SPECIFICATION COMPOSITION AND CONTROLLER SYNTHESIS FOR ROBOTIC SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Paul Glotfelter

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Robotics

Georgia Institute of Technology

May 2019

SPECIFICATION COMPOSITION AND CONTROLLER SYNTHESIS FOR ROBOTIC SYSTEMS

Approved by:

Dr. Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Dr. Jonathan Rogers
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Samuel Coogan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Jorge Cortés
School of Mechanical and Aerospace Engineering
University of California, San Diego

Date Approved: March 8, 2019

ACKNOWLEDGEMENTS

Graduate school has been a hectic and transformative process. I've learned a lot, worked a lot, and met some amazing people. Throughout my program, my advisor, Magnus Egerstedt, has been invaluable asset. His knowledge and expertise has sharpened (read created) my theoretical skill set, and he taught me not just to do research but to do research with a story and purpose. My frequent collaborator Jorge Cortés has also been an integral part of my graduate journey, and it would be fair to say that, without him, my research would certainly have been smoother (and more continuous). It's also fair to say that without my undergraduate advisor, Patrick Martin, I likely wouldn't have attended graduate school at Georgia Tech. He helped inspire my love of robotics and has continued to be supportive and encouraging throughout my graduate studies.

No one goes through graduate school alone, at least happily. On this front, it would be a complete disservice not to thank the wonderful GRITSLab members for their enlightening conversation and friendship. The diversity of thought and culture has created an environment that engenders creativity and camaraderie. Being able to discuss research freely and openly with such an amazingly smart group of people is a rarity. On that note, I owe a special shoutout to my boy Ian Buckley with whom I have enjoyed many a high-quality conversation over lattes of varying quality.

The Robotarium has created more work for me than I ever could have imagined; however, it has also been one of the most rewarding experiences of my life. The Robotarium gang (Daniel, Sidd, Li, Gennaro, Jeremy, Yousef, and Sean) has made long days dealing with anything from stepper motors to networking issues enjoyable rather than tedious. I owe a debt of gratitude to both Daniel Pickem and Li Wang for their generous conversations regarding the Robotarium and barrier functions during the early stages of my Ph.D. process.

My family has also been very supportive and encouraging through this time. Angela, Maria, Tina (Mom), and Craig (Dad), your support means the world to me. Last, but certainly not least, I want to thank my wife, Misty, for her love and support, both emotionally and financially. You've been with me through long nights, from studying to running experiments in the Robotarium, and I couldn't have asked for a better partner during these years.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

From precision agriculture to autonomous-transportation systems, robotic systems have been proposed to accomplish a number of tasks. However, these systems typically require satisfaction of multiple constraints, such as safety or connectivity maintenance, while completing their primary objectives. The objective of this thesis is to endow robotic systems with a Boolean-composition and controller-synthesis framework for specifications of objectives and constraints. Barrier functions represent one method to enforce such constraints via forward set invariance, and Lyapunov functions offer a similar guarantee for set stability. This thesis focuses on building a system of Boolean logic for barrier and Lyapunov functions by using $\min$ and $\max$ operators. As these objects inherently introduce nonsmoothness, this thesis extends the theory on barrier functions to nonsmooth barrier functions and, subsequently, to controlled systems via control nonsmooth barrier functions. However, synthesizing controllers with respect to a nonsmooth function may create discontinuities; as such, this thesis develops a controller-synthesis framework that, despite creating discontinuities, still produces valid controllers (i.e., ones that satisfy the objectives and constraints). These developments have been successfully applied to a variety of robotic systems, including remotely accessible testbeds, autonomous-transportation scenarios, and leader-follower systems.

CHAPTER 1
INTRODUCTION

From precision agriculture to autonomous transportation, the usage of robots continuous to proliferate; and in domains such as precision agriculture, autonomous transportation, and factory floors, robots are changing how humans interact with the world. In each of these areas, robots must interact with the world around them and cooperate with humans toward the accomplishment of some task, which can typically be decomposed into a series of objectives and constraints that the system must satisfy. For example, an autonomous vehicle must successfully deliver its passenger to the required destination while obeying the speed limit and avoiding collisions.

Increasing the complexity of the situation, most of these applications require robots to execute a variety of tasks that may change over time. For instance, a robot involved in precision agriculture may have to visit a different series of crop patches each day, and an autonomous vehicle may encounter different, unexpected obstacles while in transit to a number of locations. As such, a need exists for frameworks and algorithms through which robots can accept high-level directives (e.g., from an engineer or planner) but, to address the potential variability, the robot must ultimately be capable of determining the best course of action for its current specification, even if it changes.

When dealing with controlled dynamical systems (e.g., a quadrotor or a vehicle), this process is often referred to as controller synthesis, and the goal of controller synthesis is usually an algorithm that allows robots to produce a controller automatically from a series of objectives and constraints. Consider an autonomous vehicle that must deliver a passenger while obeying the speed limit. In this case, the objective is to deliver the passenger, and the constraint is to obey the speed limit. A controller-synthesis algorithm would determine an input (e.g., linear and angular velocity) for the vehicle such that these high-level specifications are satisfied.

One canonical trade-off that exists when designing these frameworks relates to the particular responsibilities of the robot. In the previously discussed example, the autonomous vehicle is responsible for determining an appropriate direction in which to move such that it can deliver a passenger and obey the speed limit. However, one may also argue that the robot should determine which passenger it delivers and to what location. This argument forces the question: at what level should the controller-synthesis framework operate? This thesis makes the assumption that robots must determine the inputs to the system (e.g., velocity, acceleration), and the high-level specifications (i.e., objectives and constraints) are determined by some exterior method, like a human operator or a planner.

In turn, this assumption creates two areas of research. The first is at the specification level. In particular, how should specifications be provided to the system, and what capabilities should be available for manipulating them? The second is at the controller level: how should the robot synthesize controllers? These two questions capture the ideas behind this thesis, and the work herein is dedicated to providing an answer to these statements.

The remainder of this introduction indicates the theoretical context of this work and makes some comparisons to pre-existing methods. Later chapters discuss, in detail, the technical aspects of this work. The following sections are organized as follows: Section 1.1 denotes this thesis' philosophy and approach for encoding and manipulating specifications. Section 1.2 discusses the chosen controller-synthesis approach. Section 1.3 outlines the contributions of this thesis. Finally, Section 1.4 notes the organization for the chapters.

1.1 Encoding and Manipulating Specifications

A major aspect of this work relates to specifying objectives and constraints for robotic systems, and this section discusses the chosen methodology of this thesis to encode and manipulate specifications. Specifically, this thesis seeks to provide an encoding that is general enough to apply to most robotic systems while being flexible enough for controller synthesis. Accordingly, this work focuses on a specification that is system agnostic, meaning that it does not change based on the considered system; provably correct, in that the specification relates to theoretically rigorous concepts; and composable, implying that the specification must eventually be combined with potentially pre-existing controllers or other components of the system.

1.1.1 Encoding Constraints

In many of the aforementioned examples, the associated constraints may be formulated in a set-based fashion (e.g., obeying a speed limit, avoiding collisions), and since this work considers dynamical systems, forward set invariance represents a provable method for guaranteeing constraint satisfaction. Mathematically, forward set invariance requires that every trajectory of the system that starts in a particular set stays in that set for all time. In the case of autonomous vehicles, forward set invariance for collision avoidance means that, as long as the vehicles begin collision free, they stay collision free for all time. Importantly, forward set invariance provides a provable guarantee on trajectories for the system.

This thesis encodes constraints with barrier functions. Barrier functions have recently reemerged as a provably correct method for guaranteeing forward set invariance in dynamical systems [1, 2, 3, 4, 5, 6, 7] and have been utilized in various practical applications, from quadrotors to remotely accessible robotics testbeds [8]. Moreover, they exhibit a number of robustness properties that make them convenient in practice, leading to their utilization in real-time scenarios. For example, the work in [8] utilizes barrier functions to prevent collisions for an 80-dimensional swarm of differential-drive robots at 100 Hz.

Theoretically speaking, barrier functions ensure forward invariance by showing positivity along trajectories, meaning that the super-zero level set of the function becomes forward invariant. Validating a particular barrier function requires examining the derivative of the barrier function along trajectories and ensuring that it satisfies a certain rate function, in a very similar fashion to Lyapunov functions (e.g., see [9, 10]).

Barrier functions exhibit a number of desirable properties, leading to their usage in this thesis. They are: system agnostic, composable, implicit, and provably correct. Barrier functions are not formulated with respect to a specific dynamical system, so the robotic system in question does not theoretically limit them. Moreover, because barrier functions are fundamentally based on satisfying a particular differential inequality, they are composable. For example, this inequality can be included as a constraint in an optimization program to combine it with other components in the control hierarchy. This differential inequality also implies that barrier functions are implicit, much like Lyapunov functions, because satisfying the inequality point-wise across the state space produces a global result. For this reason, they do not require a look-ahead (e.g., as in model-predictive control), benefiting controller-synthesis algorithms from a computational perspective. Finally, as noted before, barrier functions are provably correct, because they provide guarantees based on forward set invariance.

There are many methods for guaranteeing forward set invariance, including potential functions and PDE-based methods, and later sections explicitly cover related methods. The prior statements do not mean to insinuate that barrier functions are superior to all other forward-invariance methods in the literature. However, barrier functions do offer a number of advantages over other methods with respect to the main applications of this thesis. Two properties that prove to be critically useful are the implicitness and composability of barrier functions, a quality that later chapters demonstrate. Later chapters also compare and contrast barrier functions with other methods in a technical context.

1.1.2 Encoding Objectives

In similar fashion to constraints (see Section 1.1.1), many objectives may be encoded using sets. For example, if an autonomous vehicle must reach a particular location, then the objective may be encoded as reaching a set that denotes the desired location. Formally, this property is set reachability or, more strongly, set stability. Throughout the controls and robotics literature, Lyapunov functions have been a go-to method for encoding stability [11, 9, 10]; as such, this thesis utilizes Lyapunov functions to encode set-based objectives. Furthermore, Lyapunov functions exhibit many desirable analogous qualities to barrier functions (see Section 1.1.1). Though, multiple philosophies exist with respect to satisfying objectives.

In contrast to constraints, objectives, in the context of this thesis, admit two approaches. Fundamentally, constraints are rigid. If taken to be set-based, then to satisfy the constraint, the system must remain in the set for all time (forward invariance). In contrast, objectives maintain fewer restrictions. That is, to complete an objective, the system only has to eventually enter the set. This line of thinking results in set stability as a method to encode objectives, and this method is particularly useful if the objective's specification may be manipulated (e.g., by a planner or human operator). Additionally, encoding objectives via set stability becomes useful when formal statements must be made regarding the objective (i.e., all trajectories eventually reach the desired set).

However, in some cases, the system may not have sufficient access to the objective, or the objective may be treated as a secondary goal. That is, the system must, at all costs, satisfy the constraints and, if possible, complete an objective. This philosophy treats the objective as nominal input, one that does not necessarily have to be applied. The Robotarium, a remotely accessible swarm-robotics testbed [8], represents an application that utilizes such a formulation. In particular, the safety constraints in this system, such as avoiding inter-robot collisions and collisions with walls, must be satisfied at all costs. The objective for the system is supplied from a remote user's code that generates a series of control inputs for the system. As such, the robot does not have direct access to the objective. However, for the testbed to be successful, the system must be able to execute the user's algorithm faithfully. The Robotarium represents a major concentration of this thesis, and later chapters discuss the developments of this system in detail (see Section 1.3). Both objective-based philosophies discussed in this section have merits, and this thesis addresses both strategies.

1.1.3 Manipulating Specifications

Having introduced the encoding method for specifications in Section 1.1.1 and Section 1.1.2, the question remains on how these specifications may be manipulated. In this thesis, methods are sought that apply to objectives and constraints generally. To accomplish this goal, this thesis utilizes Boolean composition, a well-used method. Boolean composition is ubiquitous throughout robotics and allows complex specifications to be built from simple atomic propositions, lending some much-needed flexibility. For example, a robot engaged in precision agriculture may have to visit a different series of crop patches over time. If the set of atomic propositions corresponds to various crop patches, then Boolean composition can capture the variability in the objective.

In terms of the theoretical content of this work, for Lyapunov and barrier functions, a system of Boolean logic may be encoded via $-$, $\max$, and $\min$ operators, which has been demonstrated in [12, 13]. However, $\min$ and $\max$ inevitably introduce nonsmoothness into the Boolean expression; as such, this thesis utilizes nonsmooth analysis, as in [14, 15, 16] to develop an appropriate theory. Yet, to address this nonsmoothness, generalized derivatives must be utilized, which inevitably capture discontinuities in the usual derivative. In turn, when used in controller synthesis, these generalized derivatives may produce discontinuous controllers. Consequently, the theory of discontinuous dynamical systems (e.g., see [17]) may be applied to analyze these discontinuous control laws. Section 1.2 contains a high-level introduction to the controller-synthesis method considered by this work.

1.2 Controller Synthesis

As noted in Section 1.1.3, discontinuous control laws may result from the controller synthesis. This property stems from the fact that the considered barrier and Lyapunov functions are nonsmooth. Discontinuous dynamical systems have a significant body of literature devoted to them; however, this prior work does not typically pertain to controller synthesis. As such, discontinuous controller synthesis represents the second main consideration for this thesis, allowing robots to produce controllers automatically from a high-level specification encoded by a Boolean expression.

As noted in Chapter 1.1, barrier functions can represent constraints and objectives may be encoded via Lyapunov functions or a pre-existing nominal input. Accordingly, this thesis seeks a framework that can synthesize controllers with respect to Lyapunov and barrier functions but also with respect to a pre-existing nominal controller.

Lyapunov functions and barrier functions represent two cornerstones of robotics: stability and invariance, and both of these objects fundamentally rely on satisfying a particular differential inequality. If a controller can be found that simultaneously satisfies both inequalities, then the system accomplishes its objective while maintaining the constraints. The question becomes: how can one produce such a controller? One method is for the designer to hand-craft a closed-form controller that satisfies these inequalities. However, if the specification changes, then this controller must change accordingly. This variability creates a need for a controller-synthesis method that does not depend on human intervention in the face of changes to the specification.

The work in [1] showed that such a controller may be realized through an optimization program. In particular, one may include the inequalities that result from a barrier or Lyapunov function into an optimization program, where the resulting controller satisfies both inequalities. Since this optimization program is solved online, the computational complexity of finding a solution must be considered. Fortunately, control-affine systems generate a control-affine inequality, allowing a validating controller to be synthesized via a Quadratic Program (QP). Typically, most solvers can find a solution to a QP in real time, even on resource-limited systems. Moreover, an optimization-based approach can synthesize controllers if the objectives are represented with Lyapunov functions or a nominal controller. For example, consider a QP whose cost function minimizes the squared distance between the synthesized controller and the nominal controller. This approach has been utilized extensively in the barrier function literature (e.g., [13, 18, 3, 2, 1, 7]) and in the Robotarium, which critically relies on this strategy [8].

1.3 Contributions

This section denotes the contributions of this thesis and their relevant chapters. These contributions correspond to three areas of research. Addressing Section 1.1, the first contribution corresponds to encoding and manipulating constraints and determining when systems can satisfy these constraints. Chapter 2 formulates an initial system of Boolean logic for barrier functions. Moreover, this chapter provides some sufficient conditions under which trajectories of discontinuous dynamical systems satisfy these specifications. In accordance with Section 1.2, the second contribution pertains to controller synthesis. Chapter 3 formulates nonsmooth barrier functions with respect to controlled systems and provides some preliminary controller-synthesis results that show when synthesizing a discontinuous control law still ensures that the system satisfies a specification containing constraints, and Chapter 4 extends this formulation to a class of hybrid systems. Chapter 5 generalizes the results of Chapters 2,3 to general Boolean expressions of Lyapunov and barrier functions. Chapter 6 strengthens the results of prior work and chapters by proving some regularity properties of the solution to a QP. Finally, Chapter 7 discusses the Robotarium, the third contribution, in which the theoretical results of this work are applied to a large-scale system.

1.4 Organization

Following the structure of Section 1.3, the major theoretical topics of this work are: Boolean composition of barrier (Lyapunov) functions and controller synthesis. Each chapter introduces the theoretical content therein, the contributed publication (if applicable) from which the chapter stems, and relates the chapter to relevant prior work. Then, the chapter provides the main theoretical results and, potentially, a robotics experiment that applies the theoretical results to a physical system. Finally, each chapter contains a conclusion. Appendix A discusses the notation for this thesis; Appendix B provides some background on discontinuous dynamical systems; and Appendix C notes some relevant results from nonsmooth analysis in the context of this thesis: the generalized gradient and its associated calculus.

CHAPTER 2
NONSMOOTH BARRIER FUNTIONS

This chapter contains the initial formulation of Nonsmoooth Barrier Functions (NBFs), and the content herein stems from the contributed publication [12]. Specifically, this chapter covers the theoretical formulation of NBFs with respect to closed-loop differential inclusions, provides some results on verifying NBFs in practice, notes a preliminary system of Boolean logic, and contains some experimental results showcasing these theoretical developments. In effect, this chapter lays the groundwork for encoding and manipulating specifications and also provides sufficient conditions for when a discontinuous system can satisfy the specification.

Some related work in the literature is as follows. Recently, [1] utilized barrier functions for constraint satisfaction by ensuring forward invariance of a set that encodes such safety requirements, and, subsequently, barrier functions have been used to encode a variety of system constraints across different domains, such as adaptive cruise control [1, 7], collision avoidance for ground vehicles [2], unmanned aerial vehicles [5], and remote-access robotics testbeds [8].

The above-referenced literature on barrier functions addresses a single, sufficiently smooth barrier function that operates on a continuous dynamical system. Recently, [3] achieves a form of Boolean composition through products and sums of barrier functions. However, the construction in [3] forgoes the robustness qualities of the zeroing barrier functions in [7] and restricts the system to lie strictly in the interior of the invariant set. In this chapter, we retain the robustness properties associated with zeroing barrier functions (see [7]) while supporting Boolean composition of barrier functions by utilizing $\max$ and $\min$ operators of multiple component barrier functions. However, the use of $\max$ and $\min$ operators introduces points of nondifferentiablity into the composite barrier functions, preventing the existing results from applying. Though not considered with regard to barrier functions, nonsmooth Lyapunov functions have been extensively studied (e.g., [19, 20, 15, 16]). The tools developed for nonsmooth Lyapunov functions will also prove highly useful for Nonsmooth Barrier Functions (NBFs), and in this chapter, we show how to extend the previously established concepts within the smooth barrier function literature to a rich class of NBFs.

It should be noted that NBFs are not the only possible tools for composition of system-level constraints in multi-agent systems. For example, potential functions and Lyapunov-like barrier functions represent an approach that also permits some degree of composition [21, 22, 23]. The major difference between this work and these other approaches lies in the fact that the work in this chapter explicitly allows for guaranteed Boolean composition of these objects (i.e., composition with Boolean $\wedge$, $\vee$, $\neg$ operators). Additionally, the above-mentioned prior methods are often formulated with respect to a particular task (e.g., obstacle avoidance) or a particular dynamical system (e.g., differential drive robots). Another strength of this work is that the NBF framework is mathematically agnostic to the particular task under consideration. For an extended comparison to pre-exisiting methods, see Chapter 4.

This chapter provides three main results with experimental validation. First, this chapter presents a framework that permits the application of NBFs to a class of systems described by differential inclusions. Second, this chapter addresses some computational requirements imposed by the nonsmooth nature of the NBF framework, demonstrating that validation of NBFs can be feasibly performed under certain assumptions. Third, a preliminary system of Boolean logic for NBFs is achieved via $\max$ and $\min$ operators. Using these theoretical results, this chapter presents an experiment in which a group of robots must avoid inter-robot collisions and collisions with spherical obstacles in the workspace.

This chapter is organized as follows. Section 2.1 introduces the system of interest. Section 2.2 formulates candidate and valid NBFs, indicating how forward invariance can be guaranteed via NBFs. Next, Section 2.3 provides sufficient conditions to guarantee that an NBF is valid with respect to a differential inclusion. Section 2.4 notes a preliminary system of Boolean logic for NBFs and provides a controller-synthesis result for continuous systems, and Section 2.5 utilizes the theoretical results of this chapter in an experiment on the Robotarium. Finally, Section 2.6 concludes the chapter.

2.1 System of Interest

This section introduces the system of interest for this chapter. In this case, the system of interest is a closed-loop differential inclusion. Though, this chapter eventually provides some preliminary results on controlled systems.

Differential inclusions have emerged as a tool to analyze certain types of dynamical systems. For example, differential equations with discontinuous right-hand sides have been extensively studied (e.g., in [24]) by transforming the discontinuous differential equation into a differential inclusion.

When formulating NBFs, we allow for applications to differential inclusions, potentially facilitating forward-set-invariance analysis of such systems; though, these results also apply to systems modeled by continuous differential equations. Given a set-valued map $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, consider the differential inclusion represented by

$$\dot{x}(t) \in F(x(t)), x(0) = x_0. \tag{2.1}$$

We assume that F is upper semi-continuous (see Appendix B) and takes nonempty, compact, convex values. These properties ensure the existence (but not uniqueness) of solutions to (2.1) (see Appendix B). In general, this chapter focuses on guaranteeing that a set is forward invariant with respect to a differential inclusion, meaning that every solution that starts in the set stays in the set (see Appendix B). This notion of forward invariance has been called strong forward invariance in other work (cf. [17]). This chapter simply refers to this property as forward invariance.

2.2 Candidate and Valid Nonsmooth Barrier Functions

Here, we define the concepts of candidate and valid NBFs. Note that, in Definition 1, the function h is not necessarily continuously differentiable. Importantly, if a candidate NBF is a valid NBF, then the set $\mathcal{C}$, as in Definition 1, is forward invariant. Valid and candidate NBFs are defined as follows.

Definition 1. *A locally Lipschitz function* $h : \mathbb{R}^n \to \mathbb{R}$ *is a* candidate Nonsmooth Barrier Function (NBF) *if and only if the set*

$$\mathcal{C} = \{x' \in \mathbb{R}^n \mid h(x') \geq 0\}$$

is nonempty.

Remark 2.1. *Because the desired result is forward invariance, the assumption that* $\mathcal{C}$ *is nonempty is technically unnecessary, as* $\mathcal{C} =$ *is forward invariant vacuously. However, enforcing that* $\mathcal{C}$ *is nonempty is an important practical consideration.*

Definition 2. *A candidate NBF* $h : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ *is a* valid Nonsmooth Barrier Function (NBF) *for* (2.1) *if and only if* $x(0) \in \mathcal{C}$ *implies that there exists a class-*$\mathcal{KL}$ *function* $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *such that*

$$h(x(t)) \geq \beta(h(x(0)), t),\ \forall\, t \in [0, t_1],$$

for all Carathéodory solutions $x : [0, t_1] \to \mathbb{R}^n$ *of* (2.1).

2.3 Sufficient Conditions for Valid Nonsmooth Barrier Functions

This section provides sufficient conditions that allow us to determine whether a candidate NBF is in fact a valid NBF. Toward this end, the following result will be useful.

Lemma 2.1. *Let $\alpha : \mathbb{R} \to \mathbb{R}$ be a locally Lipschitz extended class-$\mathcal{K}$ function and $h : [0, t_1] \to \mathbb{R}$ be an absolutely continuous function. If $\dot{h}(t) \geq -\alpha(h(t))$, for almost every $t \in [0, t_1]$, and $h(0) \geq 0$, then there exists a class-$\mathcal{KL}$ function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $h(t) \geq \beta(h(0), t)$, and $h(t) \geq 0,\ \forall\, t \in [0, t_1]$.*

Proof. To prove this result, we utilize a differential inequality. Toward this end, let

$$\dot{z}(t) = -\alpha(z(t)),\ z(0) = h(0).$$

Because α is locally Lipschitz, solutions $z(t)$ exist and are unique, and since $z(0) \geq 0$ and the restriction of an extended class-$\mathcal{K}$ function to $\mathbb{R}_{\geq 0}$ is a class-$\mathcal{K}$ function, the solution $z(t)$ is a class-$\mathcal{KL}$ function β such that

$$z(t) = \beta(z(0), t).$$

Therefore, the solution $z(t)$ is valid over $[0, t_1]$. Then, because

$$\dot{h}(t) \geq -\alpha(h(t)),\ a.e.\ t \in [0, t_1],$$

$h(t) \geq z(t), \forall t \in [0, t_1]$, by [25, Theorem 1.10.2]. Thus,

$$h(t) \geq \beta(h(0), t),\ \forall\, t \in [0, t_1],$$

proving the first claim. Because β is a class-$\mathcal{KL}$ function, $\beta(h(0), t) \geq 0,\ \forall\, t \in [0, t_1]$; thus, $h(t) \geq 0,\ \forall t \in [0, t_1]$. □

The following result states a sufficient condition for a candidate NBF to be valid in terms of its set-valued Lie derivative when evaluated along solutions to (2.1).

Theorem 2.1. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a candidate NBF. If there exists a locally Lipschitz* *extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that*

$$\min \partial_c h(x')^\top F(x') \geq -\alpha(h(x')), \forall x' \in \mathbb{R}^n,$$

then h is a valid NBF for (2.1).

Proof. Let $x(0) \in \mathcal{C}$. By Theorem C.3, each Carathéodory solution of (2.1) satisfies

$$\dot{h}(x(t)) \in \partial_c h(x')^\top F(x'),\ a.e.\ t \in [0, t_1].$$

Thus, at $a.e.\ t \in [0, t_1]$

$$\dot{h}(x(t)) \geq \min \partial_c h(x(t))^\top F(x(t)) \geq -\alpha(h(x(t))).$$

This condition implies that at $a.e.\ t \in [0, t_1]$

$$\frac{d}{dt}(h \circ x)(t) \geq -\alpha((h \circ x)(t)),$$

when $h \circ x$ is viewed as a function of t. Since $x(0) \in \mathcal{C}$, $(h \circ x)(0) \geq 0$. Directly applying Lemma 2.1 yields that h is a valid NBF, as defined in Definition 2. $\square$

Remark 2.2. *By a similar argument, if $x(0) \notin \mathcal{C}$ (i.e., $h(0) < 0$) and the solution exists for all $t \in [0, \infty)$, then we may show that $-h(x(t)) \leq \beta(-h(x(0)), t)$ (i.e., that $x(t)$ asymptotically returns to $\mathcal{C}$). In this case, h would effectively be treated like a Lyapunov function, and some additional assumptions on h may be needed (e.g., see [9]) to ensure stability.*

The experiment result of this chapter applies NBFs to a group of mobile robots. As such, the computational requirements of verifying the NBF inequality conditions become a concern. Toward this end, the following property of the usual inner product on two convex hulls becomes of use. In the interest of space efficiency, we omit this proof and note that it follows from Carathéodory's theorem for convex hulls.

Lemma 2.2. *Let $\bar{A} \subset \operatorname{co} A \subset \mathbb{R}^n$, $\bar{B} \subset \operatorname{co} B \subset \mathbb{R}^n$. If for every $a \in A$, $b \in B$,*

$$\langle a\,, b\rangle \geq c,\ c \in \mathbb{R},$$

then for every $\bar{a} \in \bar{A}$, $\bar{b} \in \bar{B}$,

$$\langle \bar{a}\,, \bar{b}\rangle \geq c.$$

Next, we present the second of this chapter's main results. We omit the proof and note that it follows from Lemma 2.2 and Theorem 2.1.

Theorem 2.2. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a candidate NBF. Let $\Phi_F,\ \Phi_h : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be set-valued maps such that*

$$F(x') \subset \operatorname{co} \Phi_F(x'), \partial_c h(x') \subset \operatorname{co} \Phi_h(x'),$$

for all $x' \in \mathbb{R}^n$. If there exists a locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that for every $x' \in \mathbb{R}^n$, $z \in \Phi_h(x')$, and $v \in \Phi_F(x')$,

$$\langle \xi\,, v\rangle \geq -\alpha(h(x')),$$

then h is a valid NBF for (2.1).

In Chapter 2.4, Theorem 2.2 facilitates the validation of candidate NBFs that are defined by $\max$ or $\min$ operations of smooth functions by expressing these sufficient conditions in terms of the component functions.

2.4 A Preliminary System of Boolean Logic

This section covers applications of $\max$ and $\min$ functions to the Boolean composition of barrier functions. In particular, this section demonstrates that these operators encode a system of Boolean logic falling into the NBF framework in Sec. 2.2. We also cover a QP-based formulation of these Boolean NBFs with respect to a class of continuous control-affine systems.

2.4.1 Composition by Boolean Logic

Throughout this section, we assume that a finite set of functions $h_i : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$, $i \in [k]$, are candidate NBFs. Within this framework, $\max$ represents a Boolean $\vee$ operation: that is, if $h^{\max} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$h^{\max}(x') = \max_{i \in [k]} \{h_i(x')\}, \tag{2.2}$$

for $x' \in \mathbb{R}^n$, is a candidate and valid NBF for (2.1), then at each $x' \in \mathbb{R}^n$, there exists at least one $i \in [k]$ such that $h_i(x') \geq 0$. Similarly, we note that $\min$ represents a Boolean $\wedge$ operation: that is, if $h^{\min} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$h^{\min}_{[k]}(x') = \min_{i \in [k]} \{h_i(x')\}, \tag{2.3}$$

for $x' \in \mathbb{R}^n$, is a candidate and valid NBF for (2.1), then at each $x' \in \mathbb{R}^n$, $h_i(x') \geq 0,\ \forall\, i \in [k]$. Furthermore, $-h$ represents $\neg h$. These expressions allow for the application of De Morgan's laws in that $h_1 \vee h_2 = \neg(\neg h_1 \wedge \neg h_2)$, permitting full Boolean composition.

Having noted the utility of $\min$ and $\max$ as Boolean operators, we focus on the criteria that these Boolean NBFs must satisfy to be covered under the results of Section 2.3. In the interest of space efficiency, we omit the proof of this result and note that it follows from Proposition C.1 and Theorem 2.2. Proposition 2.1 holds for the $\min$ operator as well.

Proposition 2.1. *Let $h_i : \mathbb{R}^n \to \mathbb{R}$, $i \in [k]$, be a finite set of locally Lipschitz functions which are candidate NBFs, and let $h^{\max} : \mathbb{R}^n \to \mathbb{R}$ be defined as in* (2.2)*. For each $x' \in \mathbb{R}^n$, let*

$$I_h^a(x') = \{i \in [k] : h_i(x') = h^{\max}(x')\}\},$$

and consider the set-valued map $\Phi_h : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ defined by

$$\Phi_h(x') = \bigcup_{i \in I_h^a(x')} \partial_c h_i(x').$$

If $h^{\max}$ is a candidate NBF and there exists a locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that for every $x' \in \mathbb{R}^n$, $z \in \Phi_h(x')$, and $v \in F(x')$,

$$\langle z \, , v \rangle \geq -\alpha(h_{[k]}^{\max}(x')),$$

then $h^{\max}$ is a valid NBF for (2.1)*.*

2.4.2 Quadratic-Program-Based Controllers

The formulation of a smooth barrier function with respect to control-affine systems produces an affine constraint on the system, and coupling this affine constraint with the minimization of a quadratic cost, at each point in time, results in a quadratic program (e.g., [1, 3]). This section provides similar results for NBFs with respect to a class of control-affine systems. In the nonsmooth case, the component functions generate a series of constraints, rather than a single constraint, that must be enforced point-wise in time. In the interest of space, we omit the proof and note that it follows from [26, Theorem 1] and Prop. 2.1.

Proposition 2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$, $G : \mathbb{R}^n \to \mathbb{R}^{n \times m}$, and $u : \mathbb{R}^n \to \mathbb{R}^m$ be locally Lipschitz, and consider the control-affine system $\dot{x}(t) = f(x(t)) + G(x(t))u(x(t))$. Let $h^{\min} : \mathbb{R}^n \to \mathbb{R}$ be defined as in* (2.3)*, where each $h_i : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable candidate NBF with a locally Lipschitz derivative. Consider the functions*

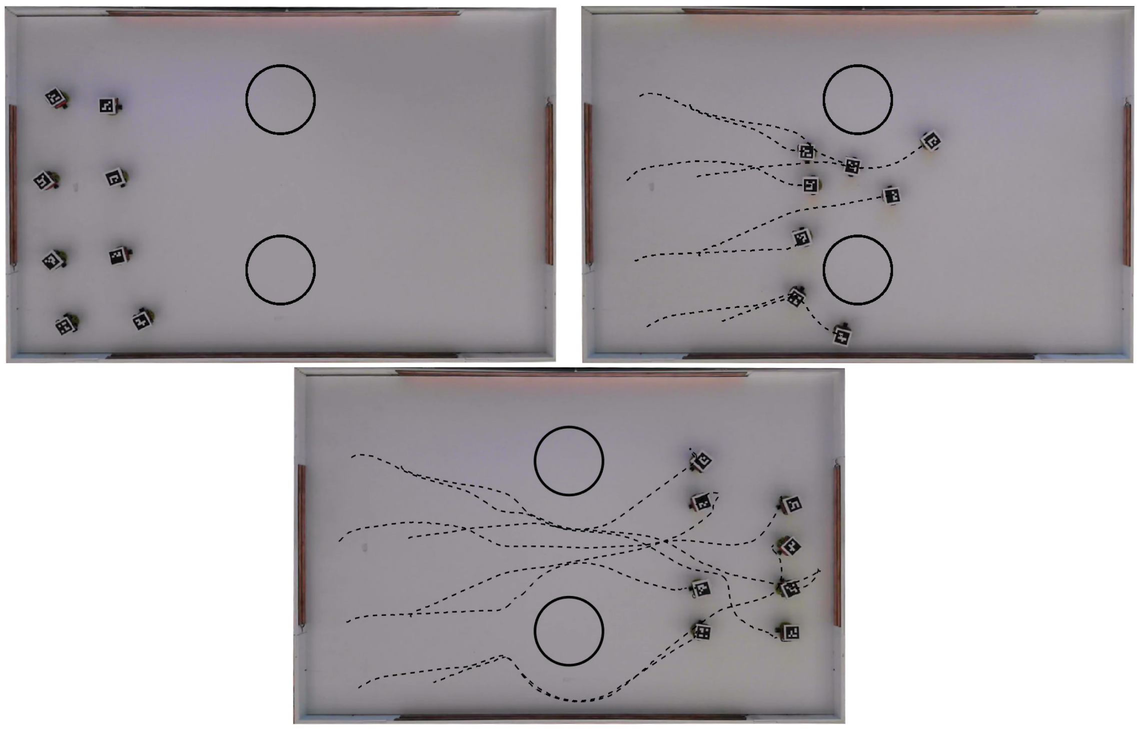

Figure 2.1: A group of 8 differential-drive robots in the Robotarium successfully navigate through a pair of obstacles (circles) to their desired destination (crosses) and avoid inter-robot collisions. This task is accomplished by solving online for a QP-based controller with respect to the NBF in (2.4) that encodes and enforces safety requirements.

