A Nonsmooth Approach to Controller Synthesis for Boolean Specifications

Paul Glotfelter, Jorge Cortés, *Fellow, IEEE*, and Magnus Egerstedt, *Fellow, IEEE*

***Abstract*—Robotic systems have been proposed as a solution to a wide array of problems, from autonomous warehousing to precision agriculture. Yet these systems typically require satisfaction of multiple constraints, such as collision avoidance and connectivity maintenance, while completing their primary objectives. Many such problems may be decomposed into stability and invariance criteria (e.g., monitor crop patches while avoiding collisions), and this article utilizes Lyapunov and barrier functions to encode stability and invariance, respectively. Barrier functions provide constraint-satisfaction guarantees, and prior results have established a Boolean composition and controller-synthesis framework for these objects via nonsmooth analysis. However, this past work has yet to address Boolean composition of Lyapunov functions directly and does not apply to all Boolean expressions. This article resolves these issues by providing a general method to encode Boolean expressions of Lyapunov or barrier functions. Moreover, this article develops an associated controller-synthesis algorithm that yields discontinuous yet validating controllers with respect to these Boolean expressions. Experimental results show the efficacy of this article in a precision-agriculture scenario, where a robot swarm must visit crop patches while avoiding collisions.**

***Index Terms*—Collision avoidance, Lyapunov methods, multiagent systems, robustness.**

I. Introduction

ROBOTIC systems are increasingly entering human-occupied spaces, from factory floors to city highways. These systems provide a number of benefits; however, they also introduce interaction-based complexity in that they typically require satisfaction of multiple constraints, such as speed limits, collision avoidance, or staying in connectivity range. Moreover, such systems may also have objectives that must be satisfied in the context of these constraints, like completing a delivery or monitoring a crop patch.

As an example, consider a precision-agriculture scenario in which a series of robots must dynamically monitor crops patches in a field (e.g., as in [1]). In this scenario, each patch must be autonomously visited by a robot, and the robots must avoid collisions with each other and potential obstacles in the field. Moreover, the location or number of these patches may change, altering the problem formulation. Thus, a framework for this application must be able to synthesize an effective control law without requiring vast theoretical modifications in the face of such changes. That is, an applicable controller-synthesis framework must address both objectives and constraints and be system agnostic, provably correct, and usable in real time. This article develops such a framework and addresses a similar precision-agriculture scenario in an experiment with real robots.

For many systems, the abovementioned constraints (e.g., collision avoidance, connectivity maintenance) and others can be encoded as forward-set-invariance requirements, and recent results have shown that barrier functions can guarantee this property while remaining conducive to controller synthesis [2]–[6]. These objects represent a valuable theoretical and practical tool, seeing success in applications ranging from teams of quadrotors to remote-access robotics testbeds [7], [8]. The versatility of barrier functions stems from the fact that they are not formulated with respect to particular systems and provide a general methodology to formally guarantee safety. Additionally, prior work demonstrates that safe controllers may be quickly synthesized. For example, the work in [8] synthesizes controllers at 100 Hz for an 80-dimensional ensemble system of differential-drive robots. These observations justify the adoption of barrier functions to encode constraints.

From a complementary perspective, Lyapunov functions have long and ubiquitously been used for stability analysis. This article uses set-stable nonsmooth Lyapunov functions (NLFs) to encode objectives, and the formulation is primarily based on [9]–[12]. Together, Lyapunov functions and barrier functions encode stability and invariance criteria, respectively, but crafting a single such function to encode a complex, evolving robotics objective, or constraint may be tedious. In this context, logical composition provides a powerful simplification tool, allowing complex behaviors to be created from simple propositions. Boolean logic represents one such method; however, Boolean operations such as conjunction and disjunction inevitably introduce nonsmoothness when encoded with $\min$ and $\max$ operations.

Manuscript received February 13, 2020; revised June 24, 2020; accepted October 17, 2020. Date of publication November 3, 2020; date of current version November 4, 2021. This work was supported by the National Science Foundation under Grant ECCS-1531195 and Grant ECCS-1917177. Recommended by Associate Editor C. M. Kellett. *(Corresponding author: Paul Glotfelter.)*

Paul Glotfelter and Magnus Egerstedt are with the Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: pglotfel@gmail.com; magnus@ece.gatech.edu).

Jorge Cortés is with the Department of Mechanical and Aerospace Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: cortes@ucsd.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2020.3035467

In our previous work, we have introduced nonsmooth barrier functions (NBFs), extending barrier functions to the nonsmooth case [13] and introducing controller-synthesis methods for them [14]. Section I-A compares and contrasts related methods, outside the area of barrier or Lyapunov functions, to this article.

This work's first main result elucidates the calculation of discontinuous but validating controllers for NBFs and NLFs represented by piecewise-smooth (PC^r) functions [15]. These results utilize discontinuous dynamical systems and nonsmooth analysis, as studied in [10], [11], [16]–[18]. Specifically, this result establishes a link between the PC^r functions of [15] with the generalized gradient of [17] to develop a new theory. To analyze PC^r functions, one typically analyzes a set of continuously differentiable functions, 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 article derives a general method for synthesizing discontinuous controllers for PC^r functions that represent an NBF or a NLF. Moreover, this article 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 article is a controller-synthesis framework that permits the combination of 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 article provides a system-independent algorithm to calculate such index sets. To show the practicality of these results, a practical 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 interrobot collisions in real time. Boolean NLFs and NBFs capture the objectives and constraints for the experiment. Section I-A includes a discussion of related work in the context of these stated contributions.

