $\mathbf{u}_{\text{nom}}$ minimally to $\mathbf{u}_{\text{rect}}$. By rectifying the control policy, the system is guaranteed to remain forward invariant for the safe set $\mathcal{S}$ due to Proposition 2. When solving for the QP, every term in the constraint is simply a numerical value except for the decision variable, the control input, which is rectified. Therefore, $h_{\text{gp}}(\mathbf{x})$ being highly non-linear and non-convex does not affect finding the rectified control input. The algorithm for computing safe control input from the synthesized Gaussian CBF is shown in Algorithm 2.

***Remark* 6.** *Note that due to the non-parametric nature of the Gaussian CBF, the algorithm above can be treated as a blackbox routine. This is a beneficial property since, if a traditional CBF is altered, then the corresponding Lie derivatives also change explicitly in their form. However, in the Gaussian CBF, the structure of the Lie derivatives remains the same, i.e, the partial derivatives are explicitly agnostic to the underlying CBF. It is characterized only by the data and the dynamical system.*

3.3.4 Gaussian CBF with Noisy Query State

Here, we extend Gaussian CBFs to handle the case when the query state, $\underline{\mathbf{x}}_*$, is stochastic and therefore a random variable.

Assumption 7. *The query state $\mathbf{x}_*$ is Gaussian distributed, $\underline{\mathbf{x}}_* \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_*}, \boldsymbol{\Sigma}_{\mathbf{x}_*})$, where $\boldsymbol{\mu}_{\mathbf{x}_*}$ is the mean and $\boldsymbol{\Sigma}_{\mathbf{x}_*}$ is its noise covariance matrix.*

This has practical significance because accurate estimates of these states are required to generate safe control actions. In practice, however, measurement uncertainty is pervasive leading to error in the state estimates, thus degrading the safety behavior. As a result, we need to modify the posterior predictions of the GP in order to account for this noise. We are now operating under assumption 7 along with assumptions 4 and 5.

The predictive equations for a Gaussian test input have been looked at before [80, 81]. Generally, if a Gaussian input is multiplied with the nonlinear GP predictive distribution, the resulting distribution is non-Gaussian,

$$p(h_{\text{gp}}(\mathbf{x}_*)) = \int p(h_{\text{gp}}(\mathbf{x}_*)|\mathbf{x}_*)p(\mathbf{x}_*|\boldsymbol{\mu}_{\mathbf{x}_*}, \boldsymbol{\Sigma}_{\mathbf{x}_*})\mathrm{d}\mathbf{x}_*. \tag{3.13}$$

In (3.13), the first probability $p(h_{\text{gp}}(\mathbf{x}_*)|\mathbf{x}_*)$ is the distribution of h_{gp} for a given query point $\mathbf{x}_*$, and the second probability is the noisy input distribution $\underline{\mathbf{x}}_* \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_*}, \boldsymbol{\Sigma}_{\mathbf{x}_*})$. By utilizing the respective density functions in (3.13), we get,

$$\begin{aligned} p(h_{\text{gp}}(\mathbf{x}_*)) = \int & \frac{1}{\sqrt{(2\pi)^n \det\left(\mathbf{K}_*\right)}} \exp\left(-\frac{\left(h_{\text{gp}}(\mathbf{x}_*) - \mu(\mathbf{x}_*)\right)^\top \left(h_{\text{gp}}(\mathbf{x}_*) - \mu(\mathbf{x}_*)\right)}{2\sigma^2(\mathbf{x}_*)} \right) \\ & \frac{1}{\sqrt{(2\pi)^n \det\left(\boldsymbol{\Sigma}_{\mathbf{x}_*}\right)}} \exp\Big(-\frac{1}{2}\left(\mathbf{x}_* - \boldsymbol{\mu}_{\mathbf{x}_*}\right)^\top \boldsymbol{\Sigma}_{\mathbf{x}_*}^{-1}\left(\mathbf{x}_* - \boldsymbol{\mu}_{\mathbf{x}_*}\right)\Big)\mathrm{d}\mathbf{x}_*, \end{aligned}$$

where $\mathbf{K}_* \in \mathbb{R}^{n_{\text{test}} \times n_{\text{test}}}$ is the posterior covariance matrix for all input points. The integral above involves integrating an exponential term with an inverse matrix operation. Furthermore, the matrix nonlinearly depends on the integration parameter $\mathbf{x}_*$, thereby making it very difficult to solve this integral analytically. To see a pictorial representation of this phenomenon, see Figure 3.11. The input has a Gaussian distribution given by $\underline{x}_* \sim \mathcal{N}(0, 0.4^2)$. We arbitrarily select 6 training samples. The resultant GP distribution is non-Gaussian as indicated in the bottom plot of Figure 3.11. This poses an issue since we want the posterior

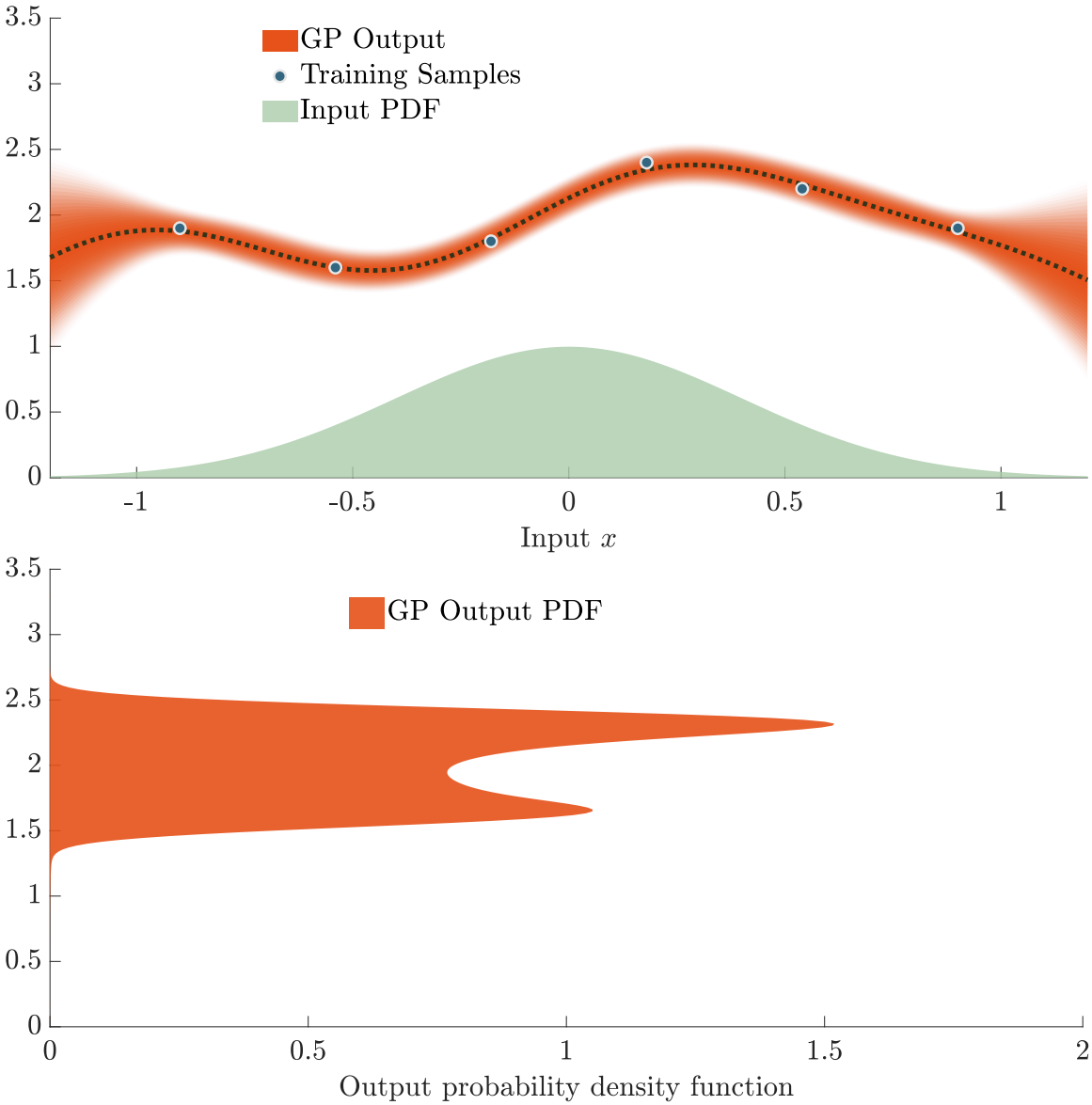

Figure 3.11: Example of 1-d GP where the input test point is noisy, $\underline{x}_* \sim \mathcal{N}(0, 0.4^2)$. The GP output and the input distributions are both shown in the top figure. The GP output pdf is plotted (bottom) using (3.13) which is non-Gaussian due to the stochastic input point.

distribution to be Gaussian. As a result, moment matching is used to derive the posterior predictions.

To determine the moments of the predictive function value, both the query distribution and the distribution of the function given by the GP are averaged over. For the SE kernel, the posterior mean and variance can be computed for the predictive distribution in (3.13)

in closed-form[2] [81]. By using the law of iterated expectations, the posterior mean with a noisy query point $\underline{\mathbf{x}}_*$ is given as follows [81],

$$\mu(\underline{\mathbf{x}}_*) = \mathbf{q}(\underline{\mathbf{x}}_*)^\top \, \overline{\mathbf{K}}^{-1} \mathbf{y}_N, \tag{3.14}$$

where $\mathbf{q} = [q_i, \ldots, q_N]^\top \in \mathbb{R}^N$ with each q_i representing the expected covariance between $h_{\text{gp}}(\underline{\mathbf{x}}_*)$ and $h_{\text{gp}}(\mathbf{x}_i)$,

$$q_i(\mathbf{x}_i, \underline{\mathbf{x}}_*) := \int k(\mathbf{x}_i, \underline{\mathbf{x}}_*) \mathcal{N}(\underline{\mathbf{x}}_* | \boldsymbol{\mu}_{\mathbf{x}_*}, \boldsymbol{\Sigma}_{\mathbf{x}_*}) \mathrm{d}\underline{\mathbf{x}}_*$$

where $\mathbf{R} = \boldsymbol{\Sigma}_{\mathbf{x}_*} + \mathbf{L}^2 \in \mathbb{R}^{n \times n}$. It is interesting to note the case for a deterministic query point $\mathbf{x}_*$, where $\boldsymbol{\Sigma}_{\mathbf{x}_*} = \mathbf{0}$ and $\boldsymbol{\mu}_{\mathbf{x}_*} = \mathbf{x}_*$. On comparing (3.14) with (3.1), the posterior mean for the noisy input results in the same posterior mean for the noise-free input, since $q_i(\mathbf{x}_i, \mathbf{x}_*)$ collapses to $k_i(\mathbf{x}_i, \mathbf{x}_*)$ in (2.4). Effectively, the noise-free input point is a special case of the noisy posterior prediction with $\boldsymbol{\Sigma}_{\mathbf{x}_*} = \mathbf{0}$ and $\boldsymbol{\mu}_{\mathbf{x}_*} = \mathbf{x}_*$.

For details on the derivation of the predictive variance for the noisy test point, see [81]. Here, we simply state the posterior predictive variance which is as follows,

$$\sigma^2(\underline{\mathbf{x}}_*) = \sigma_f^2 - \mathrm{tr}\big(\overline{\mathbf{K}}^{-1}\mathbf{V}\big) + \boldsymbol{\beta}^\top \big(\mathbf{V}\boldsymbol{\beta} - \mathbf{q}\big), \tag{3.15}$$

with $\boldsymbol{\beta} = \overline{\mathbf{K}}^{-1}\mathbf{y}_N \in \mathbb{R}^N$, and the entries of $\mathbf{V} \in \mathbb{R}^{N \times N}$ are given by,

$$v_{ij} = \frac{k(\mathbf{x}_i, \boldsymbol{\mu}_{\mathbf{x}_*}) k(\mathbf{x}_j, \boldsymbol{\mu}_{\mathbf{x}_*})}{|2\boldsymbol{\Sigma}_{\mathbf{x}_*}\mathbf{L}^{-2} + \mathbf{I}_n|^{\frac{1}{2}}} \exp\big((\mathbf{z}_{ij} - \boldsymbol{\mu}_{\mathbf{x}_*})^\top \mathbf{T} (\mathbf{z}_{ij} - \boldsymbol{\mu}_{\mathbf{x}_*})\big),$$

where $\mathbf{T} := (\boldsymbol{\Sigma}_{\mathbf{x}_*} + \frac{1}{2}\mathbf{L}^2)^{-1}\boldsymbol{\Sigma}_{\mathbf{x}_*}\mathbf{L}^{-2} \in \mathbb{R}^{n \times n}$ and each entry in $\mathbf{z}_{ij} := \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j) \in \mathbb{R}^n$. As seen in (3.14) and (3.15), both the predictive mean and variance explicitly depend on

[2]This statement holds true for all kernels, in particular the SE, polynomial, and trigonometric kernels, if the integral of the kernel multiplied with a Gaussian distribution can be solved analytically.

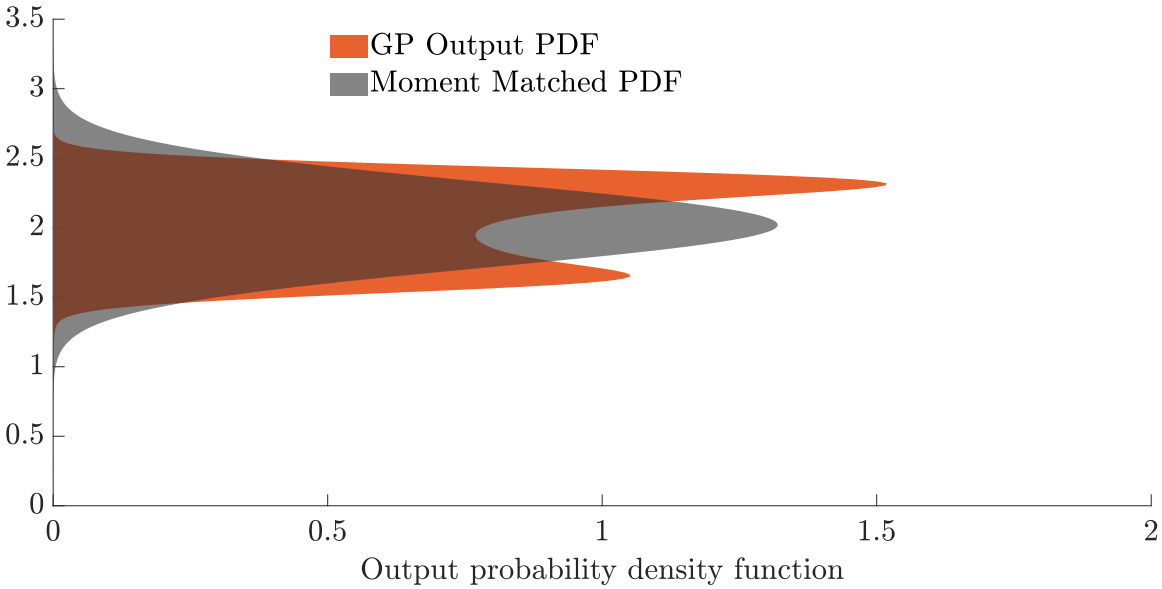

Figure 3.12: Through moment-matching, the posterior mean (3.14) and posterior variance (3.15) are derived for noisy test input $\underline{\mathbf{x}}_*$. The resulting moment-matched distribution is Gaussian as desired.

the mean $\boldsymbol{\mu}_{\mathbf{x}_*}$ and the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}_*}$ of the Gaussian distributed query state $\underline{\mathbf{x}}_*$. Using (3.14) and (3.15), we get the moment-matched Gaussian distribution as shown in Figure 3.12. In the example, $\underline{\mathbf{x}}_*$ has a large variance of 0.4^2, resulting in a non-Gaussian distribution for the GP output. In practice, the noise variance could be smaller, therefore resulting in a distribution which is closer to being a Gaussian.

After deriving analytically the posterior mean and variance for a noisy input, we are now ready to construct the Gaussian CBF. The Gaussian CBF for a noisy query point $\underline{\mathbf{x}}_*$ is then given by,

$$h_{\text{gp}}(\underline{\mathbf{x}}_*) := \mu(\underline{\mathbf{x}}_*) - \sigma^2(\underline{\mathbf{x}}_*). \tag{3.16}$$

In order to compute the Lie derivatives, the partial derivative of (3.16) with respect to $\boldsymbol{\mu}_{\mathbf{x}_*}$

is then derived as follows,

$$\left.\frac{\partial \mu(\underline{\mathbf{x}}_*)}{\partial \underline{\mathbf{x}}_*}\right|_{\boldsymbol{\mu}_{\mathbf{x}_*}} = \mathbf{y}_N^\top \, \overline{\mathbf{K}}^{-1} \left.\frac{\partial \mathbf{q}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\boldsymbol{\mu}_{\mathbf{x}_*}} \tag{3.17}$$

$$\left[\left.\frac{\partial \sigma^2(\underline{\mathbf{x}}_*)}{\partial \underline{\mathbf{x}}_*}\right|_{\mu_k}\right] = \sum_{i,j=1}^{N} \left(\frac{-1}{\bar{k}_{ij}} + \beta_i \beta_j\right) \left.\frac{\partial v_{ij}}{\partial x_k}\right|_{\mu_k} - \sum_{i=1}^{N} \beta_i \left.\frac{\partial q_i}{\partial x_k}\right|_{\mu_k}, \tag{3.18}$$

where $k = \{1, \ldots, n\}$, x_k and μ_k are the k^{th} entries of $\underline{\mathbf{x}}_*$ and $\boldsymbol{\mu}_{\mathbf{x}_*}$ respectively. β_i and β_j are the i^{th} and j^{th} entries of $\boldsymbol{\beta}$, with $i, j = \{1, \ldots, N\}$, and $\bar{k}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}$. We emphasize that $\mathbf{V}$ is a symmetric matrix which allows combining the summation with the choice of indices in (3.18). The Lie derivatives can then be computed using the equations above, similar to (3.11), to achieve safe constrained control by setting up a QP,

---

Gaussian CBF-QP with noisy input state: *Control Input modification*

$$\begin{aligned} \mathbf{u}_{\text{rect}} = \underset{\mathbf{u} \,\in\, \mathbb{R}^m}{\arg\min} \, \frac{1}{2} \left\| \mathbf{u} - \mathbf{u}_{\text{nom}} \right\|^2 \;\; \text{s.t.} \\ L_f h_{\text{gp}}(\underline{\mathbf{x}}_*) + L_g h_{\text{gp}}(\underline{\mathbf{x}}_*) \mathbf{u} + \alpha(h_{\text{gp}}(\underline{\mathbf{x}}_*)) \geq 0. \end{aligned} \tag{3.19}$$

3.4 Application Test Case: Safe Control on 3D Quadrotor

To demonstrate the efficacy of our method, we implement our proposed technique on a quadrotor system using a Crazyflie 2.1. Previously, we demonstrated safe expansion on the Crazyflie quadrotor using variance-based CBFs. Now, we utilize Gaussian CBFs where the safe sets are constructed using data fully. We run 3 separate experiments using Gaussian CBF on the quadrotor Crazyflie: (a) safety constrained control for arbitrary safe sets (b) exploring the state space safely and synthesizing the safe set online, and (c) safety constrained control in the presence of noisy states and comparing the performance with regular CBFs.

3.4.1 Online Control Rectification

We use the same quadrotor dynamics and setpoints as discussed in 2.5.1 and 2.5.2. Since Gaussian CBFs also use the posterior mean in addition to the posterior variance, we present the necessary Lie derivatives here for the system (2.19) with relative degree $\rho = 2$.

$$
\begin{aligned}
L_f h_{\text{gp}}(\mathbf{x}) &= \Big(\nabla\mu(\mathbf{x}) - \nabla\sigma^2(\mathbf{x})\Big)^\top f(\mathbf{x}), \\
L_f^2 h_{\text{gp}}(\mathbf{x}) &= f(\mathbf{x})^\top \Big(\mathbf{H}_\mu(\mathbf{x}) - \mathbf{H}_{\sigma^2}\Big) f(\mathbf{x}) + \Big(\nabla\mu(\mathbf{x}) - \nabla\sigma^2(\mathbf{x})\Big)^\top \cdot \nabla f(\mathbf{x}) \cdot f(\mathbf{x}), \\
L_g L_f h_{\text{gp}}(\mathbf{x}) &= f(\mathbf{x})^\top \Big(\mathbf{H}_\mu(\mathbf{x}) - \mathbf{H}_{\sigma^2}(\mathbf{x})\Big) g(\mathbf{x}) + \Big(\nabla\mu(\mathbf{x}) - \nabla\sigma^2(\mathbf{x})\Big)^\top \cdot \nabla f(\mathbf{x}) \cdot g(\mathbf{x}),
\end{aligned}
$$

where $\nabla\mu(\mathbf{x}) = \frac{\partial\mu(\mathbf{x})}{\partial\mathbf{x}}^\top$ and $\nabla\sigma^2(\mathbf{x}) = \frac{\partial\sigma^2(\mathbf{x})}{\partial\mathbf{x}}^\top$ are the gradients of GP mean and variance in (3.10) and $\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}$ is the Jacobian of $f(\mathbf{x})$. $\mathbf{H}_\mu(\mathbf{x})$ and $\mathbf{H}_{\sigma^2}(\mathbf{x})$ are the Hessians of GP mean and variance given by,

$$
\begin{aligned}
\mathbf{H}_\mu(\mathbf{x}) &= \bigg(\sum_i^N a_i \frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}\bigg), \\
\mathbf{H}_{\sigma^2}(\mathbf{x}) &= -2\nabla\mathbf{k}(\mathbf{x})\overline{\mathbf{K}}^{-1}\nabla\mathbf{k}(\mathbf{x})^\top - 2\bigg(\sum_i^N b_i \frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}\bigg),
\end{aligned}
$$

where a_i is the i^{th} entry of $\mathbf{y}_N^\top\overline{\mathbf{K}}^{-1} \in \mathbb{R}^{1\times N}$, b_i is the i^{th} entry of $\mathbf{k}(\mathbf{x}_*)^\top\overline{\mathbf{K}}^{-1} \in \mathbb{R}^{1\times N}$, $\nabla\mathbf{k}(\mathbf{x}) = \frac{\partial\mathbf{k}(\mathbf{x})}{\partial\mathbf{x}}^\top \in \mathbb{R}^{n\times N}$, and $\frac{\partial^2\mathbf{k}_{(i)}(\mathbf{x})}{\partial\mathbf{x}^2}$ is the partial derivative of (2.9) with respect to $\mathbf{x}$. For the case when the query state is noisy, $\underline{\mathbf{x}}_*$, we use the predictive mean and variance (3.16) as described in Section 3.3.4. Similarly, the corresponding partial derivatives are used to derive the Jacobians and Hessians from (3.17)-(3.18) to compute the Lie derivatives for the noisy query state. Given the nominal control input $\mathbf{u}_{\text{nom}} \in \mathbb{R}^3$ in (2.19), the QP below rectifies $\mathbf{u}_{\text{nom}}$ into $\mathbf{u}_{\text{rect}} \in \mathbb{R}^3$,

where $\mathcal{K} = [k_1\ k_2]^\top \in \mathbb{R}^2$ is the coefficient gain vector, and $\mathcal{H} = [L_f h_{\text{gp}}(\mathbf{x})\ h_{\text{gp}}(\mathbf{x})]^\top \in \mathbb{R}^2$ is the Gaussian Lie derivative vector. The rectified input $\mathbf{u}_{\text{rect}}$ is used to compute the rectified setpoints using (2.23), (2.24), (2.25) which are then ultimately sent to the quadrotor.

Gaussian CBF-QP: *Quadrotor Input modification*

$$\mathbf{u}_{\text{rect}} = \underset{\mathbf{u} \in \mathbb{R}^3}{\arg\min} \frac{1}{2}\left\|\mathbf{u} - \mathbf{u}_{\text{nom}}\right\|^2 \text{ s.t.} \quad (3.20)$$
$$L_f^2 h_{\text{gp}}(\mathbf{x}) + L_g L_f h_{\text{gp}}(\mathbf{x})\mathbf{u} + \mathcal{K}^\top \mathcal{H} \geq 0,$$

3.5 Hardware Experimental Verification

In this section, we discuss the implementation of our method on a hardware quadrotor. We test our proposed formulation in three different scenarios. In the first setting, we demonstrate safe constrained control, where the Gaussian CBF is used to formulate the candidate function. These safe sets are arbitrarily designed and are not limited to taking any convex shape. For the second demonstration, we synthesize the safety function online by exploring the state space while avoiding static collisions. The quadrotor performs safe control within the constructed Gaussian CBF. And for the final scenario, we revisit the constrained control problem for a given candidate function, but in the presence of noisy position states. We compare the safe controlled behavior with a regular CBF. All experiments can be seen here: `https://youtu.be/HX6uokvCiGk`.

3.5.1 Experiment Setup

We use the Crazyflie 2.1 as the hardware quadrotor. State estimation is performed onboard with the help of an external low-cost lighthouse positioning system [65]. All computations are done remotely on a ground station equipped with an Intel i7-9800X at 4.4GHz processor and 16 GB RAM. The `crazyflie_ros` API is used to communicate for interprocess communication, subscribing to pose information, and publishing setpoints over the Crazyradio PA USB dongle [66]. Positions and velocities are collected at 100Hz with a data capacity set to 300 samples. Gaussian CBF synthesis and rectification routine (4.31) are run on a parallel thread at 50Hz where solving the QP takes under 5ms. Nominal setpoint commands are sent to the Crazyflie at 100Hz with the help of a Logitech joystick

controller, which acts as the nominal controller in the QP formulation.

3.5.2 Scenario A : Safe Control for Arbitrary Safe Sets

The objective in this scenario is to demonstrate safe constrained control for any given arbitrary safe set using the Gaussian CBF formulation. We assume a high-level observer or planner provides a map from which we can sample (un)safe locations. For instance, take the example of a satellite view for a street or the indoor map of a warehouse unit, where the goal is to navigate an autonomous agent safely and provide safety specifications at the planning phase. The configuration space for position safety in such settings cannot be designed by hand effectively. By using the data driven design of Gaussian CBFs, we can construct safe sets based on the dataset allowing flexible realizations of safe sets based on information from a high-level planner or observer.

In this scenario, we construct a safety map in $2\mathrm{D}$, where the domain is chosen to be $[-0.35, 0.35]$ along each lateral axis, x and y. We uniformly sample, $N = 200$, (x, y) input points. The safety sample for each input coordinate is drawn from a uniform distribution, $y \sim \mathcal{U}(a, b)$, where $a = -1.0$ and $b = 2.0$ are the lowest and highest values respectively of the distribution. This gives a discrete $2\mathrm{D}$ safety map, where for each of the 200 (x, y) coordinates, we have an associated target safety sample. The safety maps are synthesized once and do not change during the experiment, so the sampling distance is set to $\tau = 0$. The hyperparameters are arbitrarily chosen to generate arbitrary safety maps: $\sigma_f = 1, \sigma_\omega = 0.01, \mathbf{L} = \mathrm{diag}(0.1, 0.1)$. The Gaussian CBF characterizes the posterior safety map as follows,

$$h_{\mathrm{gp}}(\mathbf{x}) := \mu(\mathbf{x}) - 4\sigma^2(\mathbf{x}). \tag{3.21}$$

We generate 3 arbitrary safe sets and run 3 separate experiments with the quadrotor always starting in the safe set as shown in Figure 3.15. For each experiment, we plot the

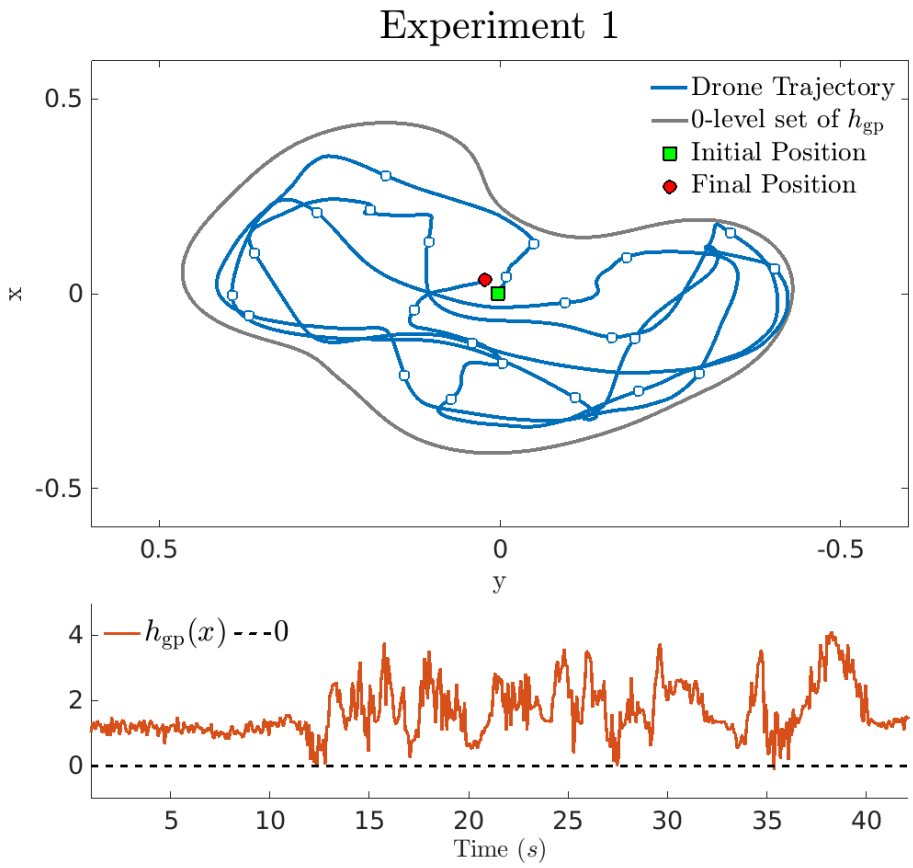

Figure 3.13: Experiment 1 uses an arbitrary safe set generated using the Gaussian CBF h_{gp} for the Crazyflie to navigate in. The initial (■), mid-flight (○), and final (●) quadrotor positions are shown. The 0-level contour line is marked (bold gray).

flight trajectory of the quadrotor, its initial and final positions, and the 0-level set of h_{gp}. The plot of $h_{\text{gp}}(t)$ is also shown for each experimental run. First, we point out that the data generates arbitrary non-convex safe sets. The posterior mean represents the safety belief in h_{gp}, whereas the posterior variance quantifies the notion of safety for regions in the state space where we have few or no samples. We can thus generate very safe realizations of candidate CBFs with high probabilistic bounds. The flight trajectory of the quadrotor always remains inside the safe set based on the QP formulation in (4.31). This can be verified by looking at the $h_{\text{gp}}(t)$ for each experiment, which is always non-negative.