$w^* : \mathbb{R}^n \to \mathbb{R}$ and $u^* : \mathbb{R}^n \to \mathbb{R}^m$ defined by

$$\begin{aligned} & w^*(x') = \min_{(u,w) \in \mathbb{R}^{m+1}} w \\ & \textit{s.t. } \nabla h_i(x')^T (f(x') + G(x')u) + \alpha(h_i(x')) - w \geq 0, \forall\, i \in [k] \end{aligned}$$

and

$$\begin{aligned} & u^*(x') = \operatorname*{arg\,min}_{u \in \mathbb{R}^m} u^T H(x') u + b(x')^T u \\ & \textit{s.t. } \nabla h_i(x')^T (f(x') + G(x')u) + \alpha(h_i(x')) \geq 0,\ \forall\, i \in [k], \end{aligned}$$

where $H : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ is locally Lipschitz, symmetric, positive definite and $b : \mathbb{R}^n \to \mathbb{R}^m$ is locally Lipschitz. If $h^{\min}$ is a candidate NBF and $w^*(x') > 0$, for all $x' \in \mathbb{R}^n$, then u^* is locally Lipschitz; and $h^{\min}$ is a valid NBF for the closed-loop system under the controller u^*.

Remark 2.3. *In this result, local Lipschitz continuity is obtained via application of [26, Theorem 1]; however, only continuity of the controller u^* is technically required to obtain this result.*

Intuitively, w^* in the above result gives some notion of the width of the feasible set of solutions. If the feasible set has non-zero width at all points, then a locally Lipschitz solution may be selected from the feasible set. In general, the computational complexity of a QP depends on the decision variables, the constraints, and the utilized solver. For an excellent survey of these methods for multi-agent systems, we refer the reader to [5].

2.5 Experimental Results

This section features a group of robots in the Robotarium (see [8]), which is a remote-access, multi-agent robotics test bed. The agents attempt to achieve a navigation objective by utilizing a given controller that accomplishes the desired goal but disregards safety measures: inter-agent collisions and static obstacles. In this experiment, a QP wraps the existing controller in an NBF framework such that it simultaneously satisfies multiple safety requirements and fulfills the intent behind the original controller. Note that, throughout this section, an explicit time dependence is dropped for clarity.

Consider a team of 8 planar, single-integrator agents each with state $x_i \in \mathbb{R}^2,\ i \in [8]$, and dynamics $\dot{x}_i = u_i(x)$. To solve the ensemble problem via QP, we stack the states and inputs into vectors $x = \begin{bmatrix} x_1^T & \dots & x_8^T \end{bmatrix}^T$, where $x \in \mathbb{R}^{16}$ and $x \mapsto u(x)$ is defined in the same fashion. The agents' objective is to drive from their initial condition to some pre-specified goal points $x^g \in \mathbb{R}^{16}$, which is accomplished by use of a locally Lipschitz proportional controller

$$u^{obj}(x) = x^g - x.$$

To avoid collisions with other agents, the following Boolean NBF applies to each pair of agents

$$h^c(x) = \bigwedge_{i=1}^{8} \bigwedge_{j=i+1}^{8} \|x_i - x_j\|^2 - (\delta^c)^2,$$

where $\delta^c > 0$. Similarly, each agent avoids collisions with two circular obstacles in the plane via the NBF

$$h^o(x) = \bigwedge_{i=1}^{8} \bigwedge_{j=1}^{2} \|x_i - o_j\|^2 - (\delta^o)^2,$$

where $o_j \in \mathbb{R}^2$ indicates the static position of an obstacle and $\delta^o > 0$. The final Boolean NBF is given by

$$h^{\min}(x) = h^c(x) \wedge h^o(x). \tag{2.4}$$

Now, we examine the derivatives of the component barrier functions of h^c and h^o. Taking a component barrier function in h^c with agents i and j yields

$$\frac{d}{dt}\left(\|x_i - x_j\|^2 - (\delta^c)^2\right) = A^{ij}(x)u.$$

Here, the superscript A^{ij} indicates that this vector describes the derivative for agents i and j. A^{ij} maps to a row vector whose indices satisfy

$$A_i^{ij}(x') = 2(x_i' - x_j')^T,\ A_j^{ij}(x') = -A_i^{ij}(x'),\ A_k^{ij}(x') = 0,$$

where $k \neq i, j$ and the subscript indicates a particular two-dimensional element of $A^{ij}(x')$. Importantly, A^{ij} is locally Lipschitz.

Similarly, each component function of h^o will have a derivative for agent i and obstacle j

$$\frac{d}{dt}\left(\|x_i - o_j\|^2 - (\delta^o)^2\right) = B^{ij}(x)u,$$

where the superscript B^{ij} indicates that this function is between agent i and obstacle j. B^{ij} maps to a row vector whose indices satisfy

$$B_i^{ij}(x') = 2(x_i' - o_j)^T,\ B_k^{ij}(x') = 0,\ k \neq i,$$

where the subscript indicates a particular two-dimensional element in $B^{ij}(x')$. In this case, B^{ij} is also locally Lipschitz.

Now, we utilize the QP formulation noted in Proposition 2.2 with the objective function $u^T u - 2u^{obj}(x)^T u$, which is equivalent to minimizing the squared norm $\|u - u^{obj}(x)\|^2$. This cost attempts, at each point in time, to minimally modify the existing controller $u^{obj}(x)$ such that the modified controller achieves the safety objectives. In this experiment, we assume that the selection $\alpha(h^{\min}(x)) = \gamma h^{\min}(x)^3$, $\gamma > 0$ makes w^*, as defined in Prop. 2.2, satisfy the condition $w^*(x') > 0$ for all $x \in \mathbb{R}^{16}$.

The QP is formulated as in Proposition 2.2 with the parameters $\gamma = 1000$, $\delta^c = 0.04$, $\delta^o = 0.1$; and we deploy the resulting controller onto the Robotarium's team of unicycle-modeled robots using the method in [27, Section 5].

Figure 2.1 displays the mobile robots during this experiment, and Figure 2.2 shows the NBF of (2.4) during the course of the experiment. The Boolean NBF in (2.4) starts positive and remains positive over the course of the experiment; thus, all component barrier functions are simultaneously satisfied. Furthermore, as a result of the minimally invasive modification, the robots also arrive at the desired goal position, satisfying their original navigation objective and the NBF. Additionally, we note that the width of the feasible set remains strictly greater than zero, validating the application of Proposition 2.2.

2.6 Conclusion

In this chapter, we have introduced a class of Nonsmooth Barrier Functions (NBFs), showing that existing results for smooth barrier functions apply to NBFs and allowing preliminary formulation of Boolean NBFs via $\max$, $\min$, and $-$ operators. Furthermore, we have provided results that illustrate some computational methods for these conditions, allowing one to validate a class of NBFs with quadratic programs. To validate these results, a Boolean NBF was deployed onto a team of mobile robots in the Robotarium. Following the theoretical developments in this chapter, two improvements are possible: extending NBFs to be explicitly controlled and expanding the Boolean composition framework. These topics are covered in Chapters 3-5.

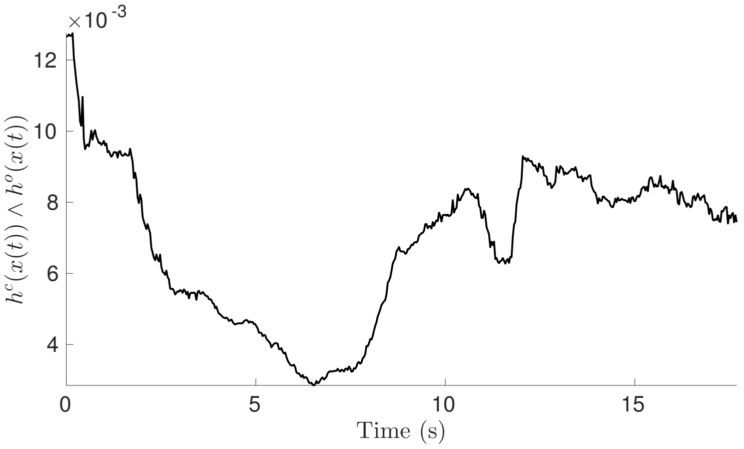

Figure 2.2: Value of Boolean NBF in (2.4) over the course of the experiment. Because the NBF remains positive over time, all safety objectives are simultaneously satisfied.

CHAPTER 3
CONTROL NONSMOOTH BARRIER FUNCTIONS

Following the formulation of NBFs in Chapter 2, this chapter, related to contributed work [13], formulates NBFs with respect to controlled systems, resulting in Control NBFs (CNBFs). This section also focuses some attention on a preliminary controller-synthesis strategy for CNBFs. In Chapter 2, the theoretical results mainly apply to closed-loop differential inclusions, but it provided some results on NBFs for continuous controlled dynamical systems in the experimental results. However, due to the nondifferentiable properties of NBFs it is not always possible to synthesize a continuous controller. That is, the assumptions in Chapter 2 are overly restrictive and cannot be guaranteed in general by this chapter. As such, this chapter dedicates some attention toward resolving this issue.

Chapter 2 extends the theory of barrier functions to include nonsmooth functions, stemming from the contributed work in [12]. Tools from nonsmooth analysis, as in [14, 15, 16], enable a Boolean logic system for these NBFs via $\max$ and $\min$ operators, which encapsulate set unions and intersections, respectively. However, this previous chapter does not explicitly consider controlled systems, a necessary condition for controller synthesis.

A related body of prior work has shown that controller synthesis via Quadratic Programs (QPs) can be used to minimally modify an existing controller such that a barrier function remains valid [1, 4, 5, 2, 3], and such methods have seen success on large-scale multi-robot systems but have yet to be extended to Boolean composition of NBFs [8]. In particular, this technique involves taking a derivative along trajectories, considering a sufficient rate function, and generating an inequality constraint for a QP. If this constraint is satisfied at all points, then the forward-set-invariance property holds. However, the nonsmooth case correspondingly requires a generalized derivative, which results in a discontinuous constraint; as such, a discontinuous control inputs may result from this process and must be considered.

By combining and extending the previous work on Boolean composition of NBFs and controller synthesis via QPs, this chapter develops a constraint-satisfaction and controller-synthesis framework that can be deployed onto multi-robot systems. As in [2, 8, 3], this framework can operate in real time and can be combined with existing controllers. For validation, the proposed framework solves a leader-follower problem, a classic problem for multi-agent systems.

To enable the above-mentioned framework, this chapter provides the following theoretical results. Considering a class of control-affine systems and allowing discontinuities in the control input, this chapter formulates NBFs with respect to this system, resulting in CNBFs, and extends the results on NBFs to CNBFs using the techniques from [14, 16, 28, 12].

Next, we focus on providing a system of Boolean logic with Boolean NBFs (BNBFs), which are Boolean combinations of NBFs. This framework leverages the work in Chapter 2 and extends it by explicitly considering discontinuous control inputs. This formulation supports the main result of this chapter: the development of an almost-active gradient for BNBFs that is suited for control synthesis via a QP. The main result proves that this object, when used as a constraint to a QP, provides a validating, though potentially discontinuous, controller.

The chapter is organized as follows. Section 3.1 introduces the problem statement and notes the system of interest. Section 3.2 develops CNBFs, notes some Boolean composability requirements, formulates the almost-active gradient, and constructs a controller-synthesis algorithm via a QP, providing the main results of this chapter. Accordingly, Section 3.3 shows the deployment of a BNBF onto a multi-robot system with leader-follower constraints, and Section 3.4 concludes the chapter.

3.1 Problem Statement and System of Interest

This section presents the particular application that this chapter seeks to solve (i.e., the leader-follower problem) and discusses the system of interest. For background on NBFs, see Chapter 2.

3.1.1 Problem Statement

As a motivating example for this chapter, consider a leader-follower team of N robots with planar states $x_i \in \mathbb{R}^2$, $i \in [N]$, where the leaders must perform a task; but, at the same time, all robots must satisfy a collection of constraints. For example, the robots must not collide. Pairwise, the inequality

$$\|x_i - x_j\| \geq \delta_{col}$$

encodes this constraint, for some $\delta_{col} > 0$. Furthermore, the function

$$h_{ij}(x_i, x_j, \delta_{col}) = \|x_i - x_j\|^2 - \delta_{col}^2, \; i, j \in [N],$$

captures this inequality (i.e., consider $h_{ij}(x_i, x_j, \delta_{col}) \geq 0$).

The robots must also avoid collisions with a fixed number, O, of obstacles, which can be captured by the function

$$h_{ij}(x_i, o_j, \delta_{obs}), \; i \in [N], j \in [O], \tag{3.1}$$

where the fixed value $o_j \in \mathbb{R}^2$ represents the known location of the obstacle and $\delta_{obs} > 0$ indicates the size of the obstacle.

The subset of leader robots, denoted by $N_L \subset [N]$, must travel to pre-specified goal points $x_{i,g} \in \mathbb{R}^2$, $i \in N_L$. The rest of the robots, the followers, denoted by $N_F \subset [N]$, must remain close to one of the leaders. Pairwise, the inequality

$$\|x_i - x_j\| \leq \delta_{con},\ i \in N_L, j \in N_F,$$

represents this criterion. In terms of (3.1), the function

$$-h_{ij}(x_i, x_j, \delta_{con})$$

encapsulates this connectivity constraints. This symbol represents a Boolean $\neg$ (NOT) operation.

Using these pairwise constraints and a system of Boolean logic, the above barrier functions may be composed to satisfy the system-wide constraints. In particular, the Boolean compositions

$$h_{col} = \bigwedge_{i=1}^{N-1} \bigwedge_{j=i+1}^{N} h_{ij}(\cdot, \cdot, \delta_{col}),\ h_{obs} = \bigwedge_{i=1}^{N} \bigwedge_{j \in O} h_{ij}(\cdot, o_j, \delta_{col})$$

encapsulate all of the collision constraints, where the large $\wedge$ symbol refers to Boolean $\wedge$ (AND) and the large $\vee$ symbol refers to Boolean $\vee$ (OR). Similarly, the Boolean composition

$$h_{con} = \bigwedge_{i \in N_F} \bigvee_{j \in N_L} \neg h_{ij}(\cdot, \cdot, \delta_{con})$$

captures the followers' connectivity constraint to the leaders, and taking

$$h = h_{col} \wedge h_{obs} \wedge h_{con} \tag{3.2}$$

yields the all-encompassing constraint.

The resulting function begs the question: how does one enforce the Boolean compositions encoded by (3.2)? As such, the main contribution of this chapter, which is contained in the publication [13], shows how to synthesize an appropriate controller from (3.2) for a team of mobile robots by considering it as an CNBF.

3.1.2 System of Interest

In this chapter, the differential inclusion

$$\dot{x}(t) \in F(x(t)),\ x(0) = x_0 \tag{3.3}$$

becomes of interest, where $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is an upper semi-continuous, nomempty, compact, convex set-valued map (see Appendix B). These conditions ensure that Cartahéodory solutions to the differential inclusion exist [17]. For a comprehensive survey of set-valued maps and discontinuous differential equations, see [17].

More specifically, we consider control-affine systems of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)), \tag{3.4}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are continuous. The controller $u : \mathbb{R}^n \to \mathbb{R}^m$ is assumed to be a feedback control; however, this treatment only requires u to be measurable and locally bounded. Note that these properties indeed capture discontinuities.

To create a system for which solutions exist, a discontinuous dynamical system, such as in (3.4), can be turned into a differential inclusion via Filippov's operator

$$\begin{aligned}\dot{x}(t) &\in K[f + gu](x(t)) = \operatorname{co} L[f + gu](x(t)) \\ &= \operatorname{co}\{\lim_{i\to\infty} f(x_i) + g(x_i)u(x_i) : x_i \to x(t),\ x_i \notin S_f, S\},\end{aligned} \tag{3.5}$$

where S_f is a particular zero-measure set that depends on the system and S is any zero-measure set. The resulting set-valued map, $K[f + gu] : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ satisfies the aforementioned sufficient conditions to permit existence of solutions to (3.3) with $F = K[f + gu]$. See Appendix B for more details.

3.2 Control Nonsmooth Barrier Functions

This section contains the main results of this chapter: formulating Control NBFs (CNBFs); providing a Boolean logic system for them, resulting in Boolean NBFs (BNBFs); and addressing controller synthesis. As such, we identify some requirements for CNBFs, which make them amenable to controller synthesis via a QP. Using these requirements, the almost-active gradient is formulated. This object, when used as a constraint to a QP, ensures that the resulting controller, which is possibly discontinuous, validates the BCNBF.

3.2.1 Boolean and Control Nonsmooth Barrier Functions

This section defines CNBFs and BNBFs. In particular, the definition of CNBFs ensures validation via Theorem. 2.1, guaranteeing the desired forward invariance property with respect to a particular differential inclusion.

Definition 3. *A candidate NBF $h : \mathbb{R}^n \to \mathbb{R}$ is a* valid Control Nonsmooth Barrier Function (CNBF) *for* (3.5) *if and only if there exists a locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ and a measurable, locally bounded function $u : \mathbb{R}^n \to \mathbb{R}^m$ such that*

$$\min \partial_c h(x')^\top K[f + gu](x') \geq -\alpha(h(x')),\ \forall x' \in \mathbb{R}^n.$$

Remark 3.1. *If a candidate NBF is a valid CNBF for* (3.5)*, then $\mathcal{C}$ is forward invariant with respect to the differential inclusion*

$$\dot{x}(t) \in K[f + gu](x(t)), x(0) = x_0.$$

Chapter 2 provides a system of Boolean logic for NBFs to create BNBFs. However, it did not address controller synthesis for general NBFs. As such, this chapter now focuses on some relevant regularity assumptions for BNBFs, toward controller synthesis in Section 3.2.2. Recall that, for a pair of candidate NBFs, $h_1, h_2 : \mathbb{R}^n \to \mathbb{R}$, a Boolean NBF

(BNBF) is given by

$$
\begin{aligned}
h(x') &= \min\{h_1(x'), h_2(x')\} \coloneqq h_1 \wedge h_2 \\
h(x') &= \max\{h_1(x'), h_2(x')\} \coloneqq h_1 \vee h_2 \\
h(x') &= -h_1(x') \coloneqq \neg h_1,
\end{aligned}
$$

at each $x' \in \mathbb{R}^n$ (cf. Chapter 2).

In general, a BNBF $h : \mathbb{R}^n \to \mathbb{R}$ can be comprised of a finite number of component functions with the above-noted Boolean operators. In this case, h is denoted

$$h = B[h_1, \dots, h_k],$$

where B represents a Boolean logic expression containing the operators in Definition 3.2.1. An important class of BNBFs are those composed of smooth functions.

Definition 4. *A candidate BNBF $h : \mathbb{R}^n \to \mathbb{R}$ defined by $h = B(h_1, \dots, h_k)$ is smoothly composed if each component candidate NBF $h_i : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable.* •

Definition 4 implies, from Proposition C.1, that, at a point $x' \in \mathbb{R}^n$,

$$\partial_c h(x') \subset \mathrm{co}\{\nabla h_i(x') : i \in I \subset [k]\} \cup \{-\nabla h_i(x') : i \in I \subset [k]\},$$

for some appropriate index set I, where ∇ denotes the usual gradient. This encapsulating set becomes particularly important when synthesizing controllers with a QP in Section 3.2.2.

3.2.2 Controller Synthesis via Quadratic Programs

To enable controller synthesis via a QP, this section defines some useful objects. These tools capture the composability requirements outlined in Section 3.2 and ensure that synthesized

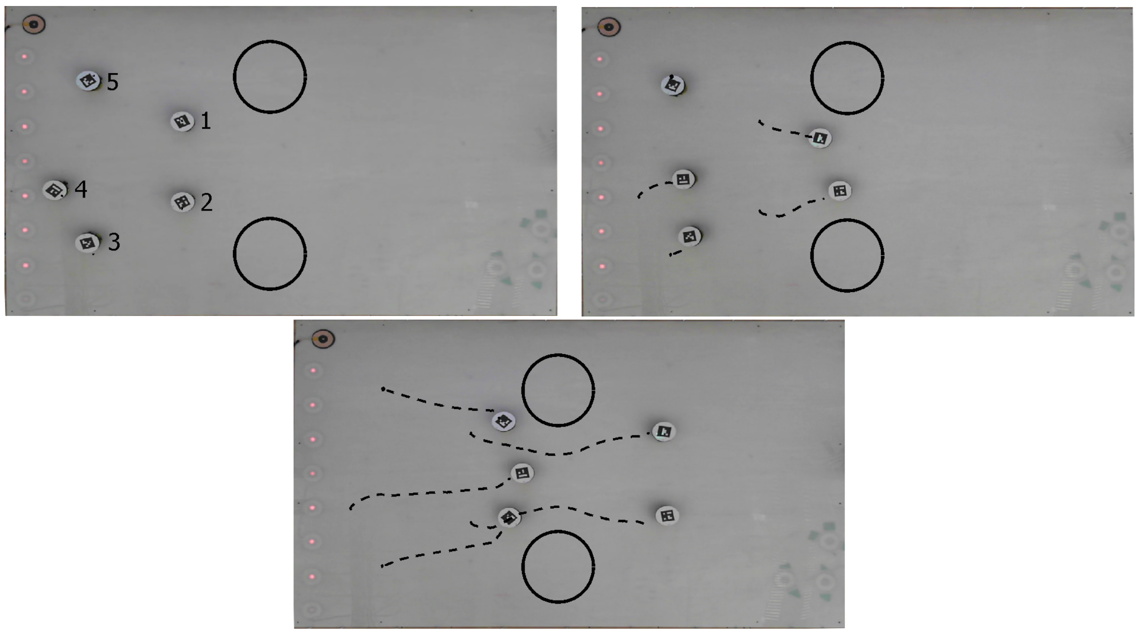

Figure 3.1: A group of 5 differential-drive robots in the Robotarium execute the experiment detailed in Section 3.3. In particular, all robots avoid inter-agent collisions and obstacle collisions; each of the three follower robots maintain connectivity to one of the leader robots; and the leader robots successfully achieve their pre-specified goal position. These results show that the synthesized controller satisfies the constraints and completes the pre-existing objective.

controllers validate the requisite BNBF. To motive the following discussion, consider the following argument. When validating CNBFs, the inequality

$$\partial_c h(x')^\top K[f+gu](x') \geq -\alpha(h(x')) \tag{3.6}$$

must be satisfied for every $x' \in \mathbb{R}^n$ for some extended class-$\mathcal{K}$ $\alpha : \mathbb{R} \to \mathbb{R}$. As such, the behavior of the controller around the point x' becomes crucial, because

$$\partial_c h(x')^\top (f(x') + g(x')u(x')) \geq -\alpha(h(x'))$$

does not imply that (3.6) does not hold.

Moreover, (3.6) combines all possible directions between the dynamics and the generalized gradient. As such, any active function in $\partial_c h(x')$, where h is a BNBF, must be included in a neighborhood of x'. As such, the generalized gradient does not effectively capture this locality, and it must be extended to provide enough regularty to satisfy (3.6). This criterion motivates the following developments.

Definition 5. *Let $\epsilon > 0$ and two candidate NBFs $h_1, h_2 : \mathbb{R}^n \to \mathbb{R}$, the almost-active set of functions for a candidate BNBF given by $h = h_1 \wedge h_2$ or $h = h_1 \vee h_2$ is defined at each $x' \in \mathbb{R}^n$ as*

$$I_\epsilon(x') = \{i : |h_i(x') - h(x')| \leq \epsilon\}.$$

The almost-active gradient of a BNBF, denoted by $\partial_\epsilon h : \mathcal{D} \subset \mathbb{R}^n \to 2^{\mathbb{R}^n}$, at a point $x' \in \mathbb{R}^n$ is

$$\partial_\epsilon h(x') = \mathrm{co} \bigcup_{i \in I_\epsilon(x')} \partial h_i(x').$$

•

The following results shows that QPs with an almost-active-gradient constraint generate validating controllers for smoothly composed BNBFs. To do so, the behavior of the almost-active gradient becomes relevant.

Lemma 3.1. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a candidate BNBF as in Definition 5, and let $\epsilon > 0$. At every $x' \in \mathbb{R}^n$, if $h_i(x') = h(x')$, then there exists $\delta > 0$ such that the almost-active set of functions satisfies $i \in I_\epsilon(y)$, for all $y \in B(x', \delta)$.*

Proof. Let $x' \in \mathbb{R}^n$, and let i be such that $h_i(x') = h(x)$. By continuity of h_i, h there exists $\delta > 0$ such that

$$|h_i(y) - h_i(x')| \leq \epsilon/2, \; |h(y) - h(x')| \leq \epsilon/2,$$

for all $y \in B(x', \delta)$. Then,

$$\begin{aligned}|h_i(x') - h(y)| &= |h_i(y) - h(y) - h_i(x') + h(x')| \\ &\leq |h_i(y) - h_i(x')| + |h(x') - h(y)| \leq \epsilon.\end{aligned}$$

Therefore, $i \in I_\epsilon(y)$, for all $y \in B(x', \delta)$. □

Applying Lemma 3.1 on a smoothly composed BNBF yields the main result on controllers resulting from QPs with the almost-active gradient as a constraint. The next theorem provides a controller-synthesis result for a restricted class of BNBFs. Later chapters extend this procedure to general BNBFs.

Theorem 3.1. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a smoothly composed candidate BNBF, as in Definition 5. If there exists $\epsilon > 0$ and a locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that the Quadratic Program (QP)*

$$\begin{aligned}u^*(x') \in \underset{u \in \mathbb{R}^m}{\arg\min}\, & u^\top A(x') u + b(x')^\top u \\ \textit{s.t. } & \langle \partial_\epsilon h(x') \,, f(x') + g(x') u \rangle \geq -\alpha(h(x')),\end{aligned}$$

with $A : \mathbb{R}^n \to \mathbb{R}^{m \times m}$, $b : \mathbb{R}^n \to \mathbb{R}^m$ continuous, has a solution for every $x' \in \mathbb{R}^n$ and u^* is measurable and locally bounded, then h is a valid CNBF for (3.5).

Proof. Let $x' \in \mathbb{R}^n$. Since h is smoothly composed,

$$\partial h(x') \subset \operatorname{co}\{\nabla h_i(x') : i \in I(x')\},$$

by Proposition C.1. By Theorem 2.2, showing that

$$\langle \nabla h_i(x') \, , L[f+gu] \rangle \geq -\alpha(h(x'))$$

for each $i \in I(x')$ suffices to achieve the desired result. Take $i \in I(x')$. By definition, $h(x') = h_i(x')$, so applying Lemma 3.1 implies that there exists $\delta > 0$ such that $i \in I_\epsilon(y)$, for all $y \in B(x', \delta)$. As such, u^* satisfies

$$\langle \nabla h_i(y) \, , f(y) + g(y)u^*(y) \rangle \geq -\alpha(h(y)),$$

for all $y \in B(x', \delta)$.

Let $v \in L[f+gu]$. Then, there exists a sequence $x_j \to x'$ such that $f(x_j)+g(x_j)u^*(x_j) \to v$. Moreover, the existence of the limit implies that the same limit holds for any subsequence. Since $x_j \to x'$, there exists a k such that $\|x_j - x'\| \leq \delta$ for all $j \geq k$ so, reusing notation, consider a subsequence $x_j \to x'$ with $j \geq k$.

Because ∇h_i, α, and $\langle \cdot \, , \cdot \rangle$ are continuous

$$\begin{aligned}
&\langle \nabla h_i(x') \, , v \rangle + \alpha(h(x')) = \\
&\langle \lim_{j\to\infty} \nabla h_i(x_j) \, , \lim_{j\to\infty} (f(x_j) + g(x_j)u^*(x_j)) \rangle + \lim_{j\to\infty} \alpha(h(x_j)) \\
&= \lim_{j\to\infty} \langle \nabla h_i(x_j) \, , f(x_j) + g(x_j)u^*(x_j) \rangle + \lim_{j\to\infty} \alpha(h(x_j)) \\
&= \lim_{j\to\infty} \left(\langle \nabla h_i(x_j) \, , f(x_j) + g(x_j)u^*(x_j) \rangle + \alpha(h(x_j)) \right) \\
&= \lim_{j\to\infty} a_j,
\end{aligned}$$

where $a_j = \langle \nabla h_i(x_j)\, , f(x_j) + g(x_j)u^*(x_j) \rangle + \alpha(h(x_j))$. By assumption, $a_j \geq 0$ for all j, since $\|x_j - x'\| \leq \delta$; therefore,

$$\lim_{j \to \infty} a_j \geq 0,$$

implying that

$$\langle \nabla h_i(x')\, , v \rangle \geq -\alpha(h(x'))$$

and completing the proof. □

The experimental results in Section 3.3 rely on a slightly generalized version of Theorem 3.1, which is not given in this chapter. The exact proof of this result would involve a generalization of the almost-active set of functions and Lemma 3.1 to BNBFs with nested component functions. In fact, Chapter 5 contains this generalized version of Theorem 3.1.

3.3 Experimental Results

This experiment solves the problem posed in Section 3.1.1, utilizing the same notation. Consider $N = 5$ robots with planar states and dynamics

$$\dot{x}_i = u_i.$$

This experiment also references the ensemble state $x \in \mathbb{R}^{2N}$ with input $u \in \mathbb{R}^{2N}$. Moreover, $N_L = \{1, 2\}$ and $N_F = \{3, 4, 5\}$.

Robots 1 and 2 travel from a specified initial condition to a pre-specified goal point $x_{i,g} \in \mathbb{R}^2$ with the controller

$$u_{i,nom}(x_i) = x_{i,g} - x_i,$$

for $i \in N_L$. Meanwhile, robots 3, 4, and 5 must remain close to either the first or the second robot. While traveling, all robots must avoid collisions with each other and a pair of obstacles with known location. The BNBF in Section 3.1.1 (i.e., h) captures these constraints.

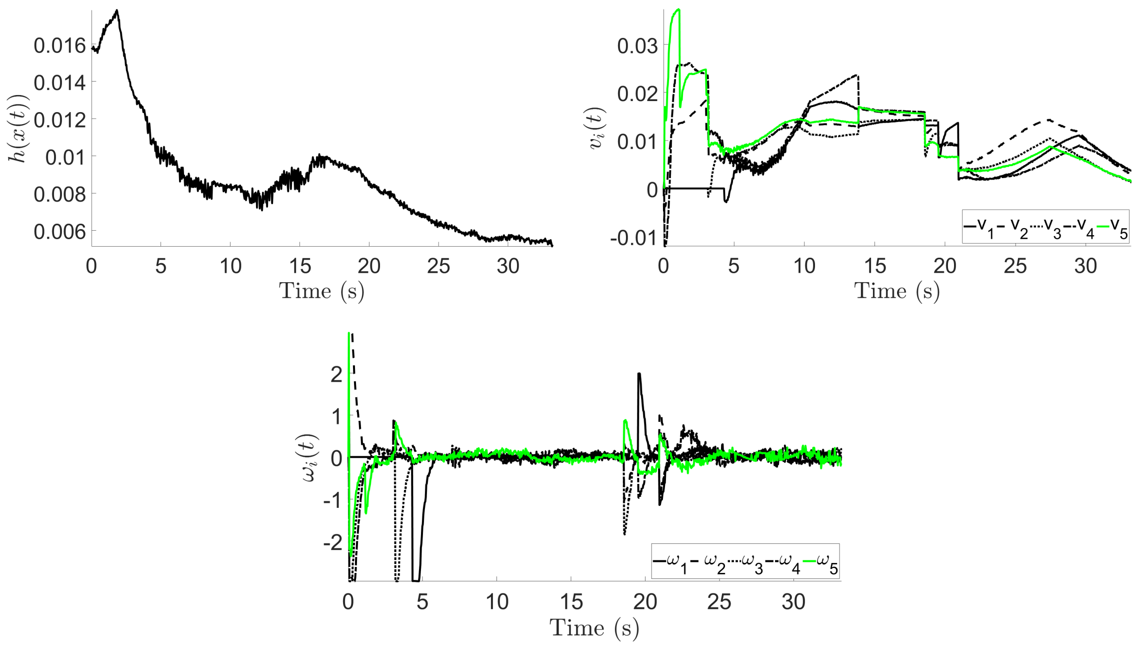

Figure 3.2: Numerical results from the team of mobile robots. The left image displays the value of the BNBF over the course of the experiment. Because the value of the BNBF is always positive, all constraints are satisfied. The middle and right images display the synthesized linear and angular velocities of the mobile robots. Though discontinuous, these control inputs ensure that the constraints are met and the objective is completed.