The organization of this article follows. Section II notes background material required for this article including the system of interest, NBFs, nonsmooth analysis, PC^r functions, and discontinuous dynamical systems. Section III begins the main contributions for this article, providing results on a class of index functions for PC^r functions and extending the results of [14]. Next, Section IV 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 V 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 VI showcases an experiment that uses the controller-synthesis framework of Section IV and the algorithm proposed in Section V to produce a safe and effective controller in a precision-agriculture scenario.

A. Related Work

Related work to the contributions proposed by Section I lies in two main areas: set invariance and logical composition of functions, and nonsmooth-analysis techniques.

1) Set-Invariance Methods: Barrier functions are not the only set-invariance method, and many other such methods exist, such as [19]–[22]. These approaches range from potential functions to PDE-based approaches to compute reachable sets. Herbert *et al.* [19] formulated a modular obstacle-avoidance framework for general dynamical systems, yet this framework is limited to the particular application of obstacle avoidance. Mitchell *et al.* [20] relies on the solution of PDEs to guarantee set invariance, and in practice, these computations can be prohibitively costly. Finally, potential functions [21] are widely used, 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. Similarly, Hoy *et al.* [22] presented a variety of collision-avoidance methods that do not generalize beyond solving navigation for mobile robots in a cluttered environment.

In general, two facts separate barrier functions from other methods in the context of this article and prior efforts. First, barrier functions are mathematically agnostic in that the formal mathematical description of their properties is independent from any particular system. Second, pointwise satisfying a particular inequality involving barrier functions across the state space produces a global invariance result. This last 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.

2) Logical Composition of Functions: It is worth noting that prior work has considered Boolean logic for real-valued functions [23], [24]. However, this literature 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 [13] and [14] becomes relatively straightforward from a synthesis perspective, at the expense of requiring nonsmooth analysis. The major limitations of [13] and [14] are that they do not apply to general Boolean expressions and do not consider NLFs in controller synthesis. This article resolves both issues by extending the formulated controller-synthesis framework to address inductive Boolean compositions of NBFs and NLFs.

With respect to barrier functions, recent work [25] uses Lyapunov-like barrier functions in the context of multiagent control, which are noticeably different in formulation to the barrier functions here. Moreover, their method of Boolean composition is a norm-like function similar to the approaches described previously. In comparison to this article, the results in [25] do not apply to differential inclusions and establish only weak forward

invariance, meaning that in the case of multiple solutions, only one is guaranteed to remain in the considered set. By contrast, the guarantees of this article apply to all solutions. Furthermore, this article provides a generic set of novel mathematical objects for synthesizing controllers with respect to Boolean compositions of functions, whereas the approach in [25] is tailored to a particular application.

3) Analysis of Solutions Along Nonsmooth Functions: Some of the new theory developed in this article aids in the analysis of PC^r functions along solutions to differential inclusions, particularly those resulting from the Filippov regularization. Usually, when considering this branch of nonsmooth analysis, one must typically use different (though analogous) tools than in the smooth case, such as so-called set-valued Lie derivatives [16]. However, many of these techniques require explicit computation of the Filippov operator or exact calculation of the generalized gradient [10], yet exact calculation of these objects can be tedious due to their limit-point representation. That is, the behavior of the system must be considered in a neighborhood around each point rather than just pointwise. This calculation presents an even larger burden in the case of optimization-based controller synthesis, as these objects cannot be calculated and held in memory, at least according to their definitions. The generalized gradient admits a convenient calculus; however, with respect to the dynamics, a calculus for the Filippov operator is less useful because it depends heavily on the particular dynamical system. Moreover, as the generalized-gradient calculus yields a superset of the generalized gradient, certain simplifying set-valued Lie derivatives cannot be applied [16].

Exacerbating the issue, one typically creates the controller (analytically or numerically) as a function of the generalized gradient, much like when creating a controller via the usual gradient of a Lyapunov function in the smooth sense, complicating the instantiation of the Filippov operator. Accordingly, there is a need for analysis techniques that allow one to circumvent the explicit calculation of these objects while retaining their provable guarantees. Some techniques exist for simplifying the calculus of Filippov regularizations, such as assuming a piecewise-smooth vector field [16], but this assumption is quite restrictive at large and introduces the tradeoff of proving that each particular vector field satisfies this property.

Toward this end, Glotfelter *et al.* [14] proposed a mathematical object that represents the generalized gradient by a finite set of known, readily computed points for a certain class of functions. Moreover, it also provides a method by which controllers synthesized with respect to this new object do not need to be Filippov regularized explicitly, a major leap forward from existing methods. The application of this technique requires no additional assumptions on the dynamics. Unfortunately, this new object applies to a restrictive class of functions along which trajectories are being considered (e.g., the Lyapunov or barrier function). Similarly, Della Rossa *et al.* [26] considered a comparable problem to [14], avoiding the calculation of the Filippov regularization; however, as in [14] and [26] applies to a restrictive class of functions and, differently from [14], requires the assumption of a piecewise-defined vector field.

This article extends [14], providing new nonsmooth-analysis techniques and constructive controller-synthesis results for a useful class of NLFs and NBFs that is closed under Boolean composition. Moreover, the resulting closed-loop systems need not be explicitly Filippov regularized, which significantly simplifies their practical application, and this circumvention of Filippov regularization requires no restrictive assumptions on the dynamics.

II. Background Material

This section contains background material, introducing the system of interest, nonsmooth analysis, and NBFs. We consider control-affine systems with potentially discontinuous control inputs and, as a result, introduce the theory of discontinuous dynamical systems. We also present nonsmooth analysis techniques necessary to formulate and validate NBFs.