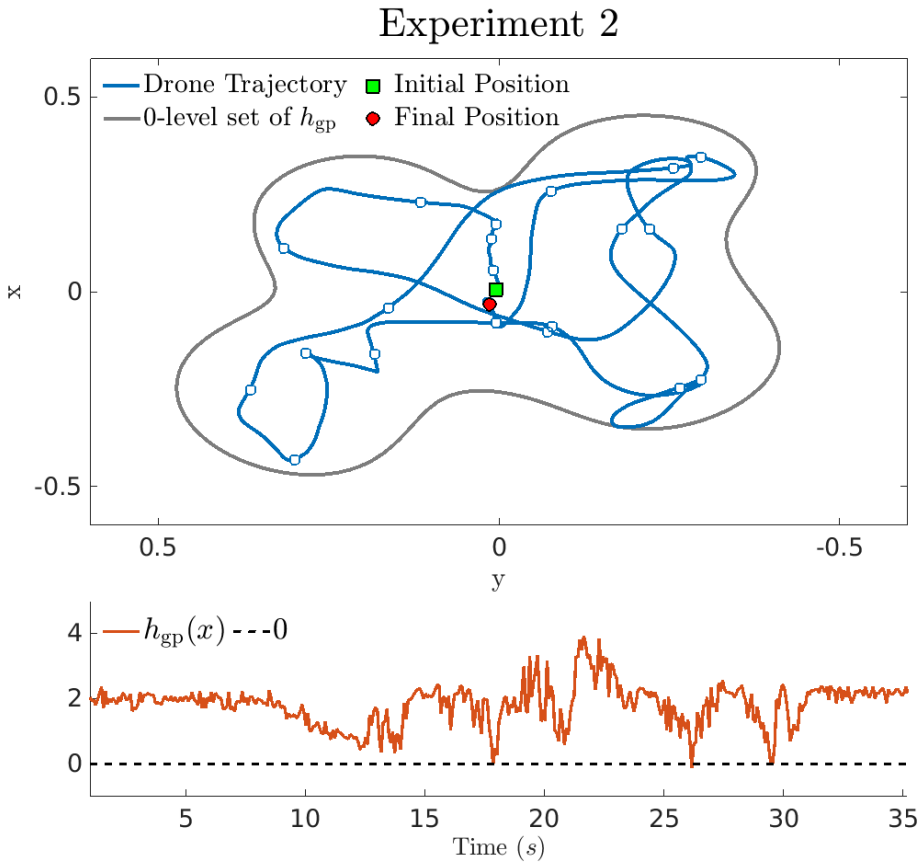

Figure 3.14: Experiment 2 uses another arbitrary safe set generated using the Gaussian CBF h_{gp} for safe control. The temporal plot of h_{gp} for both the experiments show that the quadrotor always remains inside the safe set.

3.5.3 Scenario B : Online Synthesis of Safe Set with Obstacle Avoidance

The objective is to synthesize the safety function online by exploring the state space and avoiding collisions. This has great practical significance in safe navigation since onboard sensors are limited in collecting data only within their local proximity. Therefore, we cannot know a priori the complete safety map. Moreover, the sensed data will also need to alter the safety decision boundary online. With the help of Gaussian CBFs, we can incrementally change the safe set as more data is collected. This allows expansion of the safe set in a non-convex manner, which is required in many practical scenarios involving unstructured environments.

In this scenario, $h_{\text{gp}}(\mathbf{x})$ denotes the distance between the quadrotor and the obstacle.

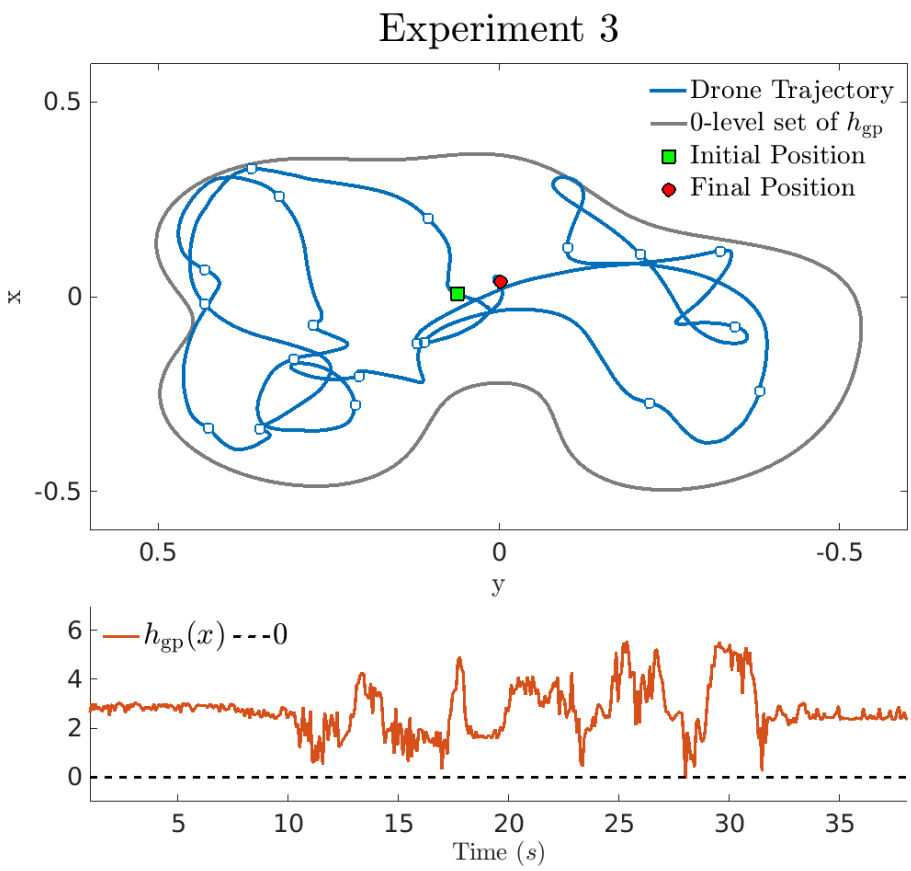

Figure 3.15: Experiment 3 run once again for quadrotor navigation using the Gaussian CBF h_{gp}. In all 3 experiments, the temporal plots of h_{gp} are non-negative, implying that the quadrotor is always safe.

Here, we use two obstacles and therefore have two separate distance measurements,

$$d_a = (\mathbf{r}_{xy} - \mathbf{p}_a)^\top (\mathbf{r}_{xy} - \mathbf{p}_a) - R_a^2,$$
$$d_b = (\mathbf{r}_{xy} - \mathbf{p}_b)^\top (\mathbf{r}_{xy} - \mathbf{p}_b) - R_b^2,$$

where $\mathbf{r}_{xy}$ is the quadrotor's lateral position, similarly $\mathbf{p}_{(\cdot)} \in \mathbb{R}^2$ is the obstacle's lateral position and $R_{(\cdot)} \in \mathbb{R}$ is the obstacle radius. The overall safety sample is taken as the noisy estimate by combining the two distance measurements,

$$y := d_a \cdot d_b + w, \quad w \sim \mathcal{N}(0, \sigma_\omega^2),$$

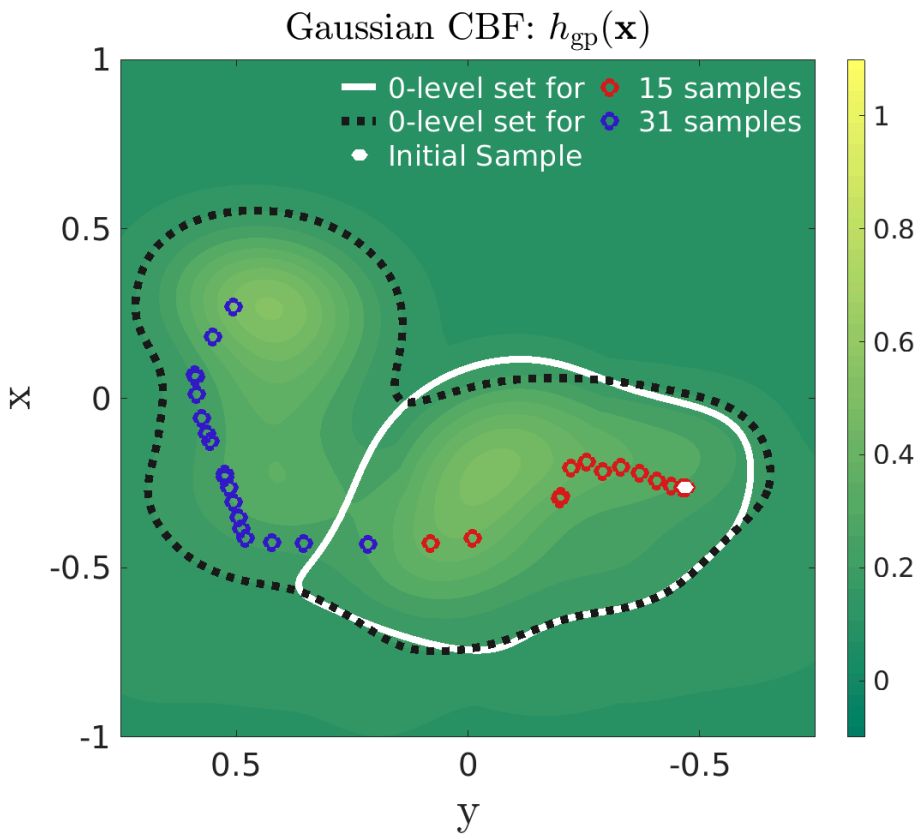

Figure 3.16: As more samples are collected, the safe set can expand arbitrarily and is not confined to a convex expansion. The contour plots are shown for the two data sample sets. The 0-level set for 15 samples is shown with bold white line and for 31 samples with black dashed line.

where $\sigma_\omega^2 \in \mathbb{R}$ is the noise variance. We take noisy sample observations to make the experiment more realistic. Moreover, we also wanted to highlight experimentally that, despite using noisy safety samples, our approach is robust enough to design non-convex safe sets online and ensure the system remains safe. If the quadrotor is closer to one of the obstacles, the product decreases, as a result reducing the safety metric. Note that, even though the obstacles are assumed to be convex, the final safe set constructed need not be convex. This is due to the noisy distance measurements observed and the posteriors being constructed online. The sampling distance is set to $\tau = 0.1$ and the hyperparameters are optimized by maximizing the log marginal likelihood using gradient methods [59]. We use the same Gaussian CBF as (3.21) in scenario A to generate the posterior safe set online.

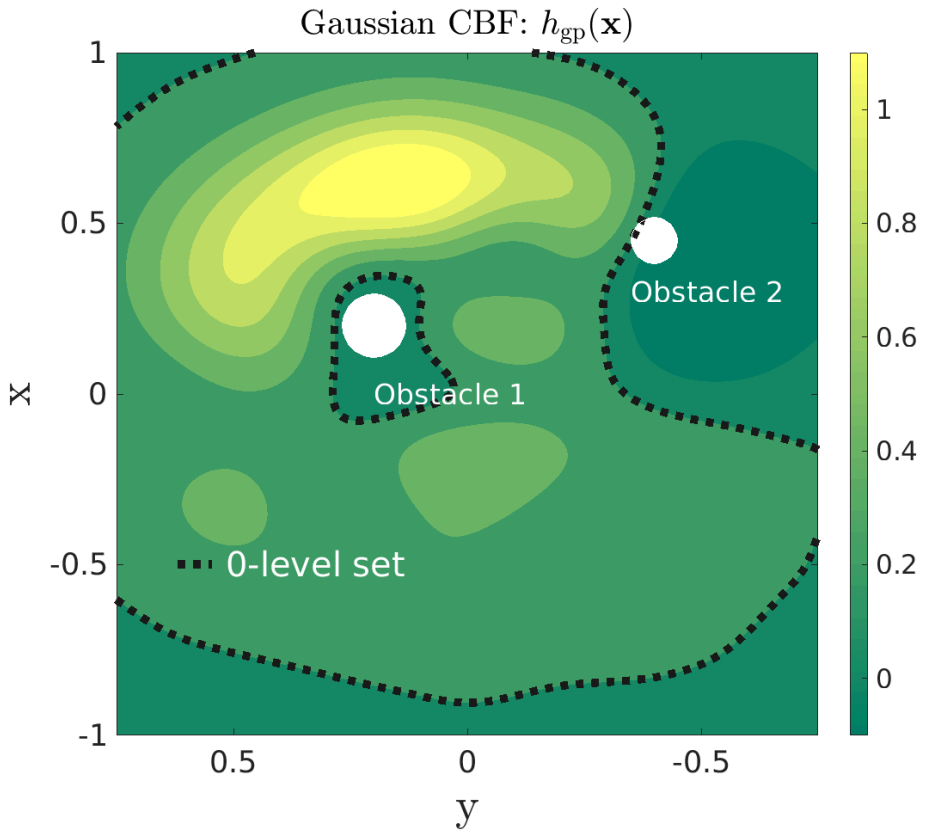

Figure 3.17: The contour plot for the Gaussian CBF with over 300 samples collected is shown. After exploring the state space, we see that the obstacles are located in the 0-sublevel sets. For the collected data set, the 0-level set is depicted with black dashed line.

The quadrotor starts in an initial safe set containing only the initial position, see Figure 3.16. The quadrotor collects safety samples y with the corresponding state $\mathbf{x}$ along its trajectory. With the data being collected, the safety function $h_{\text{gp}}(\mathbf{x})$ and its associated safe set is constructed online using (3.21). In regions where we have data, the safety belief is high and safety uncertainty is low. As more data is collected, the associated safe set expands. For unexplored regions in the state space, the safety uncertainty is high due to the high posterior variance. This aligns with the intuition that safety is not known with high confidence in unexplored spaces.

As seen in Figure 3.16, we see two sets of data samples during the quadrotor's flight. Initially, the quadrotor explored a small region in the state space containing 15 samples. The 0-level set is shown in bold white. Then the quadrotor continues exploring the state

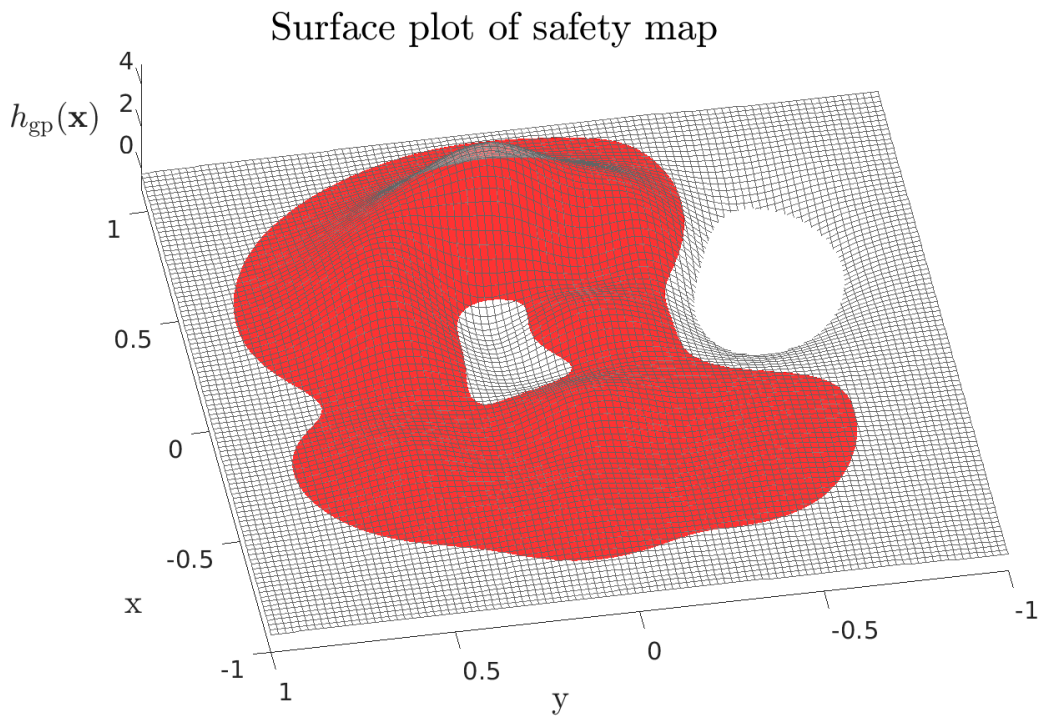

Figure 3.18: Final safe set of $h_{\mathrm{gp}}(\mathbf{x})$ for the hardware experiment with over 300 samples collected during the exploration process.

space further, thus expanding the safe set. Notice that the expansion of the safe set is not limited to any convex expansion. The 0-level set for the larger safe set of all 31 samples collected thus far is marked with dashed black line in Figure 3.16. The exploration process continues, and the quadrotor is able to detect regions in the state space which are unsafe, particularly, when it gets closer to the obstacles. The safe set constructed after collecting more than 200 samples is shown in Figure 3.17. As seen in the figure, the obstacles are located in the 0-sublevel sets which are also the unsafe regions. The final safety map for the Gaussian CBF using (3.8) is shown in Figure 3.18.

3.5.4 Scenario C : Safe Control in presence of Noisy Position State

For the final experiment, we consider the problem of safe constrained control in the presence of noisy position states. In many practical applications, measurement noise is a common occurrence which can degrade system performance and lead to unsafe consequences.

This is a particularly hard problem because we consider noise for both the input and observations to the GPs. In our current scenario, this would be noise for the safety samples and the system query state, in particular, the position state of the quadrotor. The query state is given by $\underline{\mathbf{x}}_* := [\underline{\mathbf{r}}^\top \ \dot{\mathbf{r}}^\top]^\top$, where $\underline{\mathbf{r}} \sim \mathcal{N}(\mathbf{r}, \mathbf{\Sigma}_{\mathbf{r}_*})$ is the Gaussian distributed noisy position state. We have observability of the noisy query state $\underline{\mathbf{x}}_*$ and assume knowledge of the noise covariance matrix $\mathbf{\Sigma}_{\mathbf{r}_*}$ for the position states.

The safety objective is to keep the quadrotor inside a circle of radius D. We first construct a standard CBF using the following candidate function,

$$h_{\text{cbf}} := D^2 - x^2 - y^2,$$

where $D = 0.35$ is the safety boundary radius. Next, 100 samples are sampled randomly with Gaussian noise from this candidate CBF, $y = \mathcal{N}(h_{\text{cbf}}, 0.03)$. The posterior GP is then computed, along with hyperparameter optimization, using (3.21) from this dataset which forms the Gaussian CBF. Both the CBFs as well as safety samples are shown in Figure 3.19. In this scenario we choose $\tau = 0$, since the safe sets are fixed and require no online sampling.

We perform 3 separate experiments on the quadrotor for the standard CBF using different values for the noisy position states. The different noise covariance values in the experiments are $\mathbf{\Sigma} = [0.015,\ 0.025,\ 0.04]\mathbf{I}_3 \in \mathbb{R}^{3\times 3}$. For each experiment in Figures 3.20, 3.21, and 3.22, the quadrotor starts inside the safe set and then violates safety at the boundary when subjected to noisy position states. As the value of the noise increases, the quadrotor exhibits more violation of the safety constraint by going outside the safety boundary radius. Since CBFs rectify the control input pointwise and do not account for any measurement noise in its formulation, measurement noise in the position states degrades the safety performance. For each experiment, the temporal behavior of $h_{\text{cbf}}(t)$ verifies that the safety constraint is violated due to the negative values.

We next look at the experiments using Gaussian CBFs for the same values of noise covariance used above. In Figures 3.23, 3.24, and 3.25, we see that for every experiment, the quadrotor remains confined within a more conservative safe set, which is inside the primary safety boundary radius. This occurs because in the presence of noisy input (query) to the GPs, the posterior mean uses a more conservative weighted kernel $\mathbf{q}$ in (3.16), whose entries q_i have coefficients lesser than the coefficients of the SE kernel.

The coefficient of an entry q_i given by $\sigma_f^2|\ \boldsymbol{\Sigma}\mathbf{L}^{-2} + \mathbf{I}_n\ |^{\frac{1}{2}}$ is always lesser than σ_f^2 since the eigenvalues of $\boldsymbol{\Sigma}\mathbf{L}^{-2} + \mathbf{I}_n$ are always greater than 1 ($\boldsymbol{\Sigma}\mathbf{L}^{-2}$ is a positive definite matrix

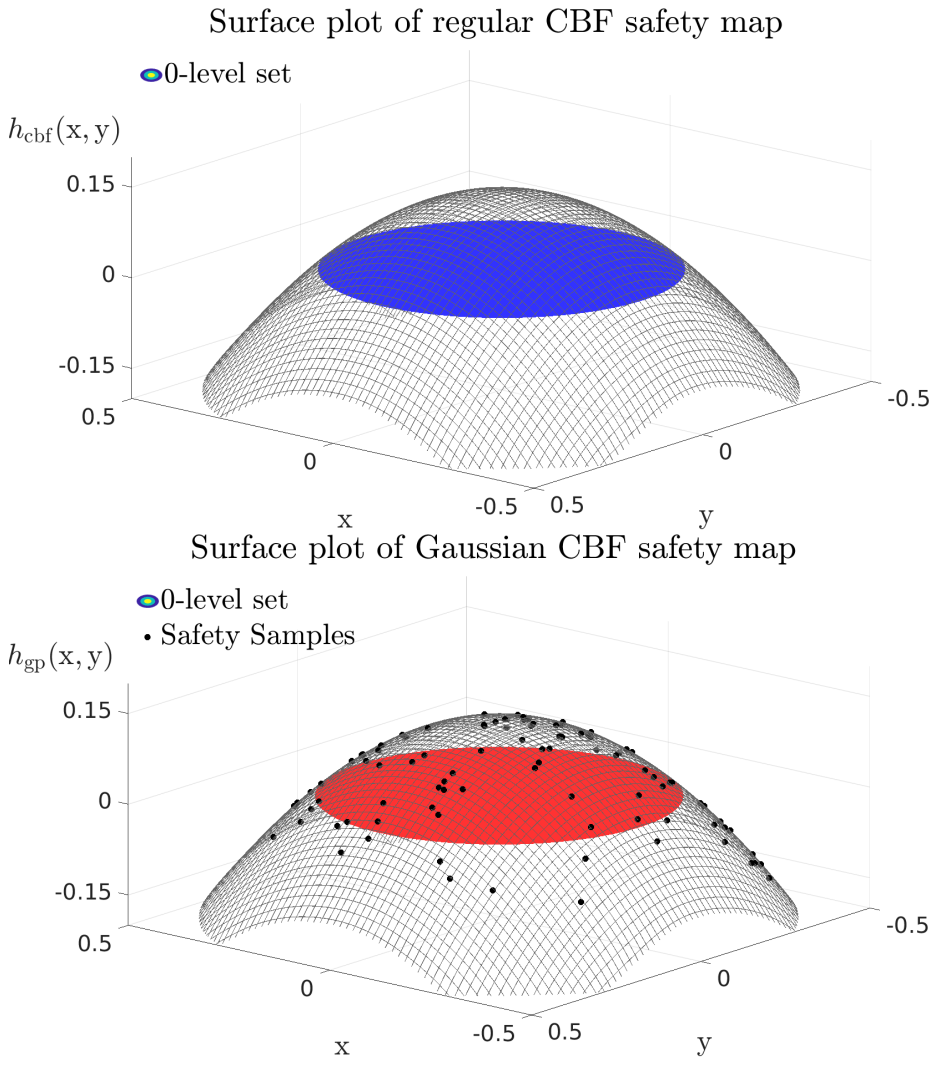

Figure 3.19: Safe sets of CBF $h_{\mathrm{cbf}}(\mathbf{x})$ (top) and Gaussian CBF $h_{\mathrm{gp}}(\mathbf{x})$ (bottom) computed by taking 100 samples (•) from h_{cbf}.

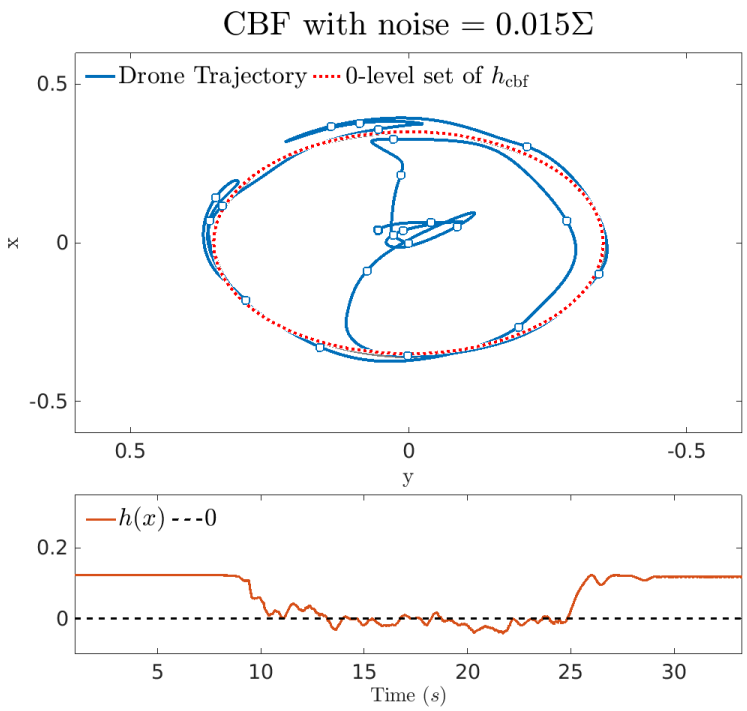

Figure 3.20: The quadrotor goes outside the boundary of the safe set $R = 0.35$ using CBF in presence of noisy position state with noise covariance of $\mathbf{\Sigma}_{\mathbf{r}_*} = 0.015\mathbf{I}_3$.

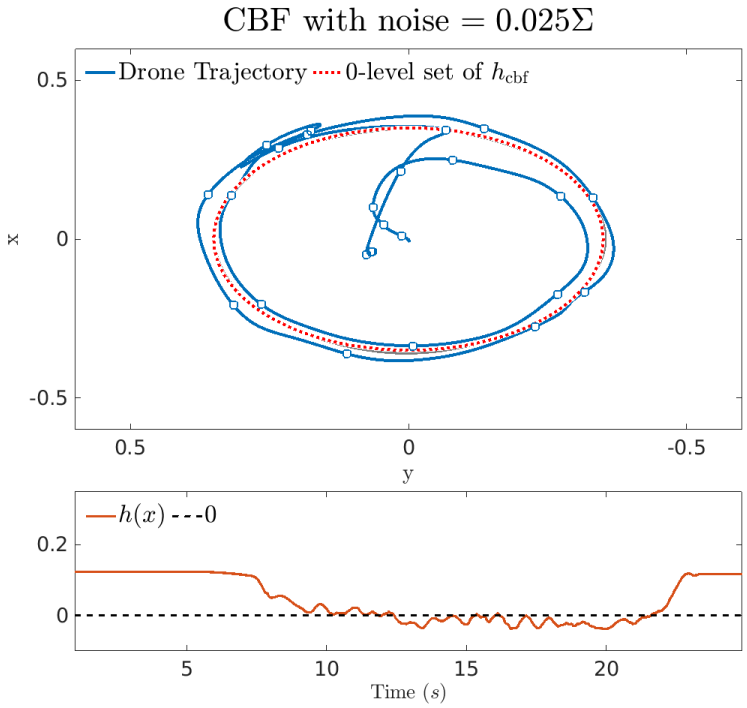

Figure 3.21: The experiment is repeated using a noise profile of $\mathbf{\Sigma}_{\mathbf{r}_*} = 0.025\mathbf{I}_3$. With regular CBF, the quadrotor violates the safety constraint of being inside the safety radius.

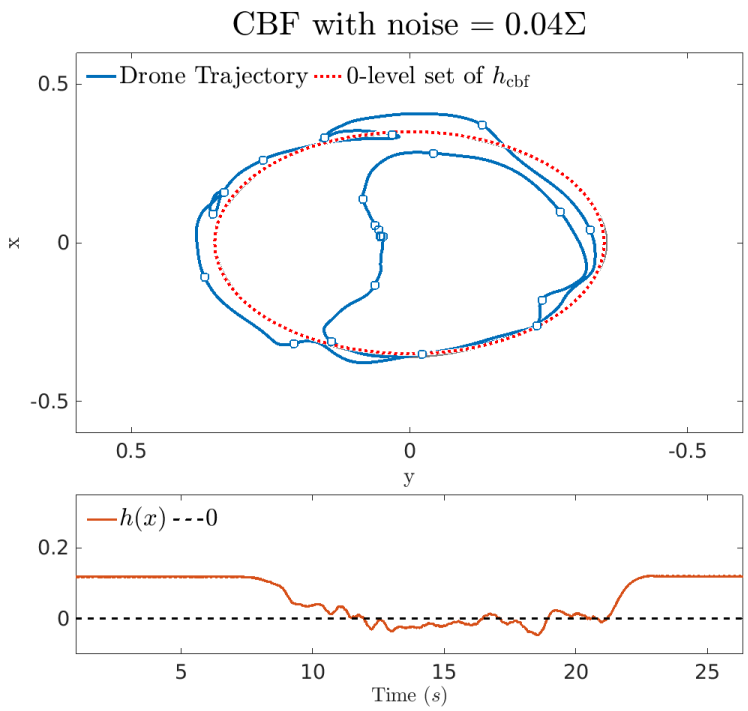

Figure 3.22: With a higher noise profile of $\mathbf{\Sigma}_{\mathbf{r}_*} = 0.04\mathbf{I}_3$, the quadrotor goes outside the boundary of the safe set $R = 0.35$ further as confirmed by the ground truth position data.