This experiment solves the QP indicated in Theorem 3.1. Since h is smoothly composed, calculating the almost-active gradient involves only the gradient of $h_{i,j}$, which is

$$\nabla_{x_i} h_{i,j}(x_i, x_j, \cdot) = 2(x_i - x_j) = -\nabla_{x_j} h_{i,j}(x_i, x_j, \cdot). \tag{3.7}$$

As required, these gradients are continuous, and the requisite QP, in the format of Theorem 3.1, is

$$u^*(x) \in \underset{u \in \mathbb{R}^{2N}}{\arg\min}\, u^\top u - 2u_{nom}^\top(x) u$$
$$\text{s.t. } \langle \partial_\epsilon h(x) \, , u \rangle \geq -\gamma h(x)^3,$$

where $\gamma > 0$ and $h(x) \to h(x)^3$ is the selected extended class-$\mathcal{K}$ function. In this case,

$\partial_\epsilon h(x)$ can be calculated by considering the active expressions and substituting an appropriate gradient in (3.7). Note that the above QP minimizes the objective function $\|u - u_{nom}\|^2$, ensuring that u^* respects the leaders' primary objective. The parameters for this experiment were chosen as

$$\delta_{con} = 0.35,\ \delta_{obs} = 0.1,\ \delta_{col} = 0.08,\ \gamma = 1000, \epsilon = 0.007.$$

For deployment, this experiment utilizes the Robotarium, a remotely accessible swarm robotics testbed [8]. The differential-drive robots utilized in the Robotarium have nonlinear unicycle dynamics, which are controlled by linear and angular velocity. However, the single-integrator model may be mapped onto such a system using a number of techniques, and this experiment employs the transformation in [27].

Figure 3.1 displays the resulting trajectories of the robots under the controller u^*. Due to the minimally invasive QP formulation and the results of Section 3.2.2, the team of robots complete the objective while respecting the desired constraints. In particular, Figure 3.2 indicates that the BNBF, h, remains positive over the course of the experiment, implying that the synthesized controller respects all of the constraints. Additionally, the leader robots successfully achieve their pre-specified goal positions.

Figure 3.2 displays the linear and angular velocities of the robots during the experiment. As expected, the control inputs are discontinuous. However, as predicted by the results of Section 3.2.2, the synthesized controller still ensures that the BNBF remains positive, meaning that all constraints are satisfied.

3.4 Conclusion

This chapter proposed and theoretically validated a framework for constraint composition and controller synthesis using barrier functions. Composition of these constraints was obtained through Boolean operators, and their application resulted in nonsmooth functions.

As such, this chapter presented CNBFs, which were formulated with respect to controlled systems. Accordingly, we developed an almost-active gradient for nonsmooth functions, and, when included as a constraint to a quadratic program, this object permitted the synthesis of discontinuous but valid controllers. Experimental results on a leader-follower team of mobile robots demonstrated the efficacy of these results. Building on these results, Chapter 4 extends CNBFs to a class of hybrid systems and Chapter 5 extends these results to general Boolean expressions of NBFs.

CHAPTER 4
HYBRID NONSMOOTH BARRIER FUNCTIONS

This chapter contains the results from the contributed publication [29], and this chapter extends the results of Chapter 2 and Chapter 3 to a class of hybrid systems. In particular, NBFs are extended to be time varying and have a countable number of jump-based discontinuities, resulting in Hybrid NBFs (HNBFs) and their analogous controlled versions. Moreover, this chapter applies the developed HNBF framework to a LIDAR-equipped differential drive robot. In particular, we present an algorithm that utilizes HNBFs to formulate a composable collision-avoidance algorithm in the local coordinates of the robot.

Prior approaches to set invariance in robotics include [30, 31, 32]; the use of barrier functions distinguishes this thesis from the existing literature. In addition to provable guarantees for forward set invariance, barrier functions offer a number of advantages, namely their implicit formulation for general dynamical systems and composability. The implicit nature of barrier functions relates to Lyapunov functions in that satisfying a particular inequality point-wise across the state space provides a global set-invariance result, meaning that barrier-function-based controller-synthesis frameworks do not necessarily require simulation of the system over an interval (e.g., as in model predictive control). The computational significance of this is demonstrated in [8], where such a framework is used to synthesize controllers for 80-dimensional ensemble systems of differential-drive robots at 100 Hz. Furthermore, barrier functions are composable, which means that arbitrary controllers, such as those generated by the methods of [31, 33], can be easily incorporated during barrier-function-based controller synthesis. For further advantages of barrier functions, see [1, 7, 34, 3, 12].

Since potential functions (e.g., [32]) represent a related method and are well used in the literature, this chapter explicitly compares this technique with barrier functions. In terms of similarities, both are predicated on guaranteeing the positivity of a function. However, potential functions are typically formulated for a particular task (i.e., navigation with obstacle avoidance) and are predicated on the system moving with the negative gradient of the potential function. Moreover, since navigation and obstacle avoidance are being combined into a single function, the potential function must be specially tuned to satisfy a series of requirements [32, Definition 1]. Conversely, barrier functions focus only on set invariance through weaker conditions (i.e., only a differential inequality), resulting in some advantages. First, barrier functions can be formulated independently from specific dynamical systems or applications. Second, barrier functions allow the system to approach the boundary of the invariant set. This property permits barrier functions to be combined with pre-existing controllers and makes them amenable to Boolean composition; this fact also implies that the usage of barrier functions does not preclude that of potential functions. That is, the pre-existing controller may come from the gradient of a potential function.

For example, consider a mobile robot that utilizes a combined approach: barrier functions for low-level collision avoidance and a potential function for navigation (via a nominal controller). Now, suppose an unexpected obstacle appears. Because the obstacle is unexpected, the less-strict requirements of barrier functions support immediate incorporation into the collision-avoidance framework. Then, the robot can avoid collisions while determining a modified potential function that incorporates the new obstacle. Using both approaches, the system can capture the flexibility of barrier functions and the guaranteed navigation of potential functions.

As for this chapters's contribution, Chapters 2 and 3 do not address time-varying NBFs, inhibiting their application in robotics scenarios where unexpected events and dynamic environments may induce changing objectives and constraints (e.g., an autonomous vehicle avoiding a previously unsensed obstacle). Accordingly, the main contribution of this chapter formulates Hybrid NBFs (HNBFs): time-varying NBFs with jumps. This chapter also addresses a practically useful class of controlled HNBFs and presents a quadratic-program-based controller-synthesis method with the aforementioned advantages. HNBFs are similar in their hybrid nature to the switching sequences presented in [35, 36] and much other work on hybrid systems. However, the analysis in this chapter differs by focusing on set invariance rather than stability.

Because of their recent renewed interest, the modern formulation of barrier functions has not been widely applied to classic robotics problems. Accordingly, to demonstrate a practical application of HNBFs in robotics, this thesis revisits an established problem: collision-avoidance for arbitrary primary objectives. Specifically, HNBFs are used to encode a collision-avoidance algorithm that composes with the primary controller of a LIDAR-equipped differential-drive robot.

This chapter is organized as follows. Section 4.1 contains background material, including tools from [14, 16, 15, 17, 13, 12]. Using these tools, the main theoretical results in Section 4.2 extend NBFs to HNBFs and applies HNBFs to controlled systems. For use in the subsequent experiment, a general measurement model and a composable collision-avoidance framework for control-affine systems are formulated. To demonstrate the proposed method, Section 4.3 instantiates this framework for a LIDAR-equipped differential-drive robot in local coordinates, and in Section 4.4, the robot uses the framework to achieve collision-free navigation. Section 4.5 concludes the chapter.

4.1 Background Material

The proposed HNBFs require tools from nonsmooth analysis and the theory of discontinuous dynamical systems, background material for which is contained in Appendix B and Appendx C. Because this chapter considers time-varying systems, this section presents the thoery of discontinuous dynamical systems and NBFs in the context of time-varying systems, which is not contained in Appendix B. Accordingly, this section introduces background material including: notation, system of interest, discontinuous dynamical systems, nonsmooth analysis, Boolean logic for barrier functions, and controller synthesis for NBFs.

4.1.1 System of Interest

Many robots can be modelled as control-affine systems (e.g., differential-drive robots, quadrotors) of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t), t), x(t_0) = x_0, t_0 \in \mathbb{R}, \tag{4.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^m$ are locally Lipschitz continuous; $u : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ is measurable and locally bounded in both arguments. In this case, u is not necessarily continuous, so these assumptions model discontinuities in the control input, which inevitably occur during controller synthesis in the context of this chapter. For additional information and definitions (e.g., locally bounded), see [17, p. 44, p. 49] or Appendix B.

For differential equations, such as (4.1), solutions may not exist. Toward analyzing discontinuous differential equations, applying Filippov's operator $K[f + gu] : \mathbb{R}^n \times \mathbb{R} \to 2^{\mathbb{R}^n}$ yields a particular system to which solutions exist. This operator maps discontinuous differential equations, as in (4.1), to a solution-bearing differential inclusion. At a point (x', t'),

$$\begin{aligned} &K[f + gu](x', t') = \\ &\text{co}\{\lim_{i\to\infty} f(x_i) + g(x_i)u(x_i, t') : x_i \to x', x_i \notin N_f, N\}, \end{aligned} \tag{4.2}$$

where N_f is a particular zero-Lebesgue-measure set and N is an arbitrary zero-Lebesgue-measure set [16, Theorem 1].

For a general differential inclusion,

$$\dot{x}(t) \in F(x(t), t),\ x(t_0) = x_0, t_0 \in \mathbb{R}, \tag{4.3}$$

on a set-valued map $F : \mathbb{R}^n \times \mathbb{R} \to 2^{\mathbb{R}^n}$, a Carathéodory solution is an absolutely continuous function $x : [t_0, t_1] \to \mathbb{R}^n$ such that $\dot{x}(t) \in F(x(t), t)$ almost everywhere on $[t_0, t_1] \ni t$ and $x(t_0) = x_0$. Such solutions exist if F is compact-, convex-valued, nonempty, and locally bounded; for each fixed t', the function $x' \mapsto F(x', t')$ is upper semi-continuous; and for each fixed x', the function $t' \mapsto F(x', t')$ is measurable [17, p. 44, p. 49]. In terms of completeness of solutions, this chapter requires that solutions exist but does not require that they exist for all time.

Combining these results, Filippov's operator in (4.2) automatically satisfies these properties. That is, Carathéodory solutions to the differential inclusion

$$\dot{x}(t) \in K[f + gu](x(t), t), x(t_0) = x_0, t_0 \in \mathbb{R}, \tag{4.4}$$

always exist. See [17] for a comprehensive coverage of discontinuous dynamical systems. Note that the assumptions for the time-varying case are similar to that of the time-invariant case, which is covered in Appendix B.

This chapter provides results for the general differential inclusion in (4.3) as they apply to the dynamical system in (4.1) via (4.2). In practice, the explicit computation of Filippov's operator is not required; instead, theoretical validation eliminates the need to calculate (4.4).

4.1.2 Nonsmooth Barrier Functions and Boolean Logic

This section presents the material of Chapter 2 in the context of time-varying systems. Most of the theory readily extends to this case; however, there are some notable differences in regard to the generalized gradient and Filippov's operator.

Barrier functions provably guarantee forward invariance of a set that captures domain-specific constraints (e.g., an autonomous vehicle staying in a lane or avoiding collisions). Denote $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ as the barrier function. The goal becomes to ensure forward invariance of the set

$$\mathcal{C} = \{(x', t') \in \mathbb{R}^n \times \mathbb{R}_{\geq t_0} : h(x', t') \geq 0\}.$$

It turns out that forward invariance can be guaranteed when

$$\dot{h}(x(t), t) \geq -\alpha(h(x(t), t), a.e.[t_0, t_1] \ni t, \tag{4.5}$$

for every Carathéodory solution, for some locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ (see Chapter 2).

However, if the barrier function is not continuously differentiable but is locally Lipschitz, as in this chapter, calculating $\dot{h}$ can no longer be done via the usual chain rule. Fortunately, the generalized gradient of [14] can be employed. At a point $(x', t') \in \mathbb{R}^n \times \mathbb{R}_{>t_0}$, the generalized gradient $\partial_c h : \mathbb{R}^n \times \mathbb{R}_{>t_0} \to 2^{\mathbb{R}^n \times \mathbb{R}}$ is defined as

$$\begin{aligned} &\partial_c h(x', t') = \\ &\text{co}\{\lim_{i \to \infty} \nabla h(x_i, t_i) : (x_i, t_i) \to (x', t'), (x_i, t_i) \notin \Omega, \Omega_h\}, \end{aligned} \tag{4.6}$$

where Ω_h designates the zero-measure set on which h is nondifferentiable, Ω is any zero-Lebesgue-measure set, and ∇h is the usual gradient [14, Theorem 2.5.1].

The generalized gradient in (4.6) is useful because it provides analysis of derivatives along Carathéodory solutions to a differential inclusion. Specifically, given a Carathéodory solution to (4.3), the time derivative of the function $t \mapsto h(x(t), t)$, satisfies

$$\dot{h}(x(t), t) \geq \min \partial_c h(x(t), t)^\top (F(x(t)) \times 1) \tag{4.7}$$

almost everywhere on $[t_0, t_1] \ni t$ [19, 15]. Importantly, the inequality in (4.7) circumvents the calculation of $\dot{h}(x(t), t)$, which is particularly useful for controller synthesis. Now, combining (4.5) with (4.7) yields the desired forward-invariance result in an analogous fashion to Chapter 2. Note that for sets $A, B \subset \mathbb{R}^n$,

$$A^\top B = \{a^\top b : a \in A, b \in B\},$$

such as in (4.7).

NBFs inherently capture Boolean composition. That is, barrier functions composed with $\wedge$, $\vee$, and $\neg$ operators. For example, an autonomous vehicle must stay in the designated lane and not violate the speed limit. The results of Chapter 2 and Chapter 3 show that $\max$, $\min$, and negation form a full system of Boolean logic with logical operators defined as

$$\begin{aligned} &h_1 \wedge h_2 := \min\{h_1, h_2\} \\ &h_1 \vee h_2 := \max\{h_1, h_2\} \\ &\neg h_1 := -h_1. \end{aligned} \tag{4.8}$$

Note that the logical operations are point-wise applied. For example,

$$(h_1 \wedge h_2)(\cdot) = \min\{h_1(\cdot), h_2(\cdot)\}.$$

This system of logic can generate complex constraints for robotic systems from basic component functions. Moreover, the generalized gradient becomes straightforward to estimate, and the corresponding controller-synthesis framework only requires knowledge of the component functions.

The generalized gradient of such an NBF, at $(x', t') \in \mathbb{R}^n \times \mathbb{R}_{>t_0}$, satisfies the inclusion

$$\partial_c h(x', t') \subset \operatorname{co} \bigcup_{i \in I(x', t')} \partial_c h_i(x', t'),$$

where $I(x', t') = \{i : h_i(x', t') = h(x', t')\}$ is the active-function map (see Proposition C.1. That is, the convex hull of the generalized gradients of the active component functions (i.e., those currently attaining the $\max$ or $\min$) encapsulates the generalized gradient of the NBF.

In Chapter 3, the NBF framework is extended to controller synthesis for Boolean compositions. In particular, Chapter 3 develops an almost-active gradient for specific NBFs, as in (4.8), that yields a validating controller when included as a constraint to a QP. Moreover, Filippov's operator on the controller does not have to be explicitly computed, and, despite the required analysis, the controller-synthesis framework can, effectively, be automatically applied.

In this case, the continuous differentiability of the component functions becomes a key property. The almost-active set for a so-called smoothly composed NBF, as in (4.8), is defined, at $(x', t') \in \mathbb{R}^n \times \mathbb{R}_{>t_0}$, as

$$I_\epsilon(x', t') = \{i : |h_i(x', t') - h(x', t')| \leq \epsilon\}.$$

Similarly, the almost-active generalized gradient is

$$\partial_\epsilon h(x', t') = \operatorname{co} \bigcup_{i \in I_\epsilon(x', t')} \partial_c h_i(x', t').$$

For such smoothly composed NBFs, [12] shows that including the almost-active generalized gradient as a constraint in an optimization program produces a potentially discontinuous but validating control input, assuming that the resulting controller is measurable and locally bounded [13, Theorem 3].

4.2 Main Results

This section contains the main results of this chapter: the formulation of HNBFs, Control HNBFs (CHNBFs), and piece-wise-constant CHNBFs, with Section 4.6 containing proofs of the theorems. The piece-wise-constant CHNBFs are particularly useful for applications with varying constraints. To demonstrate their applicability, this section also formulates a general-purpose collision-avoidance and controller-synthesis framework for the system in (4.1) in terms of a piece-wise-constant CHNBF that represents dynamically appearing obstacles. Despite the breadth of analysis required to prove the theorems, this framework essentially depends on solving a QP with distance measurements as constraints, and it is simple to apply, effective, and computationally feasible.

4.2.1 Hybrid Nonsmooth Barrier Functions

To account for the time-varying nature of the problem as well as potential jumps, the formulation of HNBFs first requires a modification to the concept of forward invariance, which is captured by the following two definitions.

Definition 6. *A sequence $\{\tau_k\}_{k=1}^{\infty}$ is a* switching sequence *for* (4.3) *if and only if it is strictly increasing, unbounded, and $\tau_1 = t_0$. With respect to $\{\tau_k\}_{k=1}^{\infty}$, let*

$$K_{t_1} = \inf\{K \in \mathbb{N} : [t_0, t_1] \subset \bigcup_{k=1}^{K} [\tau_k, \tau_{k+1}).$$

Definition 7. *A set $S \subset \mathbb{R}^n \times \mathbb{R}$ is* hybrid forward invariant *with respect to* (4.3) *and a switching sequence $\{\tau_k\}_{k=1}^{\infty}$ for* (4.3) *if and only if for every Carathéodory solution starting from x_0 at t_0,*

$$(x(\tau_k), \tau_k) \in S, \forall k \leq K_{t_1} \implies (x(t), t) \in S,$$

$\forall t \in [t_0, t_1]$.

Definition 7 captures the desired behavior of the system. Informally, if the system starts in the set and every jump remains within the set, then the system cannot leave the set. With regards to composable collision avoidance as presented in Section 4.2.2, this assumption implies that no obstacles instantaneously appear too close to the robot. The following two definitions construct candidate HNBFs as well as the conditions under which they are valid.

Definition 8. *A function* $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ *is a* candidate Hybrid Nonsmooth Barrier Function (HNBF) *for* (4.3) *if and only if there exists a switching sequence* $\{\tau_k\}_{k=1}^{\infty}$ *for* (4.3) *such that, for every* $k \in \mathbb{N}$, h *is locally Lipschitz continuous on* $\mathbb{R}^n \times [\tau_k, \tau_{k+1})$.

Remark 4.1. *In the context Definition 8, the generalized gradient of* h *is only defined on each open interval* (τ_k, τ_{k+1}), *as it requires that the function be defined on an open set; though, local Lipschitz continuity holds on* $[\tau_k, \tau_{k+1})$.

Definition 9. *A candidate HNBF* $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ *for* (4.3) *by the switching sequence* $\{\tau_k\}_{k=1}^{\infty}$ *is a* valid HNBF *for* (4.3) *if and only if the set*

$$\mathcal{C} = \{(x', t') \in \mathbb{R}^n \times \mathbb{R}_{\geq t_0} : h(x', t') \geq 0\}$$

is hybrid forward invariant via (4.3), $\{\tau_k\}_{k=1}^{\infty}$.

Definition 9 departs from the formulation of NBFs in Definition 1. Instead, Definition 9 only requires positivity of the function $t \mapsto h(x(t), t)$ and does not require $\mathcal{C}$ to be nonempty, which has been recognized to be superfluous since the forward-invariance statement is an implication. For practical purposes, one could modify Definition 8 to include the property that

$$\{x' \in \mathbb{R}^n : h(x', \tau_k) \geq 0\}$$

is nonempty for all $k \in \mathbb{N}$.

The robustness properties of asymptotic stability to C are precluded by the potential jumps in the system, and recovery of this property is left to future work. In this chapter, the set $\mathcal{C}$ still remains attractive; however, the system will not be able to reach C asymptotically due to potential jumps. Regardless, the following results utilize the above definitions to guarantee hybrid forward invariance.

Theorem 4.1. *Let* $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ *be a candidate HNBF for* (4.3) *under the switching sequence* $\{\tau_k\}_{k=1}^{\infty}$. *If for every* $k \in \mathbb{N}$, *there exists a locally Lipschitz extended class-*$\mathcal{K}$

function $\alpha^k : \mathbb{R} \to \mathbb{R}$ *such that*

$$\min \partial_c h(x', t')^\top (F(x', t') \times 1) \geq -\alpha^k(h(x', t')),$$

$\forall x' \in \mathbb{R}^n, t' \in (\tau_k, \tau_{k+1})$, *then* h *is a valid HNBF for* (4.3).

Following from Theorem 4.1, the next results concern a class of piece-wise constant HNBFs for controlled systems, and the subsequent section utilizes them to encode a series of collision-avoidance constraints that stem from range and bearing measurements. Toward this end, the following definition and result capture the notion of a CHNBF. By definition, valid CHNBFs for (4.1) are valid HNBFs for (4.4).

Definition 10. *A candidate HNBF* $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ *for* (4.1) *under the switching sequence* $\{\tau_k\}_{k=1}^{\infty}$ *is a valid* Control HNBF (CHNBF) *for* (4.1) *if and only if there exists a measurable and locally bounded function* $u : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ *such that for every* $k \in \mathbb{N}$ *there exists a locally Lipschitz extended class-*$\mathcal{K}$ *function* $\alpha^k : \mathbb{R} \to \mathbb{R}$ *satisfying*

$$\min \partial_c h(x', t')^\top (K[f + gu](x', t') \times 1) \geq -\alpha^k(h(x', t')),$$

$\forall x' \in \mathbb{R}^n$, $t' \in [\tau_k, \tau_{k+1})$.

Theorem 4.2. *Let* $h^k : \mathbb{R}^n \to \mathbb{R}$, $k \in \mathbb{N}$, *be a collection of valid CNBFs, in the sense of Definition 3, for* (4.1) *under the control laws and extended class-*$\mathcal{K}$ *functions,* $u^k : \mathbb{R}^n \to \mathbb{R}$, $\alpha^k : \mathbb{R} \to \mathbb{R}$, $k \in \mathbb{N}$*; and let* $\{\tau_k\}_{k=1}^{\infty}$ *be a switching sequence for* (4.1)*. Then, the function* $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ *defined as*

$$h(x', t') = h^k(x'), \forall k \in \mathbb{N}, \forall t' \in [\tau_k, \tau_{k+1}), \forall x' \in \mathbb{R}^n$$

is a candidate HNBF. Moreover, h *is a valid CHNBF for* (4.1) *with the input* $u : \mathbb{R}^n \times$

$\mathbb{R}_{\geq t_0} \to \mathbb{R}^m$ *defined as*

$$u(x', t') = u^k(x'), \forall k \in \mathbb{N}, \forall t' \in [\tau_k, \tau_{k+1}), \forall x' \in \mathbb{R}^n.$$

Remark 4.2. *In this theorem, the choice of the switching interval remains arbitrary. Moreover, the controller u in the proof is only defined on $\mathbb{R}^n \times \mathbb{R}_{\geq t_0}$; however, it can be extended to $\mathbb{R}^n \times \mathbb{R}$, without affecting measurability and local boundedness, by setting it to zero on the region $\mathbb{R}^n \times \mathbb{R}_{<t_0}$.*

In Theorem 4.2, the collection of CNBFs is not time varying, as is allowed for by Theorem 4.1; this choice is made so that Theorem 4.2 resembles the upcoming collision-avoidance algorithm, and such an extension (i.e., to make each $h^k : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$, $u^k : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}^m$) would be possible. Regardless, with these results, the next subsection revisits a classic problem, collision avoidance, where the collision avoidance algorithm is encoded via an HNBF as in Theorem 4.2.

4.2.2 Composable Collision Avoidance

Using the theoretical developments of Section 4.2.1, this section formulates a CHNBF for composable collision avoidance. Here, a scenario is modelled in which a sensor (e.g., LIDAR, infrared) or a map provides piece-wise-constant relative measurements to points that a robot must avoid. The system is assumed to evolve according to control-affine dynamics as in (4.1), and x' is assumed to be the position of the robot. Formally, the measurement model takes the form of a piece-wise-constant set-valued map $M : \mathbb{R}_{\geq t_0} \to 2^{\mathbb{R}^n}$ that returns a finite set of points over intervals indicated by a switching sequence $\{\tau_k\}_{k=1}^{\infty}$. That is,

$$M(t') = M(t''), \forall t', t'' \in [\tau_k, \tau_{k+1}).$$

As such, the shorthand

$$M(\tau_k) = M(t'), t' \in [\tau_k, \tau_{k+1}),$$

indicates the points over the interval $[\tau_k, \tau_{k+1})$. The set-valued map M takes a finite number of values in $\mathbb{R}^n$ with cardinality satisfying $|M(\tau_k)| = N_k$. The index $i \in N_k$ denotes a corresponding measurement $m_i \in M(\tau_k)$.

The value N_k represents the number of points to be avoided over the interval $[\tau_k, \tau_{k+1})$. For each of these points on the interval, the barrier function

$$h_i^k(x') = \|x' - m_i\|^2 - d^2,\ m_i \in M(t_k),\ i \in N_k, \tag{4.9}$$

encapsulates the constraint that the robot should remain at least a distance d away from the point m_i on the k^{th} interval. Then, the NBF given by

$$h^k = \bigwedge_{i=1}^{N_k} h_i^k$$

represents the constraint that all of these sampled points must be simultaneously avoided over a specific interval.

For controller-synthesis purposes, the almost-active gradient must be calculated. For this case, the almost-active gradient on an interval k at a point x' is given by

$$\partial_\epsilon h^k(x') = \mathrm{co} \bigcup_{i \in I_\epsilon^k(x')} \nabla h_i^k(x'),$$

where

$$\nabla h_i^k(x') = 2(x' - m_i).$$

Importantly, this procedure greatly prunes the number of constraints that must be included for controller synthesis, as only those measurements which are relatively close to the robot must be utilized.

Splicing the component functions together, the function $h : \mathbb{R}^n \times \mathbb{R}_{\geq t_0} \to \mathbb{R}$ defined as

$$h(x', t') = h^k(x'), \forall k \in \mathbb{N}, \forall t' \in [\tau_k, \tau_{k+1}), \forall x' \in \mathbb{R}^n, \tag{4.10}$$

Algorithm 1 Collision-Free Controller Synthesis

```
Input: Nominal controller: $u_k^{\text{nom}}(x(t))$
       Current measurements: $M(\tau_k)$
Output: Collision-free controller: $u(x(t),t)$.
  $h_i^k(x(t)) \leftarrow \|x(t)-m_i\|^2 - d^2, \forall m_i \in M(\tau_k)$
  $h^k(x(t)) \leftarrow \min_i\{h_i^k(x(t))\}$
  $I_\epsilon^k(x(t)) \leftarrow \emptyset$
  for $i = 1 : N_k$ do
    if $|h_i^k(x(t)) - h^k(x(t))| \le \epsilon$ then
      $I_\epsilon^k(x(t)) \cup \{i\}$
  $u(x(t),t) \leftarrow \arg\min_{u\in\mathbb{R}^m} \|u - u_k^{\text{nom}}(x(t))\|^2$
   s.t. $2(x(t)-m_i)^\top (f(x(t))+g(x(t))u) \ge -\alpha^k(h^k(x(t)))$
       $\forall i \in I_\epsilon^k(x(t))$
```

is a candidate HNBF by direct application of Theorem 4.2. To synthesize a controller for the HNBF, it suffices to consider the controller for each set of measurements, as per Theorem 4.2.

Since each of the individual gradients at each point in time is smooth, the results of Chapter 3 apply, and utilizing the almost-active gradient as a constraint to an optimization program yields a collision-avoiding controller. Let a nominal controller $u_k^{\text{nom}} : \mathbb{R}^n \to \mathbb{R}^m$ represent the primary objective, then the solution to the QP

$$\begin{aligned} u^k(x') &= \underset{u\in\mathbb{R}^m}{\arg\min}\, \|u - u_k^{\text{nom}}(x')\|^2 \\ &\text{s.t. } 2(x' - m_i)^\top(f(x') + g(x')u) \ge -\alpha^k(h^k(x')), \forall i \in I_\epsilon^k(x') \end{aligned} \tag{4.11}$$

is a validating controller on each interval, by the results in Theorem 3.1, assuming that the controller exists and is measurable and locally bounded. Specifically, each NBF h^k is a valid CNBF under u^k and α^k. Accordingly, the HNBF in (4.10) is a valid CHNBF for (4.1) by Theorem 4.2. Note that each α^k is, effectively, a design parameter. In practice, by Theorem 3.1, the almost-active gradient does not have to be included as a constraint directly. Rather, just the gradients of the component functions must be included, eliminating the convex-hull operation. In general, the QP in (4.11) is not feasible for every possible system. Rather, if the solution exists and is measurable and locally bounded, then Theorem 4.2 holds.

Algorithm 1 combines these results into a cohesive procedure for calculating collision-free controllers; a consequence of this formulation is that the synthesized controller tries to match the nominal controller but will preferably ensure collision avoidance. The QP may be solved through many software libraries such as the MATLAB optimization toolbox or the Python library CVXOPT. The complexity of Algorithm 1 depends on determining the ϵ-close points to the minimum (linear complexity) and solving a QP. Typically, strongly convex QPs, a class to which this algorithm belongs, have a runtime that is cubic in the number of decision variables. Assuming a standard conversion of inequality constraints to equality constraints, the runtime should be on the order of $O((m + N_k)^3)$ for most solvers. Finally, note that while the barrier functions, h_i^k, are hand designed, the composition is automatically computed by Algorithm 1.

4.3 Composable Collision-Avoidance for Differential-Drive Robots

Previous sections detailed the general methods and theoretical constraint-satisfaction guarantees afforded by the CHNBF controller-synthesis framework summarized in Algorithm 1. The purpose of this section is to formulate these methods for a particular choice of dynamics in order to achieve composable collision-avoidance for LIDAR-equipped differential-drive robots. Moreover, Algorithm 1 is encoded in the local coordinates of the robot, meaning that the robot does not need to estimate its global state; this formulation also makes the collision-avoidance algorithm independent from the particular objective at hand.

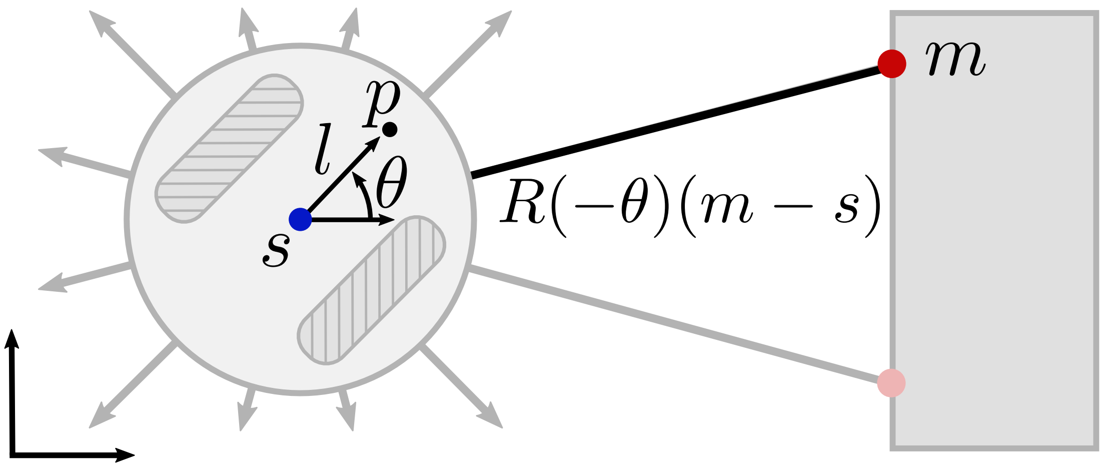

Figure 4.1: The differential-drive robot, modeled by unicycle dynamics as in (4.12), is equipped with a LIDAR. The point p is a distance l from s, the center of the robot and the LIDAR, which returns the vector $R(-\theta)(m-s)$.

4.3.1 CHNBF Controller Synthesis in Local Coordinates

The CHNBF controller-synthesis method in Algorithm 1[1] must be adapted to apply to LIDAR-equipped differential-drive robots, which are modelled by unicycle dynamics:

$$\dot{x} = v\cos(\theta) \qquad \dot{y} = v\sin(\theta) \qquad \dot{\theta} = \omega, \tag{4.12}$$

where $s = [x, y]^\top$ is the position; θ is the heading; and v, ω are the linear and angular velocity control inputs, respectively. The LIDAR is located at s.