A. Notation

The symbol $\times$ denotes the Cartesian product. For $k > 0$, the symbol $[k]$ indicates the set $\{1, \cdots, k\}$. The notation $\mathbb{R}_{\geq 0}$ represents the set of nonnegative real numbers; $a.e.$ symbolizes almost everywhere in the sense of Lebesgue measure. The operation co corresponds to the convex hull of a set. A function $\alpha : \mathbb{R} \to \mathbb{R}$ is extended class-$\mathcal{K}$ if α is continuous, strictly increasing, and $\alpha(0) = 0$. An extended class-$\mathcal{K}$ function is class-$\mathcal{K}$ 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, decreasing, and $\lim_{s\to\infty} \beta(r, s) = 0$. For a set A, $|A|$ means its cardinality, and 2^A indicates its powerset.

B. System of Interest

This article considers control-affine systems of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t)),\ x(0) = x_0 \tag{1}$$

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. This assumption covers a wide class of systems, as most robotic systems (e.g., differential-drive robots, quadcopters) are control affine. An important point is that we avoid 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 article pertains to controller synthesis, it assumes continuity of f and g. Because u is potentially discontinuous, solutions to (1) may not exist. Fortunately, Filippov's operator maps (1) into a differential inclusion to which solutions exist.

Definition II.1 ([11, Th. 1]): The Filippov operator $K[f + gu] : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ with respect to (1) at $x' \in \mathbb{R}^n$ is

$$K[f + gu](x') = \text{co}\left\{\lim_{i\to\infty} f(x_i) + g(x_i)u(x_i) : x_i \to x,' x_i \notin \bar{S} \cup S\right\}$$

where $\bar{S}$ is a fixed zero-Lebesgue-measure set and S is any set of zero Lebesgue measure.

Remark II.2: The map of limit points $L : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ defined by

$$L[f+gu](x') = \{\lim_{i\to\infty} f(x_i) + g(x_i)u(x_i) : x_i \to x,' x_i \notin \bar{S} \cup S\}$$

becomes useful later in this article. Note that for every $x' \in \mathbb{R}^n$, $K[f+gu](x') co L[f+gu](x')$. •

A general differential inclusion is formulated as

$$\dot{x}(t) \in F(x(t)),\ x(0) = x_0 \tag{2}$$

where $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is a nonempty, compact-, convex-valued map that is upper semicontinuous. A set-valued map $F : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is upper semicontinuous if for every $x' \in \mathbb{R}^n$, $\epsilon > 0$ there exists $\delta > 0$ such that

$$y \in B(x,' \delta) \Rightarrow F(y) \subset F(x') + B(0,\epsilon).$$

Sometimes, set-valued maps are written $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$, which has the same meaning as in this article.

These conditions ensure that Carathéodory solutions to (2) exist. A Carathéodory solution is an absolutely continuous function $x : [0, t_1] \to \mathbb{R}^n$ such that

$$\dot{x}(t) \in F(x(t)), x(0) = x_0 \tag{3}$$

almost everywhere on $[0, t_1] \ni t$, and $x(0) = x_0$. Note that $t_1 > 0$ and that t_1 may depend on each particular solution. In general, Carathéodory solutions to (2) exist, under these regularity conditions, but are not unique. For this article, an important fact is that Filippov's operator $K[f+gu]$ satisfies the aforementioned conditions. That is, the differential inclusion

$$\dot{x}(t) \in K[f+gu](x(t)), x(0) = x_0 \tag{4}$$

has Carathéodory solutions [9], [16]. Such solutions are sometimes referred to as Filippov solutions to (1). For extensive coverage of discontinuous dynamical systems, see [16].

The usage of Filippov regularization, rather than the similar Krasovskii regularization (e.g., see [9]), is motivated by prior work as well as the result in Definition II.1. Indeed, within the theoretical results of this article, one may immediately replace the usage of Filippov regularization with Krasovskii regularization. Sometimes Filippov regularization is denoted by $F[f+gu]$; however, to avoid confusion with (3), this article utilizes the notation in [11].

C. Nonsmooth Analysis

Previous work provides a plethora of methods for analyzing nonsmooth functions. In the case that the given nonsmooth function is at least locally Lipschitz, the generalized gradient is a well-understood object with an extensive calculus. Moreover, in the case of Boolean composition (i.e., composition with $\min$ and $\max$ operators), one can readily find a superset of the generalized gradient in terms of the component functions. The formal definition of the generalized gradient is as follows.

Definition II.3 ([17, Th. 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

$$\partial_c h(x') = \text{co}\left\{\lim_{i\to\infty} \nabla h(x_i) : x_i \to x,' x_i \notin \Omega_h \cup S\right\}$$

where Ω_h is the Lebesgue-zero-measure set where h is nondifferentiable.

Note that, per Definition II.3, the generalized gradient $\partial_c h : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ is a set-valued map. Another useful result provides a chain rule for locally Lipschitz functions. Note that the closure operation from [17, Th. 2.3.9] may be omitted (see the proof of the referenced theorem) for finite-dimensional spaces. Moreover, for two sets $A \subset \mathbb{R}^{n\times m}$, $B \subset \mathbb{R}^{m\times p}$, the set-valued multiplication is

$$AB = \{ab : a \in A, b \in B\}.$$

Theorem II.4 ([17, Th. 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

$$\partial_c h(x') \subset \text{co}\left(\bigtimes_{k=1}^{m} \partial_c g^k(x')\right) \partial_c f(g(x'))$$

where g^k denotes the kth component function of g.