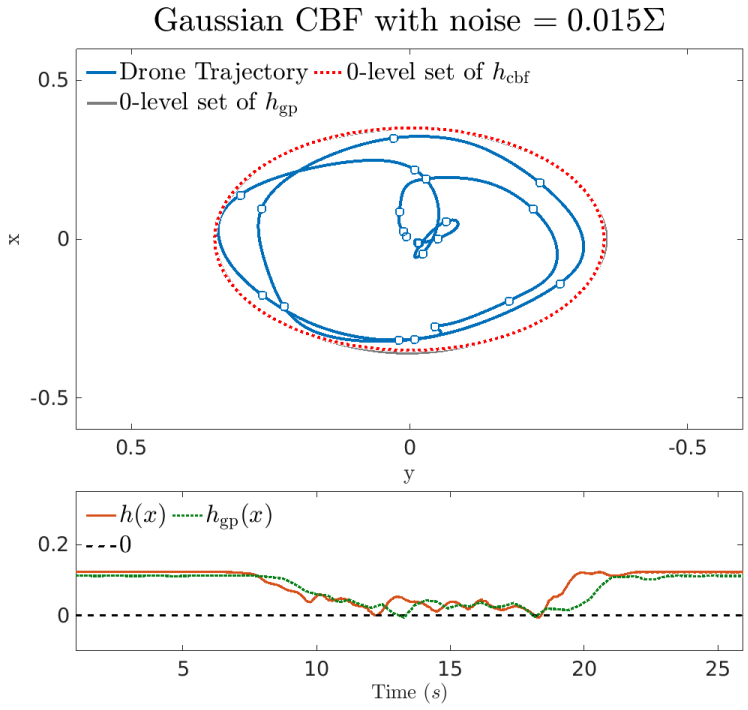

Figure 3.23: Using the same noise covariance in 3.20, the quadrotor does not go outside the circular boundary of $R = 0.35$ for the Gaussian CBF.

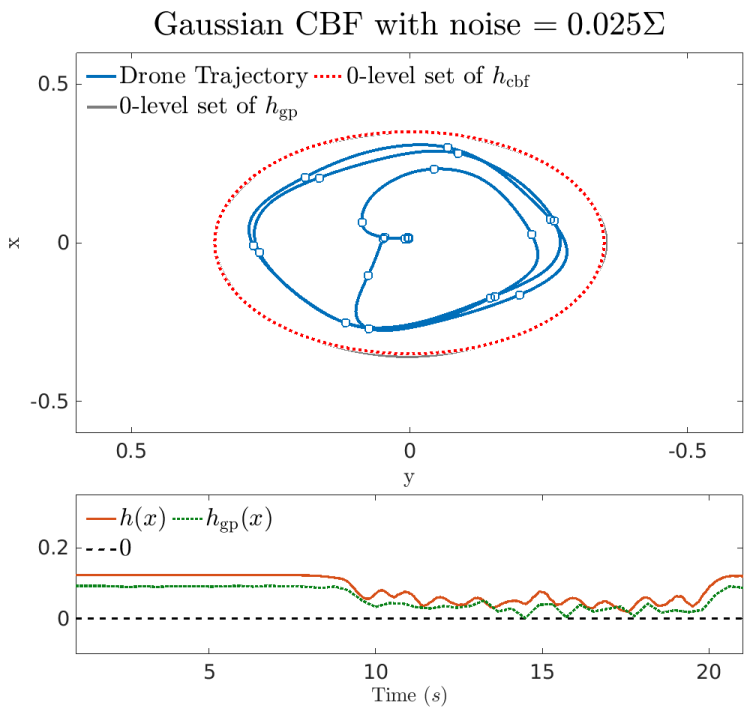

Figure 3.24: Once again, repeating the experiment with $\mathbf{\Sigma}_{\mathbf{r}_*} = 0.025\mathbf{I}_3$ as done in 3.21, Gaussian CBF constrains the quadrotor inside the safety radius.

added to the identity matrix). Therefore, the determinant is always positive and greater than unity. Intuitively, this makes sense since the GP posterior distribution is not overfitting to the noisy input query states, thus leading to a more conservative posterior estimation. We also plot h_{cbf} as a function of time using the trajectory of the quadrotor rectified under h_{gp}. The quadrotor does not get close to the circular boundary, since it is constrained conservatively by h_{gp}, thus ensuring that the original safety requirement is met. Indeed, if the noise becomes very large, then the safe set may not exist under h_{gp}. Determining the bounds on the measurement noise is currently outside the scope of this study and is left for future investigation. Here, we are primarily interested in achieving safe control in the presence of noisy query states with nonempty compact safe sets.

3.5.5 Note on Complexity

GPs are known to scale cubically with data i.e., $\mathcal{O}(N^3)$, where N is the number of data-points. This complexity arises due to the inverse operation in (3.8) for the covariance matrix

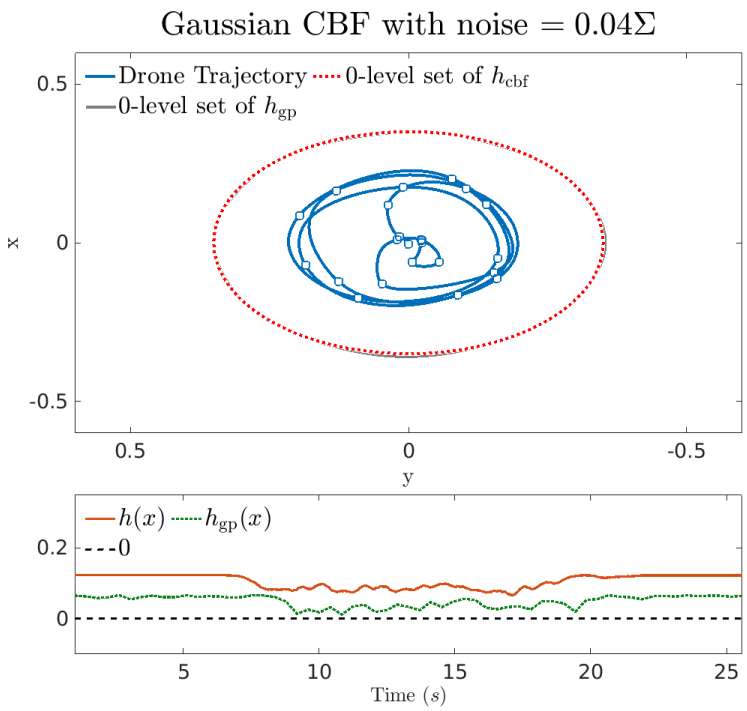

Figure 3.25: Using a higher noise covariance of $\mathbf{\Sigma}_{\mathbf{r}_*} = 0.04\mathbf{I}_3$, the quadrotor flies further away from the boundary of the safety radius because a very conservative safe set is modeled by h_{gp}. The plots of $h_{\text{gp}}(t)$ and $h_{\text{cbf}}(t)$ on the ground truth position confirms that the quadrotor is always inside the safe set.

$\overline{\mathbf{K}}$. As the number of data points increases, this could potentially cause a computational bottleneck. We address this with the help of rank-1 inverse method. For example, given 500 samples, it only takes 25ms to synthesize the Gaussian CBF. Thereafter, we handle more datapoints through rank-1 updates giving tremendous boost in computational speed. For every new data point included, it only takes 4ms to compute the inverse covariance matrix.

3.5.6 Gaussian CBF Limitations

Apart from the complexity of GPs discussed above, Gaussian CBFs have certain limitations which can restrict its applicability. The approaches discussed in Chapters 2 and 3 have only considered system dynamics in nominal operation. We do not consider any unmodeled dynamics in the system dynamical modeling. Although, we looked at noisy state (query) uncertainty for Gaussian CBFs in Section 3.3.4, there was no dynamical uncertainty. This

remains an open area for research investigation using Gaussian CBFs and could leverage methods such as stochastic CBFs [82] to account for stochastic or unmodeled dynamical components. Another limitation with Gaussian CBFs currently is the chain rule of differentiation required for dynamical systems with high relative degree. In this thesis, systems with relative degree up to 2 is only studied. Consequently, the Hessians were derived for the kernels which were needed in the Lie derivative calculations. As the relative degree goes higher, higher-order tensors will be needed which needs to be carefully derived.

3.6 Concluding Remarks

In this chapter, we proposed a framework for the synthesis of a safety function in a data-driven manner using GPs. The formulation utilizes safety samples as opposed to the traditional requirement of a smooth function. The newly formulated CBF called the Gaussian CBF was constructed by using a flexible GP prior. GPs provide the posterior mean and variance which serve as analogues for safety belief and uncertainty in our methodology. By exploiting the kernel properties in the posterior mean and variance, we were able to analytically compute the associated Lie derivatives. The Lie derivatives served as constraints in formulating a QP for rectifying the given nominal control input. We empirically verified our framework on a hardware quadrotor platform without risking any expensive system failures. We verify our approach on three different scenarios. The objective in each experiment was to synthesize a candidate safety function using GPs. We successfully show safe control for arbitrary safe sets synthesized using Gaussian CBFs, online synthesis of a Gaussian CBF as more data is collected in a collision avoidance problem, and juxtapose a Gaussian CBF with a regular CBF for constrained control in the presence of noisy position states. The quadrotor always remained inside the safe sets associated with the synthesized Gaussian CBFs.

CHAPTER 4

SAFE IMPLICIT SURFACES USING GAUSSIAN CONTROL BARRIER FUNCTIONS WITH SPARSITY

On the one hand, level set methods form the foundation of modern safety techniques such as CBFs. On the other hand, these level set methods have been successfully applied to geometric shape and surface representations as implicit surfaces using distance fields. Inspired by the success of CBFs for safety, and implicit surfaces for shape/surface estimation, we propose a unified approach where the implicit surfaces itself behave as the barrier certificates. To this end, we use GPs based implicit surface (GPIS) potentials as CBF representations while exploiting sparsity. Data in the form of *safety samples* from sensory measurements condition the GP such that the posterior mean can define an implicit surface, which forms the basis of safety belief for the CBF. GPs also provide an uncertainty estimate, quantified by the posterior variance, which can be used as a robust margin for safety. In the previous chapter 3, we did not specify the particular kind of safety samples we worked with. They were either synthetically generated or produced ad-hoc in random. In this chapter, we focus on 3D LiDAR point cloud data as the basis for our safety samples.

Although, GPs have favorable properties such as uncertainty estimation and analytical tractability, they scale cubically with data. To alleviate this issue, we focus on developing and presenting a sparse solution called *sparse Gaussian CBFs*. We validate our approach on a collision avoidance problem with two test cases: a 7DOF robot manipulator in simulation, and a 3D quadrotor in hardware. We study the test cases using Gaussian CBFs and sparse Gaussian CBFs both as an offline and online problem.

Previously, even though the quadrotor system evolved in 3D, CBFs represented using GPs were confined to 2D settings, i.e., the safe sets were in 2D. We now expand our previous work by fully characterizing a CBF using GPs in 3D. Since our approach can handle

CBF construction using data in 3D, the computational complexity is much higher compared to a standard 2D problem. To this end, we propose sparse Gaussian CBFs to account for the complexity of the data. In order to construct CBFs in 3D using data, we look at signed distance functions (SDFs) to generate implicit surfaces. SDFs are implicit surface representations, indicating whether a given datapoint $\mathbf{x}$ in the metric space for a set Ω ,characterizing the SDF, belongs to the set Ω or not. SDFs have been shown to be advantageous as a natural representation for both navigation and planning [49].

To summarize, in this chapter we propose a single unified approach for implicit surface representation and CBF with the help of GPIS. This is achieved by generating a GP based implicit surface, which serves as the candidate CBF. To the best of our knowledge, GPIS have not been used previously for demonstrating safe control. We formulate *Sparse Gaussian CBFs* by establishing sparse solution to our previous work on Gaussian CBFs in Chapter 3. In addition to sparse Gaussian CBFs retaining the attractive properties of Gaussian CBFs such as uncertainty estimation and analytical tractability, it also has lower complexity. We juxtapose Gaussian CBFs with sparse Gaussian CBFs in two test cases. First in simulation using a robot manipulator, and the second in hardware using the same Crazyflie quadrotor as done in the previous chapters. We believe this to be the first work where CBFs were synthesized completely using data in 3D as well as deployed on hardware. We perform collision avoidance using a quadrotor in hardware based on the synthesized Gaussian CBF and sparse Gaussian CBF as both offline and online problems.

4.1 Background Preliminaries

4.1.1 Implicit Surfaces

An implicit surface describes the shape of a volumetric object in an $n-$dimensional Euclidean space by means of a function. This function helps identify if each location in the space belongs to the object or not. More formally, we define the implicit surface, without loss of generality, as the 0-level set (or isosurface) of a real-valued implicit function

$f_{\text{is}} : \mathbb{R}^n \to \mathbb{R}$ for a point $\mathbf{x} \in \mathbb{R}^n$ as,

$$f_{\text{is}}(\mathbf{x}) \begin{cases} > 0, & \mathbf{x} \text{ outside the surface} \\ = 0, & \mathbf{x} \text{ on the surface} \\ < 0, & \mathbf{x} \text{ inside the surface} \end{cases} . \tag{4.1}$$

The 0-level set $f_{\text{is}}^{-1}(0)$ of an implicit function f_{is} can represent a surface $\mathcal{S}$ in $\mathbb{R}^n$. In our context, this surface $\mathcal{S}$ represents the surface of safety or safety decision boundary. Conversely, given the surface $\mathcal{S}$ in $\mathbb{R}^n$, there exists a function $f_{\text{is}} : \mathbb{R}^n \to \mathbb{R}$ with $\mathcal{S}$ as its $0-$level set such that $f_{\text{is}}^{-1}(0) = \mathcal{S}$ (Prop. 2 in [83]). As a result, given samples on the surface $\mathcal{S}$, one can construct the implicit function f_{is}. We will use a GP regressor for approximating the implicit surface of a volumetric object in $d = 3$, where $d \leq n$, which is identified as Gaussian Process Implicit Surface (GPIS) in the literature.

4.1.2 Sparse Gaussian Process Regression

Despite GPs being very powerful regressors, as the dataset grows larger, they become computationally intractable. GP prediction complexity has a cost of $\mathcal{O}(N^3)$. Even if one stores the covariance matrix to save costs, the complexity per test case is $\mathcal{O}(N)$ for predictive mean and $\mathcal{O}(N^2)$ for predictive variance. Hence, many sparse approximations of GPs have been developed to bring down the complexity cost while retaining accuracy [84, 85, 86, 87, 88]. Broadly stated, sparse approximations of GPs fall into two major categories: *approximate generative model with exact inference* and *exact generative model with approximate inference*. Unifying theories for these various frameworks are discussed in [89, 90]. We use the the *Sparse Pseudo-Input Gaussian Process* (SPGP) [87] for our work.

The starting point to any GP approximation method is through a set of so-called *inducing* or *pseudo-points* giving rise to sparsity. Consider a pseudo-dataset $\mathbf{D}_M = \left[\overline{\mathbf{X}}_M, \overline{\mathbf{y}}_M\right]$, $M \ll N$, where the pseudo-inputs are $\overline{\mathbf{X}}_M = \{\overline{\mathbf{x}}_i\}_{i=1}^M$ and pseudo-targets are $\overline{\mathbf{y}}_M =$

$\{\overline{y}_i\}_{i=1}^{M}$. The objective is to find the posterior distribution over the pseudo-targets, and then integrate out the likelihood with the help of this posterior to find the predictive distribution. The complete data likelihood is given by [87]:

$$p(y|\mathbf{X}, \overline{\mathbf{X}}, \overline{\mathbf{y}}) = \mathcal{N}(y \mid \mathbf{K}_{NM}\overline{\mathbf{K}}_M^{-1}\,\overline{\mathbf{y}}, \mathbf{\Lambda}_N + \sigma_\omega^2\mathbf{I}_N), \tag{4.2}$$

where $\mathbf{\Lambda}_N = \text{diag}(\mathbf{K}_N - \mathbf{K}_{NM}\overline{\mathbf{K}}_M^{-1}\mathbf{K}_{MN})$, $\big[\mathbf{K}_{NM}\big]_{(i,j)} = k(\mathbf{x}_N, \overline{\mathbf{x}}_M)$. By placing a Gaussian prior on the pseudo-targets, one can derive the posterior distribution using Bayes rule:

$$p(y|\mathbf{X}, \overline{\mathbf{X}}, \overline{\mathbf{y}}) = \mathcal{N}(y \mid \mathbf{K}_{NM}\overline{\mathbf{K}}_M^{-1}\,\overline{\mathbf{y}}, \mathbf{\Lambda}_N + \sigma_\omega^2\mathbf{I}_N), \tag{4.3}$$

For a complete mathematical treatment for the derivation of the predictive distribution of SPGP, see [87]. Here, we simply present the predictive mean and variance of SPGP:

$$\mu(\mathbf{x}_*) = \mathbf{k}_M^\top(\mathbf{x}_*)\,\mathbf{Q}_M^{-1}\,\mathbf{K}_{MN}\big(\mathbf{\Lambda}_N + \sigma_\omega^2\mathbf{I}_N\big)^{-1}\mathbf{y}_N \tag{4.4}$$

$$\sigma(\mathbf{x}_*)^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_M^\top(\mathbf{x}_*)\,\big(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1}\big)\,\mathbf{k}_M(\mathbf{x}_*) + \sigma_\omega^2, \tag{4.5}$$

where $\mathbf{k}_M(\mathbf{x}_*) = \big[k(\overline{x}_1, \mathbf{x}_*), \ldots, k(\overline{x}_M, \mathbf{x}_*)\big]^\top \in \mathbb{R}^M$ is the covariance vector between $\overline{X}_M$ and $\mathbf{x}_*$, $\mathbf{Q}_M = \mathbf{K}_M + \mathbf{K}_{NM}^\top\big(\mathbf{\Lambda}_N + \sigma_\omega^2\mathbf{I}_N\big)^{-1}\mathbf{K}_{NM} \in \mathbb{R}^{M\times M}$, $\mathbf{\Lambda}_N = \text{diag}\big[\mathbf{K}_N - \mathbf{K}_{NM}\,\mathbf{K}_M^{-1}\,\mathbf{K}_{MN}\big] \in \mathbb{R}^{N\times N}$ is a diagonal matrix, $\mathbf{K}_M \in \mathbb{R}^{M\times M}$ is the covariance between pairs of pseudo-inputs $\overline{\mathbf{X}}_M$, and $\big[\mathbf{K}_{NM}\big]_{(i,j)} = k(\mathbf{x}_i, \overline{\mathbf{x}}_j),\ i \in \{1, ..., N\}, j \in \{1, ..., M\}$ is the covariance matrix between $\mathbf{X}_N$ and $\overline{\mathbf{X}}_M$. Computation cost for $\mathbf{Q}_M$ is dominated by the inversion operation which is $\mathcal{O}(M^2N)$ [87]. By precomputing the inverse, the cost per test case is $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ for predictive mean and variance respectively. We can jointly optimize for the kernel hyperparameters, Θ, and pseudo-inputs, $\overline{\mathbf{X}}_M$, by maximizing the

log marginal likelihood using quasi-Newton gradient methods:

$$\log p(y \mid \mathbf{X}, \overline{\mathbf{X}}, \overline{\mathbf{y}}) = -\frac{1}{2}\Big(N \log(2\pi) - \log|\overline{\mathbf{K}}_N| - \mathbf{y}_N \overline{\mathbf{K}}_N^{-1} \mathbf{y}_N\Big). \tag{4.6}$$

4.2 Problem Statement

Given a dynamical system (2.1) and a volumetric object, we are interested in determining the surface of the object using GPs such that the system remains safe around the object. Consider that system (2.1) is given with access to its states $\mathbf{x} \in \mathbb{R}^n$ and samples of a volumetric object which represent points on the surface of the object. To accurately construct the safety surface (GPIS) of the object, we need instances of on-surface and off-surface sample points. The off-surface points consist of external and internal points to the surface. We rely on surface normals from sensor measurements to characterize points external to the surface and, conversely, taking the opposite direction of the surface normal to characterize internal points. We define the set of samples used, both on-surface and off-surface samples collectively, as *safety samples* for approximating the implicit surface. The resulting GPIS is the Gaussian CBF of interest, see Figure 4.1.

Assumption 8. *We assume access to sensor measurements that provide samples on the object's surface and surface normals, constituting the safety samples expressed in the world frame for our setting.*

This is a mild assumption since many sensors are equipped to provide point cloud and surface normal direction vectors, e.g. laser, haptic, or camera sensors [91]. The sensor measurements provide us the data in order to construct a valid safety certificate, i.e. the safe surface, which can then be used for ensuring dynamical system safety through forward invariance properties.

***Remark* 7.** *Since $d \leq n$, where the safety surface is modeled in $d = 3$, the 3D Euclidean space for instance, whereas the system state $\mathbf{x} \in \mathbb{R}^n$, the $(n - d)$ free dimensions can be*

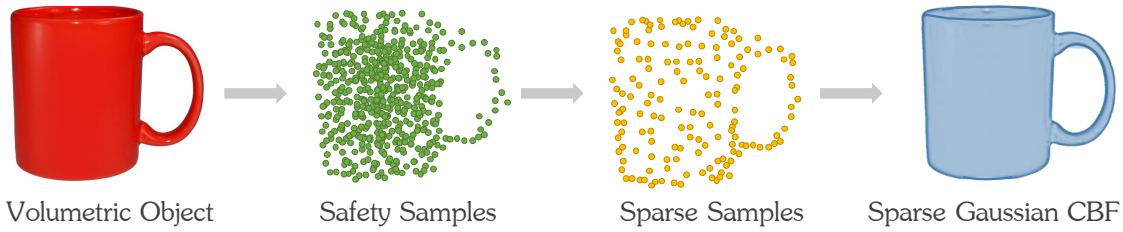

Figure 4.1: The safety surface of a volumetric object of interest is modeled as a sparse Gaussian CBF with the help of sparse safety samples.

truncated for GP training.

Our primary objective is to *synthesize a safety function* h_{sgp} *non-parametrically using measurements of the system states and safety samples online, while exploiting data sparsity, and ensure* (2.1) *remains safe*. The safety function h_{sgp} is the sparse variant of a Gaussian CBF [53] representing the implicit surface of the object we are modeling,

$$\underbrace{f_{\text{is}}(\mathbf{x})}_{\text{Implicit Surface}} = \underbrace{h_{\text{sgp}}(\mathbf{x})}_{\text{Sparse Gaussian CBF}} := \underbrace{h_{\text{b}}(\mathbf{x})}_{\text{Safety Belief}} + \underbrace{h_{\text{u}}(\mathbf{x})}_{\text{Safety Margin/Uncertainty}} . \tag{4.7}$$

The sparse Gaussian CBF is the implicit surface approximation of a volumetric object. Since we use data to model this surface, it is desirable to account for a margin of safety in the estimation of the safety function. It becomes particularly more important, since we are exploiting sparsity, to account for any uncertainty in the estimation. Intuitively, the safety belief represents the best estimation of system safety (or implicit surface representation), while the safety margin represents any uncertainty. The safety margin will be higher in the state space where sufficient data is not available. In the absence of any uncertainty, the safety belief will be the final overall desired safety.

Problem 5. *Given system* (2.1) *and online measurements of the system state* $\mathbf{x}_*$*, synthesize* $h_{\text{sgp}}(\mathbf{x})$ *to model an implicit surface with a safety belief and associated safety margin or uncertainty, conditioned on past safety samples and observations while exploiting sparsity, for a volumetric object in the dataset,* $\mathbf{D}_N = \{\mathbf{X}_N, \mathbf{y}_N\}$*, where* $\mathbf{X}_N = \{\mathbf{x}_i\}_{i=1}^N$ *and* $\mathbf{y}_N =$

$\{y_i\}_{i=1}^{N}$, *such that system (2.1) remains safe.*

We are also interested in synthesizing safe controllers similar to the Problems 2 and 4. The rectified control input $\mathbf{u}_{\text{rect}}$ is finally applied to the dynamical system (2.1), which in principle should ensure the system remains safe, i.e., does not collide with the obstacle (or volumetric object).

4.3 Proposed Methodology

Here, we present our approach to approximate the GPIS f_{is} as a sparse Gaussian CBF $h_{\text{sgp}}(\mathbf{x})$. GPs can provide uncertainty estimate in the form of posterior variance, unlike traditional neural networks which require careful design considerations for quantifying uncertainty estimation [92].

4.3.1 Data Processing

Given N_s sensor measurements in an $n-$dimensional Euclidean space, $\mathcal{P} \in \mathbb{R}^{N_s \times n} \times \mathbb{R}^{N_s \times n}$, where $\mathcal{P}$ is the set of point cloud data and surface normal directions, we first compose the training input points and targets/labels. The training inputs to the GP comprises of on-surface and off-surface points. The point cloud data in $\mathcal{P}$ is used as on-surface points. The off-surface points, which represent both external and internal points to the surface, are computed using the surface normal information. Let the on-surface and off-surface sets be given as follows,

$$\begin{aligned} \Omega_{\text{ext}} &:= \{\, \mathbf{x}_+ \;\mid\; f_{\text{is}}(\mathbf{x}_+) > 0 \} \\ \Omega_0 &:= \{\, \mathbf{x}_0 \;\mid\; f_{\text{is}}(\mathbf{x}_0) = 0 \} \quad . \\ \Omega_{\text{int}} &:= \{\, \mathbf{x}_- \;\mid\; f_{\text{is}}(\mathbf{x}_-) < 0 \} \end{aligned}$$

Since we assume having access only to the point cloud coordinates for the surface of the object and corresponding surface normals, i.e., the set $\mathcal{P}$ only, we need to form the sets Ω_0,

Ω_{ext}, and Ω_{int}.

- The set Ω_0 is formed by taking N_0 point cloud coordinates randomly from $\mathcal{P}$, where $N_0 \leq N_s$, and for each coordinate assigning the target value (or label) of 0, i.e., $\mathbf{y}_{N_0} = \{0\}_{i=1}^{N_0}$.

- The set Ω_{ext} is formed by synthetically creating N_+ coordinates randomly, where $N_+ < N_0$, in the direction of the surface normal vectors with a certain offset. Similarly, the set Ω_{int} is formed by synthetically creating N_- coordinates randomly, where $N_- \leq N_+$, in the opposite direction of the surface normal vectors.

- The corresponding targets (or labels) for Ω_{ext} and Ω_{int} are assigned $+1$ and -1, i.e, $\mathbf{y}_{N_+} = \{+1\}_{i=1}^{N_+}$, and $\mathbf{y}_{N_-} = \{-1\}_{i=1}^{N_-}$ respectively.

- The training inputs are $\mathbf{X}_N = [\mathbf{x}_0^\top, \mathbf{x}_+^\top, \mathbf{x}_-^\top]^\top \in \mathbb{R}^{N \times n}$ and training observations/labels are $\mathbf{y}_N = [\mathbf{y}_{N_0}, \mathbf{y}_{N_+}, \mathbf{y}_{N_-}]^\top$, with $N = N_0 + N_+ + N_-$, thus forming the dataset $\mathcal{D}_N = \{\mathbf{X}_N, \mathbf{y}_N\}$.

4.3.2 Sparse Gaussian Control Barrier Function

We propose the following sparse Gaussian CBF $h_{\text{sgp}}(\mathbf{x})$ which models both the safety belief and associated margin using the sparse predictive mean (4.4) and variance (4.5),

$$\begin{aligned} h_{\text{sgp}}(\mathbf{x}) &:= \mu(\mathbf{x}) + \sigma^2(\mathbf{x}) \\ &= \underbrace{\mathbf{k}_M^\top(\mathbf{x}) \overline{\mathbf{Q}}_{MN}\, \mathbf{y}_N}_{\text{safety belief}} + \underbrace{k(\mathbf{x},\mathbf{x}) - \mathbf{k}_M^\top(\mathbf{x})\, \mathbf{P}_M\, \mathbf{k}_M(\mathbf{x}) + \sigma_\omega^2}_{\text{safety margin}}, \end{aligned} \tag{4.8}$$

where $\overline{\mathbf{Q}}_{MN} = \mathbf{Q}_M^{-1}\, \mathbf{K}_{MN} \big(\mathbf{\Lambda}_N + \sigma_\omega^2 \mathbf{I}_N\big)^{-1} \in \mathbb{R}^{M \times N}$ and $\mathbf{P}_M = \big(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1}\big) \in \mathbb{R}^{M \times M}$. The sparse GP posterior mean represents the belief regarding our notion of safety while the sparse GP posterior variance quantifies the safety margin.

The admissible control space for the CBF above is given by,

$$\mathrm{K}_{\mathrm{gcbf}}^{\mathrm{sparse}} = \{\mathbf{u} \in \mathbb{R}^m \;\big|\; L_f h_{\mathrm{sgp}}(\mathbf{x}) + L_g h_{\mathrm{sgp}}(\mathbf{x})\mathbf{u} + \alpha(h_{\mathrm{sgp}}(\mathbf{x})) \geq 0\}, \tag{4.9}$$

which ensures that the dynamical system (2.1) is forward invariant in the safe set given by h_{sgp} (Proposition 2). In Section 4.3.4, we elaborate how safe control is achieved using sparse Gaussian CBFs.

***Remark* 8.** *Sparse Gaussian CBFs used for modeling the implicit surface while addressing safety can be done using sensor measurements to design the safety function, i.e., the implicit surface of the volumetric object. This allows designing safe surfaces based on data, instead of hand designing a candidate function which has been the traditional practice.*

4.3.3 Lie Derivatives of Sparse Gaussian CBF

We now derive the Lie derivatives for the sparse Gaussian CBF present in (4.8) for ensuring forward invariance in the safe set. As done previously in Sections 2.3.1 and 3.3.2, first, we take the partial derivative of (4.8) with respect to $\mathbf{x}$ at a query point $\mathbf{x}_*$,

$$\left.\frac{\partial h_{\mathrm{sgp}}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} = \left.\frac{\partial \mu(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} + \left.\frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} \tag{4.10}$$

$$= \mathbf{y}_N^\top \overline{\mathbf{Q}}_{NM} \left.\frac{\partial \mathbf{k}_M(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} - \; 2\mathbf{k}_M^\top(\mathbf{x}_*)\mathbf{P}_M \left.\frac{\partial \mathbf{k}_M(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*}, \tag{4.11}$$

where $\overline{\mathbf{Q}}_{MN}$ and $\mathbf{P}_M$ are defined in (4.8). Using (2.9), we can compute the Lie derivatives of $h_{\mathrm{sgp}}(\mathbf{x})$ by taking its time derivative as follows,

$$\begin{aligned}
\dot{h}_{\mathrm{sgp}}(\mathbf{x}) &= \frac{\partial h_{\mathrm{sgp}}(\mathbf{x})}{\partial \mathbf{x}}\dot{\mathbf{x}} \\
&= \frac{\partial h_{\mathrm{sgp}}(\mathbf{x})}{\partial \mathbf{x}} f(\mathbf{x}) + \frac{\partial h_{\mathrm{sgp}}(\mathbf{x})}{\partial \mathbf{x}} g(\mathbf{x})\mathbf{u} \\
&= L_f h_{\mathrm{sgp}}(\mathbf{x}) + L_g h_{\mathrm{sgp}}(\mathbf{x})\mathbf{u},
\end{aligned} \tag{4.12}$$

where (4.11) is used in the Lie derivatives, $L_f h_{\text{sgp}}(\mathbf{x})$ and $L_g h_{\text{sgp}}(\mathbf{x})$. Here, we used the SE kernel but later we will use the Matérn kernel and compute its partial derivatives as well.