Under unicycle dynamics, direct application of Algorithm 1 results in limited control authority because ω does not appear in the time derivative of the position of the robot. Thus, Algorithm 1 is instead formulated for a particular output of the state such that satisfying the collision-avoidance constraints for the output is sufficient to ensure collision-avoidance for the state; this output is chosen to be a point p orthogonal to the wheel axis of the robot as in Figure 4.1, which is given by the following expression: $p = s + lR(\theta)e_1$, where

[1]In the following sections, time dependence is suppressed for clarity.

$e_1 = [1, 0]^\top$ and $R(\theta)$ is the counterclockwise rotation matrix given by

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix};$$

note that $R(\theta) = R^\top(-\theta)$ and $R(-\theta) = R(\phi - \theta)R(-\phi)$.

Underpinning this choice of output is a near-identity diffeomorphism (NID), as presented in [37], which provides the following invertible mapping

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/l \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \dot{p} = LR(-\theta)\dot{p}.$$

Denote by r the radius of the robot. Now, observe that

$$\|p - m_i\| \leq \|s - m_i\| + l.$$

Then, picking $d \geq r + l$, ensures that $\|s - m_i\| \geq r$, when $\|p - m_i\| \geq d$. Considering p as the state variable, the barrier functions in (4.9) become

$$h_i^k(p) = \|p - m_i\|^2 - d^2.$$

Accordingly, it must be that

$$2(p - m_i)^\top \dot{p} \geq -\gamma(h_i^k(p))^3, \forall i \in I_\epsilon^k(p) \tag{4.13}$$

to satisfy Algorithm 1, where $h(p) \mapsto h(p)^3$ is the selected extended class-$\mathcal{K}$ function and

$\gamma > 0$. Applying identity transformations to (4.13) yields the following equations:

$$\begin{aligned}2(p-m_i)^\top \dot{p} &= 2(p-m_i)^\top R(\theta)R(-\theta)\dot{p}\\ &= 2\left(R(-\theta)(s-m_i)+le_1\right)^\top L^{-1}u;\\ \left(\|p-m_i\|^2-d^2\right)^3 &= \left(\|R(-\theta)(s-m_i)+le_1\|^2-d^2\right)^3.\end{aligned}$$

Thus, the constraint in (4.13) can be written in local coordinates, and yielding the QP

$$\begin{aligned}u^k(p) = &\operatorname*{arg\,min}_{u\in\mathbb{R}^2} \|G(u-u_k^{\text{nom}})\|^2 \\ \text{s.t. } &2\left(R(-\theta)(s-m_i)+le_1\right)^\top L^{-1}u\\ &\geq -\gamma\left(\|R(-\theta)(s-m_i)+le_1\|^2-d^2\right)^3, \forall i \in I_\epsilon^k(p),\end{aligned} \tag{4.14}$$

where G is a positive-definite weight matrix (in Section 4.4, $G = L^{-1}$) and u_k^{nom} is the nominal control input to the unicycle in local coordinates. The QP in (4.14) selects a collision-free control input and can be solved onboard the robot without global information because $R(-\theta)(s-m_i)$ is in the coordinate frame of the robot. Also, note that $R(-\theta)(s-m_i)$ is not the static LIDAR reading at time τ_k. However, denoting the position and heading of the robot at the time τ_k by s_{τ_k} and θ_{τ_k}, $R(-\theta)(s-m_i)$ can be written:

$$\begin{aligned}R(-\theta)(s-m_i) &= R(-\theta)(s-s_{\tau_k}+s_{\tau_k}-m_i)\\ &= R(-\theta)(s-s_{\tau_k})+R(-\theta)(s_{\tau_k}-m_i)\\ &= \underbrace{R(\theta_{\tau_k}-\theta)}_{\text{local change in }\theta}\Big(\underbrace{R(-\theta_{\tau_k})(s-s_{\tau_k})}_{\text{local change in }s}-\underbrace{R(-\theta_{\tau_k})(m_i-s_{\tau_k})}_{\text{LIDAR measurement}}\Big).\end{aligned} \tag{4.15}$$

Thus, $R(-\theta)(s-m_i)$ can be obtained using LIDAR measurements and local change in state (e.g., from encoders).

In summary, Algorithm 2 shows the local formulation of the CHNBF controller-synthesis

Algorithm 2 Local Collision-Free Controller Synthesis for LIDAR-Equipped Differential-Drive Robots

```
Input: Local nominal unicycle control input: $u_k^{\text{nom}}$
       Current LIDAR measurements: laser_scans
Output: Collision-free unicycle control input: $u^k$.
  for $i = 1 : N_k$ do
      $h_i^k \leftarrow \| - \text{laser\_scans}_i + le_1\|^2 - d^2$
      $\nabla h_i^k \leftarrow 2(-\text{laser\_scans}_i + le_1)$
  $h^k \leftarrow \min_i\{h_i^k\}$
  $I_\epsilon^k \leftarrow \emptyset$
  for $i = 1 : N_k$ do
      if $|h_i^k - h^k| \leq \epsilon$ then
          $I_\epsilon^k \cup \{i\}$
  $u^k \leftarrow \arg\min_{u \in \mathbb{R}^2} \|G(u - u_k^{\text{nom}})\|^2$
      s.t. $(\nabla h_i^k)^\top L^{-1} u \geq -\gamma(h_i^k)^3, \ \forall i \in I_\epsilon^k$
```

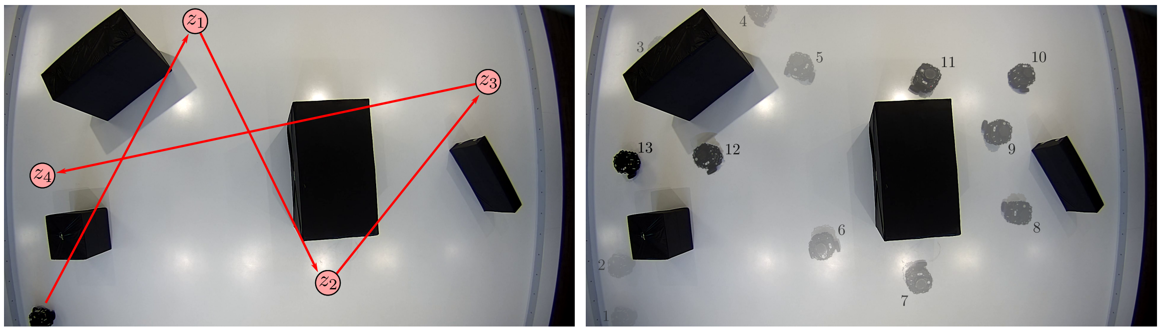

Figure 4.2: (Left) The primary objective of the robot is to sequentially visit four points in the obstacle field. (Right) Using the controller-synthesis framework in Algorithm 1 and Section 4.3, the robot navigates without collision; the numbers above the robot indicate its progress along the trajectory.

framework for LIDAR-equipped differential-drive robots[2], providing onboard composable, provably safe, and computationally straightforward collision avoidance. Because Algorithm 2 is decoupled from the nominal controller, it can be easily integrated into a ROS node, providing plug-and-play collision avoidance for arbitrary objectives.

[2] For simplicity, Algorithm 2 directly uses the LIDAR measurements under the assumption that the local changes in θ and s are negligible on the interval; the local state estimation in (4.15) can be used if this assumption is invalid.

4.3.2 Parameter Selection

The values of ϵ and γ are chosen experimentally using the following rationale. Increases in ϵ result in a larger almost-active set, generating more constraints and increasing the runtime of the QP; the increase in the runtime is tempered by the practical benefits to collision avoidance from accounting for an increased number of nearby measured points. The choice of γ is informed by its effect on $\gamma h(p)^3$: if γ is small, then $\gamma h(p)^3$ becomes small around $h(p) = 0$, preventing the robot from making progress toward the measured points; for larger γ, the robot approaches measured points more freely.

4.4 Experimental Results

The theoretical results in Section 4.2 and the application of Algorithm 2 to differential-drive robots in Section 4.3 are validated using a TurtleBot3, a LIDAR-equipped differential-drive robot; all computations are performed onboard the embedded computer of the robot using local information because the methods proposed in Section 4.2 and Section 4.3 are formulated to achieve collision avoidance independent of the primary objective. To demonstrate this point, a primary objective of sequentially visiting four points $Z = \{z_1, z_2, z_3, z_4\}$ is chosen, which is shown in Figure 4.2.

Here, the nominal controller representing this primary objective is a proportional controller that switches when the robot is within a threshold of a point, where switches concur with new measurements.

The local formulation of the CHNBF controller-synthesis method in Algorithm 2 was implemented on the embedded computer of the TurtleBot3 as a ROS node in Python. The update rate of the LIDAR is 200 ms; however, given the laser scanner measurements, Algorithm 2 runs in approximately 10 ms, suggesting that the possible performance of Algorithm 2 is approximately 100 Hz using the local estimation detailed in (4.15). In this implementation of Algorithm 2, raw laser scanner measurements, admissible because of the controlled environmental conditions of the laboratory setting, were used. Notably, due to the form of Algorithm 2, the navigation objective is independent of the ROS node, so any primary objective (i.e., nominal controller) could be integrated without modification of the ROS node.

In the experiment captured in Figure 4.2, the TurtleBot3 synthesizes a controller that minimally modifies the nominal controller while satisfying the CHNBF constraints to avoid collisions ($\epsilon = 0.2$, $\gamma = 100$, $G = L^{-1}$, and $d = 0.2$ m). The leftmost image of Figure 4.2 highlights the obvious collisions that would occur if the robot were to execute the nominal controller without accounting for obstacles. In the rightmost image of Figure 4.2, the trajectory of the robot is sampled, showing that by executing the synthesized controller, the TurtleBot3 successfully navigates the obstacle field. To quantify this success, Figure 4.3 shows that $\min_t(h(p(t),t)) \geq 0$ throughout the experiment, implying that the TurtleBot3 maintains a distance at least r from all measured obstacles.

4.5 Conclusion

This chapter presented a hybrid extension of nonsmooth barrier functions and provided an associated controller-synthesis framework. These objects can encapsulate safety constraints for robotic systems and, due to their hybrid nature, represent dynamic changes. To demonstrate the practical nature of the theoretical results, a LIDAR-equipped mobile

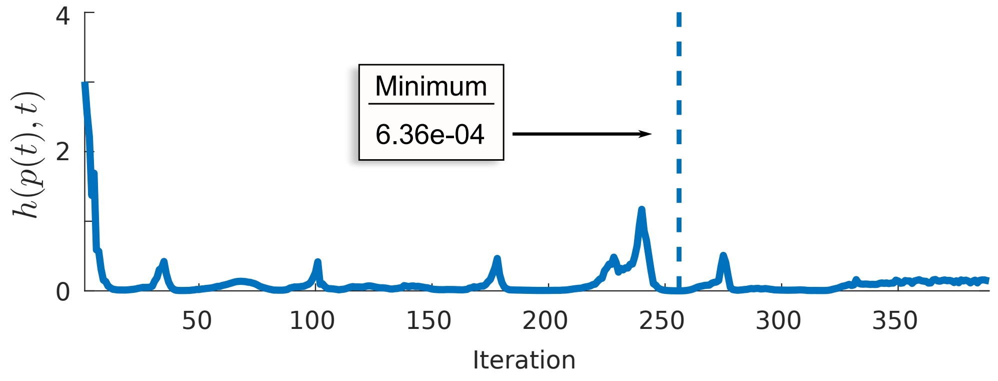

Figure 4.3: $\min_t(h(p(t),t)) > 0$ indicates that the robot maintains a distance greater than r from all measured obstacles.

robot utilized the controller-synthesis framework to generate a collision-free controller in a navigation scenario.

4.6 Proofs of Theorems 4.1 and 4.2

This section contains the proofs of Theorems 4.1 and 4.2.

Proof of Theorem 4.1

Let $x : [t_0, t_1] \to \mathbb{R}^n$ be a Carathéodory solution to (4.3) and assume that

$$(x(\tau_k), \tau_k) \in C, \forall k \leq K_{t_1}.$$

Let $t' \in [t_0, t_1]$. It remains to be shown that $(x(t'), t') \in C$. Since $\{\tau_k\}_{k=1}^{\infty}$ is a valid switching sequence, there exists a unique interval such that $t' \in [\tau_k, \tau_{k+1})$ for some k.

Consider the interval $[\tau_k, t']$. If $t' = \tau_k$, then $(x(t'), t') \in C$. Otherwise, $t' > \tau_k$. Since h is a candidate HNBF, it is locally Lipschitz on the interval $[\tau_k, t']$; thus, the function $[\tau_k, t'] \ni t \mapsto h(x(t), t)$ is absolutely continuous. Moreover, since h is a candidate HNBF, $\partial_c h$ exists on (τ_k, t'), and by assumption (4.7),

$$\begin{aligned} \dot{h}(x(t),t) &\geq \min \partial_c h(x(t), t)^{\top} (F(x(t), t) \times 1) \\ &\geq -\alpha^k(h(x(t), t)) \end{aligned}$$

almost everywhere on $(\tau_k, t') \ni t$, which is almost everywhere on $[\tau_k, t']$. Because $h(x(\tau_k), \tau_k) \geq 0$,

$$h(x(t), t) \geq 0, \forall t \in [\tau_k, t'],$$

by Lemma 2.1. Thus, $(x(t'), t') \in C$. □

Proof of Theorem 4.2

The proof proceeds by showing h is a candidate HNBF, writing $\partial_c h$ in terms of each $\partial_c h^k$, showing that u is measurable and locally bounded, and writing $K[f+gu]$ in terms of each $K[f+gu^k]$. Then, the desired result follows from the application of Theorem 4.1. In each of these five cases, notation for the switching sequence is reused.

To show that h is a candidate HNBF, consider the switching interval given by assumption, and let $k \in \mathbb{N}$. By assumption,

$$h(x',t') = h^k(x'), \forall t' \in [\tau_k, \tau_{k+1}).$$

Consequently, h is also locally Lipschitz on $\mathbb{R}^n \times [\tau_k, \tau_{k+1})$, since $t' \mapsto h(x',t')$ remains constant on this interval. Thus, h is a candidate HNBF.

Now, consider $\partial_c h(x',t')$ on the set $\mathbb{R}^n \times (\tau_k, \tau_{k+1}) \ni (x',t')$. Let Ω_{h^k} be the zero-measure set in $\mathbb{R}^n$ where h^k is nondifferentiable. Then, $\Omega = \Omega_{h^k} \times \mathbb{R}_{\geq t_0}$ is zero-measure in $\mathbb{R}^n \times \mathbb{R}_{\geq t_0}$. By (4.6),

$$\begin{aligned}&\partial_c h(x',t') = \\ &\mathrm{co}\{\lim_{i\to\infty} \nabla h(x_i,t_i) : (x_i,t_i) \to (x',t'), (x_i,t_i) \notin \Omega_h, \Omega\} = \mathrm{co} A.\end{aligned}$$

Take $z \in A$. By definition of h, $h(x_i,t_i) = h^k(x_i)$, so

$$\begin{aligned} z &= \lim_{i\to\infty} \nabla h(x_i,t_i) = \lim_{i\to\infty} \nabla h^k(x_i) = \lim_{i\to\infty} \left[\nabla_x h^k(x_i)^\top, 0\right]^\top \\ &= \left[\lim_{i\to\infty} \nabla_x h^k(x_i)^\top, 0\right]^\top \in \partial_c h^k(x') \times 0 \end{aligned}$$

This follows from (4.6), because $\partial_c h^k(x')$ can be taken as

$$\partial_c h^k(x') = \mathrm{co}\{\lim_{i\to\infty} \nabla_x h^k(x_i) : x_i \to x', x_i \notin \Omega_{h^k}\}.$$

Thus, $\partial_c h(x', t') \subset \partial_c h^k(x') \times 0$, because $\partial_c h^k(x') \times 0$ is convex and $A \subset \partial_c h^k(x') \times 0$ implies that $\operatorname{co} A \subset \partial_c h^k(x') \times 0$.

Now, consider the proposed control input u. To be a valid input, it must be measurable and locally bounded. To show that u is locally bounded, take $(x', t') \in \mathbb{R}^n \times \mathbb{R}_{\geq t_0}$. There exists a unique interval k such that $t' \in [\tau_k, \tau_{k+1})$. Since each u^k is locally bounded, there exists a $\delta_1 > 0$ such that $\|u^k(y)\| \leq M, \forall y \in B(x', \delta_1)$.

Because the switching sequence is strictly increasing, there exists a $\delta_2 > 0$ such that $[t', t' + \delta_2] \subset [\tau_k, \tau_{k+1})$. Accordingly, for every $y \in B(x', \delta_1)$, $s \in [t', t' + \delta_2]$ $\|u(y, s)\| = \|u^k(y)\| \leq M$, showing that u is locally bounded.

It remains to be shown that u is measurable. Let O be an open set in $\mathbb{R}^n$. Then, it must be shown that $u^{-1}(O) = \{(x', t') \in \mathbb{R}^n \times \mathbb{R}_{\geq t_0} : u(x', t') \in O\}$ is measurable. This set is equivalent to

$$\bigcup_{k=1}^{\infty} \{(x', t') \in \mathbb{R}^n \times [\tau_k, \tau_{k+1}) : u(x', t') \in O\}.$$

Examining any $k \in \mathbb{N}$ yields that

$$\begin{aligned} &\{(x', t') \in \mathbb{R}^n \times [\tau_k, \tau_{k+1}) : u(x', t') \in O\} \\ &= \{x' \in \mathbb{R}^n : u^k(x') \in O\} \times [\tau_k, \tau_{k+1}) \\ &= (u^k)^{-1}(O) \times [\tau_k, \tau_{k+1}), \end{aligned}$$

because $u^k(x') = u(x', t')$ on $\mathbb{R}^n \times [\tau_k, \tau_{k+1})$. Thus, if x' is such that $u^k(x') \in O$, then $u(x', t') \in O$ for every $t' \in [\tau_k, \tau_{k+1})$. Each u^k and every $[\tau_k, \tau_{k+1})$ is measurable, so the set $u^{-1}(O)$, is measurable.

Since u is measurable and locally bounded, consider $K[f + gu]$ on any interval $[\tau_k, \tau_{k+1})$. By definition, there exists a zero-measure set N_f^1 such that for any other zero-measure set

N^1,

$$\begin{aligned}&K[f+gu](x',t') = \\ &\mathrm{co}\{\lim_{i\to\infty} f(x_i)+g(x_i)u(x_i,t') : x_i \to x', x_i \notin N_f^1, N^1\},\end{aligned}$$

for every $x' \in \mathbb{R}^n$, $t' \in [\tau_k, \tau_{k+1})$. Now, consider $K[f+gu^k]$. By definition, there exists a zero-measure set N_f^2 such that for any other zero-measure set N^2

$$\begin{aligned}&K[f+gu^k](x') = \\ &\mathrm{co}\{\lim_{i\to\infty} f(x_i)+g(x_i)u^k(x_i) : x_i \to x', x_i \notin N_f^2, N^2\},\end{aligned}$$

for any $x' \in \mathbb{R}^n$.

In each of these definitions, set $N^1 = N_f^2$ and $N^2 = N_f^1$. Consequently, since $u(x',t') = u^k(x')$ for every $x' \in \mathbb{R}^n$, $t' \in [\tau_k, \tau_{k+1})$,

$$K[f+gu](x',t') = K[f+gu^k](x'),$$

$\forall x' \in \mathbb{R}^n$, $t' \in [\tau_k, \tau_{k+1})$.

It remains to be shown that h is a valid HNBF for (4.4) under the control input u. Take any $k \in \mathbb{N}$; by the above results, examining any $x' \in \mathbb{R}^n$, $t' \in (\tau_k, \tau_{k+1})$ yields

$$\begin{aligned}&\min \partial_c h(x',t')^\top \left(K[f+gu](x',t') \times 1\right) \\ &= \min \partial_c h^k(x')^\top K[f+gu^k](x') \\ &\geq -\alpha^k(h^k(x')) = -\alpha^k(h(x',t')),\end{aligned}$$

so h is a valid HNBF for (4.4) by application of Theorem 4.1. $\square$

CHAPTER 5
CONTROLLER SYNTHESIS FOR BOOLEAN SPECIFICATIONS

As noted in Chapter 1, robotic systems typically have a series of objectives and constraints that must be satisfied. Prior chapters have utilized barrier functions to encode constraints and utilized optimization programs to synthesize safe controllers with respect to some pre-existing nominal controller. However, this approach does not allow manipulation of or formal guarantees on the objectives. As such, this chapter uses barrier and Lyapunov functions to encode constraints and objectives, respectively. Specifically, this chapter extends the Boolean composition and controller-synthesis framework of Chapter 2 and Chapter 3 to include Nonsmooth Lyapunov Functions (NLFs) and to apply to general Boolean expressions.

As noted by previous chapters, barrier functions are not the only set-invariance method, and many other such methods exist, such as in [30, 31, 32, 33], to name a few. These approaches range from potential functions to PDE-based approaches to compute reachable sets. The work in [30] formulates a modular obstacle-avoidance framework for general dynamical systems, yet this framework is limited to this particular application of obstacle avoidance. The work in [33] relies on the solution of PDEs to guarantee set invariance, and in practice, these computations can be prohibitively costly. Finally, potential functions (e.g., [32]) offer a well-used approach, yet they are typically formulated with respect to a specific system or constraint (i.e., obstacle avoidance) and usually must be tuned for different objectives. As such, the system's objective and constraints may not be readily interchanged.

In general, three facts separate barrier functions from other methods in the context of this chapter (and prior chapters). The first is that barrier functions are provably correct, as the constraint satisfaction is based on forward set invariance. Second, barrier functions are mathematically agnostic in that they may be formulated independently from a particular system. Third, barrier functions are implicit. That is, point-wise satisfying a particular inequality across the state space produces a global invariance result. This quality allows barrier functions to be included in optimization programs for controller-synthesis purposes without requiring a look-ahead (e.g., as in model-predictive control), significantly reducing the computational burden.

From a complementary perspective, Lyapunov functions have long been used as a go-to method for stability, and the prior work is ubiquitous in the literature. This chapter uses set-stable NLFs to encode objectives, and the formulation is primarily based on [11, 15, 16, 9].

It is worth noting that prior work considers Boolean logic for real-valued functions [38, 39]. In fact, the authors of [38] formulate R-functions which admit a system of Boolean logic, and the Boolean logic framework in Chapters 2,3 falls into this category. One notable difference is that the literature of R-functions, as in [38], mostly focuses on smooth functions that capture conjunction and disjunction, rather than the nonsmooth $\max/\min$. Such smooth analogs are possible; however, they come at the expense of becoming significantly more difficult to differentiate. Because controller synthesis with respect to Boolean expressions inevitably involves taking derivatives, this quality complicates the synthesis process. Conversely, the approach of Chapters 2,3 becomes relatively straightforward from a synthesis perspective, at the expense of requiring nonsmooth analysis.

However, the results of Chapter 2 do not apply to general Boolean expressions, and Chapter 3 does not consider NLFs. This chapter resolves the issues posed by Chapters 2,3 by extending the formulated controller-synthesis framework to address inductive Boolean compositions of NBFs and NLFs. In regard to theory, this chapter utilizes the work in [40], which considers piece-wise-smooth (PC^r) functions, to generalize the main theorem of Chapter 3. This consideration results in two main contributions.

This chapter's first main result elucidates the calculation of discontinuous but validating controllers for NBFs and NLFs represented by PC^r functions. These results involve discontinuous dynamical systems and utilize nonsmooth analysis, as studied in [17, 14, 16, 15, 19]. Specifically, this result establishes a link between the PC^r functions of [40] with the generalized gradient of [14] to develop a new theory. To analyze PC^r functions, one typically analyzes a set of continuously differentiable function, and certain types of index sets capture the behavior of the generalized gradient. By extending these index functions to capture the locality of PC^r functions, this chapter derives a general method for synthesizing discontinuous controllers for PC^r functions that represent an NBF or a NLF. Moreover, this chapter shows that Boolean expressions on NBFs or NLFs fall into the class of PC^r functions and provides some preliminary results on Boolean composition of NLFs.

The second contribution of this chapter formulates a controller-synthesis framework that permits the combination NBFs and NLFs in an optimization program. The resulting potentially discontinuous controller guarantees that the objective is accomplished and the constraints are satisfied. This synthesis procedure relies on using an appropriate index set for the Boolean expressions, and this chapter provides a system-independent algorithm to calculate such index sets. To show the practicality of these results, a precision-agriculture-based experiment showcases that the algorithm may be applied in real time. In the considered scenario, a swarm of robots must visit a series of crop patches in a simulated farming environment while avoiding inter-robot collisions. Boolean NLFs and NBFs capture the objectives and constraints for the experiment, and the aforementioned algorithm produces an online controller that accomplishes this objective while preserving safety in the swarm.

The organization of this chapter follows. Section 5.1 notes background material required for this chapter including: the system of interest, NBFs, nonsmooth analysis, PC^r functions, and discontinuous dynamical systems. Section 5.2 begins the main contributions for this chapter, providing results on a class of index functions for PC^r functions and extending the results of [13]. Next, Section 5.3 formulates Boolean expressions of NBFs and NLFs and shows that they fall into a specific class of PC^r functions. Moreover, this section formulates an optimization-based controller-synthesis framework for NBFs and NLFs. Combining the prior two sections, Section 5.4 demonstrates some recursive methods for calculating appropriate index functions, applies these methods to Boolean expressions, and provides a straightforward algorithm for recursively calculating these index functions. To demonstrate the efficacy of the theoretical results, Section 5.5 showcases an experiment that uses the controller-synthesis framework of Section 5.3 and algorithm in Section 5.4 to produce a safe and effective controller in a precision-agriculture scenario.

5.1 Background Material

This section contains background material for this chapter, introducing the system of interest and some results on piece-wise differentiable (PC^r) functions. In particular, this chapter considers control-affine systems with potentially discontinuous control inputs and notes some results that unite PC^r functions with the generalized gradient.

5.1.1 System of Interest

This chapter considers control-affine systems of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)),\ x(0) = x_0, \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are continuous and $u : \mathbb{R}^n \to \mathbb{R}^m$ is measurable and locally bounded. Control-affine systems are amenable to controller synthesis via QPs, as will be displayed in later sections. An important point is that this chapter avoids the assumption that u is continuous, which becomes relevant for controller synthesis with respect to NBFs. That is, the synthesized controller may contain discontinuities, which is allowed for by the measurability and locally bounded assumptions.

Theoretically speaking, f and g may also be discontinuous, in the same sense as u; however, as this chapter pertains to controller synthesis, it assumes continuity of f and

g. Because u is potentially discontinuous, solutions to (5.1) may not exist. Fortunately, Filippov's operator (see Definition 19) maps (5.1) into a differential inclusion to which solutions exist.

Remark 5.1. *The map of limit points* $L : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ *defined such that*

$$K[f + gu](x') = \operatorname{co} L[f + gu](x')$$

becomes useful later in this chapter.

In particular, when considered as a differential inclusion, $K[f + gu] : \mathbb{R}^n \to 2^{\mathbb{R}n}$ satisfies the sufficient conditions for existence of solutions to a differential inclusion. A general differential inclusion is formulated as

$$\dot{x}(t) \in F(x(t)),\ x(0) = x_0, \tag{5.2}$$

where $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is a nonempty, compact-, convex-valued map that is upper semi-continuous (see Appendix B).

These conditions ensure that Carathéodory solutions to (5.2) exist. A Carathéodory solution is an absolutely continuous function $x : [0, t_1] \to \mathbb{R}^n$ such that

$$\dot{x}(t) \in F(x(t)), x(0) = x_0,$$

almost everywhere on $[0, t_1] \ni t$, and $x(0) = x_0$. In general, Carathéodory solutions to (5.2) exist, under these regularity conditions, but are not unique. See Appendix B for more discussion of discontinuous dynamical systems. For this chapter, an important fact is that Filippov's operator $K[f + gu]$ satisfies the aforementioned conditions. That is, the differential inclusion

$$\dot{x}(t) \in K[f + gu](x(t)), x(0) = x_0$$

has Carathéodory solutions. For extensive coverage of discontinuous dynamical systems, see [17].

If $h : \mathbb{R}^n \to \mathbb{R}$ is a valid NBF, then Definition 2 ensures that $h(x(t)) \geq 0, \forall t \in [0, t_1]$. Thus, $x_0 \in \mathcal{C}$ implies that $x(t) \in \mathcal{C}$ for all $t \in [0, t_1]$, for every Carathéodory solution, meaning that $\mathcal{C}$ is forward invariant. In this case, satisfying the differential inequality

$$\dot{h}(x(t)) \geq -\alpha(h(x(t))), a.e.\ t \in [0, t_1], \tag{5.3}$$

for some locally Lipschitz extended class-$\mathcal{K}$ $\alpha : \mathbb{R} \to \mathbb{R}$ and for every Carathéodory solution, ensures that h is a valid NBF (see Theorem 2.1). Toward controller synthesis, an algorithm must be able to implicitly verify (5.3) without explicitly calculating the time derivative of $h \circ x$ for every Carathéodory solution, yet the usual chain rule does not apply because h is nonsmooth.

The main advantage of the generalized gradient is that it permits analysis of nonsmooth functions along Carathéodory solutions. In particular, given a locally Lipschitz function $h : \mathbb{R}^n \to \mathbb{R}$ and a Carathédory solution $x : [0, t_1] \to \mathbb{R}^n$ to (5.2), the set-valued inner product satisfies the inequality

$$\dot{h}(x(t)) \geq \min \partial_c h(x(t))^\top F(x(t)) \tag{5.4}$$

almost everywhere on $[0, t_1] \ni t$. The strength of (5.4) is that this inequality may be used for controller-synthesis purposes, allowing the validity of the NBF to be verified spatially (i.e., over $\mathbb{R}^n$), rather than computing the time derivative explicitly.

A CNBF, as in Definition 3, automatically satisfies the requirements for a valid NBF via Theorem 2.1, but some difficulty arises in actually finding such a controller, as Filippov's operator must be applied to it. This situation creates issues for controller synthesis, because Filippov's operator, as in Definition 19, cannot feasibly be calculated in real time.

Moreover, for a controlled system,

$$\min \partial_c h(x')^\top (f(x') + g(x')u) \geq -\alpha(h(x'))$$

does not imply that

$$\min \partial_c h(x')^\top K[f + gu](x') \geq -\alpha(h(x')),$$

so the generalized gradient cannot be directly used for controller synthesis.

The work in Chapter 3 shows that, for certain well-behaved Boolean CNBFs, controller synthesis is possible by using an extended version of the generalized gradient called the almost-active gradient. Specifically, the composition of functions, such as $\max$ or $\min$, with a series of well-behaved component functions bore a significant role in the developments of the controller-synthesis framework in Chapter 3, and Chapter 3 shows that, for a particular class of Boolean CNBFs, controller synthesis was possible by including the almost-active gradient as a constraint. However, this object was limited, as it only applied to Boolean CNBFs consisting of exclusively $\wedge$ or $\vee$ operations. As such, this chapter generalizes this object to provide a controller-synthesis framework with respect to all Boolean CNBFs of interest. Fortunately, Boolean CNBFs with continuously differential component functions fall into a class of piece-wise differentiable functions. In particular, the work in [40] studies such functions and provides a number of helpful results.

5.1.2 Piece-wise-Differentiable Functions

As noted in Section 5.1.1, this chapter extends the results in Chapter 3 by formulating Boolean NBFs as piece-wise differentiable (PC^r) functions as in [40], and one of the main results of this paper, in Section 5.2, utilizes this theory to formulate controller synthesis for general Boolean expressions. Accordingly, PC^r functions represent an important class of nonsmooth functions, and this section discusses relevant background material. Formally, a PC^r function is defined as follows, where the terminology has been modified to fit this paper.

Definition 11 ([40, p. 91])**.** *A function $h : \mathbb{R}^n \to \mathbb{R}$ is PC^r if for every $x' \in \mathbb{R}^n$ there exists an open neighborhood N of x' and a finite set of C^r functions $\{h_i : i \in K_h\}$, with K_h a finite-cardinality set, such that h is a continuous selection of the h_i on this neighborhood. That is, h is a continuous function and $h(y) \in \{h_i(y) : i \in K_h\}$, $\forall y \in N$.*

Note that, in Definition 11, the finite set of component functions may vary based on the particular point in the domain, but this chapter exclusively considers PC^r functions whose component functions are defined over the entire domain. That is,

$$h(\cdot) \in \{h_i(\cdot) : i \in K_h\},$$

everywhere on $\mathbb{R}^n$, for some finite set of functions denoted by K_h. In the context of this paper, this assumption is not restrictive and does not inhibit the generality of the proposed results for controller synthesis. Throughout this chapter, we make the assumption that $r > 0$ (i.e., that all the component functions are at least C^1. Another important class of PC^r functions is piece-wise linear (PL) functions, which satisfy the following definition.

Definition 12. *A function $h : \mathbb{R}^n \to \mathbb{R}$ is Piece-Wise Linear (PL) if for every $x' \in \mathbb{R}^n$ there exists an open neighborhood N of x' and a finite set of linear functions $\{a_i^\top x' : i \in K_h\}$, $a_i \in \mathbb{R}^n$, such that h is a continuous selection of the a_i on N. That is, h is a continuous function and $h(y) \in \{a_i^\top y : i \in K_h\}$, $\forall y \in N$.*

Now, it remains to relate PC^r functions to the generalized gradient. In this case, the active index set, $I_h^a : \mathbb{R}^n \to 2^{K_h}$, for a PC^r function $h : \mathbb{R}^n \to \mathbb{R}$ defined as

$$I_h^a(x') = \{i \in K_h : h_i(x') = h(x')\}, \tag{5.5}$$

plays a role, much as in Proposition C.1. However, it may be that the active set captures irrelevant functions. As such, the essentially active set, $I_h^e : \mathbb{R}^n \to 2^{K_h}$, defined by

$$I_h^e(x') = \{i \in K_h : x' \in \text{cl}(\text{int}(\{y : h_i(y) = h(y)\}))\} \tag{5.6}$$

can be alternatively used. The essentially active index function, I_h^e, contains only functions that lie in the closure of the interior of the active set, ignoring lower-dimensional sets. Informally, a function must be active on a sequence that converges in the interior of the active set to be in I_h^e. In general, $I_h^e(\cdot) \subset I_h^a(\cdot)$ always. However, I_h^e may be more difficult to calculate, whereas $I_h^a(\cdot)$ remains straightforward to calculate in general, under the assumption that the component functions are known.