Proposition II.5 ([17, Proposition 2.3.1]): Let $h : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function. For any scalar s one has

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

The main advantage of the generalized gradient is that it permits analysis 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 (2), we have

$$\frac{d}{dt}\left(h(x(t))\right) \geq \min \partial_c h(x(t))^{\top} F(x(t)) \tag{5}$$

almost everywhere on $[0, t_1] \ni t$. The map $\partial_c h(\cdot)^{\top} F(\cdot)$ is sometimes referred to as a set-valued Lie derivative (e.g., see [16]). Explicitly, the set-valued product is the set

$$\partial_c h(\cdot)^{\top} F(\cdot) = \{a^{\top} b : a \in \partial_c h(\cdot), b \in F(\cdot)\}$$

but, for brevity, we use the relaxed notation in (5).

D. Nonsmooth Barrier Functions

This section contains background material on NBFs. For brevity, NBFs are formulated with respect to the general differential inclusion in (3) with the understanding that these results also apply to the controlled system in (1) via Filippov's operator (4).

The formulation of barrier functions is as follows. Given a function $h : \mathbb{R}^n \to \mathbb{R}$, the safe set is defined as

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

That is, $\mathcal{C}$ is the super-zero level set to h. The goal becomes to ensure forward invariance of $\mathcal{C}$ with respect to (3). This article considers a set $\mathcal{C}$ to be forward invariant with respect to a differential inclusion (3) if

$$x_0 \in \mathcal{C} \Rightarrow x(t) \in \mathcal{C}, \forall t \in [0, t_1]$$

for every Carathéodory solution $x : [0, t_1] \to \mathbb{R}^n$. A valid NBF is defined as follows.

Definition II.6: A locally Lipschitz function $h : \mathbb{R}^n \to \mathbb{R}$ is a valid NBF for (3) if, for each $x_0 \in \mathcal{C}$, there exists a class-$\mathcal{KL}$ function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that

$$h(x(t)) \geq \beta(h(x_0), t)$$

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

The abovementioned definition 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. Note that, cf., [13], satisfying the differential inequality

$$\dot{h}(x(t)) \geq -\alpha(h(x(t))), \ a.e.\ t \in [0, t_1] \tag{6}$$

for every Carathéodory solution and for some locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$, ensures that h is a valid NBF.

Toward controller synthesis, we should be able to verify (6) without explicitly calculating the time derivative of $h \circ x$ for every Carathéodory solution. This is precisely the strength of (5), as it allows one to check the validity of the NBF spatially (i.e., over $\mathbb{R}^n$), rather than computing the time derivative explicitly. Combining (5) and (6), the next results follow.

Theorem II.7: [13, Th. 2] Let $h : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function. 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 (3).

Definition II.8 ([14, Definition 4]): Let $h : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function. The function h is a valid Closed-loop NBF (CNBF) for (1) if and only if there exists a locally Lipschitz extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ and a measurable and locally bounded function $u : \mathbb{R}^n \to \mathbb{R}^m$ such that

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

Note that Definition II.8 differs in terminology from [14], which is for symmetry with upcoming definitions; the mathematical relationship is equivalent. A CNBF automatically satisfies the requirements for a valid NBF via Theorem II.7 in the sense that $\mathcal{C}$ is forward invariant with respect to (4), but some difficulty arises in actually finding such a controller, as the Filippov operator must be applied to it. This situation creates issues for controller synthesis, because the Filippov operator cannot feasibly be calculated in real time. Moreover, for a controlled system

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

does not imply that

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

so the generalized gradient cannot be directly used for controller synthesis. This implication does not hold because, at any point, $\partial_c h(x')$ and $K[f + gu](x')$ are collections of limit points and represent discontinuous objects. The work in [14] 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. However, this object is limited, as it only applies to Boolean CNBFs consisting of exclusively $\wedge$ or $\vee$ operations. Section II-E lifts this restriction.

E. Piecewise-Differentiable Functions

This article extends the results in [14] by formulating Boolean NBFs as piecewise differentiable (PC^r) functions as in [15], and one of the main results of this article, in Section III, 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.

Definition II.9 ([15, p. 91]): A function $h : \mathbb{R}^n \to \mathbb{R}$ is piecewise differentiable (PC^r) if for every $x' \in \mathbb{R}^n$ there exists an open neighborhood N of x' and a set of C^r functions $\{h_i : i \in K_h\}$, with K_h a finite index set, such that h is a continuous selection of the functions h_i on N. That is, h is continuous and $h(y) \in \{h_i(y) : i \in K_h\}, \forall y \in N$. The function h is piecewise linear (PL) if this definition holds with linear functions $h_i(y) = a_i^\top y$.

Note that, in Definition II.9, the finite set of component functions may vary based on the particular point in the domain. Here, we only consider PC^r functions whose component functions are defined over the entire domain. That is

$$h(y) \in \{h_i(y) : i \in K_h\}, \forall y \in \mathbb{R}^n$$

for some finite set of functions denoted by K_h, which is fixed. In the context of this article, this assumption is not restrictive and does not inhibit the generality of the proposed results for controller synthesis. Throughout this article, we make the assumption that $r > 0$ (i.e., that all the component functions are at least C^1).

We next relate PC^r functions to the generalized gradient defined in Section II-C. 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

$$I_h^a(x') = \{i \in K_h : h_i(x') = h(x')\} \tag{7}$$

plays a role. 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

$$I_h^e(x') = \{i \in K_h : x' \in \text{cl}(\text{int}(\{y : h_i(y) = h(y)\}))\} \tag{8}$$

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. To be in I_h^e, a function must be active on a sequence that converges in the interior of the active set. As a consequence, $I_h^e(\cdot) \subset I_h^a(\cdot)$ always. However, I_h^e may be more difficult to calculate, whereas $I_h^a(\cdot)$ remains straightforward to calculate, assuming the component functions are known.