4.3.4 Safe Control using Sparse Gaussian CBF

Given the nominal control input $\mathbf{u}_{\text{nom}}$ and a desired state $\mathbf{x}_{\text{des}}$ outside the safe set $\mathcal{S}$, the system state will go outside $\mathcal{S}$, thus violating the safety requirement. The nominal control input must be rectified in order to ensure forward invariance of the system inside the safe set. We can form an online quadratic program (QP) to rectify $\mathbf{u}_{\text{nom}}$ subject to constraints derived using the Lie derivatives in (4.12) [8]. The QP optimization routine is given by,

Sparse Gaussian CBF-QP: *Control Input modification*

$$\begin{aligned}\mathbf{u}_{\text{rect}} = \underset{\mathbf{u}\,\in\,\mathbb{R}^m}{\arg\min}\ \frac{1}{2}\left\|\mathbf{u}-\mathbf{u}_{\text{nom}}\right\|^2 \ \text{ s. t.}\\ L_f h_{\text{sgp}}(\mathbf{x}) + L_g h_{\text{sgp}}(\mathbf{x})\mathbf{u} + \alpha(h_{\text{sgp}}(\mathbf{x})) \geq 0,\end{aligned} \tag{4.13}$$

where $\mathbf{u}_{\text{rect}}$ is the rectified control input. By rectifying the control policy, the system is guaranteed to remain forward invariant for the safe set $\mathcal{S}$. The algorithm for generating the safe control input from the synthesized sparse Gaussian CBF is shown in Algorithm 3.

Algorithm 3 Sparse Gaussian CBF Synthesis & Safe Control

Input: SYSTEM (2.1)
GP PRIOR $h_{\text{sgp}}(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$
NOMINAL INPUT $\mathbf{u}_{\text{nom}}$
DATASET $\mathbf{D}_N$ & number of pseudo-points M
Output: RECTIFIED INPUT $\mathbf{u}_{\text{rect}}$

1: **procedure** SAFECONTROL
2: INITIALIZE $\mathcal{D}_M = \{\overline{\mathbf{X}}_M, \overline{\mathbf{y}}_M\}$ randomly from $\mathcal{D}_N$
3: OPTIMIZE $h_{\text{sgp}}(\mathbf{X}_N, \mathbf{y}_N; \overline{\mathbf{X}}_M, \overline{\mathbf{y}}_M)$ using (4.6)
4: SYNTHESIZE $h_{\text{sgp}}(\mathbf{X}_N, \mathbf{y}_N)$ using (4.8)
5: COMPUTE $\frac{\partial h_{\text{sgp}}(\mathbf{x})}{\partial \mathbf{x}}$ using (4.11) & (2.9)
6: SETUP QP constraint using (2.1) & (4.12)
7: RECTIFY $\mathbf{u}_{\text{nom}}$ using (4.13)
return $\mathbf{u}_{\text{rect}}$

First, a subset of the training dataset $\mathcal{D}_N$ is randomly selected to initialize the pseudo-

dataset $\mathcal{D}_M$. Hyperparameter and pseudo-input locations are optimized in step 3, followed by the synthesis of the sparse Gaussian CBF in step 4. The CBF represents the safety surface upon which we desire to perform safe control. The corresponding derivatives are computed in steps $5-6$ to set up the QP constraint. Finally, rectification is done on the nominal control input in step 7.

4.3.5 Illustrative 3D Example

Here, we discuss our proposed approach in achieving safe control by modeling a volumetric object using sparse Gaussian CBFs. We apply our method on two different objects - a convex object (sphere) and a non-convex object (solid blob). We wanted to highlight the efficacy of our method for synthesizing a safety function given any arbitrary sample points, and perform safe control.

Data Preprocessing: We synthetically generate the data for modeling the implicit surfaces of these two objects. We will later look at more concrete test cases where existing datasets with point cloud information and surface normals are used.

For the sphere, we generate vertices on the surface of a sphere in a $3-$dimensional Euclidean space for a certain radius and center. We select some samples in the interior and exterior of the sphere (by changing the radii) and label the on-surface points with 0 and off-surface points with $+1$ and -1 as described in Section 4.3.1.

For the blob, we generate the samples by using two different centers and radii of a sphere. The centers and radii are chosen such that the points are in close proximity to each other, thus generating a non-convex shaped object. The point cloud points and associated labels form the dataset $\mathcal{D}_N$ where $N = 300$ for the sphere and $N = 400$ for the blob.

Dynamical System: We take a fully actuated control affine system in $\mathbb{R}^3$ given by,

$$\dot{\mathbf{x}} = \underbrace{\mathbf{0}_3}_{f(\mathbf{x})} + \underbrace{\mathbf{I}_3}_{g(\mathbf{x})}\mathbf{u} = \mathbf{u}. \tag{4.14}$$

Based on 1, we have complete observability of the system state $\mathbf{x} \in \mathbb{R}^3$.

Sparse Gaussian CBF: Using the dataset $\mathcal{D}_N$ for the sphere and the blob, we train a sparse GP which uses one-fifth of the samples as pseudo-inputs and targets. This synthesizes the h_{sgp} for each object where the safe set is characterized by,

$$\mathcal{S} = \{\mathbf{x} \mid h_{\text{sgp}}(\mathbf{x}) \geq 0 \}.$$

The training time for modeling the sphere and the blob takes 120ms and 165ms respectively. The sparse Gaussian CBF used is,

$$h_{\text{sgp}} = \mu + \sigma^2.$$

where the posterior variance provides a safety margin.

Safe Control Synthesis: A proportional controller is selected as the nominal controller, $\mathbf{u}_{\text{nom}} = \mathbf{K}_p(\mathbf{x} - \mathbf{x}_{\text{des}})$, where $\mathbf{K}_p \in \mathbb{R}^{3\times3}$ is the gain matrix. With h_{sgp} and system (4.14), the Lie derivatives are computed using (4.11) and (4.12),

$$\begin{aligned} L_f h_{\text{sgp}} &= 0 \\ L_g h_{\text{sgp}} &= \frac{\partial h_{\text{sgp}}(\mathbf{x})}{\partial \mathbf{x}}. \end{aligned}$$

The Lie derivatives are then used as the safety constraints in the QP formulation (4.13),

$$\begin{aligned} L_f h_{\text{sgp}} + L_g h_{\text{sgp}} \mathbf{u} + \alpha(h_{\text{sgp}}) &\geq 0 \\ \frac{\partial h_{\text{sgp}}(\mathbf{x})}{\partial \mathbf{x}} \mathbf{u} &\geq -k_0 h_{\text{sgp}}. \end{aligned}$$

Even if h_{sgp} models arbitrary complex shapes or surfaces, the decision variable $\mathbf{u}$ appears linearly in the constraint. Consequently, this enables performing real-time convex optimization as a QP. We use the sparse Gaussian CBF with a proportional constant, k_0, as

the class-κ function. The solution to the QP in (4.13) provides the rectified control input $\mathbf{u}_{\text{rect}}$, which is then used on (4.14) to ensure forward invariance. The illustrations are shown in Figure 4.2.

Nominal vs. Safe Control: As seen in Figure 4.2, each surface is modeled as the boundary of the safe set using h_{sgp}. The rectified control input ensures that the system does not collide with the object. It follows the reference trajectory in the absence of any safety violates, but relaxes trajectory tracking for upholding safety by remaining away from the object's modeled surface. However, the nominal control input violates the safety requirement and follows the reference trajectory.

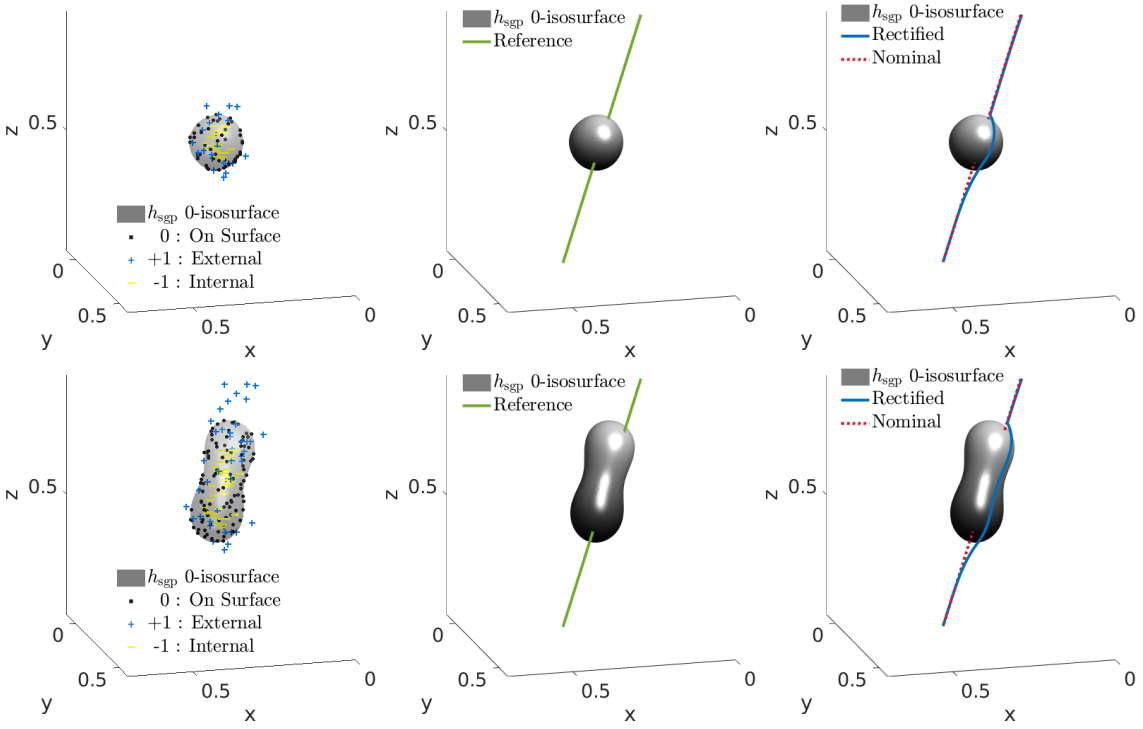

Figure 4.2: Safe control in the presence of convex and non-convex volumetric objects. (Left) The samples used for modeling the 0-isosurface of a sphere (top) and a solid blob (bottom) using sparse Gaussian CBF. (Middle) A reference trajectory is selected such that it goes through each object. (Right) The sparse Gaussian CBF rectified control prevents the system from colliding with the objects, unlike the nominal controller.

4.4 Test Case I : 7-DOF Robot Manipulator Simulation

Now, we look at using Gaussian CBFs with and without sparsity for controlling a 7 degree of freedom (DOF) robot manipulator in simulation. A robot manipulator has many interesting applications such as logistics, warehouse, surgery etc. We are interested in trajectory tracking for a manipulator while remaining safe in the presence of an obstacle. A traditional CBF is limited because majority of the volumetric objects cannot be modeled using hand-designed functions. However, using Gaussian CBFs, we can model complex geometric objects with high safety guarantees giving our approach a huge advantage over traditional approaches of hand designing CBFs.

4.4.1 Stanford Bunny Implicit Surface Modeling

As a test case, we model the surface of the Stanford bunny as the boundary of our safe set using both regular and sparse GPs. The bunny is a well known 3D test model used in computer graphics, rendering, and geometry. The bunny has sufficiently complex geometry to represent a non-convex safety boundary. The bunny model has 34817 datapoints which we down sample to $N = 2178$ datapoints randomly. We also down scale its dimensions to be within $0.45\text{m} \times 0.45\text{m} \times 0.55\text{m}$.

Offline Training: We synthesize the safety function as the implicit surface modeling the shape of the bunny. The particular form of the Gaussian CBFs is discussed in Section 4.4.3. We use the `gpml` toolbox [93] for training the GPs on an Intel i7-9800X CPU with 16 GB RAM and 4.4 GHz. One-half of the 2178 datapoints were initialized as pseudo-points and targets for sparse GPs. The training time took 14.15s on average for regular GP using 50 iterative steps to maximize the log marginal likelihood for hyperparameter optimization using (4.6), whereas, sparse GPs took 9.46s for training. The implicit surfaces for the 0-isosurface of the bunny using both Gaussian and sparse CBFs are shown in Figure 4.3. The Gaussian CBF models the bunny more accurately than the sparse Gaussian CBF. The

sparse based Gaussian CBF captures the overall shape of the bunny but does not model precisely the details on the face or torso. This is expected using the sparse Gaussian CBF since there is a trade-off between modeling accuracy and computational speed.

Evaluation Metric: To quantify the modeling performance, we use the Chamfer distance which is a symmetric metric used to measure distance between two surfaces. More precisely, it computes the distance between the two point clouds ($\mathcal{P}_1$ and $\mathcal{P}_2$) sampled from a pair of surfaces.

$$d_{\text{ch}}(\mathcal{P}_1, \mathcal{P}_2) = \sum_{x \in \mathcal{P}_1} \underset{\mathbf{y} \in \mathcal{P}_2}{\arg\min} \left\| \mathbf{x} - \mathbf{y} \right\| + \sum_{\mathbf{y} \in \mathcal{P}_2} \underset{\mathbf{x} \in \mathcal{P}_1}{\arg\min} \left\| \mathbf{y} - \mathbf{x} \right\|. \tag{4.15}$$

We are interested in computing the Chamfer distance between the point clouds $\mathcal{P}_1 = \mathcal{P}_{\text{bunny}}$ and $\mathcal{P}_2 = \mathcal{P}_{\text{gp}}/\mathcal{P}_{\text{sgp}}$. The point clouds $\mathcal{P}_{\text{gp}}$ and $\mathcal{P}_{\text{sgp}}$ are formed by selecting the vertices from the 0-isosurfaces of h_{gp} and h_{sgp} respectively. Between the point clouds $\mathcal{P}_{\text{bunny}}$ and $\mathcal{P}_{\text{gp}}$, we get a chamfer distance of $d_{\text{ch}}(\mathcal{P}_{\text{bunny}}, \mathcal{P}_{\text{gp}}) = 0.011\text{m}$, whereas between the point clouds $\mathcal{P}_{\text{bunny}}$ and $\mathcal{P}_{\text{sgp}}$, the chamfer distance is $d_{\text{ch}}(\mathcal{P}_{\text{bunny}}, \mathcal{P}_{\text{sgp}}) = 0.042\text{m}$. The results are promising since the chamfer distances are very small and also support the understanding that the GP model expresses the surface more accurately and with more detail than its sparse counterpart.

4.4.2 Robot Manipulator Kinematics

We are interested in controlling a robot manipulator's end-effector while ensuring safety around a complex geometric volumetric object such as the bunny. First, we consider the kinematics of 7-DOF robot manipulators with the state given by the joint position $\mathbf{q}$. The kinematics is given by,

$$\dot{\mathbf{q}} = \mathbf{u}, \tag{4.16}$$

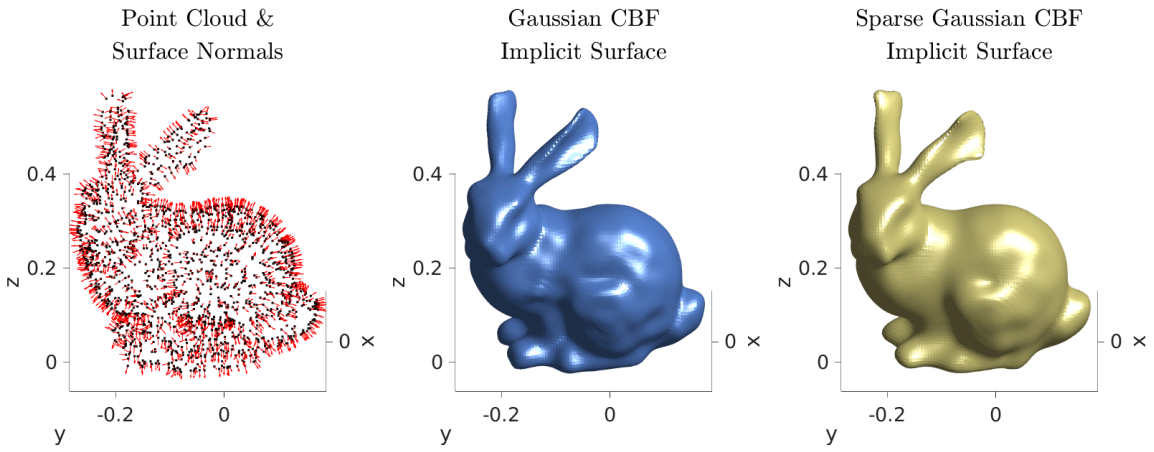

Figure 4.3: The Stanford bunny is modeled as the boundary of the safe set using Gaussian CBFs. (Left) The point cloud and surface normals are shown used for generating the training set. The bunny's implicit surface is modeled using Gaussian CBF (middle) and sparse Gaussian CBF (right).

where $\mathbf{q} \in \mathbb{R}^7$ is the joint position of the robot, and $\mathbf{u} \in \mathbb{R}^7$ is the control input. We assume to have direct control over the joint velocities, $\dot{\mathbf{q}}$, of the manipulator and perform kinematic based control. Hence, the nominal control input is the reference joint velocity trajectory, $\mathbf{u}_{\text{nom}} = \dot{\mathbf{q}}_{\text{ref}}$.

Design of the reference joint velocity is done by interpolating the homogeneous transformation matrix of the end-effector given the initial and desired poses. We simulate the Kinova Gen3 7-DOF manipulator and run our simulation experiment using MATLAB's Robotic Systems Toolbox. We generate reference trajectories such that it goes through the body of the bunny. The synthesized Gaussian CBFs with and without sparsity along with the reference trajectories are shown in Figure 4.5.

4.4.3 Safe Kinematic Control Synthesis for Manipulator

Given the nominal controller and reference trajectory, we want to rectify the nominal control subject to certain safety conditions. The safety objective is for the end-effector of the manipulator to track a reference trajectory without colliding with the bunny in the task space. We use the Gaussian CBF which has been trained to model the bunny's implicit

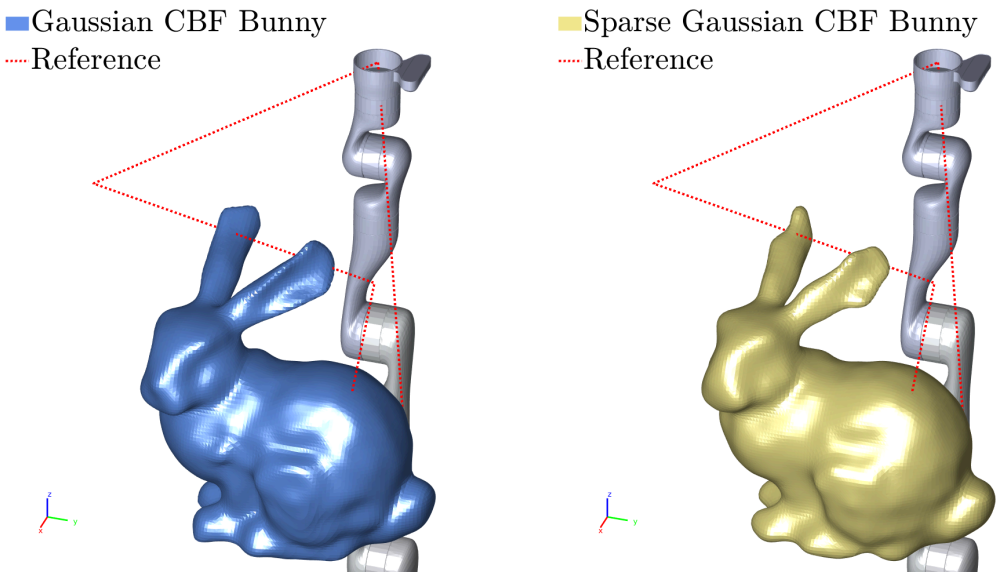

Figure 4.4: The reference trajectory passes through the bunny's back and ear lobes for both the synthesized CBFs.

surface with and without sparsity. Each candidate Gaussian CBF is given as follows,

$$h_{(\cdot)}(\mathbf{x}) = \mu + 2\sigma^2. \tag{4.17}$$

The 0-isosurface of $h_{(\cdot)}(\mathbf{x})$ characterizes the bunny with $(\cdot)$ denoting either the GP model or sparse model. It is desirable to use a high safety margin, quantified using the posterior variance, in order to avoid coming too close to the bunny. The time derivative of the CBF is computed as follows,

$$\begin{aligned}
\dot{h}_{(\cdot)}(\mathbf{x}) &= \frac{\partial h_{(\cdot)}(\mathbf{x})}{\partial \mathbf{q}}\,\dot{\mathbf{q}} \\
&= \frac{\partial h_{(\cdot)}(\mathbf{x})}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{q}}\,\dot{\mathbf{q}} \\
&= \frac{\partial h_{(\cdot)}(\mathbf{x})}{\partial \mathbf{x}}\,\mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}},
\end{aligned}$$

where $\mathbf{J}(\mathbf{q}) : \mathbb{R}^7 \to \mathbb{R}^{3\times 7}$ is the geometric Jacobian relating the joint velocity of the robot, $\dot{\mathbf{q}} \in \mathbb{R}^7$, to the linear velocity of the end-effector and $\frac{\partial h_{(\cdot)}(\mathbf{x})}{\partial \mathbf{x}}$ uses (4.11) for sparse GP and

the partial derivative of (3.1)-(3.2) for regular GP. The QP constraint is given by,

$$L_g h_{(\cdot)}(\mathbf{x})\mathbf{u} + k_0 h_{(\cdot)}(\mathbf{x}) \geq 0 \tag{4.18}$$

$$\underbrace{\frac{\partial h_{(\cdot)}(\mathbf{x})}{\partial \mathbf{x}}\,\mathbf{J}(\mathbf{q})}_{L_g h_{(\cdot)}(\mathbf{x})}\ \underbrace{\dot{\mathbf{q}}}_{\mathbf{u}} \geq -k_0 h_{(\cdot)}(\mathbf{x}). \tag{4.19}$$

With the Lie derivatives, and the decision variable, $\mathbf{u} = \dot{\mathbf{q}}$, appearing linearly in the inequality (4.19), we can rectify the nominal control, $\mathbf{u}_{\text{nom}} = \dot{\mathbf{q}}_{\text{ref}}$, using the QP formulation in (4.13).

4.4.4 Simulation Scenario A : Knowledge of Bunny a priori

The purpose of this simulation experiment is to do trajectory tracking while upholding safety, i.e., not colliding with the bunny. However, if the reference trajectory collides with the bunny, then trajectory tracking is relaxed in order to ensure safety. The control input is rectified as discussed earlier to ensure that the system remains within the safe set. The 0-isosurface of $h_{(\cdot)}(\mathbf{x})$ represents the boundary surface of the bunny, i.e., the safety boundary.

Assumption 9. *We have knowledge of the point cloud and surface normals of the bunny a priori.*

Offline Training: This means we have access to the complete data describing the bunny a priori. The advantage of having the complete information allows us to model and train the (sparse) GP offline. As stated in Section 4.4.1, with $N = 2178$ training points, the training time for computing the bunny's implicit surface took approximately 14s for the GP model with $\mathcal{O}(N^3)$ complexity. For the sparse model, it took 9s to train, with a complexity of $\mathcal{O}(M^2N)$, where half the training points were used as M pseudo-input points.

Online Rectification: The rectified control law is used on the manipulator for both the synthesized Gaussian CBFs as seen in Figure 4.5. The manipulator tracks the reference trajectory when there are no safety violations, i.e., collision with the bunny. When the ma-

nipulator gets too close to the bunny, the rectified control input relaxes reference trajectory tracking. In the bottom left and right of Figure 4.5, we see the plots of $h_{\text{gp}}(t)$ and $h_{\text{sgp}}(t)$ which are always non-negative demonstrating that the manipulator is always inside the safe set. The functions, $h_{\text{gp}}(t)$ and $h_{\text{sgp}}(t)$, both get close to 0 when the manipulator gets very close to the bunny on three occasions - bunny's back and the two ears. It took on average 8ms to compute the Gaussian CBFs at a query point where the query point is taken as the end-effector's location in 3-dimension. Rectifying the control input using QP on average took under 3ms using MATLAB's `quadprog` solver. The total computation time from calculating the CBFs at a query point and rectifying the input took on average 12ms.

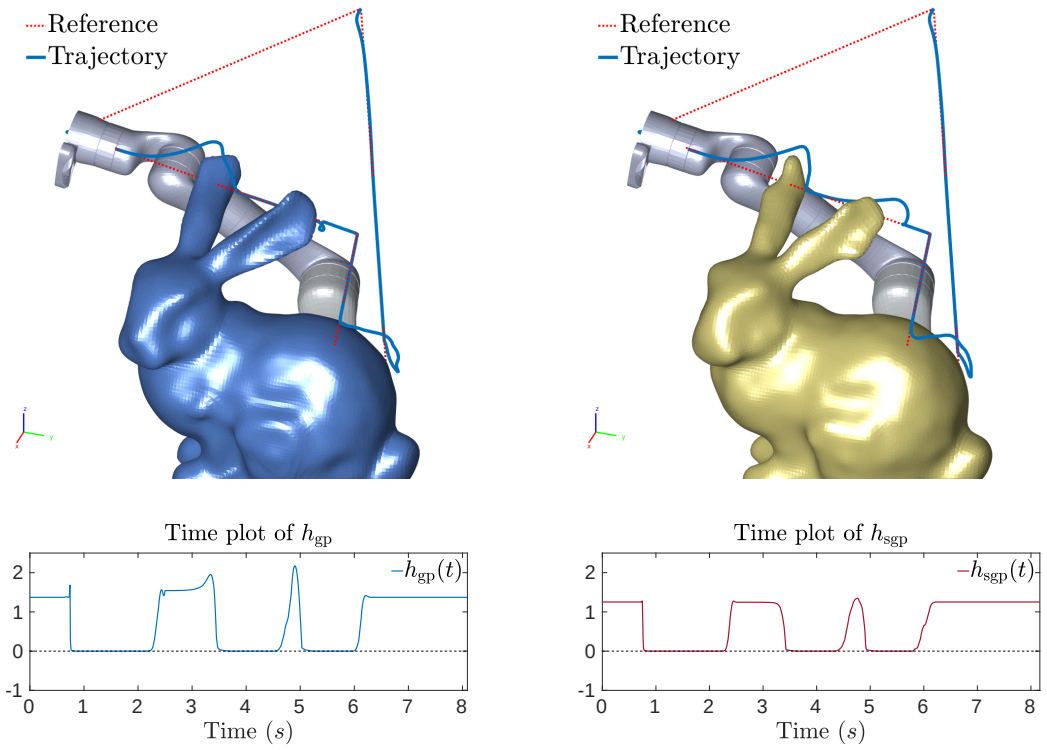

Figure 4.5: The manipulator avoids collision with the bunny but follows the reference trajectory otherwise using Gaussian CBF (top-left) and sparse Gaussian CBF (top-right). The temporal plot of $h_{\text{gp}}(t)$ (bottom-left) and $h_{\text{sgp}}(t)$ (bottom-right) shows that the CBFs are always non-negative.

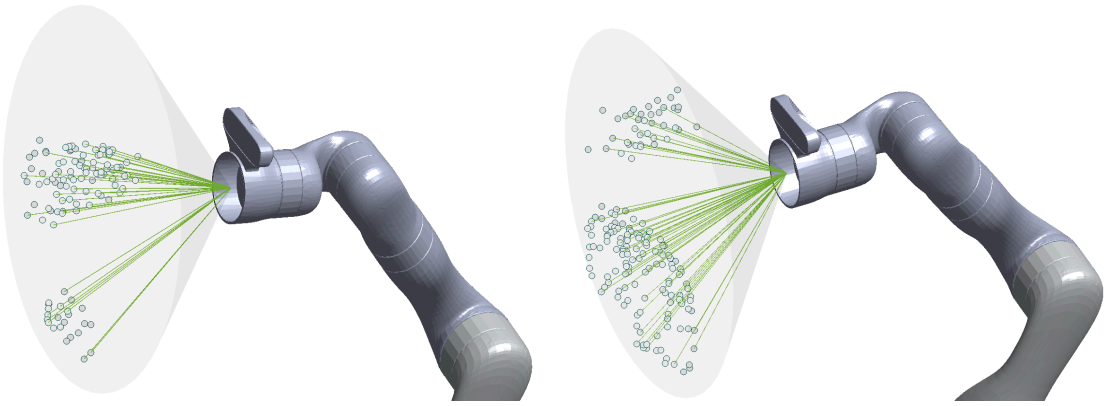

Figure 4.6: A spherical cone model is used as a proximal 3D LiDAR sensor on the robot. The samples detected within the FOV and scanning range are shown (white discs) along with the sensing rays (green lines).

4.4.5 Simulation Scenario B : Proximal Sensing of the Bunny

Now, we relax our previous assumption of having full knowledge of the point cloud a priori. In many applications, having complete knowledge of the obstacles is not always feasible. It is desirable to locally sense for the obstacles and determine the acceptable regions of safety. Moreover, in the previous setting, the implicit surfaces were trained offline which can be a limiting factor. This precludes the possibility of using Gaussian CBFs to model complex volumetric objects online. Here, we perform proximal sensing and train for safe implicit surfaces using local data online.

Assumption 10. *The proximal sensor can sense samples within its FOV and scanning range, and provide the point cloud locations and surface normals in the world frame.*

Proximal Sensing: We introduce a sensing modality to the manipulator in the form of depth scan. We use a spherical cone model to represent the proximal sensor as a 3D LiDAR sensor on the robot. The field-of-view (FOV) of the sensor is taken as 110° with a scanning range of 0.8m. The sensor is mounted on the end-effector. In Figure 4.6, two examples of the spherical cone model is shown to detect samples. A sample is detected if it lies within the FOV and scanning range of the sensor.