Tying these theories together, the following proposition relates the essentially active set I_h^e to the generalized gradient. Note that the terminology of the proposition has been modified to fit this chapter.

Proposition 5.1 ([40, Proposition 4.3.1])**.** *If U is an open subset of $\mathbb{R}^n$ and $h : U \subset \mathbb{R}^n \to \mathbb{R}$ is a PC^1-function with C^1 selection functions $h_i : N \to \mathbb{R}$, $i \in K_h$, at $x' \in N \subset U$, where N is a neighborhood of x', then*

$$\partial_c h(x') = \text{co}\{\nabla h_i(x') : i \in I_h^e(x')\}.$$

Proposition 5.1 unites the theory from PC^r functions and the generalized gradient, and this relationship is critical for the forthcoming results.

5.2 Extending the Almost-Active Set

This section begins the main results of this chapter, which relate to extending the work in Chapter 3. Specifically, the almost-active gradient, described in Chapter 3, is extended to PC^r functions through particular index sets. In turn, this development extends the controller-synthesis framework for Boolean CNBFs to general Boolean expressions.

5.2.1 Index Functions

As seen in Proposition 5.1, index functions unite PC^r functions with the generalized gradient. This section presents a class of index functions that generalize the results of Chapter 3 and are eventually useful for validating NBFs and NLFs in the presence of discontinuities in the control input. The following definition describes the index functions that capture the generalized gradient.

Definition 13. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function, and let $I_h : \mathbb{R}^n \to 2^{K_h}$ be an index function for h. Then, I_h is an encapsulating index function for h if and only if*

$$\partial_c h(x') \subset \text{co}\{\nabla h_i(x') : i \in I_h(x')\}$$

and

$$h_i(x') = h(x'), \forall i \in I_h(x'), \forall x' \in \mathbb{R}^n.$$

Remark 5.2. *For a given function, encapsulating index functions always exist via Proposition 5.1. Moreover, from the perspective of Proposition 5.1, the assumption that $h_i(\cdot) = h(\cdot)$ is nonrestrictive, because the other indices are superfluous.*

As these encapsulating index functions always exist, the goal becomes to find index functions that are relatively easy to calculate and that satisfy these definitions. in In fact, special care is taken in later sections to ensure that these index functions are readily calculated. Because this chapter studies these index functions extensively, the following notation

$$\{\nabla h_i(\cdot) : i \in I(\cdot)\} = \{\nabla h_i\}_{i \in I}(\cdot)$$

becomes useful for brevity in later results.

Recall the problem discussed in Section 5.1.1 in relation to synthesizing controllers via the generalized gradient. In this case, encapsulating index functions, as in Definition 13, still encounter this fundamental limitation. As such, it becomes necessary to further extend this object. Resolving this limitation, this section shows that capturing the locality of encapsulating index functions creates an object that is sufficient for controller-synthesis purposes. Toward this end, the following definition indicates these locality-capturing index functions.

Definition 14. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function, and let $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$ be an index function for h. Then, I_h^δ is a* δ-encapsulating index function *for h if and only if there exists an encapsulating index function for h such that for every $x' \in \mathbb{R}^n$ there exists a $\delta' > 0$ with*

$$I_h(x') \subset I_h^\delta(y), \forall y \in B(x', \delta').$$

In particular, a δ-encapsulating index function, as in Definition 14, maintains the indices of an encapsulating index function in a neighborhood of each point. Later, this section shows that this regularity is enough to circumvent troublesome values introduced by Filippov's operator.

An important point is that PC^r functions always possess an encapsulating index function via (5.6). Moreover, they also admit an associated δ-encapsulating index function. As with encapsulating index functions, the primary consideration for picking a specific δ-encapsulating index function depends on use-specific circumstances (e.g., I^e versus I^a).

Proposition 5.2. *If $h : \mathbb{R}^n \to \mathbb{R}$ is a PC^r function, then there exists a δ-encapsulating index function $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$ for h.*

Proof. Consider the index function $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$ defined by

$$I_h^\delta(x') = \{i \in K_h : |h_i(x') - h(x')| \leq \epsilon\}, \forall x' \in \mathbb{R}^n,$$

for some fixed $\epsilon > 0$ independent of x'. Let $I_h^e : \mathbb{R}^n \to 2^{K_h}$ be the essentially active index function for h as in (5.6).

Now, take $x' \in \mathbb{R}^n$ and assume that $i \in I^e(x')$. Since $i \in I_h^e(x')$, $h_i(x') = h(x')$. Moreover, by continuity of h_i and h, there exists a δ_1 and δ_2 such that

$$\|y - x'\| \leq \delta_1 \implies |h_i(y) - h_i(x')| \leq \epsilon/2, \forall y \in B(x', \delta_1)$$

and

$$\|y - x'\| \leq \delta_2 \implies |h(y) - h(x')| \leq \epsilon/2, \forall y \in B(x', \delta_2).$$

Let $\delta = \min\{\delta_1, \delta_2\}$. It remains to be shown that $i \in I_h^\delta(y)$ for all $y \in B(x', \delta)$. Let $y \in B(x', \delta)$. Then,

$$\begin{aligned} |h_i(y) - h(y)| &= |h_i(y) - h_i(x') + h(x') - h(y)| \\ &\leq |h_i(y) - h_i(x')| + |h(x') - h(y)| \\ &\leq \epsilon, \end{aligned}$$

implying that $i \in I_h^\delta(y)$. Thus, I_h^δ is a δ-encapsulating index function for h via I_h^e. □

Remark 5.3. *By the proof of this proposition, I_h^δ is also a δ-encapsulating index function for h with respect to the active index function I_h^a as in* (5.5).

5.2.2 Extending Prior Results

Now, the task remains to extend the results of [13] by using δ-encapsulating index functions. However, as this chapter also focuses on composition of objectives, the subsequent results are presented in enough generality to apply to the (forthcoming) results on NLFs as well. This next theorem represents the first main result of this chapter, demonstrating that δ-encapsulating index functions, as in Definition 14, are indeed the correct type of index function to navigate the issues caused by applying Filippov's operator.

Theorem 5.1. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function; let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a measurable and locally bounded function. If there exists a δ-encapsulating index function $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$*

for h and a continuous function $\alpha : \mathbb{R} \to \mathbb{R}$ such that

$$\min_{i \in I_h^\delta} \{\nabla h_i\}(x')^\top f(x') \geq -\alpha(h(x')), \forall x' \in \mathbb{R}^n,$$

then

$$\min \partial_c h(x')^\top K[f](x') \geq -\alpha(h(x')), \forall x' \in \mathbb{R}^n.$$

Proof. Let $x' \in \mathbb{R}^n$. Proceeding onward be means of Theorem 2.1, it remains to be shown that

$$\min \partial_c h(x')^\top K[f](x') \geq -\alpha(h(x')),$$

and by application of Lemma 2.2, it suffices to show that

$$\{\nabla h_i(x')\}_{I_h(x')}^\top L[f](x') \geq -\alpha(h(x')),$$

for any encapsulating index function I_h. As such, take $l \in L[f](x')$. Then, there exists $x_j \to x'$ such that $l = \lim_{j \to \infty} f(x_j)$.

Since I_h^δ is a δ-encapsulating index function for h, there exists an encapsulating index function $I_h : \mathbb{R}^n \to 2^{K_h}$ for h such that, for $x' \in \mathbb{R}^n$,

$$I_h(x') \subset I_h^\delta(y), \forall y \in B(x', \delta),$$

for some $\delta > 0$. Moreover, there exists an N such that for all $j \geq N$, $\|x_j - x'\| \leq \delta$. Thus, reusing notation and without loss of generality, consider only x_j such that $j \geq N$.

Now, take $i \in I_h(x')$. It remains to be shown that

$$\nabla h_i(x')^\top l \geq -\alpha(h(x')).$$

Accordingly,

$$\begin{aligned}
&\nabla h_i(x')^\top l - \alpha(h(x')) \\
&= \nabla h_i(x')^\top \lim_{j\to\infty} f(x_j) - \alpha(h(x')) \\
&= \lim_{j\to\infty} \nabla h_i(x_j)^\top \lim_{j\to\infty} f(x_j) - \lim_{j\to\infty} \alpha(h(x_j)) \\
&= \lim_{j\to\infty} \nabla h_i(x_j)^\top f(x_j) - \lim_{j\to\infty} \alpha(h(x_j)) \\
&= \lim_{j\to\infty} \left[\nabla h_i(x_j)^\top f(x_j) - \alpha(h(x_j))\right]
\end{aligned}$$

Moreover, $\|x_j - x'\| \leq \delta, \forall j$, meaning that $i \in I_h^\delta(x_j), \forall j$. Thus,

$$\nabla h_i(x_j)^\top f(x_j) - \alpha(h(x_j)) \geq 0, \forall j,$$

so

$$\lim_{j\to\infty} \left[\nabla h_i(x_j)^\top f(x_j) - \alpha(h(x_j))\right] \geq 0$$

as well, implying that

$$\partial_c h(x') K[f](x') \geq -\alpha(h(x'))$$

and yielding the desired result. □

Theorem 5.1 becomes particularly useful in the context of controller synthesis. Specifically, in a similar vein to the work of [13], including a δ-encapsulating index function as a constraint in an optimization program yields a validating but potentially discontinuous controller. Note that the proof of Proposition 5.3 is omitted, as it follows directly from Theorem 5.1.

Proposition 5.3. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a candidate NBF, and let f, g be as in* (5.1). *If there exists a δ-encapsulating index function $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$ for h; a locally Lipschitz* *extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$; and a measurable and locally bounded function $u : \mathbb{R}^m \to \mathbb{R}^n$ such that*

$$\min_{i \in I_h^\delta} \{\nabla h_i\}(x')^\top (f(x') + g(x')u(x')) \geq -\alpha(h(x')), \forall x' \in \mathbb{R}^n,$$

then h is a valid CNBF for (5.1).

5.3 Boolean Composition for Lyapunov and Barrier Functions

Having formulated the main result for δ-encapsulating index functions in the prior section, this section formulates the Boolean composition syntax and the class of Boolean expressions that this chapter considers. The first results pertain to formulating Nonsmooth Lyapunov Functions (NLFs) in the context of this chapter and some conditions under which they may be composed with Boolean operators. Then, the syntax and semantics of Boolean composition for NLFs and NBFs are discussed.

5.3.1 Control Nonsmooth Lyapunov Functions

NLFs have been studied in great detail, including in [15, 16, 11]. However, they have not been previously extended to the particular case of Boolean composition. There are many different formulations for Lyapunov functions; in this case, we use the following formulation, a combination of the definitions in [11, 15, 9], as it is amenable to Boolean composition, which two later results demonstrate.

Definition 15. *A locally Lipschitz function $V : \mathbb{R}^n \to \mathbb{R}$ is a* candidate Nonsmooth Lyapunov Function (NLF) *if and only if V satisfies $V(x') > 0, \forall x' \notin A$, where*

$$A = \{x' \in \mathbb{R}^n : V(x') = 0\};$$

A is nonempty; and

$$\{x' \in \mathbb{R}^n : V(x') \leq a\}$$

is a bounded set for every $a \in R_{\geq 0}$.

Remark 5.4. *For a candidate NLF V, as above, we always use A to denote the zero level set. Moreover, note that A is compact, because it is assumed to be bounded and is closed via continuity of V.*

Definition 16. *A candidate NLF $V : \mathbb{R}^n \to \mathbb{R}$ is a* valid Control Nonsmooth Lyapunov Function (CNLF) *for* (5.1) *if and only if there exists a continuous, positive-definite function $p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and a measurable and locally bounded function $u : \mathbb{R}^n \to \mathbb{R}^m$ such that*

$$\max \partial_c V(x')^\top K[f + gu](x') \leq -p(V(x')), \forall x' \in \mathbb{R}^n.$$

As shown by Definition 15 and Definition 16, this chapter focuses on set stability for NLFs (as in [9]), and the reasoning for this consideration follows. Boolean composition of NLFs entails intersections and unions of the zero level set. As such, restricting NLFs to a point eliminates the generality of Boolean composition, because union and intersection would only be able to generate NLFs for a point. this chapter omits the proof of Theorem 5.2 for brevity, as it is similar to many prior proofs of similar results; however, we do provide a discussion of the required tools.

Theorem 5.2. *Let $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ be a candidate NLF. If V is a valid CNLF for* (5.1)*, then the set A is uniformly globally asymptotically stable with respect to* (5.1).

Remark 5.5. *In the case that the boundedness of level sets in Definition 15 does not hold, then this definition may be modified to require an open set $D \subset \mathbb{R}^n$ containing A where the above inequality holds. In this case, a local stability result follows.*

Definition 15 and Definition 16 ensure uniform global asymptotic stability to the set A

in the sense that

$$\|x(t)\|_A \le \beta(\|x_0\|_A, t), \tag{5.7}$$

for every Carathéodory solution $x : [0, t_1] \to \mathbb{R}^n$, where

$$\|x'\|_A = \inf_{a \in A} \|x' - a\|$$

is the usual point-to-set distance and $\beta : \mathbb{R}_{\ge 0} \times \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ is a class-$\mathcal{KL}$ function.

This result of asymptotic stability follows from [11, 15, 10, 9], and this chapter does not reprove these stability results. However, we note that they mainly follow from [15, Theorem 2.2] and [11, Lemma 2.5]. In particular, one may use [11, Lemma 2.5] or [10, Lemma 4.3] to obtain class-$\mathcal{K}_\infty$ functions $\alpha_1, \alpha_2 : \mathbb{R}_{\ge 0} \to \mathbb{R}$ such that

$$\alpha_1(\|x\|_A) \le V(x) \le \alpha_2(\|x\|_A). \tag{5.8}$$

From Definition 16, every Carathéodory solution to $K[f](\cdot)$ satisfies

$$V(x(t)) \le \beta(V(x_0), t),$$

where $\beta : \mathbb{R}_{\ge 0} \times \mathbb{R}_{\ge 0} \to \mathbb{R}_{\ge 0}$ is a class-$\mathcal{KL}$ function [9, Lemma 4.2]. Then, applying (5.8) yields (5.7).

Now this chapter provides two results on Boolean composability for NLFs, where the proof for the second result is omitted for brevity. Note that, for NLFs, $\max$ represents an $\wedge$ operation whereas $\min$ represents an $\vee$ operation. Due to the requirements that the zero level set of V is bounded and V is positive, negation is not currently defined within this framework; and this chapter leaves this particular result to future efforts.

Proposition 5.4. *If V_1 and V_2 are candidate NLFs, as in Definition 15, and $A_1 \cap A_2 \neq \emptyset$,* *then* $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *defined as*

$$V(x') = \max\{V_1(x'), V_2(x')\}, \forall x' \in \mathbb{R}^n,$$

is a candidate NLF.

Proof. To show positive definiteness of V, consider $x' \in \mathbb{R}^n$. If $x' \in A$, then $x' \in A_1$ and $x' \in A_2$; thus,

$$V(x') = \max\{V_1(x'), V_2(x')\} = 0.$$

Conversely, if $x' \notin A$, then $x' \notin A_1$ or $x' \notin A_2$. In either case, $V_1(x') > 0$ or $V_2(x') > 0$, respectively. Thus, $V(x') > 0$ as well. Note that A is nonempty by assumption.

To show boundedness of level sets, let $a \in R_{\geq 0}$ and consider

$$L = \{x' \in \mathbb{R}^n : V(x') \leq a\}.$$

Since V is a maximum, $V(x') \leq a$ if and only if both $V_1(x') \leq a$ and $V_2(x') \leq a$. As such,

$$L = \{x' \in \mathbb{R}^n : V_1(x') \leq a\} \cap \{x' \in \mathbb{R}^n : V_2(x') \leq a\}.$$

Since V_1 and V_2 are candidate NLFs, L is the intersection of bounded sets, making it bounded as well. □

Proposition 5.5. *If* V_1 *and* V_2 *are candidate NLFs, then* $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *defined as*

$$V(\cdot) = \min\{V_1(\cdot), V_2(\cdot)\}$$

is a candidate NLF.

In a similar fashion to Proposition 5.3, Proposition 5.6 utilizes Theorem 5.1 to address controller synthesis for NLFs via δ-encapsulating index functions.

Proposition 5.6. *Let $V : \mathbb{R}^n \to \mathbb{R}$ be a candidate NLF, and let f, g be as in* (5.1). *If there exists a δ-encapsulating index function $I_V^\delta : \mathbb{R}^n \to 2^{K_V}$ for V; a positive-definite, continuous function $p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$; and a measurable and locally bounded function $u : \mathbb{R}^m \to \mathbb{R}^n$ such that*

$$\max_{i \in I_V^\delta} \{\nabla V_i\}(x')^\top (f(x') + g(x')u(x')) \leq -p(V(x')), \forall x' \in \mathbb{R}^n,$$

then V is a valid CNLF for (5.1).

5.3.2 Composition Syntax

This section details the Boolean expressions considered by the work. In particular, the given expressions are comprised of $\min$, $\max$, and $-$ operators. Formally, a Boolean expression on a finite number of C^r component NBFs is defined as

$$h = B[h_1, \ldots, h_k]. \tag{5.9}$$

For a composition of C^r candidate Lyapunov functions, the notation for a Boolean NLF becomes

$$V = B[V_1, \ldots, V_k]. \tag{5.10}$$

This chapter considers Boolean NBFs inductively defined with the following syntax.

$$B \coloneqq h_i \mid \neg B_1 \mid B_1 \wedge B_2 \mid B_1 \vee B_2, \tag{5.11}$$

where the logical operators are defined as

$$\neg B_1 = -B_1$$
$$B_1 \wedge B_2 = \min\{B_1, B_2\}$$
$$B_1 \vee B_2 = \max\{B_1, b_2\}.$$

The syntax for NLFs is given as follows.

$$B \coloneqq V_i \mid B_1 \wedge B_2 \mid B_1 \vee B_2,$$

where the logical operators are defined as

$$B_1 \wedge B_2 = \max\{B_1, B_2\}$$
$$B_1 \vee B_2 = \min\{B_1, B_2\}.$$

For NLFs, note that the above syntax omits the negation operator $\neg$ and that the roles of $\min$ and $\max$ are swapped (with respect to NBFs). For this chapter, we omit this operator due to the requirements of Definition 15.

For NLFs or NBFs, the component functions disambiguate the Boolean expression. For a Boolean composition B of NLFs or NBFs, the atomic component functions are denoted V_i or h_i, respectively. The notation B_i refers to the inductive component expressions, as in (5.11). The following example demonstrates this property.

Example 5.1. *An example of a Boolean NBF is as follows. Let $h : \mathbb{R}^n \to \mathbb{R}$ be point-wise defined as*

$$h(x') = \max\{\min\{h_1(x'), h_2(x')\}, h_3(x')\},$$

where each $h_i : \mathbb{R}^n \to \mathbb{R}$, $i \in [3]$, is locally Lipschitz. Then, h is a Boolean NBF. The above *expression is equivalent to*

$$h = (h_1 \wedge h_2) \vee h_3.$$

Set $B_1 = h_1$*,* $B_2 = h_2$*,* $B_3 = h_3$*. Now, define* $B_4 = B_1 \wedge B_2$*, which follows* (5.11)*. Finally, set*

$$h = B_4 \vee B_3.$$

In this example, the component function for h *are* h_1*,* h_2*, and* h_3*. For* h*, the component expressions are* B_4 *and* B_3*. For* B_4 *the component functions are* h_1 *and* h_2*; the component expressions are* B_1 *and* B_2

As expected, the following result shows that Boolean expressions are indeed PC^r. Note that the result may be applied to Boolean NBFs or NLFs, and the subsequent proposition demonstrates a controller-synthesis method utilizing δ-encapsulating index functions. The proof of Proposition 5.8 follows directly from application of Propositions 5.3,5.6 and is omitted for brevity.

Proposition 5.7. *If* $B[f_1, \dots, f_k] : \mathbb{R}^n \to \mathbb{R}$ *is a Boolean expression with* C^r *component functions* $f_i : \mathbb{R}^n \to \mathbb{R}$*,* $i \in [k]$*, then* $B[f_1, \dots, f_k]$ *is a* PC^r *function.*

Proof. Since $B[f_1, \dots, f_k]$ is a Boolean expression, it is an inductive composition of the f_i using $\min$, $\max$ or $-$ operators. As such, $B[f_1, \dots, f_k]$ is continuous, and at any $x' \in \mathbb{R}^n$, $B[f_1, \dots, f_k](x') = f_i(x')$ or $B[f_1, \dots, f_k](x') = -f_i(x')$, for some $i \in [k]$. As such,

$$\begin{aligned} &B[f_1, \dots, f_k](x') \\ &\in \{-f_i(x') : i \in [k]\} \cup \{f_i(x') : i \in [k]\}, \forall x' \in \mathbb{R}^n. \end{aligned}$$

Because each f_i and $-f_i$ is C^r, $B[f_1, \dots, f_k]$ is a continuous selection of C^r component functions, making it PC^r. □

Proposition 5.8. *Let* $h : \mathbb{R}^n \to \mathbb{R}$ *be a locally Lipschitz Boolean NBF and* $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ *be a candidate Boolean NLF. Let* $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$ *be a* δ*-encapsulating index function for* *h, and let $I_V^\delta : \mathbb{R}^n \to 2^{K_V}$ be a δ-encapsulating index function for V. Let $\alpha : \mathbb{R} \to \mathbb{R}$ be a locally Lipschitz extended class-$\mathcal{K}$ functions and $p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a continuous, positive-definite function. If $u^* : \mathbb{R}^n \to \mathbb{R}^m$ defined as*

$$
\begin{aligned}
u^*(x') \in \operatorname*{arg\,min}_{u \in \mathbb{R}m} & \; u^\top A(x') u + u^\top b(x') \\
\textit{s.t.} \; & \min_{i \in I_h^\delta} \{\nabla h_i\}(x')^\top (f(x') + g(x')u) \geq -\alpha(h(x')) \\
& \max_{i \in I_V^\delta} \{\nabla V_i\}(x')^\top (f(x') + g(x')u) \leq -p(V(x'))
\end{aligned}
$$

with $A : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ continuous, point-wise positive-definite, symmetric and $b : \mathbb{R}^n \to \mathbb{R}^m$ continuous, exists for every $x' \in \mathbb{R}^n$ and is measurable and locally bounded, then h is a valid CNBF and V is a valid CNLF.

In Proposition 5.8, the optimization program is a Quadratic Program (QP) due to the control-affine system. As such, assuming standard solvers, the runtime for solving such a program at a point is typically on the order of $O((|I_h^\delta(\cdot)| + |I_V^\delta(\cdot)| + m)^3)$ (i.e., cubic complexity in the number of constraints and decision variables). As such, it can typically solved in real time, even on computationally limited devices.

Since Boolean NLFs or NBFs fall into the class of PC^r functions, controller-synthesis algorithms for Boolean expressions focus on finding an appropriate δ-encapsulating index function. However, many such index functions exist for any given PC^r function, and not all such functions are, practically speaking, conducive to synthesis.

For example, given a Boolean NBF, h, Proposition 5.7 shows that h is PC^r with component functions h_i and $-h_i$ with $i \in [k]$ for some finite k. As such,

$$h(x') \in \{-h_i(x') : i \in [k]\} \cup \{h_i(x') : i \in [k]\}.$$

We temporarily use $-i$ to refer to $-h_i$. Accordingly, by Proposition 5.2, a suitable δ-

encapsulating index function for I_h^e is

$$\begin{aligned} I_h^\delta(\cdot) =& \{i \in [k] : |h_i(\cdot) - h(\cdot)| \leq \epsilon\} \\ & \cup \{-i \in [k] : |-h_i(\cdot) - h(\cdot)| \leq \epsilon\}. \end{aligned}$$

However, $I_h^\delta(\cdot)$ may capture too many component functions.

Consider the following example. Let $h = (h_1 \wedge h_2) \vee h_3$, where for a particular x', $h_1(x')$ is within ϵ of h_3 but $h_2(x')$ is much smaller than h_3. Then, $(h_1 \wedge h_2)(x') = \min\{h_1(x'), h_2(x')\}$ is also much smaller than h_3. Thus, intuitively, only 3 should be included in $I_h^\delta(x')$. However, using the previously noted I_h^δ, both 1 and 3 would be included in $I_h^\delta(x')$. As such, choosing I_h^δ in this manner may be too conservative. Accordingly, recursive methods to calculate an appropriate δ-encapsulating index function for h are the focus of Section 5.4.

5.4 Computing δ-Encapsulating Index Functions

Given the results in Sections 5.2-5.3, the goal becomes to calculate efficient and manageable δ-encapsulating index functions. As such, this section presents a method for recursively calculating a δ-encapsulating index function for an arbitrary composition of PC^r functions. Then, the method is specialized to the case of a Boolean NBF or NLF. To provide these results, composition of PC^r functions must first be considered.

5.4.1 Recursively Calculating δ-Encapsulating Index Functions

If $g : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$ are PC^r functions and $h = f \circ g$, then h is also a PC^r function. This result follows from the fact that h is a continuous selection of the functions $h_i = f_j \circ g_k$, where $i \in K_f \times K_g$. In this case, the component functions of h may be denoted by i or, equivalently, by the tuple (j, k) corresponding to $f_j \circ g_k$. Moreover, the gradients of each h_i at $x' \in \mathbb{R}^n$ are given by $\nabla g_k(x') \nabla f_j(g_k(x'))$.

One additional notational issue remains. In the case of Proposition 5.1, the PC^r function in question maps from $\mathbb{R}^n$ to $\mathbb{R}$; and, here, $g : \mathbb{R}^n \to \mathbb{R}^m$. As such, g is considered to be a continuous selection of PC^r functions $g^j_{i_j} : \mathbb{R}^n \to \mathbb{R}$ such that

$$g(x') \in \{ \left[g^1_{i_1}(x') \dots g^m_{i_m}(x') \right]^\top : i_j \in K_{g^j}, \forall j \in [m] \}.$$

For a given $g^j_{i_j}$, the subscript i_j represents a particular choice of a function for that component, and the superscript j represents the component of the vector-valued function g. The distinction is necessary because each component j could have a different set of selection functions. From this perspective, a particular vector-valued component function $g_i : \mathbb{R}^n \to \mathbb{R}^m$ may be viewed as $i \in K_{g_1} \times \ldots \times K_{g_m} = K_g$. As such, $g^j_{i_j}$ refers to a selection of the j^{th} component function g^j whereas g_i refers to a selection of a vector-valued component function for g.

Since h is a PC^r function, it always has an encapsulating index function; however, in the case that the component functions are known, it may be more desirable to compute this index function for h in terms of f and g. The next proof of the next lemma is omitted for brevity, and the following theorem provides one of the main results for this paper, demonstrating a recursive method for calculating encapsulating index functions.

Lemma 5.1. *Let $A_i \subset \mathbb{R}^n$, $i \in [m]$, and $B \subset \mathbb{R}^m$, then*

$$\operatorname{co}\left(\bigtimes_{i=1}^{m} \operatorname{co} A_i \right) (\operatorname{co} B) = \operatorname{co}\left(\bigtimes_{i=1}^{m} A_i \right) B$$

Theorem 5.3. *Let $g : \mathbb{R}^n \to \mathbb{R}^m$, $f : \mathbb{R}^m \to \mathbb{R}$ be PC^r functions, with $g = [g^1, \dots, g^m]^\top$, and let $h : \mathbb{R}^n \to \mathbb{R}$ be defined as $h = f \circ g$. If $I_f : \mathbb{R}^m \to 2^{K_f}$, $I_{g^k} : \mathbb{R}^n \to 2^{K_{g^k}}$ are encapsulating index functions for f and each g^k, $k \in [m]$, then $I_h : \mathbb{R}^n \to 2^{K_f \times K_g}$ defined as*

$$I_h(x') = I_f(g(x')) \times \bigtimes_{k=1}^{m} I_{g^k}(x'), \forall x' \in \mathbb{R}^n,$$

is an encapsulating index function for h.

Proof. Let I_f, I_{g^k}, $k \in [m]$, be encapsulating index functions for f and each g^k, $k \in [m]$; and let $I_h : \mathbb{R}^n \to 2^{K_f \times K_g}$ be defined as the point-wise Cartesian product

$$I_h(\cdot) = I_f(g(\cdot)) \times \bigtimes_{k=1}^{m} I_{g^k}(\cdot).$$

The above index function I_h is an acceptable index function for h, since

$$h(\cdot) \in \{f_i(g_j(\cdot)) : i \in K_f, j \in K_g\},$$

where by prior discussion, $K_g = \bigtimes_{k=1}^{m} K_{g^k}$.

It remains to be shown that I_h is an encapsulating index function for h. Let $x' \in \mathbb{R}^n$. Then,

$$\begin{aligned}
\partial_c h(x') &\subset \operatorname{co}\left(\bigtimes_{k=1}^{m} \partial_c g^k(x')\right) \partial_c f(g(x')) \\
&\subset \operatorname{co}\left(\bigtimes_{k=1}^{m} \operatorname*{co}_{j_k \in I_{g^k}} \{\nabla g^k_{j_k}\}(x')\right)\left(\operatorname*{co}_{i \in I_f}\{\nabla f_i\}(g(x'))\right) \\
&= \operatorname{co}\left(\operatorname{co}\bigtimes_{k=1}^{m} \underset{j_k \in I_{g^k}}{\{\nabla g^k_{j_k}\}}(x')\right)\left(\operatorname*{co}_{i \in I_f}\{\nabla f_i\}(g(x'))\right) \\
&= \operatorname{co}\left(\bigtimes_{k=1}^{m} \underset{j_k \in I_{g^k}}{\{\nabla g^k_{j_k}\}}(x')\right) \underset{i \in I_f}{\{\nabla f_i\}}(g(x')) \\
&= \operatorname{co}\{\left(\bigtimes_{k=1}^{m} \nabla g^k_{j_k}(x')\right) \nabla f_i(g(x')) : i \in I_f(g(x')), j_k \in I_{g^k}(x')\} \\
&= \operatorname{co}\{\nabla g_j(x') \nabla f_i(g_j(x')) : (i,j) \in I_f(g(x')) \times \bigtimes_{k=1}^{m} I_{g^k}(x')\},
\end{aligned}$$

since, for every $j \in \bigtimes_{k=1}^{m} I_{g^k}(x')$, $g_j(x') = g(x')$ by Definition 13. Thus, I_h is an encapsulating index function for h. $\square$

To separate the index sets in Theorem 5.3, the fact that $g(\cdot) = g_j(\cdot)$ must be utilized.

When formulating such a theorem for δ-encapsulating index functions, special assumptions must be made to ensure that this decomposition is possible: namely, that the outermost function in the composition is PL. The next theorem provides the main result in this chapter for calculating δ-encapsulating index functions.

Theorem 5.4. *Let $f : \mathbb{R}^m \to \mathbb{R}$ be a PL function, $g : \mathbb{R}^n \to \mathbb{R}^m$ be a PC^r function, with $g = [g^1, \ldots, g^m]^\top$. If $I_f^\delta : \mathbb{R}^m \to 2^{K_f}$, $I_{g^k}^\delta : \mathbb{R}^n \to 2^{K_g}$, $k \in [m]$, are δ-encapsulating index functions for f and each g^k, $k \in [m]$, then there exists a δ-encapsulating index function $I_h^\delta : \mathbb{R}^n \to 2^{K_f \times K_g}$ for h such that, for all $x' \in \mathbb{R}^n$,*

$$\underset{i \in I_h^\delta}{\{\nabla h_i\}}(x') = \left(\bigtimes_{k=1}^{m} \underset{j_k \in I_{g^k}^\delta}{\{\nabla g_{j_k}^k\}}(x') \right) \underset{i \in I_f^\delta}{\{a_i\}}(g(x')).$$

Proof. Let $x' \in \mathbb{R}^n$. Because I_f^δ is a δ-encapsulating index function for f, there exists an encapsulating index function $I_f : \mathbb{R}^m \to 2^{K_f}$ for f such that at $g(x')$ there exists $\delta_1 > 0$ satisfying

$$I_f(g(x')) \subset I_f^\delta(y), \forall y \in B(g(x'), \delta_1).$$

By continuity of g, let δ_2 be such that

$$y \in B(x', \delta_2) \implies \|g(y) - g(x')\| \leq \delta_1.$$

Similarly, let δ_3^k be such that

$$I_{g^k}(x') \subset I_{g^k}^\delta(y), \forall y \in B(x', \delta_3^k),$$

for each $k \in [m]$. Then, define $\delta_3 = \min_{k \in [m]} \delta_3^k$, and set

$$\delta = \min\{\delta_1, \delta_2, \delta_3\}.$$

By Theorem 5.3, $I_h : \mathbb{R}^n \to 2^{K_f \times K_g}$ defined as

$$I_h(\cdot) = I_f(g(\cdot)) \times I_g(\cdot)$$

is an encapsulating index function for h, where $K_g = \bigtimes_{k=1}^m K_{g^k}$ and $I_g(\cdot) = \bigtimes_{k=1}^m I_{g^k}(\cdot)$, and by the particular selection of δ, for every $y \in B(x', \delta)$,

$$I_f(g(x')) \subset I_f^\delta(g(y)), I_{g^k}(x') \subset I_{g^k}^\delta(y), \forall k \in [m].$$

As such,

$$\begin{aligned} I_h(x') &= I_f(g(x')) \times I_g(x') \\ &\subset I_f^\delta(g(y)) \times I_g^\delta(y), \forall y \in B(x', \delta), \end{aligned}$$

so $I_f^\delta \circ g \times I_g^\delta$ is a δ-encapsulating index function for I_h. Moreover,

$$\begin{aligned} &\{\nabla g_j(x') \nabla f_i(g_j(x'))) : (i,j) \in I_f^\delta(g(x')) \times I_g^\delta(x')\} \\ &= \{\nabla g_j(x') a_i : (i,j) \in I_f^\delta(g(x')) \times I_g^\delta(x')\} \\ &= \underset{j \in I_g^\delta}{\{\nabla g_j\}}(x') \underset{i \in I_f^\delta}{\{a_i\}}(g(x')) \\ &= \left(\bigtimes_{k=1}^m \underset{j_k \in I_{g^k}^\delta}{\{\nabla g_{j_k}^k\}}(x') \right) \underset{i \in I_f^\delta}{\{a_i\}}(g(x')), \end{aligned}$$

showing the desired relation. □

5.4.2 δ-Encapsulating Index Functions for Boolean Expressions

To apply Theorem 5.4 to Boolean composition, this section addresses the particular case of $\min$ and $\max$ operations. In fact, the next proposition shows that calculating a δ-encapsulating index set for a Boolean expression admits a convenient format. The subsequent proposition provides an result in the case of negation, and for brevity, the proof is omitted, as it follows directly from Proposition C.2.