Tying these theories together, the following result links the essentially active set I_h^e to the generalized gradient, and this relationship is critical for the forthcoming results.

Proposition II.10 ([15, 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

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

III. Evaluating Set-Valued Lie Derivatives With Encapsulating Index Functions

This section begins with the main results of this article, which relate to extending the work in [14] to PC^r functions through particular index sets. In turn, this development extends the controller-synthesis framework for Boolean CNBFs to general Boolean expressions.

A. Encapsulating Index Functions

As seen in Proposition II.10, index functions unite PC^r functions with the generalized gradient. This section presents a class of index functions that 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 III.1: 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 $h_i(x') = h(x')$ for all $i \in I_h(x')$ and all $x' \in \mathbb{R}^n$.

Note that, by Proposition II.10, encapsulating index functions always exist. Moreover, the requirement that $h_i(x') = h(x')$ for all $i \in I_h(x')$ is nonrestrictive, because indices outside this index set are superfluous. The goal then becomes to find index functions that are relatively easy to calculate and satisfy this definition. Because this article studies these index functions extensively, the following notation:

$$\{\nabla h_i(\cdot) : i \in I(\cdot)\} = \left\{\nabla h_i \atop {i \in I}\right\}(\cdot)$$

becomes useful for brevity in later results. The following example illustrates Definition III.1.

Example III.1: Let $h : \mathbb{R} \to \mathbb{R}$ be defined as $h(x') = |x'|$, which is equivalent to $\max\{-x,' x'\}$. As such, h is a PC^r function with component functions $h_1, h_2 : \mathbb{R} \to \mathbb{R}$ defined as $h_1(x') = -x'$, $h_2(x') = x'$. Accordingly, $K_h = \{1, 2\}$. Note that the generalized gradient for h is

$$\partial_c h(x') = \{-1\}, x' \in (-\infty, 0)$$
$$\partial_c h(x') = [-1, 1], x' = 0$$
$$\partial_c h(x') = \{1\}, x' \in (0, \infty).$$

Then, an encapsulating index function for h is $I_h : \mathbb{R} \to 2^{K_h}$ defined as

$$I_h(x') = \{1\}, x' \in (-\infty, 0)$$
$$I_h(x') = \{1, 2\}, x' = 0$$
$$I_h(x') = \{2\}, x' \in (0, \infty).$$

•

Recall the problem discussed in Section II-D in relation to synthesizing controllers via the generalized gradient. In this case, encapsulating index functions, as in Definition III.1, still encounter this fundamental limitation, because encapsulating index functions only capture indices that create the generalized gradient. As such, it becomes necessary to further extend the index-function developments to make sure that we can rely on the indices uniformly around any particular point. 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 defines locality-capturing index functions.

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

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

A locally encapsulating index function 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 the calculation of the Filippov operator. The following example demonstrates a locally encapsulating index function.

Example III.2: Let $I_h : \mathbb{R} \to 2^{K_h}$ be defined as in Example III.1. Then, a locally encapsulating index function $I_h^l : \mathbb{R} \to 2^{K_h}$ for I_h is

$$I_h^l(x') = \{1\}, x' \in (\infty, -\epsilon)$$
$$I_h^l(x') = \{1, 2\}, x' \in [-\epsilon, \epsilon]$$
$$I_h^l(x') = \{2\}, x' \in (\epsilon, \infty)$$

for any fixed $\epsilon > 0$. •

An important point is that PC^r functions admit locally encapsulating index functions, as we establish next. As with encapsulating index functions, the primary consideration for picking a specific locally encapsulating index function depends on use-specific circumstances (e.g., I^e versus I^a).

Proposition III.3: If $h : \mathbb{R}^n \to \mathbb{R}$ is a PC^r function, then there exists a locally encapsulating index function $I_h^l : \mathbb{R}^n \to 2^{K_h}$ for h.

Proof: Consider the index function $I_h^l : \mathbb{R}^n \to 2^{K_h}$

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

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 (8).

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

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

and

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

Let $\delta = \min\{\delta_1, \delta_2\}$. It remains to be shown that $i \in I_h^l(y)$ 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 $i \in I_h^l(y)$. Thus, I_h^l is a locally encapsulating index function for h via I_h^e. ■

Remark III.4: Note that the proof of Proposition III.3 also shows that I_h^l is a locally encapsulating index function for h with respect to the active index function I_h^a defined in (7). •

B. Evaluation of Set-Valued Lie Derivatives

The next theorem represents the first main result of this article, demonstrating that locally encapsulating index functions are indeed the correct type of index function to navigate the issues caused by employing Filippov's operator.

Theorem III.5: Let $\hat{f} : \mathbb{R}^n \to \mathbb{R}^n$ be measurable and locally bounded and $\alpha : \mathbb{R} \to \mathbb{R}$ continuous. Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function, and let $I_h^l : \mathbb{R}^n \to 2^{K_h}$ be a locally encapsulating index function for h. If

$$\min\left\{\underset{i\in I_h^l}{\nabla} h_i\right\}(x')^\top \hat{f}(x') \geq -\alpha(h(x')), \forall x' \in \mathbb{R}^n$$

then

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

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

Proof: Let $x' \in \mathbb{R}^n$. It must be shown that

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

and by application of [13, Lemma 3], it suffices to show that

$$\min\left\{\underset{i\in I_h}{\nabla} h_i\right\}(x')^\top L[\hat{f}](x') \geq -\alpha(h(x'))$$

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

$$l = \lim_{j\to\infty} \hat{f}(x_j).$$

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

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

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

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

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

$$\nabla h_i(x_j)^\top \hat{f}(x_j) + \alpha(h(x_j)) \geq 0, \forall j$$

so

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

as well, implying that

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

and yielding the desired result. ■

Theorem III.5 becomes particularly useful in the context of controller synthesis. Specifically, including a locally encapsulating index function as a constraint in an optimization program yields a validating but potentially discontinuous controller. The proof of Proposition III.6 is omitted, as it follows directly from Theorem III.5.