Online Training & Rectification: The simulation experiment is repeated for the same

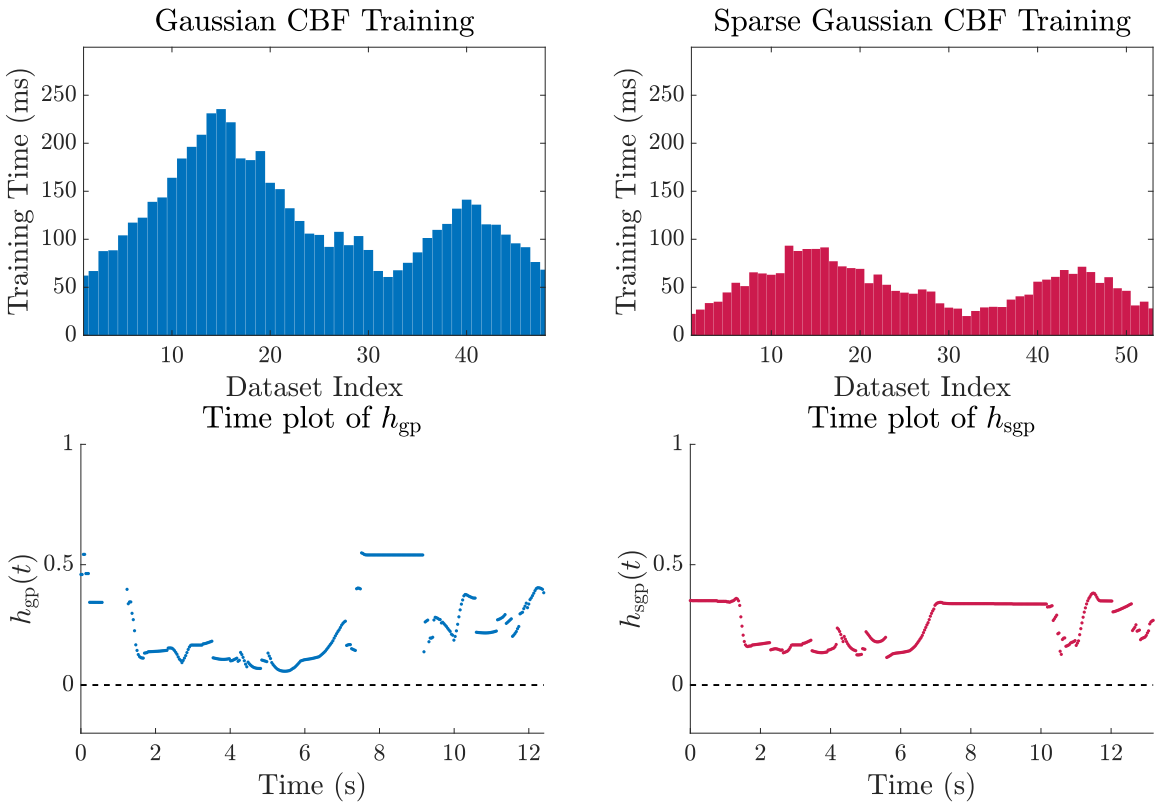

Figure 4.7: The training times per local dataset is shown for Gaussian CBF and sparse Gaussian CBF (top). The time plots of both the CBFs (bottom) are always non-negative, denoting that no safety violations occurred.

reference trajectory in Figure 4.5. Local training datasets are formed by detecting samples using the sensor. A local dataset is formed if a minimum of 200 samples are detected. Sparse Gaussian CBF uses one-half of the local training dataset as pseudo-inputs and targets. With Gaussian CBF, 48 local datasets were formed with an average training time of 124ms per dataset. For the sparse variant, 53 local datasets were formed with an average training time of 52ms per dataset. The training times for both the CBFs are shown in Figure 4.7. Gaussian CBF has $\mathcal{O}(N^3)$ complexity, where N is the number of training points per local dataset. Whereas, sparse Gaussian CBF has $\mathcal{O}(M^2N)$ complexity, where M is the number of pseudo-inputs per dataset. We compute the covariance matrix and store in memory until the next dataset is formed. This gives a prediction complexity of $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ for the GP mean and variance respectively. Similarly, for the sparse model, the complexities are $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ for the predictive mean and variance respectively.

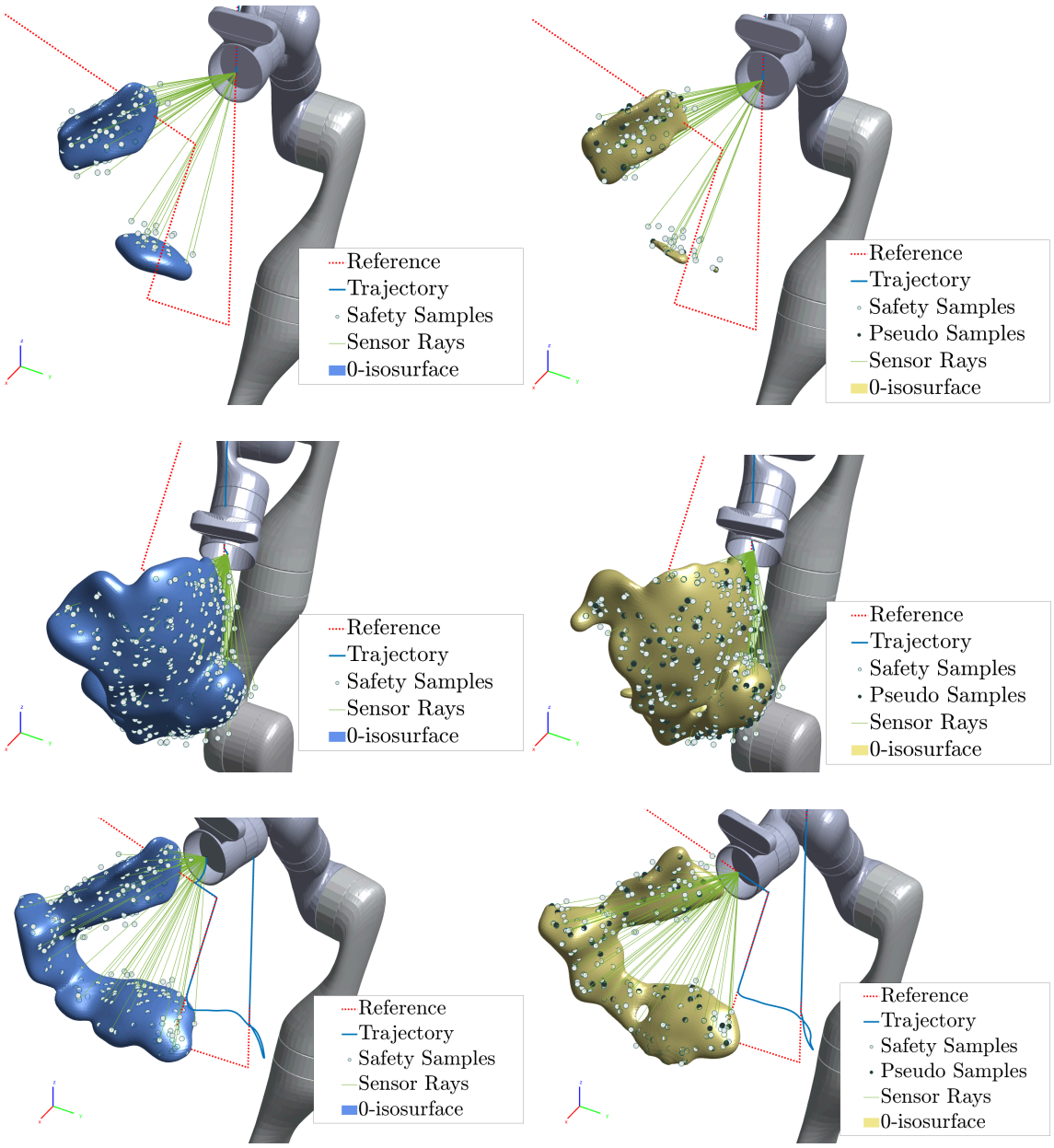

Figure 4.8: (Left) Gaussian CBFs are trained online with local training datasets. A local dataset is formed by detecting point cloud samples (white discs) using the proximal sensor. (Right) Sparse Gaussian CBFs are trained with the help of pseudo-inputs (black discs) and local point cloud data. For both the CBFs, corresponding 0-isosurfaces are shown to illustrate their modeling capability.

Figure 4.7 also shows the time plots of the CBFs, which are always non-negative, thus indicating that no safety constraint was violated. The average inference time on a query point, which was chosen to be the end-effector's position, was approximately 5ms for both

the CBFs.

Discussion: Three instances of the simulation are shown in Figure 4.8 for each data-driven CBF. Gaussian CBF generates isosurfaces with a higher granularity as opposed to sparse Gaussian CBFs as expected. For instance, when the sensor detects the neck and left ear of the bunny, the sparse Gaussian CBF models the surface with lesser precision than the Gaussian CBF. However, sparse Gaussian CBF still models the safety surface and ensures that the manipulator does not collide with the bunny. Notice that the data-driven generation of the CBFs is not limited to convex or connected surfaces. As can be seen in the figure, disconnected isosurfaces can be modeled also using GPs.

4.5 Test Case II : 3D Quadrotor Hardware

We now validate our proposed methodology for generating safe surfaces from data and performing safe control using Gaussian CBFs on a Crazyflie 2.1 hardware quadrotor. The safety objective is avoid collision with a static chair while teleoperating the quadrotor indoors. This is done by constructing the Gaussian CBF modeling a chair and achieving safe constrained control.

4.5.1 Experimental Setup

We use the Crazyflie 2.1 as the hardware quadrotor which is a versatile open-source platform weighing only $27\mathrm{g}$ [65]. State estimation is performed onboard via an external low-cost lighthouse positioning system [65]. The Crazyflie has a maximum payload capacity of $15\mathrm{g}$. Consequently, it will not be possible to equip the quadrotor with an external depth scanner or 3D LiDAR sensor to perform online sensing with a hardware sensor. Therefore, we use a virtual range sensor modeled as a spherical cone, as done in Section 4.4.5, for sensing the obstacle in Section 4.5.6.

We use an IKEA ADDE chair as the testing apparatus in our collision avoidance scenario. The point cloud data and surface normals are extracted from the chair's OBJ file

freely available online [94]. The chair comes with a large gap in its backrest along with numerous holes in the seat and backrest. These holes allow accurate state estimation for Crazyflie while flying beneath or behind the chair. The lighthouse system uses infrared rays for position state estimation, and as a result, requires having line-of-sight for the quadrotor.

We use a remote ground station for GP training, Gaussian CBF synthesis, sensor modeling, and control rectification. The ground station has an Intel i7-9800X at 4.4GHz processor and 16 GB RAM. Communication with Crazyflie happens using a bluetooth dongle where state estimates are received at 100Hz and rectified setpoint commands are sent at 100Hz with zero-order hold. Control rectification is performed at 50Hz, virtual sensing at 100Hz, and GP training at 20Hz for the online case. Nominal setpoint commands are sent to the Crazyflie at 100Hz with the help of a Logitech joystick controller. These nominal setpoints act as the nominal controller in the QP formulation (see Section 2.5.2).

4.5.2 Chair Implicit Surface Modeling

First, we discuss the modeling of the static chair for the hardware experiment. Since we are unable to use a physical depth scanner on the Crazyflie, we rely on virtual point cloud data. We first download the chair's OBJ file [94] and load it onto an open source 3D mesh processing software system such as MeshLab. Using MeshLab, we extract 10000 point cloud and surface normal datapoints for the chair. We use the Matérn kernel, which is a generalization of the SE kernel, to model the chair.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left(1 + \frac{\sqrt{3} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel}{l}\right) \exp\left(-\frac{\sqrt{3} \parallel \mathbf{x}_i - \mathbf{x}_j \parallel}{l}\right) + \delta_{ij}\sigma_\omega^2. \tag{4.20}$$

In practice, the SE kernel has very strong smoothness assumptions, and it may not be ideal for many physical phenomena [60]. The Matérn kernel is often recommended in practice since it characterizing physical processes better [60]. The class of Matérn kernels can control the smoothness properties unlike the SE kernel which is infinitely smooth.

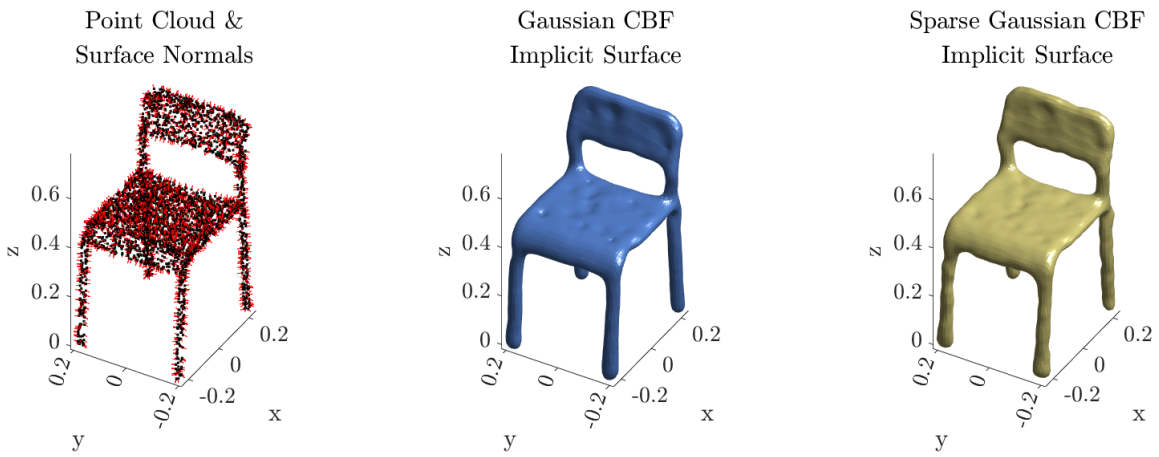

Figure 4.9: The chair is modeled as the boundary of the safe set using Gaussian CBFs. (Left) The point cloud and surface normals are shown used for generating the training set. The chair's implicit surface is modeled using Gaussian CBF (middle) and sparse Gaussian CBF (right).

We downsample the point cloud to 2201 datapoints randomly from the original 10000 datapoints and treat one-third of the datapoints as pseudo inputs and targets. The average training times were 4.8s for regular GP and 2.72s for sparse GP with 15 iterative steps to maximize the log marginal likelihood during hyperparameter optimization using (4.6). We use the following CBF,

$$h_{(.)}(\mathbf{x}) = \mu + 4\sigma^2. \tag{4.21}$$

The point cloud with surface normals and corresponding GP based implicit surfaces used as Gaussian CBFs are shown in Figure 4.9. We do not attempt to mimic the gaps and holes smaller than 0.05m in the point cloud data since the quadrotor cannot fly through those regions. The corresponding chamfer distances between $\mathcal{P}_{\text{chair}}$ and $\mathcal{P}_{\text{gp}}$/$\mathcal{P}_{\text{sgp}}$ are 0.078m and 0.0952m respectively using (4.15). Although the chamfer distances are still small, the reason for their relatively larger values compared to the bunny is due to a much higher variance margin used.

4.5.3 Matérn Kernel and Partial Derivatives

Here, we provide the equations of Matérn kernel's Jacobian and Hessian for parameter $\nu = 3/2$ with respect to $\mathbf{x} \in \mathbb{R}^n$. The kernel is expressed for $\nu = 3/2$ as follows [59],

$$k_{\nu=3/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \left(1 + \frac{\sqrt{3}}{l}\|\mathbf{x}_i - \mathbf{x}_j\|\right) \exp\left(-\frac{\sqrt{3}}{l}\|\mathbf{x}_i - \mathbf{x}_j\|\right),$$

for $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$, $i, j \in \{1, \ldots, N\}$, N being the number of samples, and $\sigma_f \in \mathbb{R}$ and $l \in \mathbb{R}$ being the signal variance and characteristic length scale hyperparameters respectively for the kernel. We rewrite the equation above as follows,

$$k_{\nu=3/2}(t) = \sigma_f^2 \big(1 + t\big) \exp\big(-t\big), \tag{4.22}$$

where $t(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sqrt{3}}{l}\|\mathbf{x}_i - \mathbf{x}_j\| \in \mathbb{R}$. We now derive the first and second order partial derivatives of t with respect to $\mathbf{x}$ at a query point $\mathbf{x}_*$,

$$\left.\frac{\partial t(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} = -\frac{\sqrt{3}}{l}\frac{(\mathbf{x} - \mathbf{x}_*)^\top}{\|\mathbf{x} - \mathbf{x}_*\|} \tag{4.23}$$

$$\left.\frac{\partial^2 t(\mathbf{x})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}_*} = \frac{\sqrt{3}}{l}\frac{\mathrm{diag}(\mathbf{1})}{\|\mathbf{x} - \mathbf{x}_*\|} + \frac{\sqrt{3}}{l}\frac{(\mathbf{x} - \mathbf{x}_*)(\mathbf{x} - \mathbf{x}_*)^\top}{\|\mathbf{x} - \mathbf{x}_*\|^3}, \tag{4.24}$$

where $\frac{\partial t(\mathbf{x})}{\partial \mathbf{x}} \in \mathbb{R}^{1\times n}$, $\frac{\partial^2 t(\mathbf{x})}{\partial \mathbf{x}^2} \in \mathbb{R}^{n\times n}$, and $\mathrm{diag}(\mathbf{1}) \in \mathbb{R}^{n\times n}$ is a diagonal matrix of ones. The Matérn kernel's Jacobian and Hessian in (4.22) with respect to $\mathbf{x}$ at a query point $\mathbf{x}_*$ are then computed by,

$$\left.\frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*} = -\sigma_f^2\, t \exp(-t) \left.\frac{\partial t(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*}, \tag{4.25}$$

$$\left.\frac{\partial^2 \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}_*} = -\sigma_f^2\, t \exp(-t) \left.\frac{\partial^2 t(\mathbf{x})}{\partial \mathbf{x}^2}\right|_{\mathbf{x}_*} + \sigma_f^2 (t-1) \exp(-t) \left.\frac{\partial t(\mathbf{x})}{\partial \mathbf{x}}^\top\right|_{\mathbf{x}_*} \left.\frac{\partial t(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}_*}, \tag{4.26}$$

where (4.23) and (4.24) are used.

4.5.4 Safe Control Synthesis for Quadrotor with Sparsity

The associated Lie derivatives for the Gaussian CBF (with and without sparsity) in (4.21) are,

$$L_f h_{(.)}(\mathbf{x}) = \Big(\nabla\mu(\mathbf{x}) - \nabla\sigma^2(\mathbf{x})\Big)^\top f(\mathbf{x}), \tag{4.27}$$

$$L_f^2 h_{(.)}(\mathbf{x}) = f(\mathbf{x})^\top \Big(\mathbf{H}_\mu(\mathbf{x}) - \mathbf{H}_{\sigma^2}(\mathbf{x})\Big) f(\mathbf{x}) + \Big(\nabla\mu(\mathbf{x}) - \nabla\sigma^2(\mathbf{x})\Big)^\top \nabla f(\mathbf{x}) f(\mathbf{x}), \tag{4.28}$$

$$L_g L_f h_{(.)}(\mathbf{x}) = f(\mathbf{x})^\top \Big(\mathbf{H}_\mu(\mathbf{x}) - \mathbf{H}_{\sigma^2}(\mathbf{x})\Big) g(\mathbf{x}) + \Big(\nabla\mu(\mathbf{x}) - \nabla\sigma^2(\mathbf{x})\Big)^\top \nabla f(\mathbf{x}) g(\mathbf{x}), \tag{4.29}$$

where $\nabla\mu(\mathbf{x}) = \frac{\partial\mu(\mathbf{x})}{\partial\mathbf{x}}^\top$ and $\nabla\sigma^2(\mathbf{x}) = \frac{\partial\sigma^2(\mathbf{x})}{\partial\mathbf{x}}^\top$ are the gradients of the respective GP mean and variance with and without sparsity, and $\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}$ is the Jacobian of $f(\mathbf{x})$. $\mathbf{H}_\mu(\mathbf{x})$ and $\mathbf{H}_{\sigma^2}(\mathbf{x})$ are the Hessians of the respective Gaussian CBFs. For the Gaussian CBF, the Hessians are as follows.

$$\text{GP Hessian } \begin{cases} \mathbf{H}_{\mu_{\text{gp}}}(\mathbf{x}) & = \Big(\sum_i^N a_{\text{gp}}^i \frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}\Big), \\ \mathbf{H}_{\sigma^2_{\text{gp}}}(\mathbf{x}) & = -2\nabla\mathbf{k}(\mathbf{x})\overline{\mathbf{K}}^{-1}\nabla\mathbf{k}(\mathbf{x})^\top - 2\Big(\sum_i^N b_{\text{gp}}^i(\mathbf{x}) \frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}\Big), \end{cases}$$

where a_{gp}^i is the i^{th} entry of $\mathbf{y}_N^\top \overline{\mathbf{K}}^{-1} \in \mathbb{R}^{1\times N}$, $b_{\text{gp}}^i(\mathbf{x})$ is the i^{th} entry of $\mathbf{k}(\mathbf{x})^\top \overline{\mathbf{K}}^{-1} \in \mathbb{R}^{1\times N}$, and $\nabla\mathbf{k}(\mathbf{x}) = \frac{\partial\mathbf{k}(\mathbf{x})}{\partial\mathbf{x}}^\top \in \mathbb{R}^{n\times N}$ is the kernel Jacobian using (4.25). For the sparse Gaussian CBF, the Hessians are given by,

$$\text{Sparse GP Hessian } \begin{cases} \mathbf{H}_{\mu_{\text{sgp}}}(\mathbf{x}) & = \Big(\sum_i^M a_{\text{sgp}}^i \frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}\Big), \\ \mathbf{H}_{\sigma^2_{\text{sgp}}}(\mathbf{x}) & = -2\nabla\mathbf{k}_M(\mathbf{x})\big(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1}\big)\nabla\mathbf{k}_M(\mathbf{x})^\top \\ & \quad -2\Big(\sum_i^M b_{\text{sgp}}^i(\mathbf{x}) \frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}\Big), \end{cases}$$

where a^i_{sgp} is the i^{th} entry of $\mathbf{y}_N^\top \big(\mathbf{Q}_M^{-1}\ \mathbf{K}_{MN}\big(\mathbf{\Lambda}_N + \sigma_\omega^2 \mathbf{I}_N\big)^{-1}\big)^\top \in \mathbb{R}^{1\times M}$, $b^i_{\text{sgp}}(\mathbf{x})$ is the i^{th} entry of $\mathbf{k}_M(\mathbf{x})^\top \big(\mathbf{K}_M^{-1} - \mathbf{Q}_M^{-1}\big) \in \mathbb{R}^{1\times M}$, $\nabla \mathbf{k}_M(\mathbf{x}) = \frac{\partial \mathbf{k}(\mathbf{x})}{\partial \mathbf{x}}^\top \in \mathbb{R}^{n\times M}$ is the kernel Jacobian using (4.25), and $\frac{\partial^2 \mathbf{k}_{(i)}(\mathbf{x})}{\partial \mathbf{x}^2}$ is the kernel Hessian using (4.26). Given the nominal control input $\mathbf{u}_{\text{nom}} \in \mathbb{R}^3$ in (2.20)-(2.22), the QP below rectifies $\mathbf{u}_{\text{nom}}$ into $\mathbf{u}_{\text{rect}} \in \mathbb{R}^3$,

---

(Sparse) Gaussian CBF-QP: *Quadrotor Input modification*

$$\begin{aligned}\mathbf{u}_{\text{rect}} = \underset{\mathbf{u}\ \in\ \mathbb{R}^3}{\arg\min}\ \frac{1}{2}\big\|\mathbf{u} - \mathbf{u}_{\text{nom}}\big\|^2 \ \text{ s. t.}\\ L_f^2 h_{(\cdot)}(\mathbf{x}) + L_g L_f h_{(\cdot)}(\mathbf{x})\mathbf{u} + \mathcal{K}^\top \mathcal{H} \geq 0,\end{aligned} \tag{4.30}$$

---

where $\mathcal{K} = [k_1\ k_0]^\top \in \mathbb{R}^2$ is the coefficient gain vector, and $\mathcal{H} = [L_f h_{(\cdot)}(\mathbf{x})\ h_{(\cdot)}(\mathbf{x})]^\top \in \mathbb{R}^2$ is the Lie derivative vector. The rectified input $\mathbf{u}_{\text{rect}}$ is used to compute the rectified setpoints using (2.23)-(2.25) which are then sent to Crazyflie 2.1.

4.5.5 Scenario A: Safe Teleoperation with Knowledge of Chair

In this scenario, we conduct safe teleoperation where the safety objective is for the Crazyflie 2.1 to fly through and around the chair without colliding with it. A human operator controls the Crazyflie 2.1 using a Logitech Joystick providing nominal control inputs. The control input is then rectified using the scheme discussed in Section 4.5.4. We operate under the following assumption for this scenario.

Assumption 11. *We have knowledge of the point cloud and surface normals of the chair a priori.*

Offline Training: We train the GP model with and without sparsity to generate the 0-isosurface of the chair. We used 2201 datapoints to train the GP model, and one-third of the datapoints were initialized as pseudo inputs and targets to train the sparse model. After training, the 0-isosurface serves as the (sparse) Gaussian CBF in our experiment. For the complete GP model, the training time was 4.8sec with a complexity of $\mathcal{O}(N^3)$ for $N = 2201$ training points, whereas for the sparse GP model, it was 2.72sec, with a

complexity of $\mathcal{O}(M^2N)$, for $M = 734$ pseudo inputs. Hyperparameter optimization was performed using 15 iterative steps to maximize the log marginal likelihood. We did not witness a noticeable performance improvement in the chamfer distance of the chair by increasing the number of iterations during hyperparameter selection.

Online Rectification: We conduct separate experiments using (4.21) for Gaussian CBF and sparse Gaussian CBF. The quadrotor trajectories with rectification for each experiment are shown in Figure 4.11. For each flight experiment, as seen in the figure, the trajectory of the quadrotor does not collide with the chair. This can be verified by looking at the temporal plot of $h_{\text{gp}}(t)$ and $h_{\text{sgp}}(t)$ for each flight trajectory, where both the Gaussian CBFs are always non-negative. The average computation time for the Gaussian CBF, which also includes Jacobian and Hessian calculations, along with QP rectification is also shown in Figure 4.11. Sparse Gaussian CBFs have an average computation time of 12.3ms, whereas regular Gaussian CBF takes 24.27ms. This is because the complexity of sparse GPs is

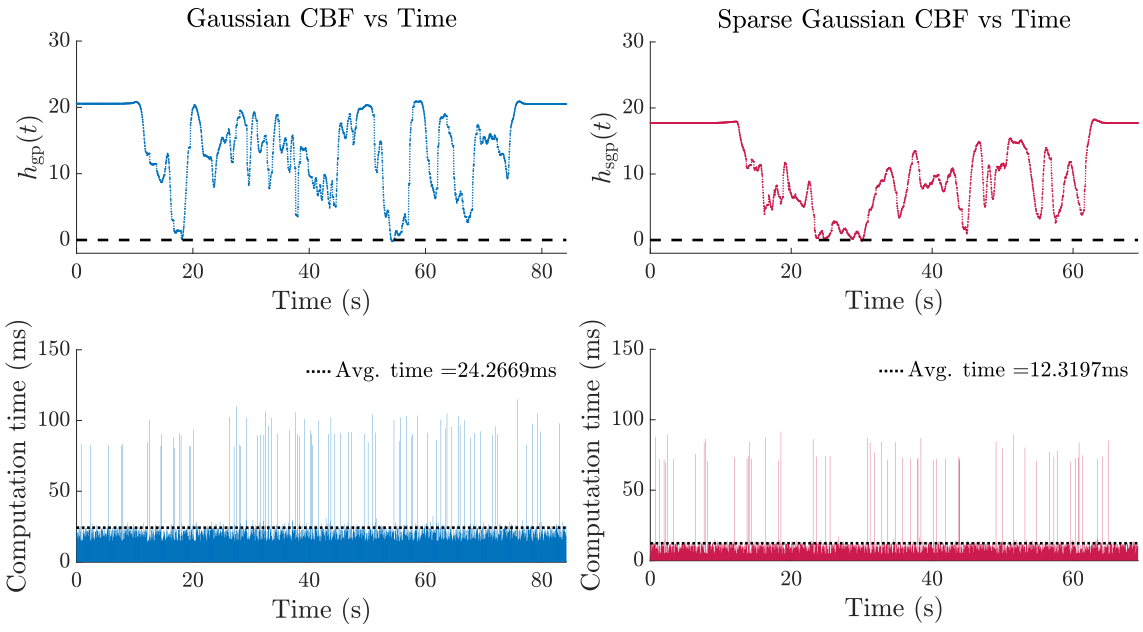

Figure 4.10: Safe control on Crazyflie 2.1 using Gaussian CBF (left) and sparse Gaussian CBF (right) on two separate experiments. The CBFs ensure that Crazyflie does not collide with the chair, as seen by $h_{\text{gp}}(t)$ and $h_{\text{sgp}}(t)$ being non-negative at all times (top). Sparse Gaussian CBF exploits sparsity and has a faster average computation time compared to Gaussian CBF (bottom).

$\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ for predictive mean and variance, where M is the pseudo inputs, after inverse pre-computations and storage. For regular GPs the complexity is $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ for the predictive mean and variance, where N is the number of training points.

Discussion: As the human operator would bring the quadrotor close to the 0-isosurface of the chair, the rectification routine would drive the quadrotor away. Since the human operator is not an expert in maneuvering the quadrotor, using Gaussian CBFs as a safety layer for collision avoidance has practical significance. Repeated flight experiments can be done in a safe manner without incurring any expensive hardware failures. Gaussian CBFs provide a way to characterize complex geometrical shapes using data, thus alleviating the need to hand-design candidate CBFs. In the bottom plot of Figure 4.11, we see spikes in computation times. This could be attributed to the use of a general purpose Linux operating system as opposed to a real-time operating system (RTOS). With RTOS, there would be finer precision control on routine executions, multi-processing threads, and clock cycles which we deem fit for future work. On comparing the two flight experiments, sparse Gaussian CBFs are computationally faster at the expense of reduced modeling accuracy. However, both the CBFs ensured that the Crazyflie does not collide with the chair.