Proposition 5.9. *Let $g : \mathbb{R}^n \to \mathbb{R}^m$ be a PC^r function, with each component $g^i : \mathbb{R}^n \to \mathbb{R}$, $i \in [m]$, and let $f : \mathbb{R}^m \to \mathbb{R}$ be a PL function with component functions e_i, $i \in [m]$, where each e_i is the i^{th} standard basis vector. Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function satisfying*

$$h(x') \in \{e_i^\top g(x') : i \in [m]\}, \forall x' \in \mathbb{R}^n.$$

If $I^\delta_{g^i}$, $i \in [m]$, are δ-encapsulating index functions for each g^i and I^δ_f is a δ-encapsulating index function for f. Then, there exists a δ-encapsulating index function $I^\delta_h : \mathbb{R}^n \to 2^{K_f \times K_g}$ for h such that

$$\underset{i \in I^\delta_h}{\{\nabla h_i\}}(x') = \left\{ \underset{j_i \in I^\delta_{g^i}}{\{\nabla g^i_{j_i}\}}(x') : i \in I^\delta_f(g(x')) \right\}.$$

Proof. Let $x' \in \mathbb{R}^n$, and let each $I^\delta_{g^j}$ and I^δ_f be as assumed.

Consequently, by application of Theorem 5.4, there exists a δ-encapsulating index function I^δ_h such that

$$\begin{aligned}\underset{i \in I^\delta_h}{\{\nabla h_i\}}(x') &= \left(\bigtimes_{k=1}^{m} \underset{j_k \in I^\delta_{g^k}}{\{\nabla g^k_{j_k}\}}(x') \right) \underset{i \in I^\delta_f}{\{e_i\}}(g(x')) \\ &= \left\{ \underset{j_i \in I^\delta_{g^i}}{\{\nabla g^i_{j_i}\}}(x') : i \in I^\delta_f(g(x')) \right\},\end{aligned}$$

which follows because multiplication by any e_i leaves only the i^{th} component function. □

Remark 5.6. *This proposition applies to Boolean expressions including $\vee$ and $\wedge$, as in* (5.11)*, because* max *or* min *may be written as a multiplication with a standard basis vector e_i. For example, the function* max *is a PL function, with component functions e_i, where e_i*

is the i^{th} standard basis vector. Because at any $x' \in \mathbb{R}^n$, $\max$ may be written as

$$\max_{i \in [m]} \{B_i(x')\} = e_i^\top \left[B_1(x'), \ldots, B_m(x')\right]^\top,$$

for some $i \in [m]$.

Proposition 5.10. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function. If $I_h^\delta : \mathbb{R}^n \to 2^{K_h}$ is a δ-encapsulating index function for h, then, for any scalar s, I_h^δ is a δ-encapsulating index function for $\bar{h} : \mathbb{R}^n \to \mathbb{R}$ defined as*

$$\bar{h}(x') = sh(x'), \forall x' \in \mathbb{R}^n.$$

Remark 5.7. *Proposition 5.10 may be used to calculate the negation of Boolean expressions B as in* (5.11) *by calculating a δ-encapsulating index function for the negated expression. That is, if $B = \neg B_1$, then $I_{B_1}^\delta$ is also a δ-encapsulating index function for B. However, each index $i \in I_{B_1}^\delta$ now refers to a negated component function. For convenience, the notation $-I_{B_1}^\delta$ refers to negated component functions but only for Boolean expressions.*

Using Proposition 5.10 and Theorem 5.4, Algorithm 3 calculates a δ-encapsulating index function for a Boolean composition as in (5.9) or (5.10). Importantly, Algorithm 3 does not change based on the particular Boolean expression or system under consideration. Note that, due to Remark 5.6, for Boolean expressions only, the index calculation can be significantly simplified. In particular, given a Boolean expression $B[f_1, \ldots, f_k]$ with component expressions $B_1[f_1, \ldots, f_k]$, $B_2[f_1, \ldots, f_k]$, B, B_1, and B_2 are PC^r functions which all have component functions f_i, $i \in [k]$. As such, the intermediate indices do not have to be preserved when calculating a δ-encapsulating index function for B, because all of the Boolean expressions have the same component functions. Another noteworthy point is that Algorithm 3 requires a δ-encapsulating index function for $\wedge$/$\vee$ functions. For instance, the proposed δ-encapsulating index function from Proposition 5.2 is such a function. The following explicitly shows this calculation.

Example 5.2. *This example demonstrates a calculation of $I^{\delta}_{\wedge/\vee}(B(x'))$ as in Algorithm 3. Let $B : \mathbb{R}^n \to \mathbb{R}$ be a Boolean expression with component expressions $B_i : \mathbb{R}^n \to \mathbb{R}$, $i \in [k]$, be defined as*

$$B = \bigwedge_{i=1}^{k} B_i = \min_{i \in [k]} B_i$$

or

$$B = \bigvee_{i=1}^{k} B_i = \max_{i \in [k]} B_i$$

Then, with a slight abuse of notation, a δ-encapsulating index function $I^{\delta}_{\wedge/\vee} : \mathbb{R}^n \to 2^{K_B}$ for the $\wedge$ or $\vee$ operation, according to Proposition 5.2, evaluated at $B(x')$ is

$$I^{\delta}_{\wedge/\vee}(B(x')) = \{i \in [k] : \|B_i(x') - B(x')\| \leq \epsilon\}, \forall x' \in \mathbb{R}^n,$$

for any fixed $\epsilon > 0$.

Algorithm 3 δ_ENCAPSULATING

```
Input: Boolean expression: $B[f_1, \dots, f_k]$
        $\delta$-encapsulating index function for $\wedge/\vee$: $I^{\delta}_{\wedge/\vee}$
        Argument: $x' \in \mathbb{R}^n$
Output: Evaluated $\delta$-encapsulating index function for $B$: $I^{\delta}_B(x')$
  $I^{\delta}_B(x') \leftarrow \emptyset$
  if $B$ is $f_i$, for $i \in [k]$ then
      $I^{\delta}_B(x') \cup \{i\}$
      return $I^{\delta}_B(x')$
  if $B$ is $\neg B_1$ then
      $I^{\delta}_B(x') \cup -\delta$_ENCAPSULATING$(B_1, x')$
      return $I^{\delta}_B(x')$
  for $i \in I^{\delta}_{\wedge/\vee}(B(x'))$ do
      $I^{\delta}_B \cup \delta$_ENCAPSULATING$(B_i, x')$
  return $I^{\delta}_B(x')$
```

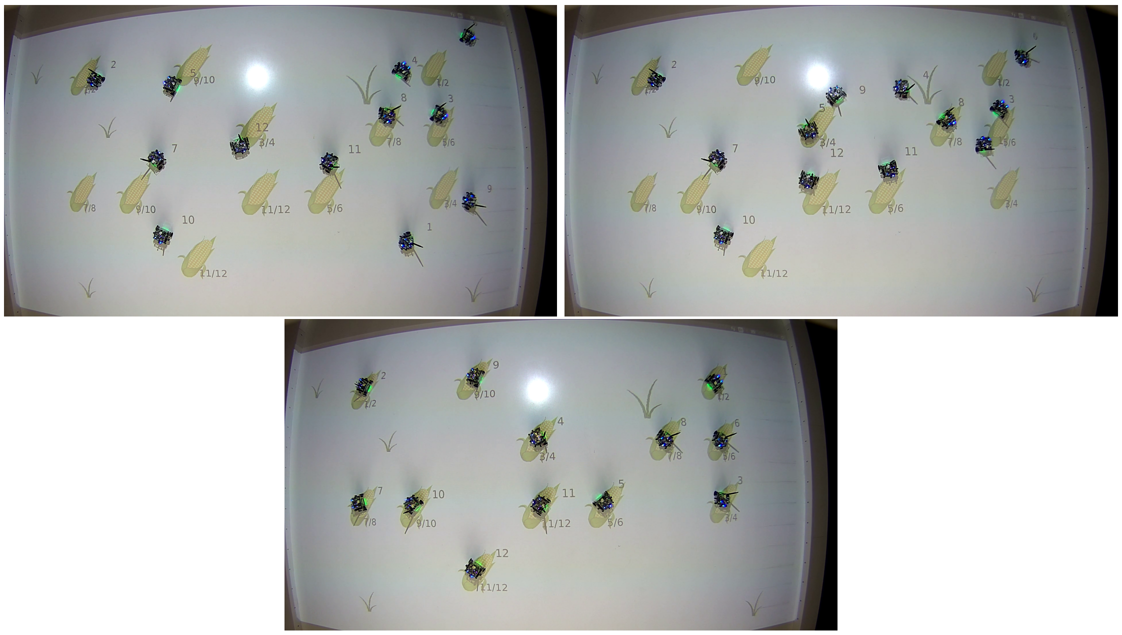
Figure 5.1: This figure shows the completion of the precision-agriculture experiment described in Section 5.5. Each pair of robots (e.g., 1 and 2) must visit a pair of crop patches, which are labeled accordingly, while avoiding collisions. This figure shows that the robots successfully visit each crop patch and avoid collisions, completing the objectives and satisfying the constraints.

5.5 Experimental Results

This section contains the experimental results of the paper. In particular, the experiment imitates a precision-agriculture scenario wherein a team of robots must visit a series of crop patches in a field while avoiding collisions with neighboring agents. The objectives and constraints are encoded using the methods discussed in Section 5.3, and the optimization program noted in Section 5.3 synthesizes a controller that satisfies the objectives and the constraints, where the δ-encapsulating index function is provided via Algorithm 3. In this section, an explicit dependence on time has been dropped for clarity.

5.5.1 Experiment Formulation and Results

The formulation of this experiment is as follows. Consider an even number N of differential-drive robots in $\mathbb{R}^2$, which represents the field. For simplicity, assume that the robots have

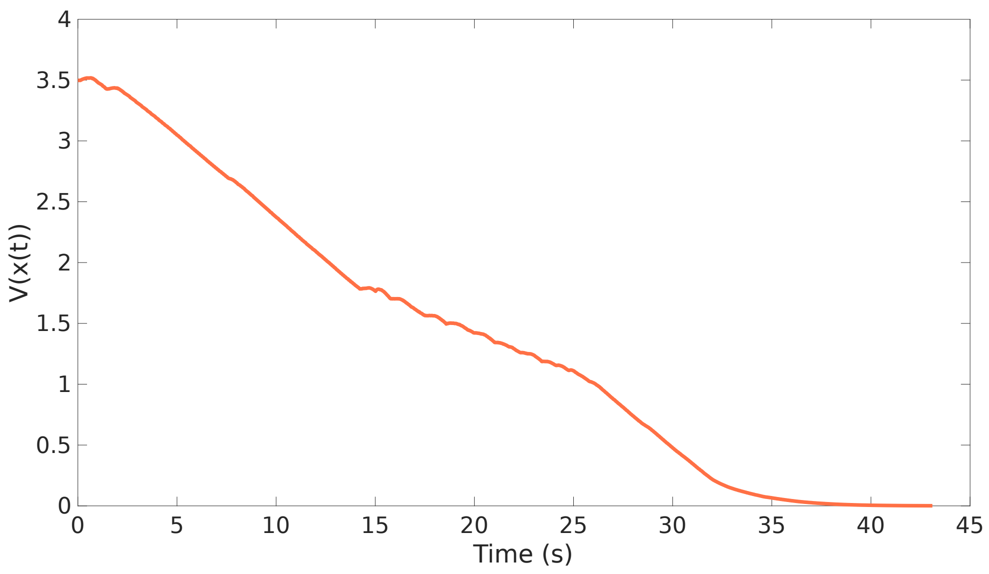

Figure 5.2: Value of objective-encoding NLF in Section 5.5. The value of the NLF goes to zero as time increases, indicating that the objective has been completed for all robots. That is, all crop patches have been visited.

state $x_i \in \mathbb{R}^2$, $i \in [N]$, and dynamics $\dot{x}_i = u_i$. The ensemble state and input is written as $x \in \mathbb{R}^{2N}$, $u \in \mathbb{R}^{2N}$. Later, this chapter utilizes the method in [27] to map the single-integrator input onto the full nonlinear differential-drive dynamics. In this section, the time dependence has been dropped for brevity.

This experiment requires that all robots avoid collisions, and this constraint may be encoded via the C^1 pair-wise collision-avoidance constraint

$$h_{ij}(x_i, x_j) = \|x_i - x_j\|^2 - d^2,$$

where $d > 0$ indicates the diameter of the robot. As such, the ensemble collision-avoidance constraint is given by the Boolean NBF

$$h = \bigwedge_{i=1}^{N} \bigwedge_{j=1+1}^{N} h_{ij},$$

where the large $\wedge$ symbol represents conjunction. Note that

$$\nabla_{x_i} h_{ij}(x_i, x_j) = 2(x_i - x_j),\ \nabla_{x_j} h_{ij}(x_i, x_j) = 2(x_j - x_i),$$

and $\nabla_x h_{ij}$ may be calculated by substituting $\nabla_{x_i} h_{ij}$ and $\nabla_{x_j} h_{ij}$ into the i^{th} and j^{th} indices.

The objectives for the robots are as follows. A pre-existing planner has determined that robots i and $i+1$ must visit crop patches $p_i \in \mathbb{R}^2$ and $p_{i+1} \in \mathbb{R}^2$, for $i = 1, 3, \ldots, N-1$, where the specific robot-to-patch assignment is unspecified for each pair. As such, this specification holds for each consecutive pair of robots. For example, robots 1 and 2 must visit patches p_1 and p_2 while robots 3 and 4 must visit p_3 and p_4.

Now, we formulate the corresponding objectives. Note that the parameterized function

$$V_{i,p}(x_i) = (x_i - p)^\top (x_i - p),$$

yields a candidate NLF for robot i for the patch $p \in \mathbb{R}^2$. Then,

$$\nabla_{x_i} V_{i,p}(x_i) = 2(x_i - p),$$

and $\nabla_x V_{i,p}$ may be calculated by substituting $\nabla_{x_i} V_{i,p}$ into the i^{th} component. For robots i and $i+1$ the objective that the robots visit p_i and p_{i+1} may be captured as

$$(V_{i,p_i} \wedge V_{i+1,p_{i+1}}) \vee (V_{i,p_{i+1}} \wedge V_{i+1,p_i}). \tag{5.12}$$

The above expression captures the specification that the robots must be at both points, but the order does not matter. The above expression is a candidate NLF; however, Proposition 5.5 cannot be directly applied as V_{i,p_i} is not a candidate NLF for the subsystem containing x_i and x_{i+1} (V_{i,p_i} does not have bounded level sets with respect to x_{i+1}). That is,

$$\{(x_i, x_{i+1}) : V_{i,p_i}(x_i) \leq a\} = A_{i,p_i} \times \mathbb{R}^2$$

is unbounded. However, the conjunction operation resolves this issue. Thus,

$$(V_{i,p_i} \wedge V_{i+1,p_{i+1}}) \tag{5.13}$$

is a candidate NLF, and by Proposition 5.5, (5.12) is a candidate NLF.

As such, the overall objective for the system may be encoded as the candidate NLF

$$V = \bigwedge_{i=1}^{N/2} (V_{2i-1,p_{2i-1}} \wedge V_{2i,p_{2i}}) \vee (V_{2i-1,p_{2i}} \wedge V_{2i,p_{2i-1}}),$$

and for the same reason as (5.13), V is also a candidate NLF.

Now that the system's constraint and objective has been formulated, the δ-encapsulating index functions for use with Algorithm 3 must be provided. This experiment utilizes the index function discussed in Proposition 5.2. That is,

$$I^{\delta}_{\wedge/\vee} = \{i \in K_B : |B_i(\cdot) - B(\cdot)| \leq \epsilon\},$$

for some fixed $\epsilon > 0$. The function $I^{\delta}_{\wedge/\vee}$ is used as a δ-encapsulating for every Boolean expression in this experiment.

In the spirit of Proposition 5.8, the QP for this experiment is given by

$$\begin{aligned} & u^*(x) \in \operatorname*{arg\,min}_{u \in \mathbb{R}m} u^\top u \\ & \text{s.t. } \nabla h_i(x)^\top (f(x) + g(x)u) \geq -\gamma h(x)^3, \forall i \in I^{\delta}_h(x) \\ & \nabla V_i(x)^\top (f(x) + g(x)u) \leq -\min\{c, V(x)\}, \forall i \in I^{\delta}_V(x), \end{aligned} \tag{5.14}$$

where $\gamma, c > 0$. Note that $V(x) \mapsto \min\{c, V(x)\}$ is the selected positive-definite function, as in Definition 16, and $h(x) \mapsto \gamma h(x)^3$ is the selected extended class-$\mathcal{K}$ function, as in Definition 3.

For the experiment, we utilize Algorithm 3 to calculate a δ-encapsulating index set for h and V. Then, combining h and V with these index functions into a QP, as in Proposition 5.8 yields a controller that ensures the robots visit the required locations and avoid collisions. For this experiment, we assume that the remaining assumptions of Proposition 5.8 hold. Namely, that $u^\star$ exists and is measurable and locally bounded. At each point, applications of Algorithm 3 calculate I_h^δ and I_V^δ, and MATLAB's optimization toolbox is utilized to solve the QP.

It is by no means guaranteed that, for any objective and constraint, h and V may be simultaneously solved in the QP. In this case, the compatibility is shown experimentally. In general, some techniques exist to ensure that this solution exists, such as the inclusion of slack variables (e.g., as in [1, 7]).

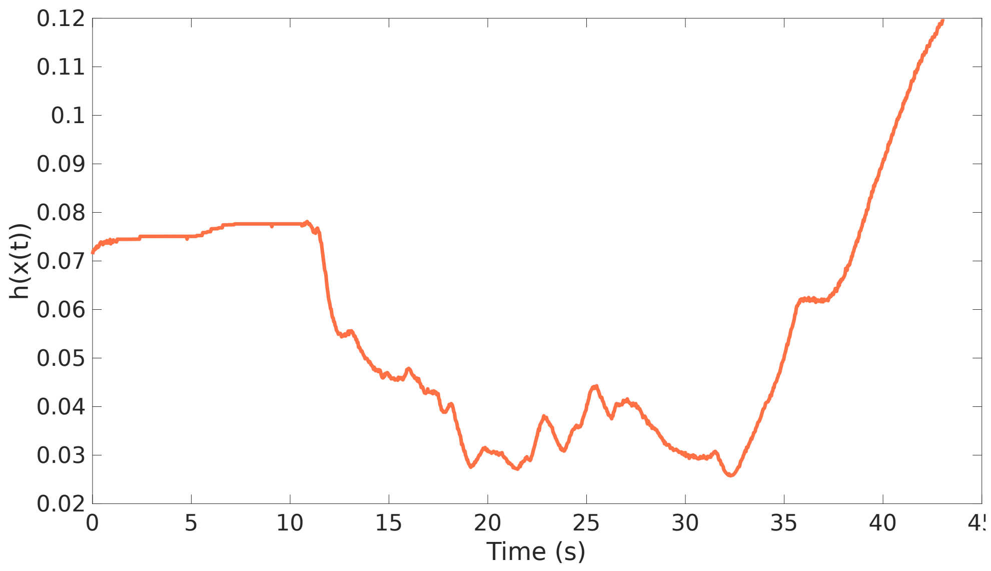

Figure 5.3: Value of the NBF that encodes the collision-avoidance constraints for the experiment described in Section 5.5. The value of the NBF remains positive over the course of the experiment, showing that all of the constraints are satisfied. That is, no robots collide.

To show the efficacy of these results on a real system, the controller $u^\star$ from (5.14) is deployed onto $N = 12$ differential drive robots in the Robotarium, a remotely accessible swarm-robotics testbed [8]. Figure 5.1 shows the robots over the course of the experiment. The projected pictures of corn and numbers on the testbed mark the assigned crop patches for each pair of robots as well as the robots' identifiers. Figure 5.1 shows that all robots reach their designated locations while avoiding collisions. Specifically, Figure 5.3 shows the value of h over the course of the experiment. The Boolean NBF h remains positive, indicating that all constraints are satisfied. Figure 5.2 shows the value of V during the experiment. The value of V decreases to 0, ensuring that each location is visited, which completes the objective.

5.5.2 Discussion of Parameters

For the experiment in Section 5.5.1, the parameters are

$$d = 0.15, \gamma = 10000, c = 0.3, \epsilon = 0.05,$$

and a discussion of these selections follows. The number d denotes the diameter of the Robotarium's differential-drive robots. The parameter γ controls the flatness (around the origin) of the extended class-$\mathcal{K}$ function $h(x) \mapsto \gamma h(x)^3$. This function is flat around 0, attentuating the rate at which the system can approach the boundary of the safe set. The parameter γ adjusts this rate: the larger γ is, the quicker robots can approach each other. For V, c controls how quickly the system must reduce the NLF. As such, c is chosen to ensure that the magnitude of $u^\star$ remains within the physical limits of the robots.

The parameter ϵ pertains to the δ-encapsulating index function and controls how many indices are included at each point. For example, for h, the δ-encapsulating index function is

$$I_h^\delta(x) = \{i : |h_i(x) - h(x)| \leq \epsilon\},$$

where each i corresponds to a collision constraint between a pair of robots. In effect, h represents the pair(s) of robots which are the closest, and ϵ controls how close other robots must be before being included in the QP (see (5.14)). Intuitively, making ϵ smaller reduces the number of constraints that must be included in the QP. Conversely, larger values of ϵ increase the number of constraints (i.e., nearby robots) that are included in the QP. A similar line of reasoning holds for V and I_V^δ.

Theoretically speaking, as long as $\epsilon > 0$, I_h^δ is indeed a δ-encapsulating index function. Though, practically speaking, since this implementation is inherently digital, increasing the value of ϵ can increase the robustness of the actual implementation by ensuring the constraints are included in the QP early enough. Again, a similar line of reasoning holds for V and I_V^δ.

5.6 Conclusion

Barrier functions and Lyapunov functions may be used to represent constraints and stability objectives for dynamical systems, respectively, and this chapter presented a new class of nonsmooth barrier functions and nonsmooth Lyapunov functions using the theory of piecewise smooth (PC^r) functions. Moreover, this chapter showed that Boolean combinations of barrier and Lyapunov functions fell into this class of PC^r functions. PC^r functions depend heavily on their corresponding index functions, and by utilizing a particular class of index functions, this chapter proved that one may synthesize controllers that are discontinuous yet, nonetheless, theoretically guarantee the validity of the barrier and Lyapunov functions. The experimental results utilized the theoretical contributions of this chapter to generate safe controllers that also accomplish an objective for a swarm of physical robots in a precision-agriculture scenario.

CHAPTER 6
REGULARITY OF CONTROLLER SYNTHESIS

This chapter discusses some results on controller synthesis in the context of Chapters 2-5. These chapters develop the theory of Nonsmooth Barrier Functions (NBFs) and associated controller-synthesis strategies that are based on an optimization program. Specifically, the sufficient conditions to ensure a valid NBF (see Theorem 2.1) involve satisfying an inequality that includes a set-valued product between the generalized gradient and the dynamics of the system of interest. If the system is control-affine, then the optimization program becomes a Quadratic Program (QP). The results of these chapters have assumed that the controller resulting from this QP is measurable and locally bounded. However, since these controllers are produced online and not in closed form, verifying these properties for a particular application becomes difficult. To strengthen the results in this thesis, this chapter investigates guaranteeing that the QP-based controller is measurable and locally bounded with an upper-semi continuous set-valued map as a constraint.

Previous work has focused on synthesizing continuous controllers to optimization programs, and most of these results come with some restrictive assumptions [26]. These assumptions typically relate to differentiability assumptions on the objective function and constraints. In the case of controller synthesis with respect to nonsmooth functions, this thesis utilizes the generalized gradient. As this object inherently represents a discontinuous object, these differentiability assumptions do not hold, so the results of this prior work cannot be utilized.

This chapter focuses on the regularity properties of solutions to a QP when a set-valued map is included as a constraint. In particular, this chapter shows that the assumptions on the constraints and objective function can be significantly relaxed, and the measurability and local boundedness of the solution to the QP can still be shown.

Set-valued analysis has been extensively utilized to analyze properties of the solution to an optimization program, and this paper takes advantage of these prior results. Specifically, the results herein utilize set-valued analysis stemming from [41]. Because the desired regularity properties relate to measurability, the theory of measurable set-valued maps is utilized, which, in turn, greatly relaxes the necessary assumptions on the optimization program.

The organization of this chapter is as follows. Section 6.1 notes some preliminary results useful in the main result of this chapter. Section 6.2 formulates the problem statment for the chapter. Section 6.3 contains the proofs for the regularity properties of synthesized controllers, constituting the main results of this chapter. Finally, Section 6.5 concludes the chapter and notes some potential future directions of work.

6.1 Preliminary Results

This section contains some relevant background material on set-valued maps, a more thorough coverage of which can be found in [41]. The most relevant topics to this chapter are covered including: measurable set-valued maps, measurable selections of set-valued maps, and Carathéodory functions. This thesis does not cover measure theory, as the theory behind the Lebesgue measure is not particularly relevant for the results herein. For background on Lebesgue measure, see [41, Chapter 8].

A set-valued map $F : \mathbb{R}^n \to 2^{\mathbb{R}}$ takes values in $\mathbb{R}^n$ and maps them to subset of $\mathbb{R}^n$. Measurability always refers to Lebesgue measurability, and measurability of set-valued maps is defined as follows.

Definition 17. *A set-valued map $F : \mathbb{R}^n \to 2^{\mathbb{R}}$ is* measurable *if the pre-image of every $open$ set is measurable. That is,*

$$F^{-}(O)$$

is measurable for every open set $O \subset \mathbb{R}^n$.

A set-valued map has closed images if $F(x')$ is closed for every $x' \in \mathbb{R}^n$. Carathédory functions are of particular interest to this chapter. Functions of this type are relevant in analysis of Carathéodory solutions to differential equations; however, they are also useful in this context.

Definition 18. *A function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is* Carath'eodory *if and only if for each fixed $u \in \mathbb{R}^m$ $x' \mapsto f(x', u)$ is measurable and, for each fixed $x' \in \mathbb{R}^n$, $u \mapsto f(x', u)$ is continuous.*

The next lemma provides a preliminary results that plays a role in showing measurability of the solution to a QP in Section 6.3.

Lemma 6.1. *Let $I : \mathbb{R}^n \to 2^K$ be a nonempty upper semi-continuous set-valued map, where K is a finite-cardinality index set; and let $a_i : \mathbb{R}^n \to \mathbb{R}^m$, $b_i : \mathbb{R}^n \to \mathbb{R}$, $i \in K$, be measurable functions. Then, the set-valued map $C : \mathbb{R}^n \to 2^{\mathbb{R}^m}$ defined as*

$$C(x') = \{u \in \mathbb{R}^m : a_i(x')^\top u + b_i(x') \geq 0, \forall i \in I(x')\}$$

is closed-valued and measurable.

Proof. Note that, at each $x' \in \mathbb{R}^n$,

$$C(x') = \bigcap_{i \in I(x')} \{u \in \mathbb{R}^m : a_i(x')^\top u + b_i(x') \geq 0\}.$$

Because $u \mapsto a_i(x')^\top u + b_i(x')$ is continuous, for each fixed x', each preimage

$$\{u \in \mathbb{R}^m : a_i(x')^\top u + b_i(x') \geq 0\}$$

is closed. Since $C(x')$ is the intersection of closed sets, $C(x')$ is also closed.

Now, it remains to show measurability of C. For each $i \in K$, define the set-valued map

$C_i : \mathbb{R}^n \to 2^{\mathbb{R}^m}$ be defined as

$$C_i(x') = \{u \in \mathbb{R}^m : a_i(x')^\top u + b_i(x') \geq 0\}.$$

By [41, Theorem 8.2.9], C_i is measurable and closed-valued. Let, for each $i \in K$, $A_i \subset \mathbb{R}^n$ be defined as

$$A_i = \{x' \in \mathbb{R}^n : I(x') \cap \{i\} \neq \varnothing\}.$$

Because I is upper semi-continuous, A_i is measurable, since it is the lower pre-image of a closed set (see [41, Proposition 1.4.4] or [41, Proposition 8.2.1]). As such, $\mathbb{R}^n/A_i$ is also measurable. Now, for each $i \in K$, define the set-valued map $\bar{C}_i : \mathbb{R}^n \to 2^{\mathbb{R}^m}$ as

$$\bar{C}_i(x') = \begin{cases} C_i(x') & x' \in A_i \\ \mathbb{R}^m & x' \notin A_i. \end{cases}$$

Note that $\bar{C}_i$ is a measurable set-valued map, since for any nonempty open set $Y \subset \mathbb{R}^m$

$$\{x' \in \mathbb{R}^n : \bar{C}_i(x') \cap Y \neq \varnothing\} = \{x' \in \mathbb{R}^n/A_i : \bar{C}_i(x') \cap Y \neq \varnothing\} \cup \{x' \in A_i : C_i(x') \cap Y \neq \varnothing\}$$

$$= \mathbb{R}^n/A_i \cup \left(C_i^-(Y) \cap A_i\right),$$

which is measurable, since C_i is a measurable set-valued map and A_i is a measurable set. Note that if Y is empty, then $F^{-1}(Y) = \varnothing$, which is measurable.

Combining the $\bar{C}_i$ yields that

$$C(x') = \bigcap_{i \in K} \bar{C}_i(x'),$$

because, at any $x' \in \mathbb{R}^n$, $i \in I(x')$ implies that $\bar{C}_i(x') = C_i(x')$, otherwise $\bar{C}_i(x') = \mathbb{R}^m$. By [41, Theorem 8.2.4], C is the intersection of measurable, closed-valued set-valued maps; as such, C is also measurable and closed-valued. $\square$

6.2 Problem Formulation

This section formulates the Quadratic Program (QP) which this chapter addresses. In particular, this chapter considers the QP

$$\begin{aligned} u^*(x') = \underset{u \in \mathbb{R}^m}{\arg\min}\, & u^\top P(x') u + q(x')^\top u \\ \text{s.t. } & a_i(x')^\top u + b_i(x') \geq 0, \forall i \in I(x') \\ & \|u\|_\infty \leq u_{max}, \end{aligned} \tag{6.1}$$

where $P : \mathbb{R}^n \to \mathbb{R}^{m \times m}$, $q : \mathbb{R}^n \to \mathbb{R}^m$ are measurable and locally bounded and P is positive-definite and symmetric at each $x' \in \mathbb{R}^n$. We moreover assume that each $a_i(x') : \mathbb{R}^n \to \mathbb{R}^m$, $b_i : \mathbb{R}^n \to \mathbb{R}$ is continuous and that $I : \mathbb{R}^n \to 2^K$ is upper semi-continuous, where K is a finite-cardinality index set.

The condition that G is positive-definite and symmetric at each $x' \in \mathbb{R}^n$ ensures that (6.1) has a unique solution at every $x' \in \mathbb{R}^n$. This property is important with for the regularity of $u^\star$. These assumptions can be weakened, along with correspondingly weaker guarantees. See Section 6.3 for more information on this topic.

6.3 Regularity Properties of the Solution

This section contains the main results of this chapter: showing that the solution to (6.1) is measurable and locally bounded. The result on measurability follows from Lemma 6.1 and the results of [41]. The property of local boundedness essentially results from the formulation of the QP in Section 6.2.

Theorem 6.1. *Let $u^* : \mathbb{R}^n \to \mathbb{R}^n$ be the solution to* (6.1)*, then u^* is measurable and locally bounded.*

Proof. The function u^* is clearly locally bounded, since the solution to (6.1) is bounded as a result of the constraint $\|u\|_\infty \leq u_{\max}$. It remains to show measurability.