Proposition III.6: Let $h : \mathbb{R}^n \to \mathbb{R}$ be a candidate NBF, and let f, g be as in (1). If there exists a locally encapsulating index function $I_h^l : \mathbb{R}^n \to 2^{K_h}$ 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

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

then, h is a valid CNBF for (1).

IV. Boolean Composition for Lyapunov and Barrier Functions

This section builds on the concept of locally encapsulating index functions to formulate the Boolean composition syntax and the class of Boolean expressions considered in this article. The first results pertain to formulating NLFs in the present context and some conditions under which they may be composed with Boolean operators. Then, we discuss the syntax and semantics of Boolean composition for NLFs and NBFs.

A. Closed-Loop NLFs

NLFs have been studied in great detail, including [9]–[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, as it is amenable to Boolean composition.

Definition IV.1: A locally Lipschitz function $V : \mathbb{R}^n \to \mathbb{R}$ is a candidate NLF if and only if, with

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

$V(x') > 0$, for all $x' \notin A$, A is nonempty, and

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

is bounded for every $a \in \mathbb{R}_{\geq 0}$.

Remark IV.2: For a candidate NLF V, 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 IV.3: A candidate NLF $V : \mathbb{R}^n \to \mathbb{R}$ is a valid closed-loop NLF (CNLF) for (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

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

Note that Definition IV.3 creates some mathematical differences from similar existing literature (e.g., that of control Lyapunov functions). For example, Definition IV.3 assumes the existence of a measurable and locally bounded controller; whereas, in the literature, one typically assumes a pointwise decrease condition. That is, pointwise, there is some value of $u \in \mathbb{R}^m$ that generates a decrease. Then, one must prove that a stabilizing controller exists. This work focuses on constructively synthesizing validating controllers with respect to index sets, as previously seen for CNBFs. As such, Definition IV.3 strives to indicate this pursuit. Moreover, the pointwise condition makes a statement about a fixed value $u \in \mathbb{R}^m$, complicating application of the Filippov operator, which requires a function. Accordingly, one may think of Definition IV.3 as assuming the existence of controller such that the candidate NLF is a valid NLF for the closed-loop system. Some future work lies in the investigation of a pointwise condition for Definition IV.3 with respect to an appropriate index set.

As shown by Definitions IV.1 and IV.3, this article focuses on set stability for NLFs. The reason for this is as follows. Boolean composition of NLFs entails intersections and unions of the zero level set. As such, restricting NLFs to a point limits the generality of Boolean composition, because union and intersection would only be able to generate NLFs for a point.

Theorem IV.4: Let $V : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ be a candidate NLF. If V is a valid CNLF for (1), then the set A is uniformly globally asymptotically stable with respect to (4).

We omit the proof of Theorem IV.4 for brevity, as it is similar to many prior proofs of similar results; however, we do provide a discussion of the required tools. The existence of a controller such that A is (uniformly globally) asymptotically stable is also referred to as stabilizability (though, typically, stabilizability refers to the ability to find such a controller). In this article, a valid CNLF explicitly requires the existence of such a controller. Accordingly, as a stabilizing controller immediately exists, one may consider the closed-loop system with respect to this particular controller.

Remark IV.5: In the case that the boundedness of level sets in Definition IV.1 does not hold, then this definition may be modified to require an open set $D \subset \mathbb{R}^n$ containing A where the abovementioned inequality holds. In this case, a local stability result follows. •

Definitions IV.1 and IV.3 ensure uniform global asymptotic stability to the set A in the sense that

$$\|x(t)\|_A \leq \beta(\|x_0\|_A, t) \tag{9}$$

for every Carathéodory solution $x : [0, t_1] \to \mathbb{R}^n$ to $K[f + gu](\cdot)$, where

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

is the usual point-to-set distance and $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class-$\mathcal{KL}$ function. Note that, for this discussion, we assume that the system has been equipped with the validating controller and consider the closed-loop system. For the discussion below, consider this particular controller as u_s.

This result of asymptotic stability follows from [9], [10], [12], [27] by noting that one can obtain (cf., [9, Lemma 2.5] or [27, Lemma 4.3]), class-$\mathcal{K}_\infty$ functions $\alpha_1, \alpha_2 : \mathbb{R}_{\geq 0} \to \mathbb{R}$ such that

$$\alpha_1(\|x\|_A) \leq V(x) \leq \alpha_2(\|x\|_A). \tag{10}$$

From Definition IV.3, every Carathéodory solution to $K[f + gu_s](\cdot)$

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

where $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class-$\mathcal{KL}$ function [12, Lemma 4.2]. Then, applying (10) yields (9). Note that this discussion depends on equipping the system with the stabilizing controller.

The next result studies how candidate NLFs can be formed through $\max$ and $\min$ operations, a fact that becomes particularly useful when discussing Boolean composition of NLFs later. Its proof is straightforward, and we omit it for brevity.

Proposition IV.6: Let $V_1, V_2 : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ be candidate NLFs. Then, $V_{\min} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ defined as

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

is a candidate NLF. If, in addition, $A_1 \cap A_2 \neq \emptyset$, then $V_{\max} : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ defined as

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

is a candidate NLF.