4.5.6 Scenario B: Safe Teleoperation with Promixal Sensing of Chair

Next, we look at the same study but perform proximal sensing of the chair. In order to achieve real time safe control with at least $20\mathrm{Hz}$, the GP training cannot exceed $50\mathrm{ms}$. This limits the number of samples used only to a few hundred datapoints. Consequently, we conduct this experiment only using Gaussian CBF due to the poor scalability of sparse GPs at few datapoints.

Relying on Assumption 10, we use a virtual sensor to proximally sense the virtual point cloud and surface normals of the physical chair in the world frame. The sensor is represented using a spherical cone model with its cone axis aligned with Crazyflie's body-frame x-axis. Similar to the previous experiment, a human operator teleoperates the Crazyflie pro-

viding the nominal control input which is then rectified using locally synthesized Gaussian CBF.

Online Training & Rectification: As the quadrotor flies, the virtual sensor detects point cloud and surface normal datapoints. These datapoints are used to construct local datasets. During our hardware experiments, we found that a minimum of 100 samples were needed to get good implicit surface estimation using GP. On another thread, we perform Gaussian CBF synthesis and control rectification at 50Hz. We limit the local dataset to a 100 datapoints, and randomly sample 100 datapoints whenever the sensor detects large numbers of point cloud samples. In Fig. 4.12, we plot the Gaussian CBF as a function of time along with computation times for synthesizing the Gaussian CBF and training the GP. The temporal plot of Gaussian CBF is always non-negative as seen in the figure. The average time for synthesizing the CBF and rectify the control input was under 6.5ms, while the average training time for the GP was 48ms.

Two instances of the experiment's online training and rectification are visualized in Fig. 4.13. Data recorded using `rosbag` is visualized to show Crazyflie's body-frame axis

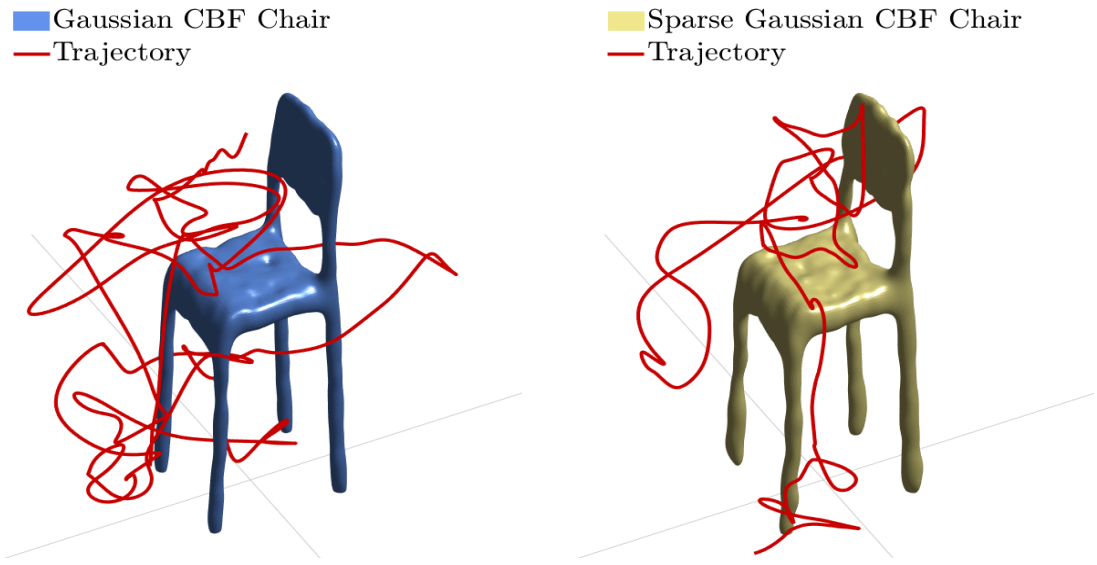

Figure 4.11: Safe teleoperated flight done on the Crazyflie 2.1 using Gaussian CBF (left) and sparse Gaussian CBF (right). The experimental hardware trajectories are shown along with the chair's 0-isosurface.

(RGB triad), flight trajectory, samples detected, and local implicit surface estimates. The virtual point cloud information and samples detected with the virtual proximal sensor are highlighted. The sensor locally detects, from which local datasets are formed and trained online to get the 0-isosurface. Control rectification done using the Gaussian CBFs allows the quadrotor to perform safe control and avoid colliding with the chair during runtime.

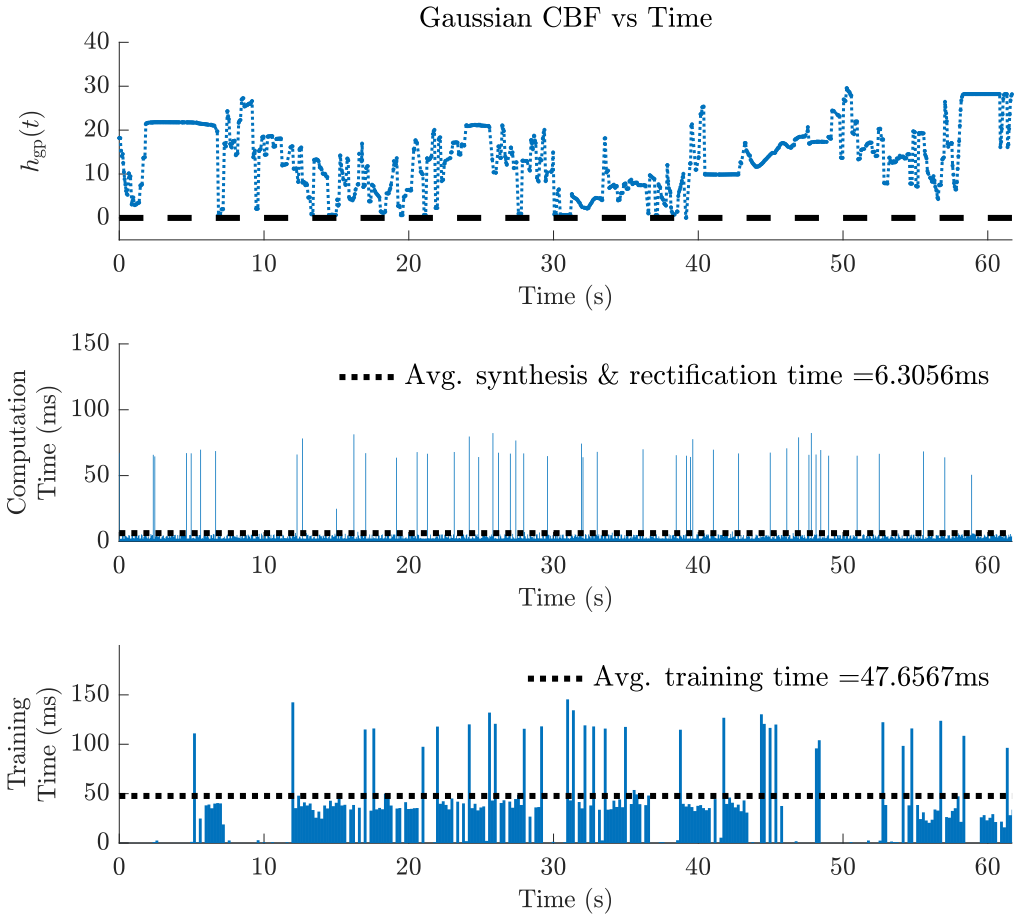

Figure 4.12: (Top) The Gaussian CBF is non-negative verifying that the quadrotor does not collide with the chair during runtime. (Middle) The average time for synthesis and QP rectification was 6.3ms, while training the GP for implicit surface estimation was under 48ms per dataset.

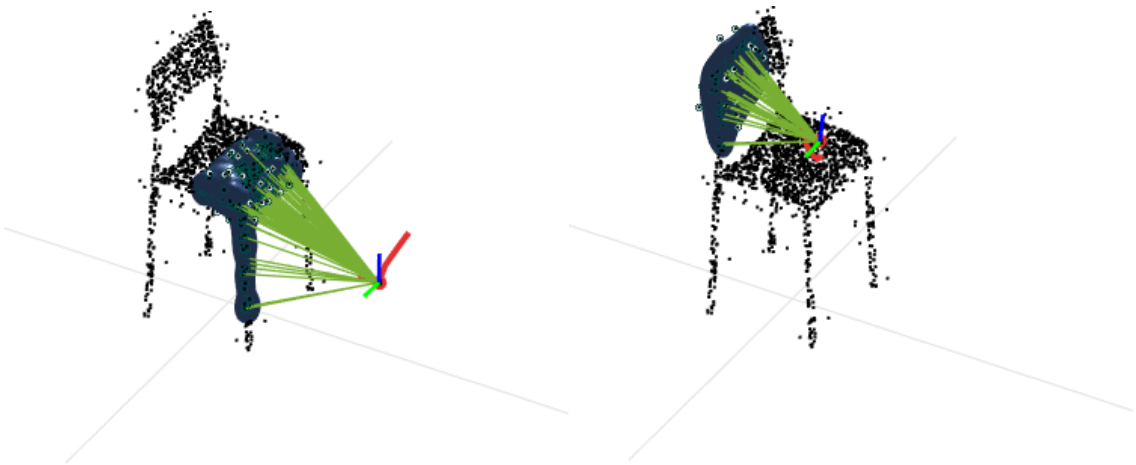

Figure 4.13: Gaussian CBF is trained online with local dataset. Local datasets are capped to 100 samples to achieve online training. The body-frame axes of Crazyflie 2.1 is shown with RGB triad.

4.5.7 Scenario A: Safe Autonomous Navigation

In this scenario, the Crazyflie is subjected to unsafe reference trajectories which go through the chair. The objective is to track the reference trajectory while considering the safety objective of not colliding with the chair. We design velocity setpoints using a proportional-integral (PI) controller. The nominal control input is given by $\mathbf{u}_{\text{nom}} = \dot{\mathbf{r}}_{\text{des}} \in \mathbb{R}^3$, with the dynamics given by a single integrator, i.e, $\dot{\mathbf{r}} = \mathbf{u}$. Since the relative degree is $\rho = 1$, we use the following QP setup,

---

(Sparse) Gaussian CBF-QP: *Quadrotor Velocity Input modification*

$$\begin{aligned} \mathbf{u}_{\text{rect}} = \underset{\mathbf{u} \in \mathbb{R}^3}{\arg\min} \frac{1}{2}\left\|\mathbf{u} - \mathbf{u}_{\text{nom}}\right\|^2 \ \text{ s. t.} \\ L_f h_{(.)}(\mathbf{x}) + L_g h_{(.)}(\mathbf{x})\mathbf{u} + \alpha h_{(.)}(\mathbf{x}) \geq 0. \end{aligned} \tag{4.31}$$

We use Gaussian CBFs and sparse Gaussian CBFs on 3 separate runs. Each run uses a different unsafe reference trajectory. The reference trajectory is designed by giving a final desired position and then the PI controller calculates the desired setpoint for every sampling time. Gaussian CBF rectification is performed at 50Hz.

The first experimental run is to fly the Crazyflie diagonally through the legs of the

chair. The flight behavior is shown in Figure 4.14. Given a desired final position of $\mathbf{r}_{\text{des}} = [0.5, 0.5, 0.25]^\top \text{m}$, with an initial position of $\mathbf{r}_{\text{init}} = [-0.5, -0.5, 0]^\top$, the Crazyflie relaxes trajectory tracking in order to avoid collision. The temporal plots of $h_{\text{gp}}(t)$ and $h_{\text{sgp}}(t)$ shown in Figure 4.14 verifies that the CBF is non-negative throughout the motion, thus verifying the safety performance of the Gaussian CBFs. The computation time is only 10ms for Gaussian CBF, and almost half of that for the sparse Gaussian CBF at 6ms.

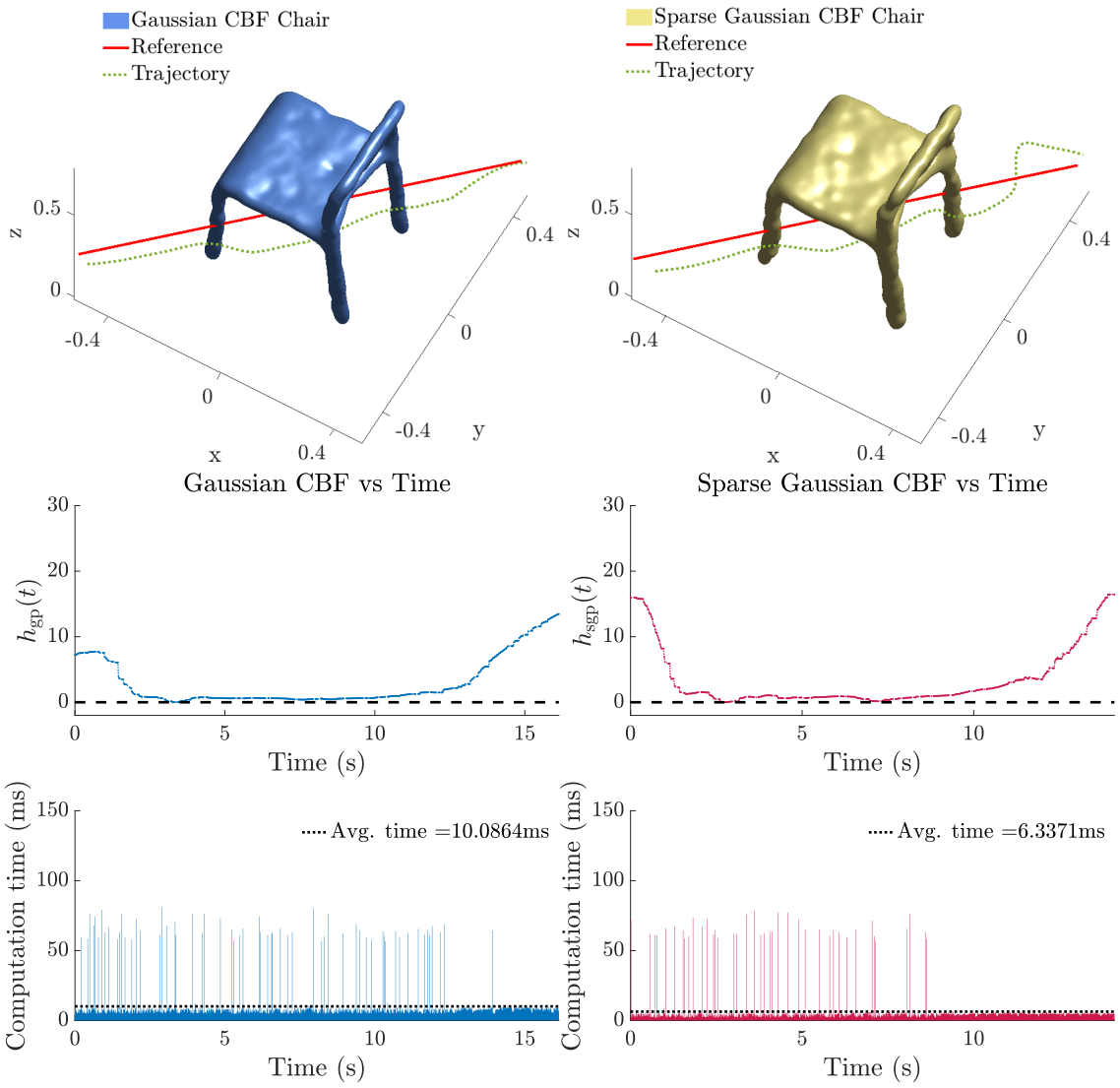

Figure 4.14: Experiment run 1 where the Crazyflie flies diagonally through the front-right and rear-left legs of the chair. Both the reference and actual trajectories are shown (top). The time plots of h_{gp} and h_{sgp} show that the CBFs are always non-negative (middle), and the computation time per iteration is plotted (bottom).

The second experimental run is shown in Figure 4.15 where the reference trajectory goes straight through the front-left and rear-left legs of the chair. We give a desired final position of $\mathbf{r}_{\text{des}} = [0.5, 0.2, 0.25]^\top \text{m}$ with an initial position of $\mathbf{r}_{\text{init}} = [-0.75, 0.2, 0.25]^\top$. For both the Gaussian CBF formulations, the Crazyflie successfully avoids colliding with the chair. The average computation times are similarly 10ms and 6.5ms for the Gaussian CBFs with and without sparsity respectively.

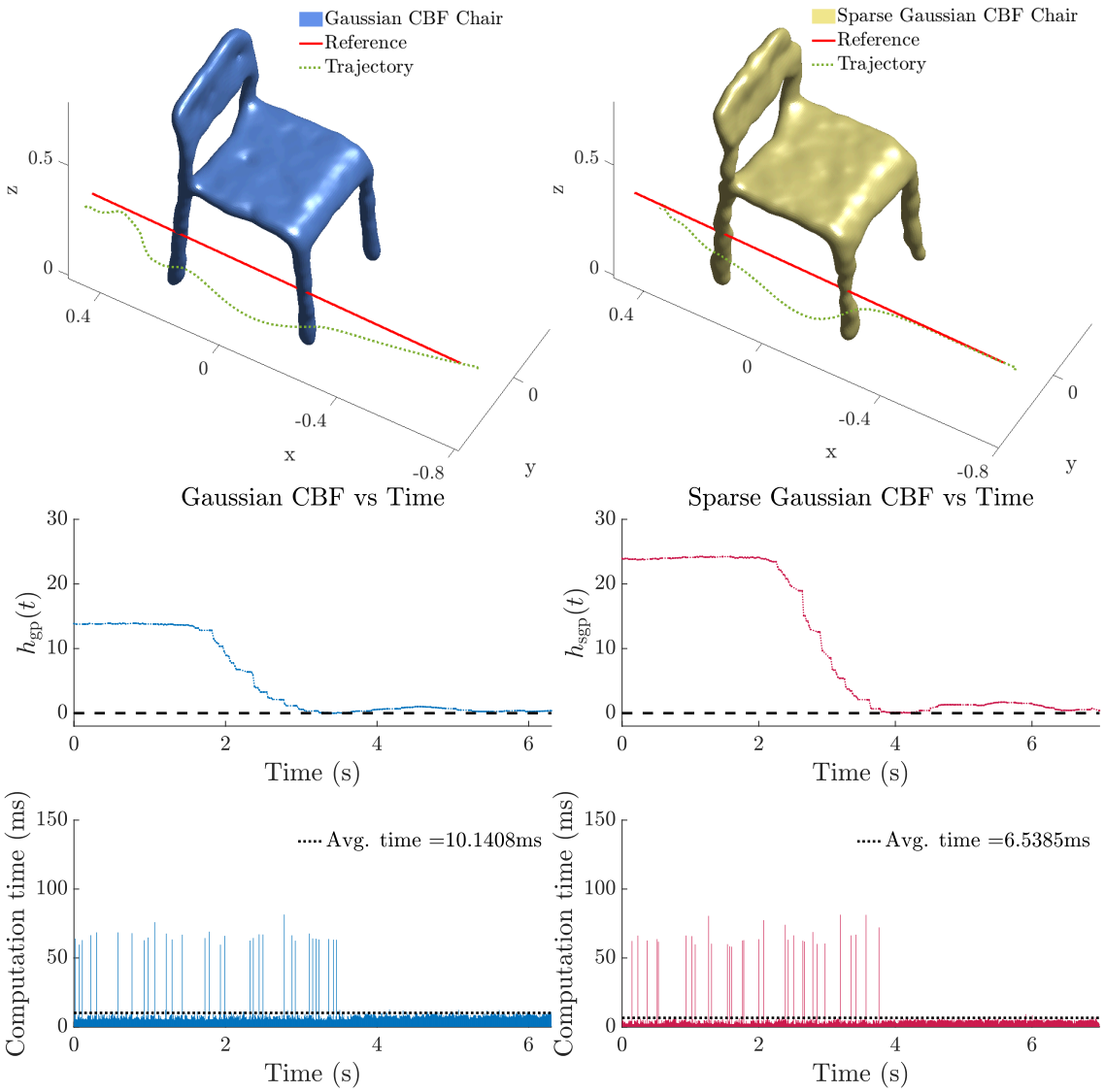

Figure 4.15: Experiment run 2 where the Crazyflie flies straight through the front-left and rear-left legs of the chair. Both the reference and actual trajectories are shown (top). The time plots of h_{gp} and h_{sgp} show that the CBFs are always non-negative (middle), and the computation time per iteration is plotted (bottom).

In the final experimental run, the Crazyflie's reference trajectory is made to go through the headrest of the chair. The initial hovering position is set as $\mathbf{r}_{\text{init}} = [-0.75, 0, 0.65]^\top$ with a final desired hovering position of $\mathbf{r}_{\text{des}} = [0.5, 0, 0.65]^\top$. In Figure 4.16, the flight behavior is shown along with the CBFs temporal plots and average computation times for each CBF per iteration. Crazyflie executes a motion where it goes below the headrest by relaxing reference tracking. The corresponding h_{gp} and h_{sgp} plots show that the safety value gets close to 0 but does not become negative.

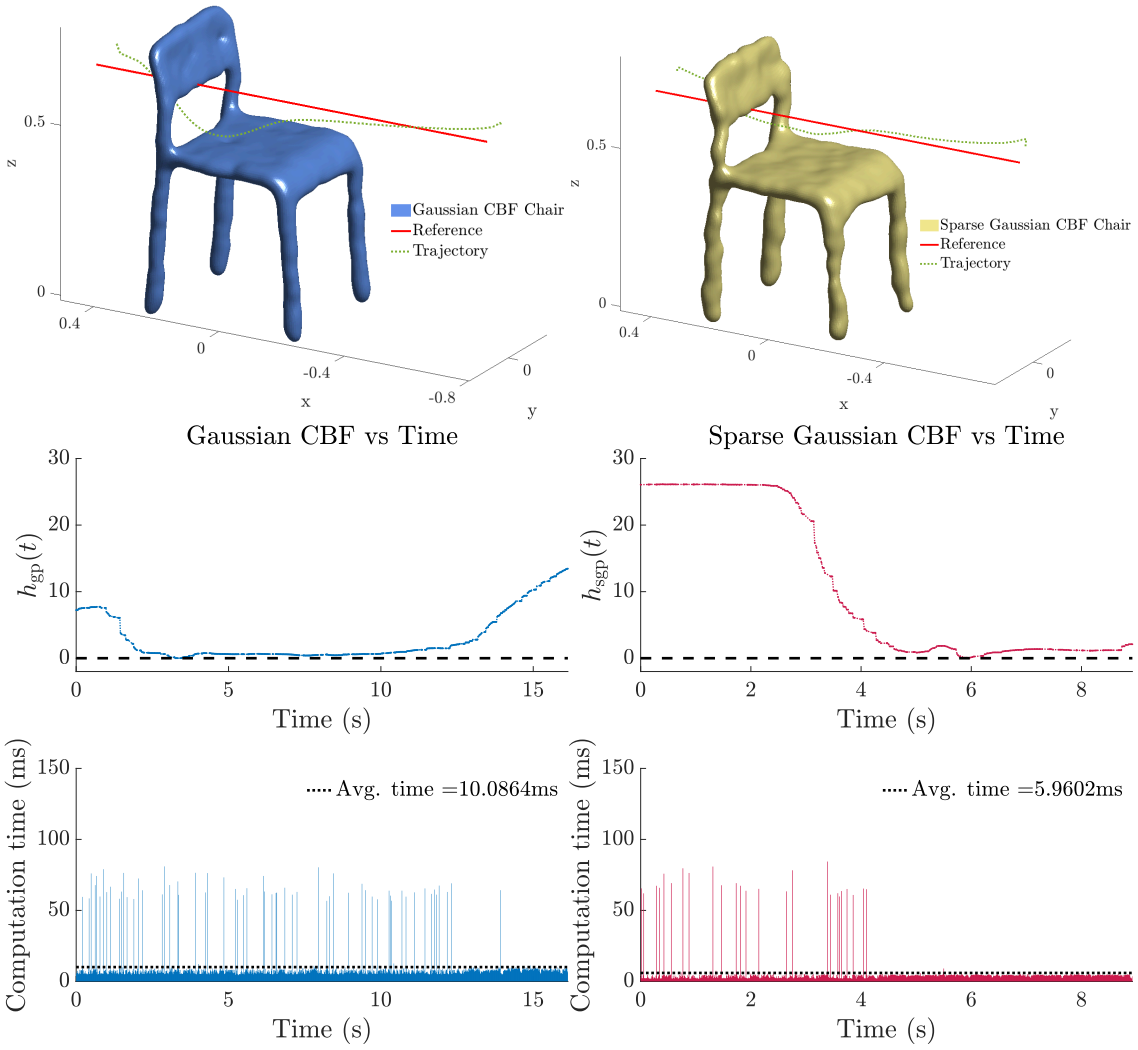

Figure 4.16: Experiment run 3 where the Crazyflie flies straight through the headrest of the chair. The reference and actual trajectories are plotted (top). The time plots of h_{gp} and h_{sgp} show that the CBFs are always non-negative (middle), and the computation time per iteration is also plotted (bottom).

4.6 Concluding Remarks

In summary, we use Gaussian CBFs to construct safe implicit surfaces. The implicit surface for a volumetric object is designed as the boundary of the safe set. We additionally derive sparse Gaussian CBFs to reduce the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^2N)$, where N are the total number of training points and M are the number of pseudo training points. We present several robotic test cases, firstly, in simulation for a 7-DOF manipulator, and secondly, for a 3D hardware quadrotor. For the manipulator problem, we estimate the safety boundary for the Stanford bunny as Gaussian CBFs with and without sparsity. Data for the Stanford bunny is represented as 3D point cloud and surface normals. The safe implicit surface here is the surface of the bunny. We first assume having complete knowledge of the point cloud to train the model offline and then demonstrate safe constrained control with rectification done online. Next, we perform proximal sensing of the point cloud to achieve online training. In both the cases, the manipulator did not collide with the bunny.

For the next test case, we conduct safe navigation in 3D using the Crazyflie. We use a physical IKEA ADDE chair for which we use an OBJ file to extract point cloud and surface normal data. First, we conduct safe teleoperation as both offline and online studies. For the offline problem, both the Gaussian CBFs (with and without sparsity) ensured that the Crazyflie did not collide with the chair. However, when conducting proximal sensing using a virtual sensor, only Gaussian CBF was able to model locally. Sparse Gaussian CBFs did not scale well for few datapoints. Second, we demonstrate safe autonomous navigation using both the Gaussian CBFs where unsafe reference trajectories would collide the drone with the chair. We perform numerous experimental runs using Gaussian CBFs and sparse Gaussian CBFs. For all the experimental runs, Crazyflie did not collide with the chair.

CHAPTER 5

MULTI-SPARSE GAUSSIAN PROCESS FOR LEARNING-BASED SEMI-PARAMETRIC CONTROL

A key challenge with controlling complex dynamical systems is to accurately model them. However, this requirement is very hard to satisfy in practice. Data-driven approaches such as Gaussian processes (GPs) have proven quite effective by employing regression based methods to capture the unmodeled or residual dynamical effects. However, GPs scale cubically with the number of data points N, and it is often a challenge to perform real-time regression. In this chapter, we propose a semi-parametric framework exploiting sparsity for learning-based control of a dynamical system. We combine the parametric model of the system with weighted sparse GP models to capture any unmodeled dynamics. We term this formulation as *Multi-Sparse Gaussian Process (MSGP)*. MSGP uses multiple sparse models with unique hyperparameters for each, thereby, preserving the richness and uniqueness of each sparse model. For a query point, a weighted sparse posterior prediction is performed based on W neighboring sparse models. Hence, the prediction complexity is significantly reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(WU^2P)$, where P and U are the number of local inputs and pseudo-inputs respectively per sparse model.

In this chapter, we present a semi-parametric framework using sparsity of GPs to estimate the residual dynamics of the dynamical system. To this end, multiple sparse GPs are used to divide the space of inputs and observations into sparse regional models. Semi-parametric methods have been applied for inverse dynamics [95], [96], system identification [97], and forward dynamics [98]; all using standard GPs. To the best of our knowledge, no prior work has merged semi-parametric modeling using sparse approximations of GPs. We create multiple sparse approximations of the original GP clustered into regionally sparse models without making any global assumptions. Local models hinder prediction

accuracy at the benefit of reduced complexity. To overcome this, each sparse model is optimized for its own hyperparameters and a weighted sparse posterior prediction is performed.

We validate the learning performance of MSGP both in simulation and hardware for the quadrotor case. We address sparse based learning on a quadrotor while considering the complete 3D dynamics in simulation. Comparison with GP, sparse GP, and local GP in simulation shows that MSGP has a higher prediction accuracy than sparse GP and local GP, and marginally better or similar performance compared to full GP, with significantly lower time complexity than all three. Learning-based control *especially using sparsity for a safety-critical system such as a quadrotor* has not been demonstrated before to the best of our knowledge in hardware. We validate MSGP on a real quadrotor setup for unmodeled mass, inertia, and disturbances (video link: `https://youtu.be/zUk1ISux6ao`).

5.1 Problem Statement

We consider a nonlinear, continuous-time system,

$$\dot{\mathbf{x}} = \underbrace{f(\mathbf{x}(t), \mathbf{u}(t))}_{\text{parametric}} + \underbrace{d(\mathbf{x}(t), \mathbf{u}(t))}_{\text{non-parametric}}, \tag{5.1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state and $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input at time t. The system dynamics is divided into a known parametric model $f(\mathbf{x}, \mathbf{u})$ and an unknown residual or non-parametric model $d(\mathbf{x}, \mathbf{u})$. The latter contains the unmodeled dynamics. Often times in practice, we use physics based first principles to derive the equations for the parametric component. However, it is not always possible to perfectly model the system resulting in unmodeled or residual dynamics, which we label as the non-parametric component.