By Lemma 6.1, the set-valued map $C : \mathbb{R}^n \times \mathbb{R}^m \to 2^{\mathbb{R}^m}$ given by

$$C(x', u) = \bigcap_{i \in I(x')} \{u \in \mathbb{R}^m : a_i(x')^\top u + b_i(x') \geq 0\}$$

is measurable and closed-valued. Consider the objective function $o : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ defined as

$$o(x', u) = u^\top P(x') u + q(x')^\top u.$$

For each fixed x', $u \mapsto o(x', u)$ is continuous. Moreover, for each fixed u, the function $x' \mapsto o(x', u)$ is measurable and locally bounded. As such, $o(x', u)$ is Carathéodory.

Note that u^* may be written as

$$u^* = \{u \in C(x') : o(x', u) = \min_{\bar{u} \in C(x')} o(x', \bar{u}),$$

where the box constraints may be readily incorporated into the constraint map $C(x')$. By [41, Theorem 8.2.11], u^* is a measurable set-valued map. Moreover, it is closed-valued, since $u \mapsto o(x', u)$ is continuous for each fixed x and $C(x')$ is closed-valued.

Because u^* is a measurable, closed-valued set-valued map, [41, Theorem 8.1.3] applies, and u^* has a measurable selection. Because the solution to the QP is unique, u^* is single-valued, and, thus, u^* must be measurable. $\square$

Remark 6.1. *Theorem 6.1 assumes that G is positive-definite and symmetric at each $x' \in \mathbb{R}^n$. These assumptions guarantee uniqueness of the solution $u^*(x')$, and the proof of the theorem critically relies on this property. However, this property can be weakened, and the statement of the theorem changes to that a measurable solution exists (i.e., rather than is guaranteed).*

6.4 Relation to Previous Work

The results in Section 6.3 show that the solution to the QP in (6.1) is measurable and locally bounded. This result, contained in Theorem 6.1, effectively strengthens Chapters 2-5, because these regularity properties of the solution do not have to be assumed (e.g., in Theorem 3.1). Moreover, Theorem 6.1 is of practical use, because online synthesized controllers cannot typically be easily validated. Thus, it is important that these regularity properties be guaranteed.

The QP in (6.1) also relates to work outside of this thesis, and such QPs appear in a variety of fields that depend on performing controller synthesis. For example, robust controller synthesis often assumes that a disturbance (potentially set-valued) is added to the constraints. In this case, the QP in (6.1) could be modified to contain the addition of a set-valued map, so the results in this chapter apply to areas beyond those considered by this thesis.

6.5 Conclusion

This chapter presented results on the regularity of a solution to a constrained optimization program. In particular, for quadratic programs with very light regularity assumptions, this chapter showed that the resulting solutions is measurable and locally bounded. This result has applications in the rest of the thesis, from a theoretical and practical perspective. Moreover, these results also related to other fields, such as robust control. Future efforts could focus on extended the results of this chapter to these alternative areas, such as robust control.

CHAPTER 7
THE ROBOTARIUM

This chapter contains a discussion of the Robotarium, a remotely accessible swarm-robotics testbed. While the theory developed in this thesis may apply to robotics at large, much of it has been inspired by solving problems pertaining to this testbed. This chapter contains results from the contributed publication [8] as well as recent efforts.

Interest in robotics research has grown in recent times, and increasing effort and funds are being correspondingly allocated toward this pursuit. However, a number of problems lie in the wake of this growing interest. The cost of starting a robotics lab can be prohibitively expensive, effort is often duplicated between similar robotics labs, and hardware differences can obfuscate research-based contributions.

First, the cost of starting a robotics lab, in terms of personnel and equipment, can be prohibitively expensive for researchers, from a new professor to a high-school student. Second, effort is often duplicated between robotics labs. For example, to start a swarm-robotics lab, a large number of differential-drive robots are typically required, and most of these systems incur a significant cost, in terms of both time and money, to establish a software pipeline. Moreover, this software setup remains relatively static between these labs, so this effort, which can potentially take years, greatly overlaps with similar labs. Third, hardware differences can obfuscate the contributions of research. Many contributions in robotics rely on demonstrating the efficacy of the chosen approach on a physical systems. Often, as should be the case, these experiments are compared with past efforts. However, hardware improvement can disrupt this comparison. For example, a recent paper may have access to better, more advanced hardware that, on a physical level, outperforms older hardware, making the newer contribution seem more effective than it really is. As such, an important goal in robotics research should be to normalize these testbeds to allow research to be compared properly.

The Robotarium solves these growing problems by offering free, remote access to a swarm of robots located at Georgia Tech. The Robotarium eliminates the cost of starting a robotics lab, eliminates duplicated effort by providing a software pipeline, and allows research to be tested on an equivalent testbed, ensuring that comparisons between research efforts utilize similar hardware. Users interact with the system by prototyping their algorithm in the MATLAB or Python programming languages and submitting their code via an online submission process. After their experiment is executed, users receive a video of their experiment as well as any data they chose to save.

The rest of this chapter is organized as follows. Section 7.1 details the interaction process for users. Section 7.2 discusses how safety of the physical devices in the Robotarium is ensured in the context of remote submissions from users as well as potentially malicious experiments. Next, Section 7.4 describes the software backend of the Robotarium. This structure is of relevance to this thesis because the Robotarium critically relies on composition of barrier functions with a stream of user inputs. Section 7.5 contains some examples of real user submissions for the platform. Finally, Section 7.6 concludes the chapter.

7.1 Interacting with the System

This section details the users' interaction process with the Robotarium. The Robotarium provides two primary interaction methods: MATLAB and Python simulators. Specifically, users utilize the MATLAB and Python simulators to encode their algorithm. After the algorithm has been encoded into the simulator, the user may submit their MATLAB or Python code via an online interface.

The Python and MATLAB simulators allow users to control a group of differential-drive robots that the Robotarium abstracts with the well-used unicycle model. In particular, each robot obeys the dynamics

$$\dot{x} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix},$$

where v and ω are the linear and angular velocities, respectively. Though this model is a faithful representation of the robots' actual dynamics, it is an under-actuated, nonlinear model. As such, not all users may be experienced with using this nonlinear model (e.g., high-school students). Even in technical research areas, many papers assume simplified dynamics, such as single- or double-integrator dynamics. With this idea in mind, the Robotarium provides a host of utilities to control groups of differential drive robots, including single-integrator-to-unicycle mappings, go-to-goal controllers, and plotting utilities.

7.1.1 Abstractions

Oftentimes, algorithms abstract dynamics to simplify analysis, particularly in swarm robotics. The level at which the abstraction takes place differs among applications. For example, a swarm-robotics algorithm may make a dynamical abstraction and assume that each of the robots obeys single-integrator dynamics, making the algorithm simpler to analyze mathematically. On the other hand, a planning algorithm may abstract the motion of the robot itself, assuming that the system has a built-in go-to-goal behavior. That is, given a supplied point (or pose) the system automatically regulates to this value. To facilitate these scenarios, the Robotarium provides a number of utilities, from single-integrator-to-unicycle mappings to pose controllers.

It is important to note that the provision of these utilities is critical to the operation of the Robotarium. Usually, the significant amount of required knowledge to control a group of robots limits potential users. To control a swarm of robots, one must typically posses an expertise in software, networked communications (e.g., TCP, UDP), and a knowledge of nonlinear control techniques to map the controller onto the system. This required knowledge can pose an issue to new roboticists, as acquiring this information can be a daunting task; moreover, it may not be particularly related to the research at hand. To truly lower this barrier, it is critical that the Robotarium provide tools for mapping abstracted algorithms onto nonlinear systems. In fact, usage of the Robotarium by non-traditional robotics groups, such as biologists, suggests that the provision of these utilities has enabled these users.

With regard to dynamical abstractions, the Robotarium provides implementations of the techniques discussed in [27, 37]. Each of these papers provides a different technique for mapping single-integrator algorithms to unicycle-modeled (or differential-drive) systems. These techniques are widely used and have been validated in practice by the Robotarium. Moreover, the Robotarium provides some go-to-point and go-to-pose controllers. These controllers represent another common type of abstraction. In this case, the robots' velocities are not directly controlled; rather, they track a point (or pose) supplied by the user.

7.2 Ensuring Safety

A major area of research in the Robotarium is hardware safety. The maximum velocity of the robots is roughly $0.4\ m/s$, which means that a full-speed collision could easily damage the arena or, more likely, the robots' hardware, so preventing collisions is a priority. Moreover, as automation represents a major goal for this system, this method must not require human intervention. However, a third objective remains. Since this collision-avoidance algorithm is always enabled, it may modify users submissions, but users submissions should still complete successfully if possible. As such, the collision-avoidance algorithm must be minimally invasive.

To create such an algorithm, the Robotarium utilizes barrier functions, which feature heavily in this thesis. Barrier functions are a natural fit for the Robotarium, because they can be formulated independently from a user's experiment. That is, safe controllers can be synthesized for effectively arbitrary submissions. There are many formulations of barrier functions, with various advantages and disadvantages, and this section presents two formulations utilized by the Robotarium. In practice, the choice between them varies based on the submission's details.

7.2.1 A Centralized Approach

As noted in Section 7.1.1, the Robotarium allows interaction at a number of levels of abstraction. For example, the robots can be modeled as single integrators with dynamics

$$\dot{x}_i = u_i,$$

where $i \in [N]$, for a group of N robots; then, this abstracted model may be mapped onto the full nonlinear unicycle dynamics using any of the techniques discussed in Section 7.1.

In this case, a pair-wise collision-avoidance constraint may be encoded as

$$h_{ij}(x_i, x_j) = \|x_i - x_j\|^2 - r^2,$$

where r denotes the diameter of the robot. Note that

$$\nabla_{x_i} h_{ij}(x_i, x_j) = 2(x_i - x_j);$$

$$\nabla_{x_j} h_{ij}(x_i, x_j) = 2(x_j - x_i);$$

and

$$\nabla_{x_k} h_{ij}(x_i, x_j) = 0, k \neq i, j.$$

As such, $\nabla_x h_{ij}(x_i, x_j)$ can be written as

$$\nabla_x h_{ij}(x_i, x_j) = \left[\nabla_{x_1} h_{ij}(x_i, x_j)^\top \ldots \nabla_{x_N} h_{ij}(x_i, x_j)^\top\right]^\top.$$

That is,

$$\nabla_{x_k} h_{ij}(x_i, x_j) = \begin{cases} 2(x_i - x_j) & k = i \\ 2(x_j - x_i) & k = j \\ 0 & \text{o.w.} \end{cases}.$$

Guaranteeing that

$$\nabla_x h_{ij}(x_i, x_j)^\top u \geq -\gamma h(x_i, x_j)^3,$$

for every $i \in [N-1]$, $j \in \{i+1, \ldots, N\}$, guarantees that robots i and j avoid collisions (e.g., see Chapter 2). As such, the following Quadratic Program (QP) produces a controller that guarantees that all robots remain collision-free

$$\begin{aligned} u^*(x) &= \underset{u \in \mathbb{R}^m}{\arg\min} \|u - u^{\text{user}}\|^2 \\ \text{s.t. } &\nabla_x h_{ij}(x_i, x_j)^\top u \geq -\gamma h(x_i, x_j)^3, \forall i \in [N-1], j \in \{i+1, \ldots, N\}, \end{aligned} \tag{7.1}$$

where u^* is the collision-free controller. Note that, in the above program, u^{user} represents the controller provided by a user's algorithm, so u^* represents the controller that avoids collisions while remaining as close as possible to the user's intent.

The program in (7.1) represents a critical component in the Robotarium's infrastructure. Not only does the resulting controller prevent collisions, but it is also minimally invasive, meaning that it alters the user's controller as infrequently as possible. This property means that the Robotarium's hardware is protected while also ensuring that the user's algorithm runs effectively.

From a control perspective, an important aspect of the strategy in (7.1) is that it is a fully centralized controller. The QP requires information from all pairs of robots and coordinates the controllers accordingly. In a real-world setting, such information may not be available to all agents; however, Robotarium can take advantage of the laboratory setting to solve such programs. In fact, this centralization has some nice practical effects. Namely, it ensures maximum coordination, allowing the user's algorithm to be executed faithfully.

Figure 7.1 shows an example of an experiment being executed on the prototype version of the Robotarium with barrier-function-based collision avoidance. In this hypothetical submission, the robots, arranged on an ellipse, attempt to swap positions. Without barrier functions enabled, this algorithm clearly causes collisions. With barrier functions, the robots avoid collisions and successfully complete the algorithm.

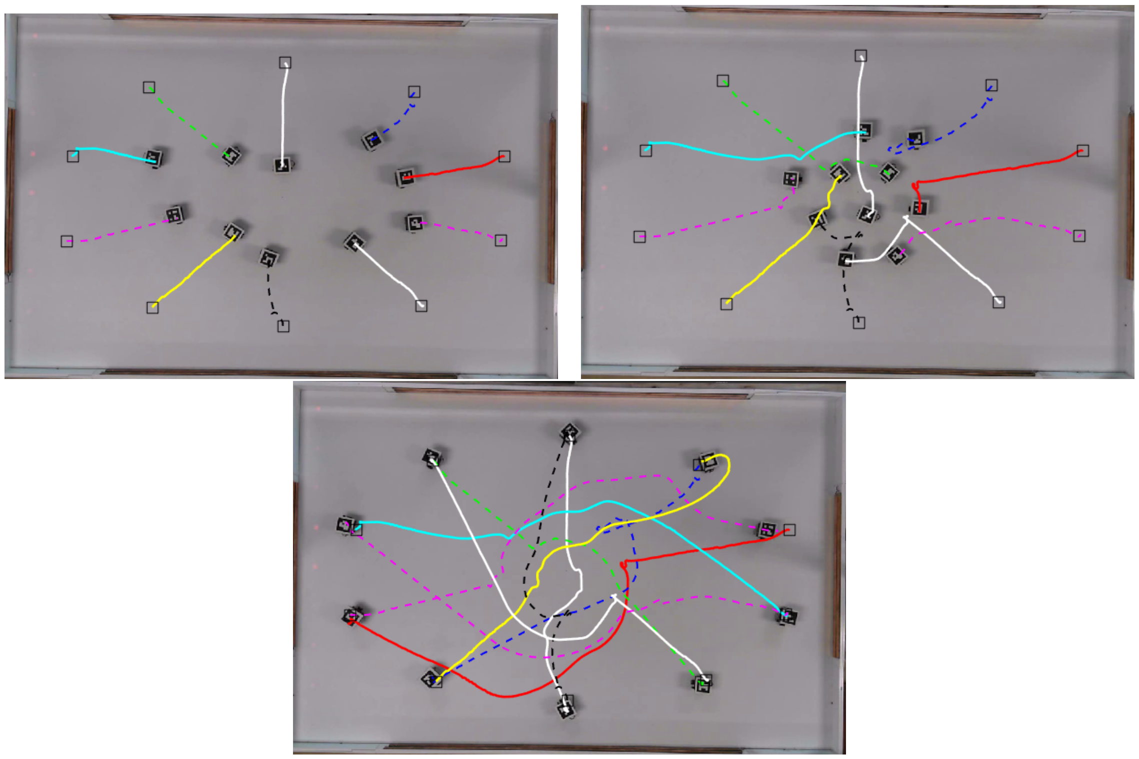

Figure 7.1: A group of differential-drive robots attempt to switch places on an ellipse. The Robotarium utilizes barrier-function-based collision avoidance, as in (7.1), to protect the robots and to ensure completion of the experiment.

7.2.2 A Decentralized Approach

From a computational perspective, the QP in (7.1) has been solved at 100 Hz for 40 robots with substantially under-optimized code. For larger numbers of robots, the runtime may be improved by decentralizing (7.1), allowing the program to be solved in parallel. The authors of [8] show that a decentralized version of the QP in (7.1) may be solved at 200 Hz for over 100 robots. Moreover, this parallelization decreases the effects of adding more robots.

The decentralized QP is formulated as follows. Note that (7.1) depends on satisfying the inequality

$$\nabla_x h_{ij}(x_i, x_j)^\top u \geq -\gamma h(x_i, x_j)^3,$$

for each $i \in [N-1]$, $j \in \{i+1, \ldots, N\}$. If every constraint is included in the QP, then this QP becomes centralized. However, note that

$$\nabla_x h_{ij}(x_i, x_j)^\top u = \nabla_{x_i} h_{ij}(x_i, x_j) u_i + \nabla_{x_j} h_{ij}(x_i, x_j) u_j.$$

The centralized QP simultaneously chooses u_i and u_j to satisfy the barrier-function constraint. However, the following alternate formulation separates the selection of u_i and u_j. Consider the QP

$$\begin{aligned} u_i^*(x) &= \underset{u \in \mathbb{R}^2}{\arg\min} \, \|u - u^{\text{user}_i}\|^2 \\ &\text{s.t. } 2(x_i - x_j)^\top u_i \geq -\frac{1}{2}\gamma h_{ij}(x_i, x_j)^3, \forall j \in [N]/\{i\}. \end{aligned} \tag{7.2}$$

If every agent solves (7.2), then

$$\begin{aligned} &\nabla_x h_{ij}(x_i, x_j)^\top u = \nabla_{x_i} h_{ij}(x_i, x_j) u_i + \nabla_{x_j} h_{ij}(x_i, x_j) u_j \\ &\geq -\frac{1}{2}\gamma h_{ij}(x_i, x_j)^3 - \frac{1}{2}\gamma h_{ij}(x_i, x_j)^3 \\ &\geq -\gamma h_{ij}(x_i, x_j)^3. \end{aligned}$$

Accordingly, all robots remain collision-free. Note that this formulation still requires that all robots have information about all other robots, yet the QP in (7.1) may be solved in parallel (i.e., each agent may solve the QP individually).

7.2.3 Centralized versus Decentralized

The Robotarium typically utilizes the centralized approach in (7.1), because it can be solved quickly enough for most submissions; additionally, the centralization affords an increased degree of coordination, allowing the users' code to execute more freely. However, if truly large numbers of robots are required (e.g., over 50), then the decentralized approach may be selected. It is worth noting that the vast majority of submissions require less than 50 robots.

The decentralized formulation, while quickly solved, is prone to a form of deadlock, where no progress in the submitted algorithm can be made (i.e., the robots simply stop moving). The increased coordination induced by the centralized formulation can help mitigate this condition in practice. As such, most submissions utilize the centralized formulation in (7.1).

7.3 Code Submission Process

The software infrastructure represents a major component of the Robotarium. To handle remote submissions, this infrastructure must handle the usual tasks, like fetching submissions from a database, but must also enable control of mobile robots. Accordingly, the software infrastructure can be broken into two equally important halves: the backend code for fetching, validating, and running submissions; and the communication infrastructure for the robots.

The first half, containing more mundane components, is mainly centered around the submission of the actual code from users. The main tasks for the backend code involve fetching submissions from a database, validating submissions through simulation, and deploying the algorithms onto the robots. In particular, when a user submits an experiment using the online submissions process, the submission, or job, is placed into a database. When the Robotarium is active, it fetches the most recently submitted experiment and begins the validation process. Figure 7.2 displays this pipeline.

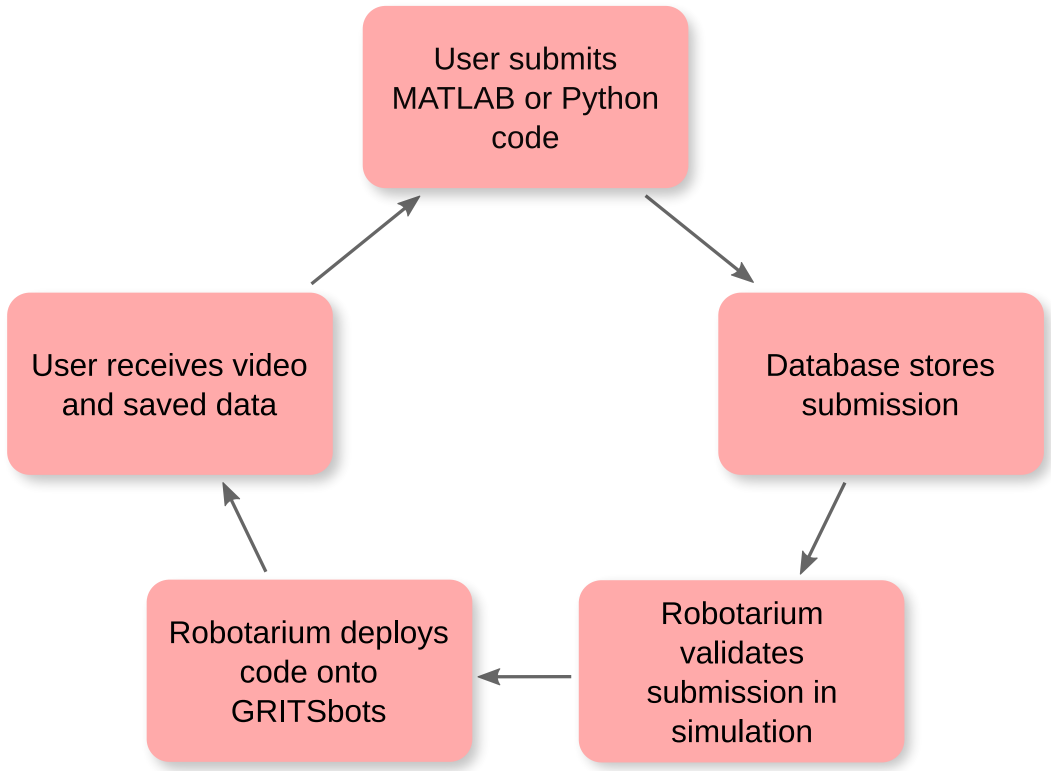
User submits MATLAB or Python code
Database stores submission
Robotarium validates submission in simulation
Robotarium deploys code onto GRITSbots
User receives video and saved data

Figure 7.2: A flow chart of a user's interaction with the Robotarium.

Because the Robotarium is openly accessible and aims to be automated, the system validates submitted experiments. This process ensures that the user-provided simulations meet some safety criteria before deploying it onto the system, such as number of collisions, time outside the boundaries, and surpassing of actuator limits. If too many of these errors are encountered in the simulation, then the submission fails the validation test, and the system does not execute the code on the physical robots. This validation step helps preserve the integrity of the physical hardware but also ensures that the user's submission has a higher likelihood of executing correctly.

Underpinning this portion of the software is Docker[1]. Docker is a containerization platform, allowing small, isolated, light-weight components to run in a cross-platform manner. Much like a virtual machine, Docker allows one to encapsulate build requirements into

[1]https://www.docker.com/

a portable container, which can be run on any docker-supported machine with the same processing architecture. In the case of differing processor architectures, the container can often be readily cross-compiled for a specific architecture (e.g., AMD64 to ARM).

In the context of the Robotarium, Docker has two crucial benefits. The first is that it allows the software to be portable. Since the Robotarium relies on many different types of systems, from embedded Linux-based computers to enterprise servers, this portability allows code to be developed independently from the platform under consideration.

Secondly, docker provides isolation of the software components, which is critical for an openly accessible platform. Since users may submit (almost) arbitrary code, written in Python or MATLAB, it becomes dangerous to execute this code directly on the Robotarium's servers. Docker allows this execution to be encapsulated by a container, meaning that any malicious or poorly written code only affects the local state of the container and cannot harm the host machine. This container-based isolation provides another benefit: users' submissions are all run in a fresh container, so all submissions begin with the same environment, settings, and capabilities. As such, containerization ensures fair execution by guaranteeing that all submissions have the same computing platform.

7.4 Communication Infrastructure

The second critical component of the Robotarium's software is the communication infrastructure. In the Robotarium, many pieces of hardware must communicate, from charging hubs to robots. As such, the method by which the various systems communicate becomes critical. The goal of the communication infrastructure is to accommodate point-to-point communications and remain accessible from a variety of programming languages.

7.4.1 Communication Method

Standard WiFi is a ubiquitous communication mechanism, and the Robotarium's devices all utilize WiFi to communicate. This decision allows the system to utilize many pre-built libraries and software packages. Moreover, this decision does not limit the system, as even small microcontrollers, such as the ESP8266 or ESP32, can be WiFi enabled with modern technology. Moreover, the controlled laboratory setting of the Robotarium provides a stable WiFi connection.

Applications, in practice, may require different communication mechanisms, such as long-distance RF signals. However, the Robotarium remains focused on the testing of robotics algorithms, which may include communication topologies; however, the means by which the agents communicate, while important, is not a focus area of the Robotarium.

Given the communication mechanism of WiFi, at an OS level, the available socket-based communication mechanisms are essentially TCP and UDP. However, these protocols may be too basic. For example, using exclusively UDP typically requires one to enrich the UDP-based communications with extra capabilities (e.g., a keep-alive). Additionally, TCP connections efficiently handle many desirable properties (e.g., establishing a connection, re-transmitting data); however, TCP is not effective in terms of data separation. For example, if three robots are communications over TCP connections, then each robot must either maintain a connection to the other two robots, or the transmitted packets must be modified to include a device identifier.

7.4.2 Publish-Subscribe Architecture

Publish-subscribe frameworks, such as in ROS [42], continue to grow in popularity, as they provide a convenient abstraction to connection-based communication. Specifically, they allow data to be conveniently multiplexed over a single socket according to string-based identifier. For example, the topic `"robots/1/battery_data"` may designate that this topic contains battery data for robot 1. Typically, topics are hierarchically structured. In a publish-subscribe framework, each client can subscribe to a number of topics, receiving data for all of these topics over a single socket-based connection. This architecture allows data to be sorted appropriately and solves the problem of communicating to many different devices over a single socket. However, this scheme causes an increase in complexity, as a central server must handle all subscriptions, message sorting, and message passing. However, the utility of the publish-subscribe architecture outweighs the cost for the Robotarium's requirements.

The Robotarium uses MQTT[2], which is an established publish-subscribe protocol that is implemented across many languages. The fact that most major languages implement an MQTT client benefits the Robotarium, because it requires many different components, including the individual robots, charging stations, pose-estimation software, and the Docker contains running the users' code. During the submission process (see Section 7.1), all of these components must communicate to ensure that the experiment executes successfully.

In the Robotarium, a central server runs an MQTT broker on a local WiFi network. All communicating systems connect to this broker, allowing them to pass messages to the rest of the system. All messages are encoded with JSON (Java Script Object Notation), a platform-independent message specification. Though potentially verbose, JSON is easily extensible and most major languages have a native JSON encoder and decoder, making it a portable message format. Moreover, encoded JSON is human readable, making debugging and monitoring of the message streams convenient.

Once a submission has been validated with the process in Section 7.1, a Docker container executes the experiment, sending control commands to the robots over MQTT. Figure 7.3 displays the communication between the relevant components during experiment execution. During an experiment, a container that runs tracking software sends the robots' poses to the execution container. Depending on the type of submission, this container either executes MATLAB or Python code. In either case, the container generates control commands for the robots and sends them over each robot's designated topic. Moreover, the executor container also retrieves the state of the robots (e.g., battery voltage) to ensure that the robots operate correctly.

[2]https://www.mqtt.org

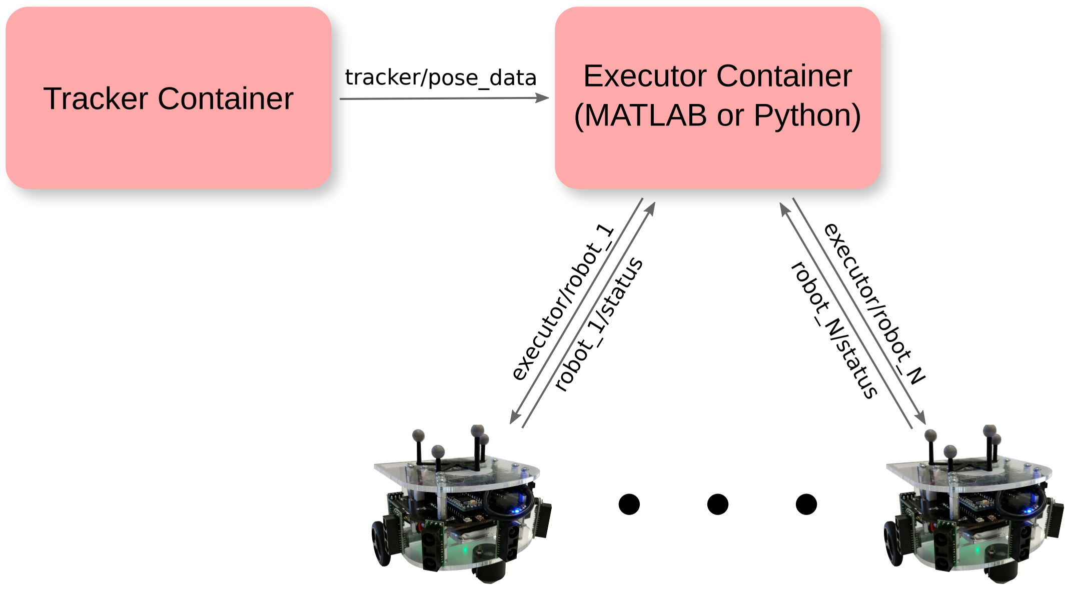

Figure 7.3: Communication architecture of the Robotarium during experiment execution. The various components communicate using MQTT and JSON-encoded messages. In the Robotarium, the tracker container sends pose information to an executor container that runs MATLAB or Python code. In turn, the executor container sends velocity information, which is generated by the user's algorithm, to the robots. This container also gathers information from the robots to ensure that they are operating correctly.

7.5 User Submissions

This section presents a selection of external user experiments that have been developed using a Robotarium-provided simulator and executed on the Robotarium using the software infrastructure shown in Figure 7.2. These examples were chosen as representative samples since they highlight the breadth of algorithms that can be executed on the Robotarium but also validate the remote access aspect of the Robotarium.

Distributed Formation Control of Cyclic Formations from the University of Texas, Dallas

This experiment instantiated a distributed formation control algorithm for regular polygonal formations, originally presented in [43]. The controller uses relative position measurements in local coordinate frames and prohibits any inter-agent communication. Note that [43] assumes agents to be points in the plane and does not consider collision avoidance. The

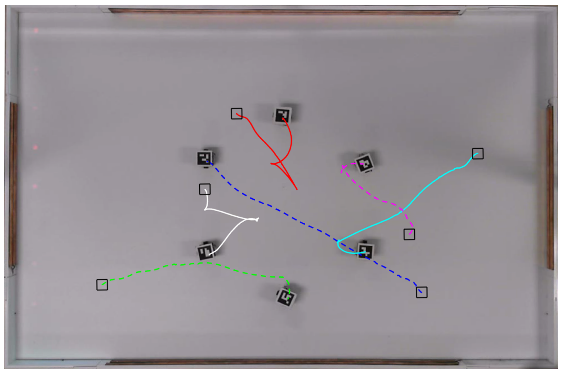

Figure 7.4: Distributed formation control of six GRITSBots assembling a regular hexagon starting at random initial positions.

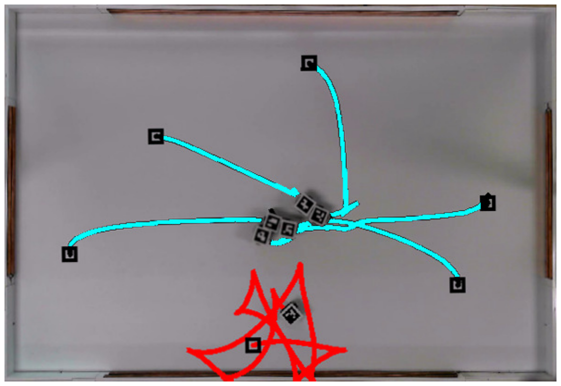

Figure 7.5: Fault-tolerant consensus with five collaborating robots (shown in blue) and one malicious robot (shown in red). Consensus is achieved despite the malicious robot's efforts to prevent rendezvous.

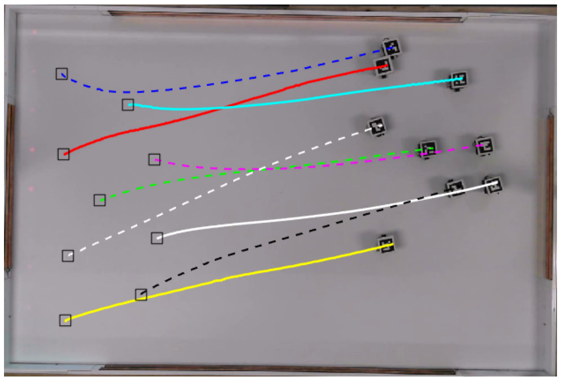

Figure 7.6: Passivity-based attitude synchronization implemented on a team of nine GRITSBots.

Figure 7.7: Experimental data from external user experiments rendered onto an image of the Robotarium testbed setup. The square markers and curves are initial positions and trajectories of the GRITSBots, respectively.

successful execution on the Robotarium therefore depended on the use of Robotarium-provided barrier certificates shown in Section 7.2 and a mapping from single-integrator dynamics to the unicycle dynamics of the robots. Figure 7.4 shows the results of this experiment with six robots.

7.5.1 Fault-tolerant Rendezvous from the University of Illinois Urbana-Champaign

A second experiment instantiated a fault-tolerant version of the rendezvous algorithm, originally presented in [44]. In this work, agents achieve consensus by moving towards points within a safe set, while maintaining connectivity through extendable sensing capabilities [44]. Because this algorithm models agents as points in the plane and contains no native collision avoidance, the successful execution utilize the single-integrator-to-unicycle mapping and barrier certificates provided by the Robotarium. Figure 7.5 shows the results of this experiment with six robots.