Remark IV.7: The assumption that $A_1 \cap A_2$ is nonempty is necessary in the case of $\max$, as the zero level set of V must be nonempty, and this property cannot be otherwise guaranteed. The same assumption is not true for $\min$, as $A = A_1 \cup A_2$ and is nonempty by candidacy of V_1 and V_2. •

In a parallel fashion to Proposition III.6, the next result utilizes Theorem III.5 to address controller synthesis for NLFs via locally encapsulating index functions.

Proposition IV.8: Let $V : \mathbb{R}^n \to \mathbb{R}$ be a candidate NLF, and let f, g be as in (1). If there exists a locally encapsulating index function $I_V^l : \mathbb{R}^n \to 2^{K_V}$ 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

$$\max \left\{ \bigtriangledown_{i \in I_V^l} V_i \right\} (x')^\top (f(x') + g(x')u(x'))$$
$$\leq -p(V(x')), \forall x' \in \mathbb{R}^n$$

then, V is a valid CNLF for (1).

B. Composition Syntax

This section details the Boolean composition syntax and resulting expressions considered in this work. Formally, a Boolean expression on a finite number of C^r component functions $f_1, \dots, f_k : \mathbb{R}^n \to \mathbb{R}$ is symbolically defined as

$$B[f_1, \cdots, f_k]. \tag{11}$$

The component functions may be NBFs or NLFs, and the given expressions are comprised of $\min$, $\max$, and $-$ operators. Specifically, this article considers Boolean NBFs inductively defined with the following syntax:

$$B := h_i \mid \neg B_1 \mid B_1 \wedge B_2 \mid B_1 \vee B_2 \tag{12}$$

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 Boolean NLFs is given as follows:

$$B := V_i \mid B_1 \wedge B_2 \mid B_1 \vee B_2$$

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 abovementioned syntax omits the negation operator $\neg$ and that the roles of $\min$ and $\max$ are swapped with respect to NBFs. We omit this operator due to the requirements of Definition IV.1 (i.e., positivity of the candidate NLF).

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 (12). The following example demonstrates this property.

Example IV.1: An example of a Boolean NBF is as follows. Let $h : \mathbb{R}^n \to \mathbb{R}$ be pointwise 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 abovementioned 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 (12). 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.

Proposition IV.9: If $B[f_1, \cdots, f_k] : \mathbb{R}^n \to \mathbb{R}$ is a Boolean expression with C^r component functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i \in [k]$, then $B[f_1, \cdots, f_k]$ is a PC^r function.

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

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

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

From this result, note that the Boolean composition of Boolean expressions is also PC^r, since the composite expression is a selection of the component functions of the component expressions. The next result describes a controller-synthesis method using locally encapsulating index functions.

Proposition IV.10: 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 $I_h^l : \mathbb{R}^n \to 2^{K_h}$ be a locally encapsulating index function for h, and let $I_V^l : \mathbb{R}^n \to 2^{K_V}$ be a locally encapsulating index function for V. Let $\alpha : \mathbb{R} \to \mathbb{R}$ be a locally Lipschitz extended class-$\mathcal{K}$ function and $p : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a continuous, positive-definite function. If $u^\star : \mathbb{R}^n \to \mathbb{R}^m$ defined as

$$u^\star(x') \in \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} u^\top A(x')u + u^\top b(x')$$
$$\text{s.t. } \min\left\{\underset{i \in I_h^l}{\nabla} h_i\right\}(x')^\top (f(x') + g(x')u) \geq -\alpha(h(x'))$$
$$\max\left\{\underset{i \in I_V^l}{\nabla} V_i\right\}(x')^\top (f(x') + g(x')u) \leq -p(V(x'))$$

with $A : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ continuous, pointwise 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.

This result is a direct consequence of Propositions III.6–IV.9, and its proof is omitted for brevity. In Proposition IV.10, 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 $O((|I_h^l(\cdot)| + |I_V^l(\cdot)| + m)^3)$ (i.e., cubic complexity in the number of constraints and decision variables). As such, it can typically be solved in real time, even for relatively large problems (e.g., [8]). The existence of solutions to this QP is predicated on the dynamics as well as the candidate NLF and NBF. In this case, uniqueness is guaranteed by the convexity of the constraint set and strict convexity of the objective function; though, existence is the key concern for this article.

Since Boolean NLFs or NBFs fall into the class of PC^r functions, controller-synthesis algorithms for Boolean expressions focus on finding an appropriate locally 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 instance, given a Boolean NBF, h, Proposition IV.9 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 III.3, a suitable locally encapsulating index function for I_h^e is

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

However, $I_h^l(\cdot)$ may capture too many component functions. The following example illustrates this point.

Example IV.2: Consider the Boolean NBF give in Example IV.1. Assume that, for a particular x', $h_1(x')$ is within $\epsilon > 0$ of $h_3(x')$ but $h_2(x')$ is much smaller than $h_3(x')$. Then, $(h_1 \wedge h_2)(x') = \min\{h_1(x'), h_2(x')\}$ is also much smaller than $h_3(x')$. Thus, intuitively, only 3 should be included in $I_h^l(x')$. However, using the previously noted I_h^l, both 1 and 3 would be included in $I_h^l(x')$. As such, choosing I_h^l in this manner may be too conservative. •

Given the point illustrated in Example IV.2, the following section explores recursive methods to calculate an appropriate locally encapsulating index functions for Boolean expressions.

V. Recursive Computation of Locally Encapsulating Index Functions for Boolean Expressions

Given the results in Sections III–IV, the goal becomes to calculate efficient and manageable locally encapsulating index functions. As such, this section presents a method for recursively calculating a locally 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.