The objective is to estimate and learn the unmodeled/residual nonlinearities in (5.1) in a *semi-parametric manner* through the use of GP priors. More specifically, we want to do this by placing multiple GP priors on the non-parametric component and exploit the sparsity of GPs. The mean function for these sparse GPs account for the physical knowledge of the

system, i.e., the parametric component. This is equivalent to a semi-parametric model,

$$\dot{\mathbf{x}} \sim f(\mathbf{x}, \mathbf{u}) + \sum_{i}^{L} \mathcal{SGP}\left(\mathbf{0},\ k(\mathbf{x}, \mathbf{x}')\right), \tag{5.2}$$

$$\sim \sum_{i}^{L} \mathcal{SGP}\left(\ f(\mathbf{x}, \mathbf{u})\ ,\ k(\mathbf{x}, \mathbf{x}')\right), \tag{5.3}$$

Comparing the resulting dynamics in (5.3) with (5.1) and (5.2) makes it clear that *the objective of MSGP is to model the residual dynamics using multiple sparse GPs*. This problem has been addressed before using standard GPs. We differ in our motivation to achieve the same using multiple sparse approximations to GPs, without comprising speed and accuracy for growing datasets. We assume that we can measure, $\hat{d}(\mathbf{x}, \mathbf{u}) = \dot{\mathbf{x}} - f(\mathbf{x}, \mathbf{u}) + \mathcal{N}(0, \sigma_n^2)$, which are corrupted by zero-mean, independent, and bounded noise. In practice, measuring or observing $\hat{d}$ can be quite difficult especially if higher-order derivatives are considered for (5.1). In this chapter, we work with acceleration dynamics and assume to have observability of acceleration data, albeit noisy. We also assume a nominal controller $\mathbf{u}_{\text{nom}}(t)$ exists that drives the parametric model $f(\cdot)$ to the zero equilibrium point.

5.2 Multi-Sparse Gaussian Process Regression

We discuss our proposed methodology inspired from the theoretical developments of SPGP and architecture of LGP. At the onset, MSGP can be seen simply as the combination of SPGP and LGP, however, it outperforms both SPGP and LGP which is very counter intuitive. MSGP at its core is different from LGP by using multiple sparse models (instead of full GP models) and unique hyperparameters in each model. Note that, although we choose SPGP as the sparse representative, the architectural nature of MSGP is agnostic to the underlying sparse approximation.

5.2.1 Multi-Sparse Model Clustering

The entire dataset of N training points is divided into L local models (randomly or deterministically), each with $P \approx N/L$ data points, such that $L \cdot P \approx N$, $P \ll N$. Every i^{th} local model is formed with a corresponding data matrix $\mathbf{D}_P^{(i)} = [\mathbf{X}_P^{(i)}, \mathbf{y}_P^{(i)}]$ and a corresponding centroid. The centroid is taken to be the mean of the input points in the local data matrix, $\mathbf{c} = \frac{1}{2}\sum_{j=1}^{P} \mathbf{x}_j$, for each model.

Next, sparsity is introduced in each local model by selecting a set of pseudo-inputs $\bar{\mathbf{X}}_U^{(i)}$, where $U \ll P$. These pseudo-inputs are selected arbitrarily at first for each sparse model; to be optimized later in the hyperparameter tuning phase in Section 5.2.2. Thus, each sparse model in MSGP (see Figure 5.1) is specified as $\{\mathbf{X}_P, \mathbf{y}_P, \mathbf{c}, \bar{\mathbf{X}}_U\}^{(i)}$ with a unique kernel. We use the SE kernel for this study.

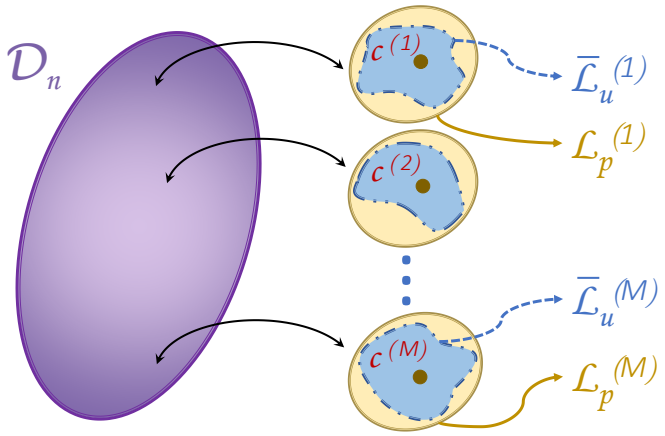

Figure 5.1: The original dataset $\mathbf{D}_N$ (purple) is divided into L local models (yellow) with approximately P data points each and a corresponding center $\mathbf{c}$. Each local model is further approximated into its sparse representation (blue), with $U \ll P$ local pseudo-inputs.

5.2.2 Localized Hyperparameter Tuning

Each sparse model is parameterized by $\bar{\Theta}^{(i)}$ that best fits its own dataset since we perform an optimization procedure on each model. The marginal likelihood for each model is,

$$\begin{aligned} p\big(\mathbf{y}_P|\mathbf{X}_P, \bar{\mathbf{X}}_U, \bar{\Theta}\big) &= \mathcal{N}\big(\mathbf{y}_p \mid \mathbf{0}\ ,\ \mathbf{L}_P + \mathbf{\Lambda}_p + \sigma_n^2\mathbf{I}_p\big) \\ &= \mathcal{N}\big(\mathbf{y}_p \mid \mathbf{0}\ ,\ \bar{\mathbf{K}}_P\big), \end{aligned} \tag{5.4}$$

where $\mathbf{L}_P := \mathbf{K}_{PU}\overline{\mathbf{K}}_U^{-1}\mathbf{K}_{UP}$, $\big[\mathbf{K}_{PU}\big]_{(i,j)} = k(\mathbf{x}_i, \bar{\mathbf{x}}_j), i \in \{1,\dots,P\}, j \in \{1,\dots,U\}$ is the covariance between local inputs $\mathbf{X}_p$ and pseudo-inputs $\bar{\mathbf{X}}_U$, $\overline{\mathbf{K}}_U \in \mathbb{R}^{U\times U}$ is the covariance between pairs of local pseudo-inputs. $\mathbf{\Lambda}_P = \operatorname{diag}\big[\mathbf{K}_P - \mathbf{L}_P\big]$ is a diagonal matrix, and $\bar{\mathbf{K}}_P := \mathbf{L}_P + \mathbf{\Lambda}_P + \sigma_n^2\mathbf{I}_P$.

By maximizing the log marginal likelihood of (5.4), we can jointly optimize for the hyperparameters, $\bar{\Theta}^{(i)}$, and pseudo-inputs, $\bar{\mathbf{X}}_U^{(i)}$, for each sparse model using quasi-Newton gradient methods. The log marginal likelihood of each locally sparse model is given by,

$$\log p(\cdot) = -\frac{1}{2}\Big(p\log(2\pi) - \log|\overline{\mathbf{K}}_P| - \mathbf{y}_P\overline{\mathbf{K}}_P^{-1}\mathbf{y}_p\Big). \tag{5.5}$$

The training complexity in MSGP has been significantly reduced to $\mathcal{O}(U^2P)$ from GP's $\mathcal{O}(N^3)$, where $U \ll P \approx N/L$, for a large number of L models. Whereas in LGP, the training complexity only reduces to the complexity results in $\mathcal{O}(N_{sub}^3)$. Moreover, we optimize the hyperparameters along with the pseudo-inputs for each model unlike LGP. This preserves the richness and uniqueness of each sparse model. This completes the creation of multiple sparse models which are uniquely optimized during the training phase. Next, we look at sparse posterior predictions for a query point $\mathbf{x}_*$.

5.2.3 Multi-Sparse GP Posterior Prediction

Optimizing hyperparameters for each sparse model may give rise to overfitting during the prediction phase. To remedy this, the posterior prediction in MSGP uses a weighted averaging over W neighboring sparse predictions $\hat{\mu}$ for a query point $\mathbf{x}_*$. The idea of weighted averaging for predictors was first introduced in LWPR; also used by LGP for its predictions. Akin to LWPR, we also perform weighted averaging, but using weighted sparse posterior predictions instead. The W nearest sparse models can be determined quickly using the SE kernel:

$$w_j(\mathbf{x}_*, \mathbf{c}^{(j)}) = \exp\Big(-\frac{1}{2}(\mathbf{x}_* - \mathbf{c}^{(j)})^\top \mathbf{L}_{(j)}^{-2}(\mathbf{x}_* - \mathbf{c}^{(j)})\Big), \tag{5.6}$$

where $\mathbf{c}^{(j)}$ is the centroid of j^{th} local model and $\mathbf{L}_{(j)}$ is the characteristic length scale diagonal matrix from the j^{th} local model respectively, with $j \in \{1, \ldots, N\}$. Finally, the mean posterior prediction of MSGP is,

$$\hat{\mu}(\mathbf{x}_*) = \frac{\sum_{j=1}^{N} w_j \mu_j(\mathbf{x}_*)}{\sum_{j=1}^{N} w_j}, \tag{5.7}$$

$$\mu_j(\mathbf{x}_*) = \mathbf{k}_{U*}^\top \, \mathbf{Q}_U^{-1} \, \mathbf{K}_{UP}\big(\mathbf{\Lambda}_P + {}^{(j)}\sigma_n^2 \mathbf{I}_P\big)^{-1}\mathbf{y}_P, \tag{5.8}$$

where $\mathbf{k}_{U*} = \big[k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_U, \mathbf{x}_*)\big]^\top \in \mathbb{R}^U$ gives the covariance between the local pseudo-inputs in $\bar{\mathbf{X}}_P$ and query point $\mathbf{x}_*$, and ${}^{(j)}\sigma_n^2$ is the local noise variance. Hence, the predictive mean complexity in MSGP is $\mathcal{O}(WU^2P)$ compared to LGP's complexity of $\mathcal{O}(WP^3)$. Note that the same approach can be taken to compute the weighted posterior variance $\hat{\sigma}^2(\mathbf{x}_*)$ using (4.5) in place of $\mu_j(\mathbf{x}_*)$ in (5.7). Here, we only use the weighted posterior mean to approximate the residual dynamics $\hat{d}$ in (5.1).

5.3 Application Test Case: Quadrotor Learning & Control

Here, we demonstrate the applicability of MSGP on a safety-critical quadrotor system. Most modern controllers require accurate knowledge of the model for improved trajectory tracking. By learning the unmodeled component using MSGP, we demonstrate improved trajectory tracking for a quadrotor in $SE(3)$. We use the geometric dynamical model for the quadrotor discussed in 2.5.1. We use a geometric tracking controller in $SE(3)$ for the quadrotor. We then discuss the augmentation of MSGP with the geometric controller for improved tracking in the presence of unmodeled dynamics and uncertainties.

5.3.1 Geometric Controller Tracking in SE(3)

The geometric controller used for trajectory tracking presented in [99] has almost global exponential stability. This implies the quadrotor can reach any desired state in the state-space from any initial configuration. For a complete mathematical treatment for the nominal controller, see [99]. Here, we just present the equations for nominal $F_{\text{nom}} \in \mathbb{R}$ and $\boldsymbol{\tau}_{\text{nom}} \in \mathbb{R}^3$:

$$F_{\text{nom}} = \underbrace{(-k_r \mathbf{e}_r - k_v \mathbf{e}_v)\mathbf{R}\mathbf{e}_3}_{\text{Feedback}} + \underbrace{mg\mathbf{e}_3 + m\ddot{\mathbf{r}}_{\text{des}})^\top \mathbf{R}\mathbf{e}_3}_{\text{Feedforward}} \tag{5.9}$$

$$= F_{\text{Feedback}} + F_{\text{Feedforward}}$$

$$\boldsymbol{\tau}_{\text{nom}} = \underbrace{-k_{\mathbf{R}}\mathbf{e}_{\mathbf{R}} - k_{\boldsymbol{\Omega}}\mathbf{e}_{\boldsymbol{\Omega}}}_{\text{Feedback}} + \underbrace{\boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega} - \mathbf{J}(\boldsymbol{\Omega}^\times \mathbf{R}^\top \mathbf{R}_{\text{des}}\boldsymbol{\Omega}_{\text{des}} - \mathbf{R}^\top \mathbf{R}_{\text{des}}\dot{\boldsymbol{\Omega}}_{\text{des}})}_{\text{Feedforward}} \tag{5.10}$$

$$= \boldsymbol{\tau}_{\text{Feedback}} + \boldsymbol{\tau}_{\text{Feedforward}}, \tag{5.11}$$

where $k_{(\cdot)} \in \mathbb{R}_{>0}$ are positive constants, the error terms are $\mathbf{e}_r = \mathbf{r} - \mathbf{r}_{\text{des}}$, $\mathbf{e}_v = \dot{\mathbf{r}} - \dot{\mathbf{r}}_{\text{des}}$, $\mathbf{e}_{\mathbf{R}} = \frac{1}{2}(\mathbf{R}_{\text{des}}^\top \mathbf{R} - \mathbf{R}^\top \mathbf{R}_{\text{des}})^\vee$, and $\mathbf{e}_{\boldsymbol{\Omega}} = \boldsymbol{\Omega} - \mathbf{R}^\top \mathbf{R}_{\text{des}}\boldsymbol{\Omega}_{\text{des}}$. The desired position, velocity, attitude, and angular acceleration are $\mathbf{r}_{\text{des}}, \dot{\mathbf{r}}_{\text{des}}, \mathbf{R}_{\text{des}}$, and $\dot{\boldsymbol{\Omega}}_{\text{des}}$ respectively. $(\cdot)^\vee$ is the inverse of $(\cdot)^\times$, i.e. $(\mathbf{a}^\times)^\vee = \mathbf{a}, \forall \mathbf{a} \in \mathbb{R}^n$.

5.3.2 Learning based Control using MSGP

The dynamical model in (2.15) - (2.18) and the controller presented in (5.9) deal with a precise model of the quadrotor. However, it is often difficult to accurately parameterize a dynamical system using physics first principles. Moreover, the model (2.15) - (2.18) does not consider aerodynamic drag, damping, wind effects, or time-varying changes to mass and inertia. We rewrite the geometric dynamics model with residual dynamics as follows,

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{5.12}$$

$$m\dot{\mathbf{v}} = -mg\mathbf{e}_3 + F_{\text{nom}}\mathbf{R}\mathbf{e}_3 + \mathbf{d}_1, \tag{5.13}$$

$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\Omega}^{\times}, \tag{5.14}$$

$$\mathbf{J}\dot{\boldsymbol{\Omega}} = \boldsymbol{\tau}_{\text{nom}} - (\boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega}) + \mathbf{d}_2, \tag{5.15}$$

where $\mathbf{d}_{\cdot} \in \mathbb{R}^3$ are the residual unmodeled components in linear and rotational accelerations. Here, we will use MSGP to capture and learn any unmodeled effects in the dynamics expressed as (5.13) and (5.15). Since unmodeled nonlinearities appear in the dynamics (5.13,5.15) for each component, we use a total of six MSGPs, placing a prior on each dimension of the unmodeled state-space as shown below,

$$m\dot{\hat{\mathbf{v}}} = mg\mathbf{e}_3 - F_{\text{nom}}\mathbf{R}\mathbf{e}_3 \; + \begin{bmatrix} \mathcal{MSGP}_1\Big(0, k(\mathbf{q}, \mathbf{q}')\Big) \\ \mathcal{MSGP}_2\Big(0, k(\mathbf{q}, \mathbf{q}')\Big) \\ \mathcal{MSGP}_3\Big(0, k(\mathbf{q}, \mathbf{q}')\Big) \end{bmatrix} \tag{5.16}$$

$$\mathbf{J}\dot{\hat{\boldsymbol{\Omega}}} = \boldsymbol{\tau}_{\text{nom}} - (\hat{\boldsymbol{\Omega}} \times \mathbf{J}\hat{\boldsymbol{\Omega}}) + \begin{bmatrix} \mathcal{MSGP}_4\Big(0, k(\mathbf{q}, \mathbf{q}')\Big) \\ \mathcal{MSGP}_5\Big(0, k(\mathbf{q}, \mathbf{q}')\Big) \\ \mathcal{MSGP}_6\Big(0, k(\mathbf{q}, \mathbf{q}')\Big) \end{bmatrix}. \tag{5.17}$$

The input to MSGPs are $\mathbf{q} = [\mathbf{r}^\top, \dot{\mathbf{r}}^\top, \boldsymbol{\Omega}^\top]^\top$ and the target observations are given by the difference between (2.16, 2.18) and (5.13, 5.15), i.e., $\hat{\mathbf{y}} = [m(\dot{\hat{\mathbf{v}}} - \dot{\mathbf{v}})^\top, \mathbf{J}(\dot{\hat{\boldsymbol{\Omega}}} - \dot{\boldsymbol{\Omega}})^\top]^\top +$

$\mathcal{N}(0, \sigma_n^2)$. Given the input samples and noisy observations, this constitutes a proper regression problem. Note that $\hat{\mathbf{y}} \in \mathbb{R}^6$, and we consider each element of $\hat{\mathbf{y}}$ as a supervised learning problem. In other words, we assume independence in the targets or observations. After learning the residual dynamics using MSGPs, we can perform prediction at a new query point $\mathbf{q}_*$, where the sparse predictive mean of the unmodeled dynamics is calculated using (5.7). This predictive mean is then used to modify the controller (5.9) with the learned dynamics as shown below,

$$
\begin{aligned}
F_{\text{input}} &= F_{\text{Feedback}} + F_{\text{Feedforward}} + \underbrace{\Big(- m \, [\, \hat{\mu}_1(\mathbf{q}_*),\ \hat{\mu}_2(\mathbf{q}_*),\ \hat{\mu}_3(\mathbf{q}_*) \,]^\top \mathbf{R}\mathbf{e}_3 \Big)}_{\text{MSGP}} \qquad (5.18) \\
&= F_{\text{Feedback}} + F_{\text{Feedforward}} + F_{\text{MSGP}} \\
\boldsymbol{\tau}_{\text{input}} &= \boldsymbol{\tau} + \underbrace{\Big(- \mathbf{J} \, [\, \hat{\mu}_4(\mathbf{q}_*),\ \hat{\mu}_5(\mathbf{q}_*),\ \hat{\mu}_6(\mathbf{q}_*) \,]^\top \Big)}_{\text{MSGP}} \qquad (5.19) \\
&= \boldsymbol{\tau}_{\text{Feedback}} + \boldsymbol{\tau}_{\text{Feedforward}} + \boldsymbol{\tau}_{\text{MSGP}}
\end{aligned}
$$

The controller design consideration above is that of feedback linearization. By plugging F_{input} and $\boldsymbol{\tau}_{\text{input}}$ into (5.13)-(5.15), the idea is to cancel out the nonlinearities or residual dynamics as much as possible with the augmented MSGP input terms in (5.18) and (5.19).

5.4 Simulation Results

We validate the MSGP semi-parametric learning framework by modifying the nominal controller's feedforward component on several test cases. We compare the MSGP's learning performance against the nominal, standard GP, SPGP, and LGP based controllers empirically. The `GPML` library in MATLAB is used for hyperparameter tuning and covariance calculations [93].

Desired trajectories are sinusoids where position reference is $\mathbf{r}_{\text{des}(t)} = [x_d,\ y_d,\ z_d]^\top = [4\sin(0.8t), 5\sin(0.4t), 2\sin(0.4t)]^\top$ and desired yaw is $\psi_d(t) = \text{atan2}(y_d, x_d)$, for $t \in$

$[t_0, t_f]$. Nominal parameters are $m = 1.25\text{kg}$, $\mathbf{J} = \text{diag}[1.1,\ 1.1,\ 2.2]\text{kgm}^2$. Controller gains are $k_r = 5$, $\mathbf{k_v} = \text{diag}[0.5, 0.5, 2.0]$, $\mathbf{k_\Omega} = \text{diag}[5, 10, 20]$, $k_\mathbf{R} = 30$. In simulation, the quadrotor is subjected to these trajectories under model uncertainties using a low-gain nominal controller (5.9). As a result, the learned controller has a stronger effect to compensate for unmodeled dynamics by adjusting controller's feedforward component.

The unmodeled dynamics are broadly classified into three categories for investigation: *parametric*, *non-parametric*, and combined *parametric and non-parametric*. For each category, we collect over 15500 samples for training and over 6000 samples for testing. One-tenth of the samples are chosen as pseudo-inputs for SPGP. Localization of data samples, into local models for LGP and MSGP are done in the same state space as inputs to the respective GPs, i.e. $\mathbf{q} = [\mathbf{r}^\top, \dot{\mathbf{r}}^\top, \mathbf{\Omega}^\top]^\top$. Each local model consists of a maximum of 750 samples resulting in over 20 regional models. Each regional model is further sparsed using one-fifth of the local samples as local pseudo-inputs to perform MSGP training and prediction.

5.4.1 Parametric Unmodeled Dynamics

Parametric unmodeled dynamics deal with changes or perturbations made to the quadrotor parameters, such as mass or inertia. Parameters are changed to the extent that the nominal controller can still achieve stable flight, although with performance degradation in trajectory tracking. Tracking error is determined in the position space of the quadrotor. The trajectory tracking error of the quadrotor when subjected to changes in the parameters is shown in Figure 5.2. In the training phase, the quadrotor is trained for each GP by introducing step changes to the mass and inertia at different time instances.

In Table 5.1, m and J are the nominal mass and inertia, $\hat{m}$ and $\hat{J}$ are the perturbed mass and inertia, $t_0 = 0$ and $t_f = 16$ seconds. In the testing phase, the controllers are compared by changing the magnitude of $\hat{m}, \hat{J}$, and time intervals. From the normalized mean squared error (NMSE) plot in Figure 5.2, it is clear that the nominal controller has a higher NMSE

Table 5.1: Parametric step changes made to mass and inertia over different time instances.

Initial time (t_0)	Final time ($\leq t_f$)	Mass ($\hat{m}$)	Inertia ($\hat{\mathbf{J}}$)
0	2	$1.00 \cdot m$	$\mathbf{J} + \text{diag}[0.75\ 0.75\ 0.75]$
2	6	$1.15 \cdot m$	$\mathbf{J} + \text{diag}[0.02\ 0.02\ 0.02]$
6	9	$0.85 \cdot m$	$\mathbf{J} + \text{diag}[1.31\ 1.31\ 1.61]$
9	12	$1.13 \cdot m$	$\mathbf{J} + \text{diag}[0.31\ 0.01\ 0.03]$
12	16	$1.05 \cdot m$	$\mathbf{J} + \text{diag}[0.55\ 0.55\ 0.82]$

along z. This is expected since changing the mass has more pronounced effect on the altitude. The GP controller performs better than the nominal and SPGP controllers, while LGP outperforms all three controllers. MSGP on the other hand demonstrates superior tracking performance with the lowest NMSE among all the controllers. MSGP achieves better tracking performance compared to the other learning based controllers due to unique hyperparameters and weighted sparse posterior prediction. Moreover, changing dynamical effects at different time instances are better captured with different hyperparameters as opposed to a global set of hyperparameters as in the case of GP, SPGP, and LGP.

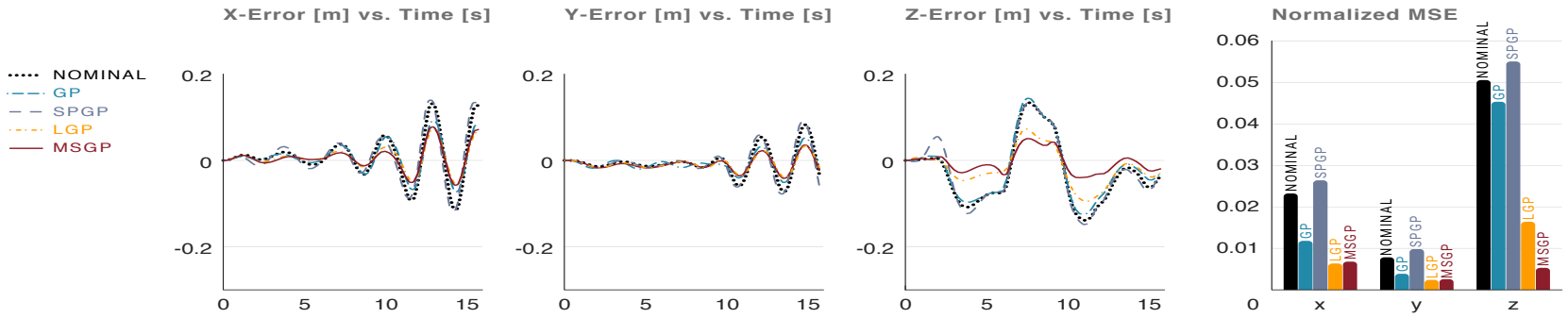

Figure 5.2: *Parametric effects*: Tracking error between nominal and learning-based controllers (GP, SPGP, LGP, MSGP) for varying mass and inertia.

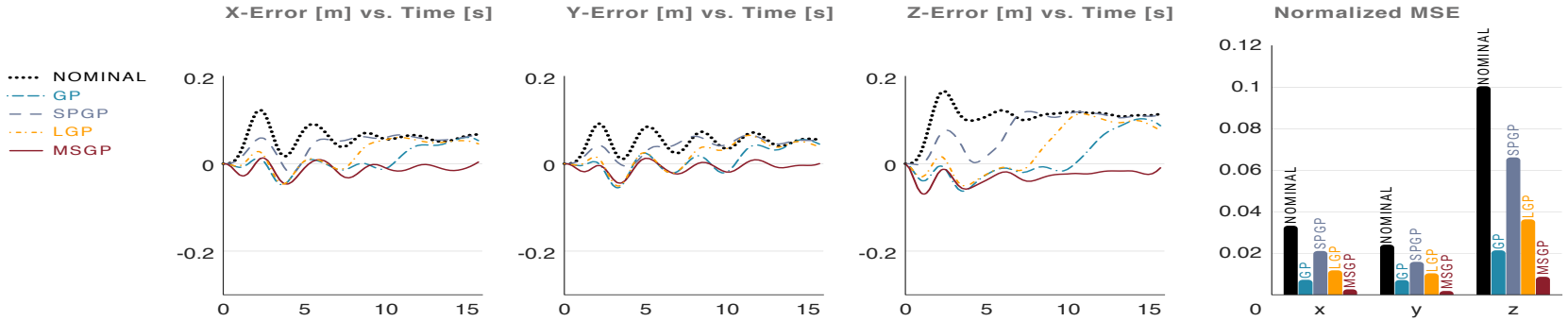

Figure 5.3: *Non-parametric effects*: Tracking error between nominal and learning-based controllers (GP, SPGP, LGP, MSGP) for varying wind.

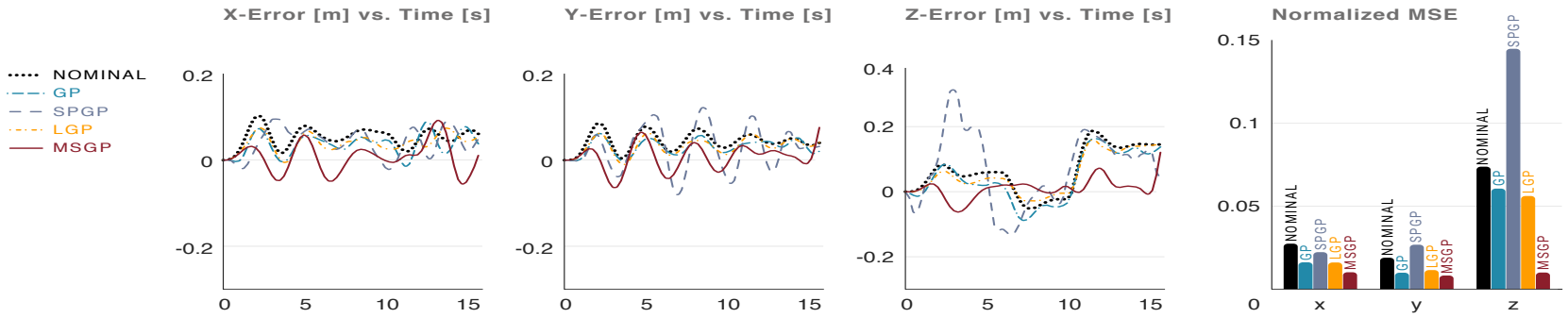

Figure 5.4: *Parametric & Non-parametric*: Tracking error between nominal and learning-based controllers (GP, SPGP, LGP, MSGP) for mass, inertia, and wind.

5.4.2 Non-Parametric Unmodeled Dynamics

Here we look at non-parametric effects introduced in the dynamics such as unmodeled aerodynamics. The quadrotor is subjected to the same unknown wind effects for training each GP: $\mathcal{W} = [0.17 \;\; 0.18 \;\; 0.16]^\top g$. During testing, a similar wind is introduced having different magnitudes along each dimension for comparing the tracking performance. Figure 5.3 shows the tracking performance of the quadrotor in presence of non-parametric aerodynamic disturbances.

The nominal controller incurs the highest NMSE along each dimension. This is expected since the unmodeled dynamics cannot be handled by the nominal controller that relies on model knowledge for feedforward compensation. GP performs significantly better than the nominal controller in presence of such effects. SPGP performs better than the nominal case, but it does not compensate as effectively as GP. LGP on the other hand outperforms both the nominal and SPGP controllers, but underperforms compared to GP. Finally, MSGP incurs the lowest NMSE, outperforming all the controllers including GP.

From the error versus time plots in Figure 5.3, it can be seen that each learning based controller eventually fails to compensate for the wind effects with the exception of MSGP, which holds out the longest among all the controllers. GP is able to compensate for the wind longer than both SPGP and LGP. SPGP gives in first to the unmodeled dynamical effects since it is only a sparse approximation of GP, while LGP, being a locally clustered approximation of GP, holds out longer than SPGP. Despite MSGP having multiple sparse approximations of GP, it is consistently able to compensate since each sparse model's uniqueness is preserved as described in Section 5.2.2.

5.4.3 Parametric & Non-Parametric Effects

Next, we study the combined effects of unmodeled dynamics in both parametric and non-parametric form. Note that the mass, inertia, and wind effects introduced here are different from the previous experiments to show performance against varied conditions. The para-

Table 5.2: Both parametric changes to mass and inertia and non-parametric changes in the form of external wind are introduced.

Initial time (t_0)	Final time ($\leq t_f$)	Mass ($\hat{m}$)	Inertia ($\hat{\mathbf{J}}$)	Wind ($\mathcal{W}$)
0	6	$1.00 \cdot m$	$\mathbf{J} + \text{diag}[0.75\ 0.75\ 0.75]$	$[0.31\ \ 0.32\ \ 0.15]^\top g$
6	10	$1.23 \cdot m$	$\mathbf{J} + \text{diag}[0.02\ 0.02\ 0.02]$	$[0.31\ \ 0.32\ \ 0.15]^\top g$
10	16	$0.81 \cdot m$	$\mathbf{J} + \text{diag}[1.31\ 1.31\ 1.61]$	$[0.31\ \ 0.32\ \ 0.15]^\top g$

metric and non-parametric changes for the training phase are shown below.