7.5.2 Passivity-based Attitude Synchronization from the Tokyo Institute of Technology

A passivity-based attitude synchronization algorithm, originally presented in [45], was implemented on the Robotarium. Utilizing the passivity property of general rigid-body motion in $SE(3)$, this algorithm was designed to achieve attitude synchronization for a group of rigid bodies with only local information exchanges, i.e., $v_i = v_j, \lim_{t \to \infty} \theta_i(t) - \theta_j(t) = 0, \forall i \neq j$. The successfully execution of this algorithm relies on the following capabilities provided by the Robotarium: 1) specification of a local information exchange graph (i.e., a cycle graph C_N); 2) a mapping from single-integrator to unicyle dynamics. Figure 7.6 shows the results of this experiment with eight robots.

7.6 Conclusion

This chapter discussed the Robotarium, a remotely accessible swarm-robotics testbed, which has been a major application for the theoretical developments of this thesis. The

Robotarium provides an openly accessible robotics testbed to those without the resources to maintain a cutting-edge robotics lab and also levels the playing field for robotics algorithms. The software infrastructure and use of barrier functions are critical to the Robotarium, and this chapter contained information on these topics, particularly, in illustrating how barrier functions can be used in a practical application that preserves the robots' hardware while also ensuring that users' algorithms complete successfully.

Appendices

APPENDIX A
NOTATION

The symbol $\times$ stands for the Cartesian product. We denote by $\mathbb{R}_{\geq 0}$ the set of nonnegative real numbers. For an integer $k > 0$, we use the shorthand notation $[k]$ to denote the set $\{1, \ldots, k\}$. The symbol $\circ$ denotes function composition. The abbreviation $a.e.$ stands for almost everywhere in the sense of Lebesgue measure. The expression $\langle \cdot\,, \cdot \rangle$ represents the inner product of two vectors. The abbreviation co stands for the convex hull of a set. A function $\alpha : \mathbb{R} \to \mathbb{R}$ belongs to extended class-$\mathcal{K}$ if α is continuous, strictly increasing, and $\alpha(0) = 0$. The function α is a class-$\mathcal{K}$ function when restricted to $\mathbb{R}_{\geq 0}$. A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is class-$\mathcal{KL}$ if it is class-$\mathcal{K}$ in its first argument and, for each fixed r, $\beta(r, \cdot)$ is continuous, strictly decreasing, and $\lim_{s \to \infty} \beta(r, s) = 0$.

APPENDIX B
DISCONTINUOUS DYNAMICAL SYSTEMS

This appendix discusses discontinuous dynamical systems in relation to the controller-synthesis results of this work. Controller synthesis, with respect to nonsmooth Lyapunov or barrier functions intevitably produces a discontinuous controller. As such, the theory of discontinuous dynamical systems is required to analyze discontinuous controller synthesis.

Section B.1 introduces the systems of interest for this thesis. However, ODEs with discontinuities in the state variable typically lack solutions. As such, Section B.2 discusses differential inclusions, which have solutions under lighter regularity conditions than ODEs. This section also discusses how discontinuous ODEs may be mapped into differential inclusions for which solutions exist. Differential inclusions require some set-valued analysis, so Section B.3 notes some relevant results in this area. Finally, Section B.4 discusses some related work to discontinuous systems (e.g., hybrid, switched systems).

B.1 System of Interest

This thesis considers control-affine system of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)), x(0) = x_0, \tag{B.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are continuous and $u : \mathbb{R}^n \to \mathbb{R}^m$ is measurable and locally bounded. The value x_0 denotes the initial condition for the system. Most mechanical systems fall into this category of systems (e.g., quadcopters and differential-drive robots). Note that in (B.1) f and g are assumed to be continuous, but u is only assumed to be measurable and locally bounded, which very generally captures discontinuities. Theoretically speaking, f and g may be assumed to also be measurable and locally bounded, but as this work typically discusses (B.1) with respect to controller synthesis, the continuity assumption is maintained for brevity.

One may argue whether the assumption, namely that u is measurable and locally bounded, is an appropriate way to analyze discontinuous dynamical systems. Indeed, other methods exist in the literature two of which are switched systems and hybrid systems. See Section B.4 for a comparison among these methods.

For some parts of this work, the time-varying ODE

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t),t), x(t_0) = x_0, t_0 \in \mathbb{R}. \tag{B.2}$$

may also be referenced when relevant. In general, the theory developed for (B.1) also holds for systems described by (B.2) by including time as a state.

One side effect of assuming a discontinuous model is that solutions to such a system may not exist. Indeed, even for simple systems, the systems in (B.1) and (B.2) may not have solutions. Consider the following example.

Example B.1. *Let*

$$\dot{x}(t) = f(x(t)),$$

where $f : \mathbb{R} \to \mathbb{R}$. Define f as the function

$$\begin{cases} f(x') = 0 & x' \neq 0 \\ f(x') = 1 & x' = 0. \end{cases}$$

Figure B.1 displays this function. Then, solutions do not exist for every initial condition. For example, consider $x_0 = 0$.

With the above example in mind, the question arises of which type of solution to use. In the case that f, g, and u are locally Lipschitz continuous, solutions may assumed to be continuously differentiable. Indeed, under these assumptions, solutions always exists (over

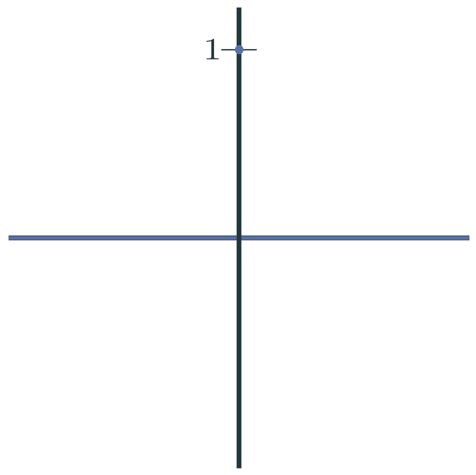

Figure B.1: A function for which the corresponding differential equation does not have a solution for every initial condition.

some interval) and are unique. However, in the case of (B.1), u is discontinuous. Thus, solutions cannot be continuously differentiable. Accordingly, the idea of a solution to an ODE must be expanded in order to apply to discontinuous systems.

This line of thought introduces the question: what are the essential qualities of a solution? In this case, we seek a continuous function whose derivative exists and is integrable. Moreover, the integration results in the function itself (i.e., the fundamental theorem of calculus applies). As such, the right notion of continuity is absolute continuity. In particular, a function $x : [a, b] \to \mathbb{R}$ is absolutely continuous on the interval $[a, b]$ if for every $\epsilon > 0$ there exists a $\delta > 0$ such that whenever

$$\sum_{i=1}^{\infty}(b_i - a_i) \leq \delta,$$

for any finite number of disjoint subsets of $[a, b]$, then

$$\sum_{i=1}^{\infty} \|x(b_i) - x(a_i)\| \leq \epsilon.$$

It turns out that a function is absolutely continuous if any only if the derivative $\dot{x}$ exists almost everywhere, in the sense of Lebesgue measure, is a Lebesgue integrable function, and

$$x(c) = x(a) + \int_a^c \dot{x}(s)ds,$$

for all $c \in [a, b]$. As such, if one seeks a continuous solution to an ODE, an absolutely continuous functions is, mathematically, the weakest possible kind of solution. In the differential equations literature, this type of solution is called a Carathéodory solution. Indeed, the following statement may be made. Consider the ODE,

$$\dot{x}(t) = f(x(t), t), x_0 = x(0),$$

if $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is locally bounded in both arguments, $t' \mapsto f(x', t')$ is measurable for each fixed $x' \in \mathbb{R}^n$, $x' \mapsto f(x', t')$ is continuous for each fixed $t' \in \mathbb{R}$, then there exist Carathéodory solutions for every initial condition. A function f is locally bounded if for every $(x', t') \in \mathbb{R}^n \times \mathbb{R}$ there exists a $\delta_1 > 0$ and an integrable function $m : [t', t' + \delta_2] \to \mathbb{R}_{\geq 0}$ such that

$$\|f(y, s)\| \leq m(s), \forall y \in B(x', \delta_1), s \in [s, s + \delta_2].$$

In this case, uniqueness is forsaken; however, no results of this thesis require uniqueness of solutions.

However, this situation does not capture the ODEs in (B.1) and (B.2), because the discontinuities occur in the state variable. Thus, discontinuous ODEs may not appropriate from an analysis perspective. As such, the subsequent section discusses how to map a discontinuous ODE to a differential inclusion for which solutions exist.

B.2 Differential Inclusions

This section covers some results on differential inclusions. As previously noted, some difficulty arises when an ODE is discontinuous in the state variable (see (B.1)). Again considering Example B.1 shows that the presented ODE also has no Carathéodory solutions for certain initial conditions (i.e., $x_0 = 0$). Indeed, this situation arises because the function f in Example B.1 is discontinuous in the state variable; thus, it does not meet the sufficient conditions for existence of solutions. To address systems, as in (B.1). It turns out that existence of solutions may be guaranteed upon a further relaxation of the concept of a solution.

Indirectly, the theory of differential inclusions is utilized. A differential inclusion, in contrast to a differential equation, is given by

$$\dot{x}(t) \in F(x(t)), x(0) = x_0, \tag{B.3}$$

where $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is a set-valued map. To introduce this theory, Section B.3 contains some notes on set-valued analysis, which holds some value for understanding differential inclusions. Indeed, the conditions for existence to solution for (B.3) are weaker than that of (B.1). In particular, Carathéodory solutions can be guaranteed when the set-valued map F is nonempty, convex-, compact-valued and upper semi-continuous. The set-valued map is nonempty, convex-, compact-valued if at each $x' \in \mathbb{R}^n$, $F(x')$ is nonempty, compact, and convex. A set-valued map is upper semi-continuous if for every $\epsilon > 0$, at every $x' \in \mathbb{R}^n$, there exists a $\delta > 0$ such that

$$\|y - x'\| \leq \delta \implies F(y) \subset F(x') + B(0, \epsilon),$$

where $F(x') + B(0, \epsilon)$ is meant to be understood as a set-valued addition. A Carathéodory solution to a differential equation is an absolutely continuous function $x : [0, t_1] \to \mathbb{R}^n$ such that $\dot{x}(t) \in F(x(t))$ almost everywhere on $[0, t_1]$. Just as in (B.2), differential inclusions may also be formulated for time-dependent systems. In particular,

$$\dot{x}(t) \in F(x(t), t), x(t_0) = x_0, t_0 \in \mathbb{R}.$$

The question remains: how can the discontinuous ODE in (B.1) be related to the differential inclusion in (B.3)? This question has been studied in [24] and, subsequently, applied to controlled systems in [15, 16], which develops nonsmooth Lyapunov theory for discontinuous dynamical systems. The work in [24] formulates Filippov's operator, which maps discontinuous differential equations to differential inclusions for which solutions exist.

Definition 19 ([16, Theorem 1]). *Given a measurable and locally bounded function $f : \mathbb{R}^n \to \mathbb{R}^n$, Filippov's operator $K[f] : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is defined at a point $x' \in \mathbb{R}^n$ as*

$$K[f](x') = \mathrm{co}\{\lim_{i\to\infty} f(x_i) : x_i \to x', x_i \notin N, N_f,$$

where N_f is a particular zero-measure set and N is an arbitrary zero-measure set.

As such, one may consider the differential inclusion resulting from Filippov's operator as

$$\dot{x}(t) \in K[f](x(t)), x(0) = x_0. \tag{B.4}$$

Moreover, $K[f]$ automatically satisfies the sufficient conditions for existence of Carathéodory solutions. That is, there exist Carathéodory solutions to the differential inclusions described by (B.4). In the literature, these solutions are called Filippov solutions to (B.1).

For an example, return to Example B.1. In this case, Caratheéodory solutions to the differential equation do not exist; however, Carathéodory solutions to the corresponding differential inclusions via Filippov's operator do exist. Intuitively, Filippov's operator is independent of the vector field at a particular point; as such, it can remove pathological points in the vector field, ensuring that solutions always exist. For an example of this phenomenon, consider the following example, which builds on Example B.1.

Example B.2. *Let $f : \mathbb{R} \to \mathbb{R}$ be as in Example B.1. Then, $K[f] : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is given by*

$$K[f](x') = \{0\}, \forall x' \in \mathbb{R}.$$

Thus, for any initial condition x_0 the solution is the constant solution $x(t) = x_0$. In fact, $K[f]$ is single-valued everywhere, which, in this case, may be interpreted as the zero function.

For robotics applications, differential equations, as in (B.1), stem from a physical interpretation of the system. As such, one may inquire if substituting the differential inclusion in (B.4) for (B.1) represents a physically meaningful operation. This theorem posits in the affirmative. In fact, one may argue that truly discontinuous systems, as in (B.1), are not physically meaningful, particularly in robotic systems. Filippov's operator regularizes discontinuous dynamical systems by eliminating points that have no physical interpretation (e.g., Example B.2). For a comprehensive survey of discontinuous dynamical systems, see [17].

A major topic of this thesis is forward set invariance via barrier functions. There are different formulations of this concept in the literature, stemming from nonuniqueness of solutions. However, this thesis uses the so-called strong version of forward set invariance noted next.

Definition 20. *A set $\mathcal{C} \subset \mathbb{R}^n$ is forward invariant, with respect to* (B.3)*, if $x_0 \in \mathcal{C}$ implies that $x(t) \in \mathcal{C}$, for every $t \in [0, t_1]$, for every Carathéodory solution of* (B.3) *starting from x_0.*

B.3 A Primer on Set-Valued Analysis

This section contains a very brief primer on set-valued analysis, pertaining to the work in this thesis. This thesis concentrates on both discontinuous dynamical systems and nonsmooth analysis, whih both require set-valued analysis. Set-valued analysis deals with objects that map points to sets. In this thesis, set-valued maps are denoted $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, where $2^{\mathbb{R}^n}$ denotes the set of subsets (i.e., the power set) of $\mathbb{R}^n$. That is, F maps points in $\mathbb{R}^n$ to subsets of $\mathbb{R}^n$.

In many senses, set-valued maps are analogous to functions; however, some extra conditions typically apply. One important property of set-valued functions is upper semi-continuity.

Definition 21. *A set-valued map $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is* upper semi continuous *if for every $\epsilon > 0$ and for every $x' \in \mathbb{R}^n$, there exists $\delta > 0$ such that*

$$\|y - x'\| \leq \delta \implies F(y) \subset F(x') + B(0, \epsilon).$$

The set-valued addition

$$F(x') + B(0, \epsilon)$$

is interpreted as

$$\{z + e : z \in F(x'), e \in B(0, \epsilon)\}.$$

This work does not distinguish the symbols for usual addition and set-based addition. Rather, we allow context to define these operations, following the notation in [41]. The same philosophy applies to multiplication as well.

Pre-images of function play a critical role in analysis of functions, and continuity can be formulated in terms of the pre-image of a function. For set-valued analysis, two pre-images play a role: the upper and lower pre-image, which are defined as follows. Interestingly, these pre-images coincide when F is a function (i.e., single-valued and nonempty everywhere), yet they endow F with distinctly different properties.

Definition 22. *Let $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be a set-valued map, and let $A \subset \mathbb{R}^n$. Then, the* upper pre-image *$F^+(A)$ is defined as*

$$F^+(A) = \{x' \in \mathbb{R}^n : F(x') \subset A\}.$$

Definition 23. *Let $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be a set-valued map, and let $A \subset \mathbb{R}^n$. Then, the* lower pre-image $F^{+}(A)$ *is defined as*

$$F^{-}(A) = \{x' \in \mathbb{R}^n : F(x') \cap A \neq \varnothing\}.$$

The notion of upper semi-continuity, as in Definition 21, can be formulated in terms of the upper pre-image, with certain extra assumptions on F. See [41] for a significantly more thorough coverage of set-valued analysis.

B.4 Related Work

This appendix presents one potential approach to analyze discontinuous dynamical systems, and several other approaches have received considerable attention. Two that have received significant attention are hybrid systems and switched systems. Hybrid systems have been considered extensively in the literature (e.g., as in [35]). Hybrid systems combine discrete and continuous elements into a single system. In particular, part of the state is governed by a differential equation, and part of the state is controlled by a state automaton. This formulation is similar to the formulation that this thesis utilizes, but there are some noticeable differences. In particular, hybrid systems may consider solutions that are discontinuous (i.e., that have jumps). Hybrid systems without jumps bear a resemblance to the differential inclusions considered in this thesis. For example, consider so-called dwell-time systems with no jumps, which enforce the constraint that the system must spend a certain amount of time in each state or mode. Such systems may be considered as differential equations that are measurable in the time variable. As such, dwell-time systems may fall into the class of systems that this paper considers.

Indeed, dwell-time systems with no jumps are related to switched systems, and switched systems represent another class of discontinuous system [36, 46]. This theory specifies that the dynamical system is controlled by a collection of differential equations, called modes, between which the system switches. The switches are controlled by the so-called switching signal. Typically, switching signals are constrained to contain countably many switches, meaning that this class of systems also falls into the discontinuous differential equations of this thesis.

APPENDIX C
NONSMOOTH ANALYSIS

This appendix discusses some tools from nonsmooth analysis as required for the theoretical developments of this thesis. Nonsmooth analysis represents an important tool in this thesis, because Boolean composition of barrier and Lyapunov functions, as formulated herein, introduces points of nondifferentiability. Luckily, nonsmooth analysis has been a well-studied topic, particularly with respect to controlled systems, and many mathematical tools have been developed to analyze nonsmooth functions along trajectories of a dynamical system.

A primary tool of nonsmooth analysis is the generalized gradient [11]. The generalized gradient is a set-valued map that captures all possible gradients around a point in an analogous fashion to Filippov's operator in Appendix B. The generalized gradient applies to locally Lipschitz functions and always exists under these regularity conditions. Due to Rademacher's theorem, the usual derivative of a locally Lipschitz functions exists almost everywhere, and the generalized gradient combines these possible derivatives into a set. It turns out that the generalized gradient posses enough regularity to capture derivatives along trajectories, making it suitable for analysis of nonsmooth Lyapunov and barrier functions. This appendix discusses the generalized gradient of [11] and its applications to this work. Specifically, Section C.1 discusses the rational behind the selection of the generalized gradient as the primary nonsmooth-analysis tool of this paper, and Section C.2 presents the generalized gradient and its corresponding calculus.

C.1 Choosing a Generalized Derivative

This section notes some rationale for the selection of a generalized derivative, as more than one exist in the literature. With respect to controlled systems, prior work on stability and invariance (e.g., Lyapunov functions) usually makes smoothness assumptions to ease analysis and ensure existence of solutions to the differential equation of interest. This thesis utilizes nonsmooth functions, which correspondingly require nonsmooth analysis. In this case, the particular branch of nonsmooth analysis corresponds to generalized derivatives because, just as in the smooth case, analyzing nonsmooth functions along trajectories to dynamical systems is a useful tool to assess qualities of the system over time.

In the case of nonsmooth functions, the usual derivative does not apply, and there are a plethora of modifications to the usual derivative which all exist under particular assumptions. In fact, in this case, the regularity assumptions determine the appropriate object that should be used. As noted in Appendix B, the integrability of derivatives of the functions remains paramount. To see this, consider a function $h : \mathbb{R}^n \to \mathbb{R}$, which may capture some desirable property (e.g., invariance, stability). Let $x : [0, t_1] \to \mathbb{R}^n$ be a Carathéodory solution (see Appendix B) to a system (e.g., the differential inclusion in (B.3)). Now, one wishes to analyze the composition $t \mapsto h(x(t))$.

Ideally, the relationship

$$h(x(t)) = h(x_0) + \int_0^t \dot{h}(x(s))ds$$

holds so that properties of $\dot{h}(x(t))$ are readily passed to $t \mapsto h(x(t))$. Because x is a Carathéodory solution, continuously differentiability is impossible. Moreover, the desired integrability property requires that the composition $t \mapsto h(x(t))$ is absolutely continuous. By definition, Carathéodory solutions x are absolutely continuous, but absolutely continuous functions are not closed under composition. As such, assuming h is absolutely continuous is not sufficient. One method of guaranteeing absolute continuity of the composition is to assume that h is locally Lipschitz continuous. The goal now is to select a generalized derivative for locally Lipschitz functions that captures the usual derivative when applicable.

Capturing the usual derivative means the following. Theories such as Lyapunov are critically based on analyzing a function along trajectories, as previously noted. In the smooth case, the usual chain rule may be used to spatially verify that the derivative satisfies the desired property. That is, given a smooth function h and a continuously differentiable trajectory x, one may analyze the product

$$\nabla h(x')^\top f(x'),$$

where $f(\cdot)$ represents the dynamics for x, over a the domain $\mathbb{R}^n \ni x'$. To appropriately analyze the nonsmooth case, the selected generalized derivative must provide a means to spatially analyze the system.

C.2 The Generalized Gradient

One such object that has an extensive calculus is the generalized gradient of [14]. Moreover, this object applies to locally Lipschitz functions. As such, it is a natural fit for this class of nonsmooth functions. Moreover, the calculus of [14] is particularly applicable to the case of Boolean composition, which is discussed below. Toward introducing this theory, the generalized directional derivative is defined below.

Definition 24 ([14]). *The* generalized directional derivative *of a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x' \in \mathbb{R}^n$ in the direction v is given by*

$$f^\circ(x'; v) = \limsup_{\substack{y \to x' \\ h \downarrow 0}} \frac{f(y + hv) - f(y)}{h}$$

.

The generalized directional derivative, as in Definition 24, always exists due to the $\limsup$ operation. However, some regularity assumptions on f are required to make it usable for analysis purposes. In particular, when f is locally Lipschitz, f° obtains desirable properties, and, in finite dimensions, permits the definition of the generalized gradient as follows.

Definition 25 ([14, Chapter 2, p. 27]). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz. The* generalized gradient *at a point $x' \in \mathbb{R}^n$ of f is defined as*

$$\partial f(x') = \{z \in \mathbb{R}^n : \langle z\,, v\rangle \leq f^\circ(x', v), \forall v \in \mathbb{R}^n\}.$$

The locally Lipschitz assumption on the function, within Definition 25, ensures that the set $\partial h(x')$ is nonempty, convex, and compact (in finite dimensions). Moreover, as a set-valued map $\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is upper semi-continuous [14, Proposition 2.1.5], and the following relationship between h° and $\partial_c h$ holds

$$h^\circ(x'; v) = \max_{z \in \partial_c h} z^\top v, \tag{C.1}$$

for every $x', v \in \mathbb{R}^n$ [14, Proposition 2.1.2].

As an aside, one may pick a relaxed derivative to represent $\dot{h}(x(t))$, such as a one-sided derivative or a Dini derivative. However, since this thesis has already noted a need for h to be locally Lipschitz continuous, using a weaker derivative becomes unnecessary. Moreover, in the finite-dimensional case, the generalized gradient admits a simpler definition than in Definition 25.

Theorem C.1 ([14, Theorem 2.5.1]). *Let $h : \mathbb{R}^n \to \mathbb{R}$ be Lipschitz near x', and suppose S is any set of Lebesgue-measure zero in $\mathbb{R}^n$. Then, the generalized gradient of the function $\partial_c h(x')$ is*

$$\partial_c h(x') = \text{co}\{\lim_{i\to\infty} \nabla h(x_i) : x_i \to x', x_i \notin \Omega_h, S\},$$

where Ω_h is the Lebesgue-zero-measure set where h is nondifferentiable.

The generalized gradient also has a developed calculus that becomes invaluable in this work. This authors of [14] cover this system extensively, but a few of the most relevant properties are included here. These propositions have been modified slightly to fit the terminology of this work.

Proposition C.1 ([14, Proposition 2.3.12])**.** *Let $\{h_i\}$ be a finite collection of functions ($i = 1, 2, \ldots, k$) Lipschitz near x'. Then, the function h defined by*

$$h(x') = \max_{i \in [k]} \{h_i(x')\}$$

is Lipschitz near x' as well. Let $I(x')$ denote the set of indices i for which $h_i(x') = h(x')$. Then,

$$\partial_c h(x') \subset \mathrm{co}\{\partial_c h_i(x') : i \in I(x')\}.$$

Theorem C.2 ([14, Theorem 2.3.9])**.** *Let $h = f \circ g$, where $g : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$ are locally Lipschitz functions, and let $x' \in \mathbb{R}^n$. Then*

$$\partial h(x') \subset \mathrm{co}\left(\bigtimes_{k=1}^{m} \partial_c g^k(x') \right) \partial_c f(g(x')),$$

where g^k denotes the k^{th} component function of g.

Proposition C.2 ([14, Proposition 2.3.1])**.** *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function. For any scalar s one has*

$$\partial_c (sh)(x') = s\partial_c h(x'), \forall x' \in \mathbb{R}^n.$$

Recapping the above thoughts, nonsmooth functions must be analyzed along Carathéodory solutions to differential inclusions. Given a locally Lipschitz function, the composition with a trajectory remains absolutely continuous. However, the derivative will not exist at every point in time. Fortunately, the generalized gradient spatially captures the derivative, much like the usual chain rule in the smooth case.

Theorem C.3. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function, and let $x : [0, t_1] \to \mathbb{R}^n$ be a Carathéodory solution to* (B.3)*. Then,*

$$\dot{h}(x(t)) \in \partial_c h(x(t))^\top F(x(t)),$$

almost everywhere on $[0, t_1] \ni t$.

Proof. From the proof of [19, Lemma 1] and (C.1), the derivative of $t \mapsto h(x(t))$ satisfies

$$\dot{h}(x(t)) \leq \max \partial_c h(x(t))^\top F(x(t))$$
$$\dot{h}(x(t)) \geq \min \partial_c h(x(t))^\top F(x(t))$$

almost everywhere on $[0, t_1] \ni t$. Moreover, since the usual inner product on $\mathbb{R}^n$ is continuous and $\partial_c h$ and $F(x(t))$ are convex valued, the set $\partial_c h(x(t))^\top F(x(t))$ is connected for every $t \in [0, t_1]$. Moreover, since it is a subset of the real line, it must be that

$$\dot{h}(x(t)) \in \partial_c h(x(t))^\top F(x(t)),$$

almost everywhere on $[0, t_1] \ni t$. $\square$

The reuslts of this work critically rely on Theorem C.3, because it allows properties of a system to be spatially verified with respect to nonsmooth functions. That is, rather than calculating $\dot{h}(x(t))$ explicitly, the algorithm or theorem may verify qualities of the set-valued product

$$\partial_c h(x')^\top F(x')$$

over the domain $\mathbb{R}^n \ni x'$ and then apply Theorem C.3.

REFERENCES

[1] 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*, Dec. 2014, pp. 6271–6278.

[2] “Control barrier certificates for safe swarm behavior,” *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68 –73, 2015, Analysis and Design of Hybrid Systems ADHS.

[3] L. Wang, A. D. Ames, and M. Egerstedt, “Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec. 2016, pp. 2659–2664.

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

[5] L. Wang, A. D. Ames, and M. Egerstedt, “Safety barrier certificates for collisions-free multirobot systems,” *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, Jun. 2017.

[6] X. Xu, “Constrained control of input-output linearizable systems using control sharing barrier functions,” *Automatica*, vol. 87, pp. 195 –201, 2018.

[7] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, “Robustness of control barrier functions for safety critical control**this work is partially supported by the national science foundation grants 1239055, 1239037 and 1239085.,” *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54 –61, 2015, Analysis and Design of Hybrid Systems ADHS.

[8] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, “The robotarium: A remotely accessible swarm robotics research testbed,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1699–1706.

[9] Y. Lin, E. Sontag, and Y. Wang, “A smooth converse lyapunov theorem for robust stability,” *SIAM Journal on Control and Optimization*, vol. 34, no. 1, pp. 124–160, 1996.

[10] H. K. Khalil, “Nonlinear systems,” 2002.

[11] F. Clarke, Y. Ledyaev, and R. Stern, “Asymptotic stability and smooth lyapunov functions,” *Journal of Differential Equations*, vol. 149, no. 1, pp. 69 –114, 1998.

[12] P. Glotfelter, J. Cortés, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE Control Systems Letters*, vol. 1, no. 2, pp. 310–315, Oct. 2017.

[13] P. Glotfelter, J. Cortés, and M. Egerstedt, "Boolean composability of constraints and control synthesis for multi-robot systems via nonsmooth control barrier functions," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, Aug. 2018, pp. 897–902.

[14] F. Clarke, *Optimization and Nonsmooth Analysis*. Soc. for Ind. and Appl. Math., 1990.

[15] D. Shevitz and B. Paden, "Lyapunov stability theory of nonsmooth systems," *IEEE Trans. Autom. Control*, vol. 39, no. 9, pp. 1910–1914, Sep. 1994.

[16] B. E. Paden and S. S. Sastry, "A calculus for computing Filippov's differential inclusion with application to the variable structure control of robot manipulators," in *25th IEEE Conf. Decision and Control*, Dec. 1986, pp. 578–582.

[17] J. Cortés, "Discontinuous dynamical systems," *IEEE Control Syst. Mag.*, vol. 28, no. 3, pp. 36–73, Jun. 2008.

[18] P. Glotfelter, I. Buckley, and M. Egerstedt, "Hybrid nonsmooth barrier functions with applications to provably safe and composable collision avoidance for robotic systems," *IEEE Robotics and Automation Letters*, pp. 1–1, 2019.

[19] A. Bacciotti and F. Ceragioli, "Stability and stabilization of discontinuous systems and nonsmooth Lyapunov functions," *ESAIM: COCV*, vol. 4, pp. 361–376, 1999.

[20] ——, "Nonpathological Lyapunov functions and discontinuous Carathéodory systems," *Automatica*, vol. 42, no. 3, pp. 453–458, Mar. 2006.

[21] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 4, Oct. 2003, 3546–3551 vol.3.

[22] J. O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, Jun. 1992.

[23] D. Panagou, D. M. Stipanovi, and P. G. Voulgaris, "Multi-objective control for multi-agent systems using lyapunov-like barrier functions," in *52nd IEEE Conference on Decision and Control*, Dec. 2013, pp. 1478–1483.

[24] A. F. Filippov, "Differential equations with discontinuous right-hand side," *Mat. Sb. (N.S.)*, vol. 51(93), pp. 99 –128, 1960.

[25] V. Lakshmikantham and S. Leela, *Differential and Integral Inequalities: Theory and Applications: Volume I: Ordinary Differential Equations*. Academic press, 1969, vol. 55.

[26] B. Morris, M. J. Powell, and A. D. Ames, "Sufficient conditions for the lipschitz continuity of qp-based multi-objective control of humanoid robots," in *52nd IEEE Conference on Decision and Control*, Dec. 2013, pp. 2920–2926.

[27] J. Cortés, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[28] A. Bacciotti and F. Ceragioli, "Nonsmooth optimal regulation and discontinuous stabilization," in *Abstract and Applied Analysis*, Hindawi Publishing Corporation, vol. 20, 2003, pp. 1159–1195.

[29] P. Glotfelter, I. Buckley, and M. Egerstedt, "Hybrid nonsmooth barrier functions with applications to provably safe and composable collision avoidance for robotic systems," *IEEE Robotics and Automation Letters*, pp. 1–1, 2019.

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

[31] M. Hoy, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robotica*, vol. 33, no. 3, 463497, 2015.

[32] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robotics and Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

[33] I. M. Mitchell, S. Kaynama, M. Chen, and M. Oishi, "Safety preserving control synthesis for sampled data systems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 63 –82, 2013, Special Issue related to IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 12).

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

[35] M. S. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Trans. Autom. Control*, vol. 43, no. 4, pp. 475–482, 1998.

[36] J. P. Hespanha, "Uniform stability of switched linear systems: Extensions of LaSalle's invariance principle," *IEEE Trans. Autom. Control*, vol. 49, no. 4, pp. 470–482, Apr. 2004.

[37] R. Olfati-Saber, "Near-identity diffeomorphisms and exponential epsilon-tracking and epsilon-stabilization of first-order nonholonomic SE(2) vehicles," in *IEEE Proc. Amer. Control Conf.*, vol. 6, 2002, pp. 4690–4695.

[38] V. Shapiro, "Semi-analytic geometry with r-functions," *Acta Numerica*, vol. 16, 239303, 2007.

[39] A. Balestrino, A. Caiti, E. Crisostomi, and S. Grammatico, "R-composition of Lyapunov functions," in *2009 17th Mediterranean Conference on Control and Automation*, Jun. 2009, pp. 126–131.

[40] S. Scholtes, *Introduction to piecewise differentiable equations*. Springer Science & Business Media, 2012.

[41] J.-P. Aubin and H. Frankowska, *Set-valued analysis*. Springer Science & Business Media, 2009.

[42] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[43] K. Fathian, D. I. Rachinskii, T. H. Summers, and N. R. Gans, "Distributed control of cyclic formations with local relative position measurements," in *IEEE Conference on Decision and Control (CDC), 2016 (to appear)*, 2016.

[44] H. Park and S. Hutchinson, "An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 358–365.

[45] Y. Igarashi, T. Hatanaka, M. Fujita, and M. W. Spong, "Passivity-based attitude synchronization in se (3)," *IEEE transactions on control systems technology*, vol. 17, no. 5, pp. 1119–1134, 2009.

[46] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Proceedings of the 38th IEEE Conference on Decision and Control*, vol. 3, Dec. 1999, 2655–2660 vol.3.