In Table 5.2, $\hat{m}$ and $\hat{J}$ are the affected mass and inertia parameters respectively, $\mathcal{W}$ is the unmodeled wind, $t_0 = 0$ and $t_f = 16$ seconds. Since it is a combination of multiple unmodeled effects, the simulation setup is very challenging because the learning based quadrotor controller needs to deal with a highly inaccurate model. During training, each GP algorithm is trained on the above combinations. During testing, the magnitudes and respective time intervals are altered to test the generalizability of the learning based controllers.

The tracking error performance is shown in Figure 5.4. Among all the controllers, SPGP performs the worst in terms of tracking error followed by the nominal controller. The sparse approximation tends to over compensate for the introduced nonlinearities in the dynamics, thus exaggerating its effects in the feedforward controller. Both GP and LGP demonstrate comparable performance and perform better than the nominal controller. MSGP has the lowest NMSE among all the controllers with comparable performance to GP and LGP along the x and y dimension. In the altitude domain however, there is a significant reduction in NMSE for MSGP compared to any other controller.

5.4.4 Training and Prediction Time Comparison

We now analyze the average time taken by different GP algorithms for posterior predictions. Since GP is a global regressor and is trained using a portion or the complete dataset, the total number of data points used by GP will typically be higher than its sparse (SPGP) or local (LGP/MSGP) approximations. SPGP takes a sampled subset of GP and approximates the original GP likelihood. Whereas, LGP and MSGP divides the original dataset into re-

gional models. For different training sizes ($3000, 5807, 8613, 11419, 14225$), we train each GP individually subjected to non-parametric wind disturbances. In the case of SPGP, we take one-tenth of each training dataset as pseudo-inputs. For LGP and MSGP, we assume the number of local data points to be linearly proportional to each training dataset. We take one-fifth of each training set to form local models, forming 5 regional models. Although in practice, the regional models need only have $250 - 500$ data points each. However, we let each regional model hold a fairly high number of local data points for comparison. Subsequently, for MSGP, we further take one-fifth of each local model's dataset as local pseudo-inputs. For LGP and MSGP, all the neighboring models are considered for computing the weighted posterior prediction, i.e., $N = 5$.

LGP and MSGP take the least time to train due to the reduced order model compared to GP. LGP takes over 25s to train each cluster in CPU time (i7-9800HQ). MSGP on the other hand takes under 6s for each cluster in CPU time. We also benchmark GPU training time using the `GPyTorch` library on a RTX 2080 Ti [100]. MSGP takes roughly 1s for 12 dimensional input and 6 dimensional output for each cluster.

The CPU prediction time comparison, including matrix inversion, is shown in Figure 5.5. GP's time complexity drastically grows with increasing training points as expected; since it cubically scales with the number of training points. SPGP scales very well compared to GP. For over 14000 points, SPGP computes under 5s. LGP has the least computational cost with fewer training points (under 5000) and marginally grows with increasing size. It is faster than SPGP and takes under 2s with over 14000 points. MSGP is similar to LGP but performs better as the number of training inputs increase due to sparsity in each model. MSGP takes approximately 1s for predictions with over 14000 training points. Note that precomputations can be made to improve the speed for all the methods. In practice, one can save, $\boldsymbol{\alpha} := (\mathbf{K} + \sigma^2\mathbf{I})^{-1}y$, where $\mathbf{K}$ denotes the covariance matrix for observed inputs and $\mathbf{y}$ is the set of target observations. Rank-1 approximations are then made for computing inverses. This results in tremendous boost in computational speed, thus achiev-

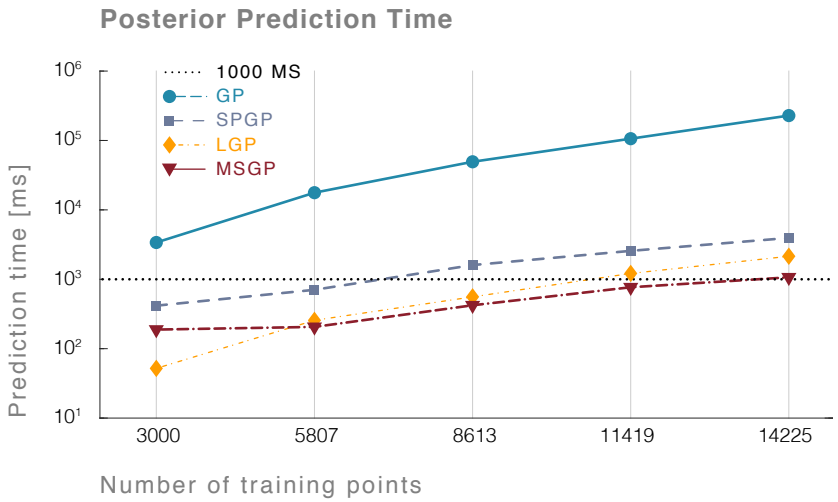

Figure 5.5: Posterior prediction time in milliseconds against different training sizes. The prediction time is computed on a query point for GP, SPGP, LGP, and MSGP. The dashed black line marks 1000 milliseconds.

Table 5.3: Space and time complexity comparison for GP, SPGP, LGP, and MSGP, ignoring the time taken to create M clusters for local methods. The last column assumes saving necessary matrices for each method.

Method	Storage	Training	Mean	Mean (w/ saving)
GP	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$
SPGP	$\mathcal{O}(NM)$	$\mathcal{O}(M^2N)$	$\mathcal{O}(M^2N)$	$\mathcal{O}(M)$
LGP	$\mathcal{O}(NP)$	$\mathcal{O}(N^3)$	$\mathcal{O}(WP^3)$	$\mathcal{O}(WP)$
MSGP	$\mathcal{O}(NU)$	$\mathcal{O}(U^2P)$	$\mathcal{O}(WU^2P)$	$\mathcal{O}(WU)$

ing faster predictions. Doing so results in a prediction time of only 0.8ms for MSGP in CPU time. The space and time complexity for the various GPs are tabulated in Table 5.3.

5.5 Hardware Experimental Verification

5.5.1 Experimental Setup

We use the same experimental setup as discussed in 2.6.1, 3.5.1, and 4.5.1. We use a position controller to generate the commanded thrust using a feedforward hovering thrust and a feedback PD controller. Desired attitude is maintained through an attitude controller

generating commanded roll, pitch, and yaw-rates. The gains selected are, $k_r = 50, k_v = 100$, under which stable flight is maintained within nominal conditions. Control inputs are sent at 100Hz while states are recorded at 100Hz. The experiment video can be seen at: `https://youtu.be/zUk1ISux6ao`.

The objective is to maintain stable flight particularly outside nominal conditions. Training data is collected by adding a payload of approximately 3g to the 32g Crazyflie. MSGP input is $\mathbf{q} = [\mathbf{r}^\top, \dot{\mathbf{r}}^\top, \ddot{\mathbf{r}}^\top, \phi, \theta, \psi, F] \in \mathbb{R}^{13}$, where $\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}, \phi, \theta, \psi, F$ are positions, velocities, accelerations, roll, pitch, yaw, and commanded thrust. The observations are computed by subtracting the measured system accelerations from the nominal system (2.16,2.18). Over 5000 training points were collected, with each sparse model randomly assigned 250 points. Each model is further sparsed using one-fourth of the points as pseudo-inputs. Training each cluster takes less than 0.16s in CPU time. For weighted sparse prediction, $N = 5$ neighboring models are used.

5.5.2 Experiment 1 : Altitude Hover

We first test the nominal and MSGP based controller for a simple task of stable hovering in the presence of an additional payload. The reference altitude is set at 0.8m. Figure 5.6 shows the tracking comparison for MSGP based controller and nominal controller. Since these are two separate experiments, the transition point for adding the payload is synchronized for visualizing the plot better. When there is no added mass, both MSGP and nominal controller exert similar hovering thrust demonstrating they operate well within nominal conditions. Once the payload is added, MSGP immediately exerts a compensating feedforward thrust as seen in Figure 5.7. If the quadrotor goes above the desired altitude, compensation is relaxed. Due to this relaxation, the quadrotor then descends below the desired altitude. Then a larger compensating thrust is computed taking the drone closer to the desired altitude. The nominal controller, however, begins exerting maximal feedback thrust due to the altitude drop experienced. For the nominal controller to eliminate steady

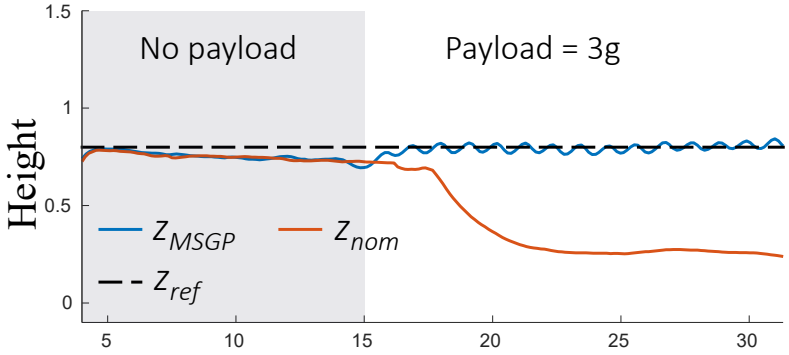

Figure 5.6: Quadrotor altitude hold with a payload of $3g$

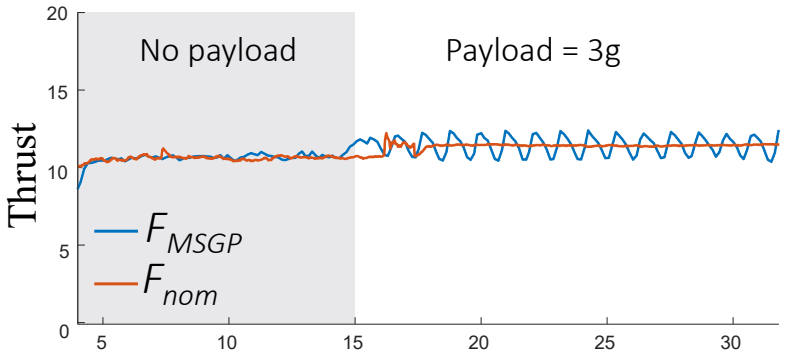

Figure 5.7: Commanded thrust for altitude hold with a payload of $3g$

state error, the gains need to be adapted or tuned accordingly. This issue is alleviated in the case of the MSGP based controller. We also note that better design of PD gains will reduce the oscillations experienced by MSGP; however, proper design of optimal gains is outside the scope of this work.

5.5.3 Experiment 2 : External Disturbances

In this experiment, we aggressively disturb the system to test the robustness and generalizability of the MSGP learning algorithm. The system is disturbed in three ways: 1) hitting the payload inducing unmodeled inertial moments, 2) hitting the quadrotor physically off the reference, 3) pulling the quadrotor down with the mass. The disturbances are meant to induce thrust and attitude compensation by MSGP.

The altitude tracking performance in presence of the disturbances is shown in Figure

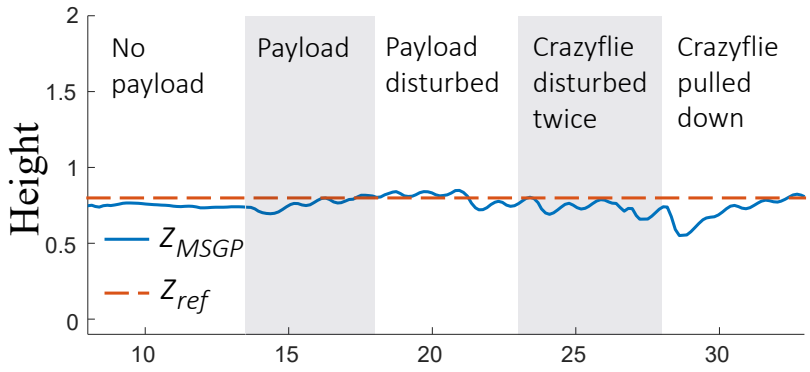

Figure 5.8: Altitude of the system in presence of aggressive unmodeled disturbances and a payload of 3g using MSGP controller

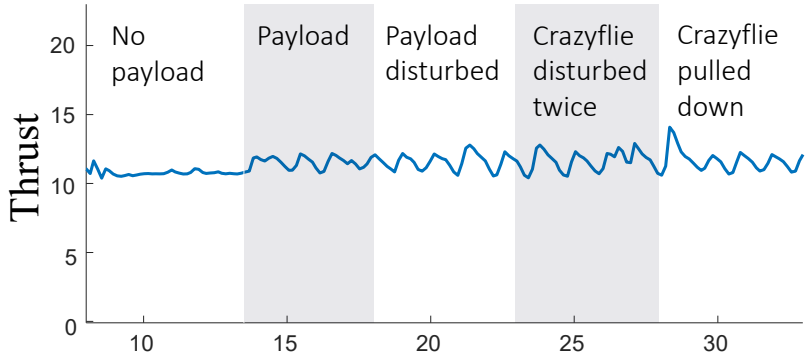

Figure 5.9: Commanded thrust of the system in presence of aggressive unmodeled disturbances and a payload of 3g using MSGP controller

5.8. Note that MSGP has not been trained a priori for these disturbances. MSGP performs good tracking when there is no payload and also compensates well when the mass is added. The transient behavior can be seen in Figure 5.9 for the feedforward thrust. Thereafter, the system is severely disturbed by first hitting the mass which induces an off-axis inertial moment which MSGP needs to address. Additionally, the Crazyflie is then hit twice to go off its current trajectory. Finally, the Crazyflie is pulled down along with the mass. For all these unmodeled disturbances, MSGP generates the necessary thrust and commanded attitudes to hold stable flight and steady altitude.

5.6 Concluding Remarks

In this chapter, we proposed a semi-parametric based control with sparsity by exploiting multiple GP sparse models. The sparse models are separately optimized for their hyperparameters, thereby, retaining their own uniqueness. A weighted sparse posterior predictor is adopted for a query point to avoid overfitting and any discontinuities. The proposed framework is tested on a geometric quadrotor controller in simulation with complete 3D dynamics. Simulations are performed extensively for unmodeled parametric, non-parametric, and combined dynamical effects. We empirically demonstrated the efficacy of our proposed approach to generalize to step changes despite being a locally sparse approximator. We also rigorously tested our proposed framework against standard GP, sparse GP, and local GP on both prediction quality and time complexity. MSGP demonstrated better prediction accuracy in the form of improved trajectory tracking with reduced prediction time compared to other GPs. Lastly, we experimentally performed sparsity based control on a quadrotor platform. We validated MSGP's effectiveness on the quadrotor when dealing with unknown mass and disturbances.

CHAPTER 6
CONCLUSION

With more and more tasks being outsourced to intelligent robots, the need for safety increases simultaneously. In order to fully realize the potential of these robots, we need an efficient and reliable manner of quantifying safety designs both algorithmically and practically. In conclusion, this thesis aimed at providing a non-parametric paradigm to safety which leverages Bayesian inference in its formulation. In order to bring safety-critical systems in the real world, we need to develop methods that can be adaptive. The design of Gaussian control barrier functions (Gaussian CBFs) use Gaussian processes to place a prior on the desired safety function candidate. To synthesize the candidate function, we rely on the use of safety samples where these samples may come from sensors, such as distance sensors, LiDAR, or data from computer aided design software. This allows a flexible parametrization of the barrier function since the data fully informs the level set characteristics. Furthermore, we implemented and showcased the tools developed in this dissertation on a hardware quadrotor system. We presented several numerical test cases where the quadrotor always remained inside the safe sets designated by the Gaussian CBFs.

We first presented the literature background in Section 1.1 of the first chapter. The second chapter presents a concept of uncertainty in the safety design by leveraging Gaussian process posterior variance. The core contribution of this thesis is presented in Chapter 3 where we present the theoretical, algorithmic, and practical realization of Gaussian CBFs. Our proposed barrier functions can also have sparse posterior realizations which can become a key enabler for practical computations under limited budget. We present this in Chapter 4 along with the synthesis of safe implicit surfaces. Implicit surfaces allow us to synthesize safety functions in 3-dimensional Euclidean space representing very complex volumetric objects. We also discuss a semi-parametric learning based framework in the

final chapter of this thesis in Chapter 5.

This dissertation has many possible future research directions. We elaborate on some of the possible directions below:

- The work presented here for Gaussian CBFs did not consider dynamical uncertainty in the system dynamics. Although, the theory presented is general enough to incorporate dynamical system uncertainty, this still remains to be addressed. One possible direction is to inform the barrier function synthesis by including uncertainty measurement data or observations in the training set.

- We only studied single agent cases in this dissertation. The Gaussian CBFs can be informed based on the data collected from multiple sources and the field of multi-agent systems is a viable future direction. This also allows the possibility of designing a decentralized or distributed class of Gaussian CBFs.

- Gaussian CBFs are ultimately Gaussian processes which themselves are stochastic realizations. This thesis did not delve in the stochasticity of Gaussian CBFs and this is an interesting avenue for future research. One may consider taking the barrier function realizations drawn from the GP and perform Fourier analysis on them instead of simply using the posterior mean and variance. Stochastic CBFs [82] deal with stochastic dynamical systems. It would be interesting to see a fully stochastic formulation where both the dynamics and barrier functions are stochastic in nature.

- The theory of Gaussian CBFs can be further explored especially in relation to reproducing kernel Hilbert spaces [59, 101]. The Gaussian CBF resides in an RKHS while the canonical version of CBFs live in a sufficient continuously differentiable function space. This means that Gaussian CBFs are contained inside the function space of CBFs. It would be interesting to consider a deeper functional analysis of the barrier functions.

- For checking forward invariance of the quadrotor or the robot manipulator, we only addressed the case where the state is a point mass, e.g., the center of mass of the quadrotor or the end-effector of the manipulator. A potential direction is to consider a volumetric representation for the system whereby the entire body of the quadrotor or manipulator remains inside the safe set. Extent compatible barrier functions [102] merged with Gaussian CBFs is another interesting avenue for future research.

- Since GPs suffer from a computational bottleneck of cubic complexity, we presented sparse realizations in this thesis. However, this can be further improved by performing online sparse approximations as discussed in [103, 104]. This would greatly improve the odds of performing large-scale real-time deployment of performing barrier function synthesis in a non-parametric manner using GPs.

REFERENCES

[1] S. Gaskill and S. Went, "Safety issues in modern applications of robots," *Reliability Engineering & System Safety*, vol. 53, no. 3, pp. 301–307, 1996.

[2] M. Valenti, D. Dale, J. How, and J. Vian, "Mission health management for 24/7 persistent surveillance operations," vol. 2, Aug. 2007.

[3] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: A review," *Precision Agriculture*, vol. 13, no. 6, pp. 693–712, Dec. 2012.

[4] S. Morante, J. G. Victores, and C. Balaguer, "Cryptobotics: Why robots need cyber safety," *Frontiers in Robotics and AI*, vol. 2, p. 23, 2015.

[5] J. Fryman and B. Matthias, "Safety of industrial robots: From conventional to collaborative applications," in *ROBOTIK 2012; 7th German Conference on Robotics*, VDE, 2012, pp. 1–5.

[6] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robotics and Autonomous Systems*, vol. 94, pp. 43–52, 2017.

[7] F. Blanchini and S. Miani, *Set-theoretic methods in control*. Springer, 2008.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[9] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[10] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *International Conference on Robotics and Automation*, IEEE, 2017, pp. 3293–3298.

[11] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *2015 American Control Conference*, IEEE, 2015, pp. 4542–4548.

[12] A. Anand, K. Seel, V. Gjærum, A. Hakansson, H. Robinson, and A. Saad, "Safe learning for control using control lyapunov functions and control barrier functions: A review," *Procedia Computer Science*, vol. 192, pp. 3987–3997, 2021.

[13] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods," *arXiv preprint arXiv:2202.11762*, 2022.

[14] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, vol. 24, pp. 551–559, 1942.

[15] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[16] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[17] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 6271–6278.

[18] A. Mehra, W.-L. Ma, F. Berg, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Adaptive cruise control: Experimental validation of advanced controllers on scale-model cars," in *2015 American Control Conference (ACC)*, 2015, pp. 1411–1418.

[19] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference*, IEEE, 2016, pp. 322–328.

[20] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *2019 IEEE 58th conference on decision and control (CDC)*, IEEE, 2019, pp. 474–479.

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

[22] W. Liu, Y. Gao, F. Gao, and S. Li, "Trajectory adaptation and safety control via control barrier functions for legged robots," in *2021 China Automation Congress (CAC)*, IEEE, 2021, pp. 5571–5576.

[23] S. Teng, Y. Gong, J. W. Grizzle, and M. Ghaffari, "Toward safety-aware informative motion planning for legged robots," *arXiv preprint arXiv:2103.14252*, 2021.

[24] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 827–834.

[25] Q. Nguyen, X. Da, J. Grizzle, and K. Sreenath, "Dynamic walking on stepping stones with gait library and control barrier functions," in *Algorithmic Foundations of Robotics XII*, Springer, 2020, pp. 384–399.

[26] M. Rauscher, M. Kimmel, and S. Hirche, "Constrained robot control using control barrier functions," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 279–285.

[27] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames, "Safety-critical manipulation for collision-free food preparation," *arXiv preprint arXiv:2205.01026*, 2022.

[28] A. Singletary, S. Kolathaya, and A. D. Ames, "Safety-critical kinematic control of robotic systems," *IEEE Control Systems Letters*, vol. 6, pp. 139–144, 2022.

[29] L. Wang, A. Ames, and M. Egerstedt, "Safety barrier certificates for heterogeneous multi-robot systems," in *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 5213–5218.

[30] Y. Emam, P. Glotfelter, and M. Egerstedt, "Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 3984–3990.

[31] M. Machida and M. Ichien, "Consensus-based control barrier function for swarm," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 8623–8628.

[32] T. Ibuki, S. Wilson, J. Yamauchi, M. Fujita, and M. Egerstedt, "Optimization-based distributed flocking control for multiple rigid bodies," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1891–1898, 2020.

[33] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[34] A. Robey, L. Lindemann, S. Tu, and N. Matni, "Learning robust hybrid control barrier functions for uncertain systems," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 1–6, 2021.

[35] A. Robey *et al.*, "Learning control barrier functions from expert demonstrations," in *2020 59th IEEE Conference on Decision and Control*, 2020, pp. 3717–3724.

[36] N. Gaby, F. Zhang, and X. Ye, "Lyapunov-net: A deep neural network architecture for lyapunov function approximation," *arXiv preprint arXiv:2109.13359*, 2021.

[37] H. Tsukamoto and S.-J. Chung, "Neural contraction metrics for robust estimation and control: A convex optimization approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 211–216, 2020.

[38] H. Zhao, X. Zeng, T. Chen, and Z. Liu, "Synthesizing barrier certificates using neural networks," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–11.

[39] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.

[40] N. Csomay-Shanklin, R. K. Cosner, M. Dai, A. J. Taylor, and A. D. Ames, "Episodic learning for safe bipedal locomotion with control barrier functions and projection-to-state safety," in *Learning for Dynamics and Control*, PMLR, 2021, pp. 1041–1053.

[41] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*, PMLR, 2020, pp. 708–717.

[42] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *International Conference on Robotics and Automation*, IEEE, 2018, pp. 2460–2465.

[43] F. Castaneda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics," in *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 3683–3690.

[44] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 585–590.

[45] A. A. Ahmadi and A. Majumdar, "Some applications of polynomial optimization in operations research and real-time decision making," *Optimization Letters*, vol. 10, no. 4, pp. 709–729, 2016.

[46] A. A. Ahmadi and A. Majumdar, "Dsos and sdsos optimization: More tractable alternatives to sum of squares and semidefinite optimization," *SIAM Journal on Applied Algebra and Geometry*, vol. 3, no. 2, pp. 193–230, 2019.

[47] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 7139–7145.

[48] K. Long, C. Qian, J. Cortés, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.

[49] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS 2016 workshop: geometry and beyond-representations, physics, and scene understanding for robotics*, University of Michigan, 2016.

[50] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3387–3395.

[51] M. Ohnishi, L. Wang, G. Notomista, and M. Egerstedt, "Barrier-certified adaptive reinforcement learning with applications to brushbot navigation," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1186–1205, 2019.

[52] Y. Emam, P. Glotfelter, Z. Kira, and M. Egerstedt, "Safe model-based reinforcement learning using robust control barrier functions," *arXiv preprint arXiv:2110.05415*, 2021.

[53] M. Khan, T. Ibuki, and A. Chatterjee, "Safety uncertainty in control barrier functions using gaussian processes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 6003–6009.

[54] M. A. Khan, T. Ibuki, and A. Chatterjee, "Gaussian control barrier functions: Non-parametric paradigm to safety," *IEEE Access*, vol. 10, pp. 99 823–99 836, 2022.

[55] M. Khan, A. Patel, and A. Chatterjee, "Multi-sparse gaussian process: Learning based semi-parametric control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5327–5334.

[56] A. Thirugnanam, J. Zeng, and K. Sreenath, "Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions," in *IEEE International Conference on Robotics and Automation*, 2022.

[57] S. He, J. Zeng, and K. Sreenath, "Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions," in *IEEE International Conference on Robotics and Automation*, 2022.

[58] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 7623–7628.

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

[60] M. L. Stein, *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[61] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen, "Derivative observations in gaussian process models of dynamic systems," in *Advances in neural information processing systems*, 2003, pp. 1057–1064.

[62] D. Eriksson, K. Dong, E. Lee, D. Bindel, and A. G. Wilson, "Scaling gaussian process regression with derivatives," in *Advances in Neural Information Processing Systems*, 2018, pp. 6867–6877.

[63] B. Xu and K. Sreenath, "Safe teleoperation of dynamic uavs through control barrier functions," in *2018 IEEE International Conference on Robotics and Automation*, 2018, pp. 7848–7855.

[64] D. W. Mellinger, "Trajectory generation and control for quadrotors," 2012.

[65] *Crazyflie 2.1 : Bitcraze*, https://www.bitcraze.io/crazyflie-2-1/, (Last Accessed Sep. 8, 2021).

[66] W. Hönig and N. Ayanian, "Flying multiple uavs using ros," in *Robot Operating System (ROS): The Complete Reference (Volume 2)*, A. Koubaa, Ed. Springer International Publishing, 2017, pp. 83–118, ISBN: 978-3-319-54927-9.

[67] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[68] R. L. Eubank, *Nonparametric regression and spline smoothing*. CRC press, 1999.

[69] L. C. Marsh and D. R. Cormier, *Spline regression models*. Sage, 2001.

[70] G. Kauermann and M. Wegener, "Functional variance estimation using penalized splines with principal component analysis," *Statistics and Computing*, vol. 21, no. 2, pp. 159–171, 2011.

[71] L. M. Sangalli, J. O. Ramsay, and T. O. Ramsay, "Spatial spline regression models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 4, pp. 681–703, 2013.

[72] A. Liu, T. Tong, and Y. Wang, "Smoothing spline estimation of variance functions," *Journal of Computational and Graphical Statistics*, vol. 16, no. 2, pp. 312–329, 2007.

[73] J. T. Pohlmann and D. W. Leitner, "A comparison of ordinary least squares and logistic regression (1)," *The Ohio journal of science*, vol. 103, no. 5, pp. 118–126, 2003.

[74] B. Craven and S. M. Islam, "Ordinary least-squares regression," *The SAGE dictionary of quantitative management research*, pp. 224–228, 2011.

[75] D. J. MacKay, "Bayesian methods for backpropagation networks," in *Models of neural networks III*, Springer, 1996, pp. 211–254.

[76] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[77] G. S. Kimeldorf and G. Wahba, "A correspondence between bayesian estimation on stochastic processes and smoothing by splines," *The Annals of Mathematical Statistics*, vol. 41, no. 2, pp. 495–502, 1970.

[78] C. J. Paciorek, "Nonstationary gaussian processes for regression and spatial modelling," Ph.D. dissertation, Carnegie Mellon University, 2003.

[79] I. J. Schoenberg, "Metric spaces and completely monotone functions," *Annals of Mathematics*, pp. 811–841, 1938.

[80] A. Girard, C. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," *Advances in neural information processing systems*, vol. 15, 2002.

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

[82] A. Clark, "Control barrier functions for stochastic systems," *Automatica*, vol. 130, p. 109 688, 2021.

[83] M. P. Do Carmo, *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

[84] T. P. Minka, "Expectation propagation for approximate bayesian inference," in *Proceedings of the 17th conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 2001, pp. 362–369.

[85] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[86] R. Herbrich, N. D. Lawrence, and M. Seeger, "Fast sparse gaussian process methods: The informative vector machine," in *Advances in Neural Information Processing Systems*, 2003, pp. 625–632.

[87] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds., MIT Press, 2006, pp. 1257–1264.

[88] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[89] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[90] T. D. Bui, J. Yan, and R. E. Turner, "A unifying framework for gaussian process pseudo-point approximations using power expectation propagation," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 3649–3720, 2017.

[91] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 2845–2850.

[92] J. Gawlikowski *et al.*, "A survey of uncertainty in deep neural networks," *arXiv preprint arXiv:2107.03342*, 2021.

[93] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (gpml) toolbox," *Journal of machine learning research*, vol. 11, no. Nov, pp. 3011–3015, 2010.

[94] *Ikea chair adde obj model*, https://3dwarehouse.sketchup.com/model/abfb8bdb-da5c-4807-98ca-a92d84638bc8/IKEA-ADDE-10219178, (Last Accessed Sep. 14 2022).

[95] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, "Local gaussian process regression for real time online model learning," in *Advances in Neural Information Processing Systems*, 2009, pp. 1193–1200.

[96] D. Romeres, M. Zorzi, R. Camoriano, and A. Chiuso, "Online semi-parametric learning for inverse dynamics modeling," in *Conference on Decision and Control*, IEEE, 2016, pp. 2945–2950.

[97] T. Wu and J. Movellan, "Semi-parametric gaussian process for robot system identification," in *International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 725–731.

[98] D. Romeres, D. K. Jha, A. DallaLibera, B. Yerazunis, and D. Nikovski, "Semiparametrical gaussian processes learning of forward dynamical models for navigating in a circular maze," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 3195–3202.

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

[100] 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.

[101] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[102] M. Srinivasan, M. Abate, G. Nilsson, and S. Coogan, "Extent-compatible control barrier functions," *Systems & Control Letters*, vol. 150, p. 104 895, 2021.

[103] H. Bijl, J.-W. van Wingerden, T. B. Schön, and M. Verhaegen, "Online sparse gaussian process regression using fitc and pitc approximations," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 703–708, 2015.

[104] H. Bijl, T. B. Schön, J.-W. van Wingerden, and M. Verhaegen, "Online sparse gaussian process training with input noise," *stat*, vol. 1050, p. 29, 2016.