A. Locally Encapsulating Index Functions for Compositions of PC^r Functions

We start this section by considering the composition of PC^r 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 II.10, the PC^r function in question maps from $\mathbb{R}^n$ to $\mathbb{R}$; instead, here, $g : \mathbb{R}^n \to \mathbb{R}^m$. As such, we consider g to be a continuous selection of PC^r functions $g_{i_j}^j : \mathbb{R}^n \to \mathbb{R}$ such that

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

For a given $g_{i_j}^j$, 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 $i \in K_{g_1} \times \cdots \times K_{g_m} = K_g$. As such, $g_{i_j}^j$ refers to a selection of the jth 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 result, one of the main results of this article, demonstrates a recursive method for calculating encapsulating index functions.

Theorem V.1: Let $g : \mathbb{R}^n \to \mathbb{R}^m$, $f : \mathbb{R}^m \to \mathbb{R}$ be PC^r functions, with $g = [g^1, \cdots, g^m]^\top$, 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, $k \in [m]$, respectively, then $I_h : \mathbb{R}^n \to 2^{K_f \times K_g}$ defined as

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

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 pointwise Cartesian product

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

The abovementioned 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

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

$$= \text{co}\{\nabla g_j(x')\nabla f_i(g_j(x')) : (i,j) \in I_f(g(x')) \times \bigtimes_{k=1}^{m} I_{g^k}(x')\}$$

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

To separate the index sets in the proof of Theorem V.1, the fact that $g(x') = g_j(x')$ for every $j \in \bigtimes_{k=1}^{m} I_{g^k}(x')$ must be utilized. When formulating a similar result for locally 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.

Theorem V.2: 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 $g = [g^1, \cdots, g^m]^\top$. If $I_f^l : \mathbb{R}^m \to 2^{K_f}$, $I_{g^k}^l : \mathbb{R}^n \to 2^{K_g}$, $k \in [m]$, are locally encapsulating index functions for f and each g^k, $k \in [m]$, then there exists a locally encapsulating index function $I_h^l : \mathbb{R}^n \to 2^{K_f \times K_g}$ for h such that, for all $x' \in \mathbb{R}^n$

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

where, for each $i \in K_f$, $a_i = \nabla f_i$.

Proof: Let $x' \in \mathbb{R}^n$. Because I_f^l is a locally 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

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

By continuity of g, let δ_2 be such that

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

Similarly, let δ_3^k be such that

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

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 V.1
$I_h : \mathbb{R}^n \to 2^{K_f \times K_g}$ 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 $y \in B(x,' \delta)$

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

As such

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

so $I_f^l \circ g \times I_g^l$ is a locally encapsulating index function for I_h. Moreover

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

showing the desired relation. ■

B. Locally Encapsulating Index Functions for Boolean Expressions

Here, we discuss the computation of locally encapsulating index functions for Boolean expressions. To apply Theorem V.2 to Boolean composition, we start by addressing the particular case of $\min$ and $\max$ operations. In fact, the next result shows that calculating a locally encapsulating index set for a Boolean expression admits a convenient format.

Proposition V.3: 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 ith 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 $I_{g^i}^l, i \in [m]$, are locally encapsulating index functions for each g^i and I_f^l is a locally encapsulating index function for f. Then, there exists a locally encapsulating index function $I_h^l : \mathbb{R}^n \to 2^{K_f \times K_g}$ for h such that

$$\{\nabla h_i\}_{i \in I_h^l}(x') = \left\{ \{\nabla g_{j_i}^i\}_{j_i \in I_{g^i}^l}(x') : i \in I_f^l(g(x')) \right\}.$$

Proof: Let $x' \in \mathbb{R}^n$, and let each $I_{g^j}^l$ and I_f^l be as assumed. Consequently, by application of Theorem V.2, there exists a locally encapsulating index function I_h^l such that

$$\begin{aligned} \{\nabla h_i\}_{i \in I_h^l}(x') &= \left(\bigtimes_{k=1}^{m} \{\underset{j_k \in I_{g^k}^l}{\nabla g_{j_k}^k}\}(x')\right)\{\underset{i \in I_f^l}{e_i}\}(g(x')) \\ &= \left\{ \{\nabla g_{j_i}^i\}_{j_i \in I_{g^i}^l}(x') : i \in I_f^l(g(x')) \right\} \end{aligned}$$

which follows because multiplication by any e_i leaves only the ith component function. ■

Remark V.4: Proposition V.3 applies to Boolean expressions including $\vee$ and $\wedge$, as in (12), 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 ith standard basis vector. Because

Algorithm 1: δ_ENCAPSULATING.

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

at any $x' \in \mathbb{R}^n$, max may be written as

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

for some $i \in [m]$. •

The next result deals with the case of negation. The proof follows directly from Proposition II.5.

Proposition V.5: Let $h : \mathbb{R}^n \to \mathbb{R}$ be a PC^r function. If $I^l_h : \mathbb{R}^n \to 2^{K_h}$ is a locally encapsulating index function for h, then, for any scalar s, I^l_h is a locally 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 V.6: Proposition V.5 may be used to calculate the negation of Boolean expressions B as in (12) by calculating a locally encapsulating index function for the negated expression. That is, if $B = \neg B_1$, then $I^l_{B_1}$ is also a locally encapsulating index function for B. However, each index $i \in I^l_{B_1}$ now refers to a negated component function. For convenience, the notation $-I^l_{B_1}$ refers to negated component functions but only for Boolean expressions. •

Using Theorem V.2 and Propositions V.3 and V.5, Algorithm 1 calculates a locally encapsulating index function for a Boolean composition as in (11). Importantly, Algorithm 1 does not change based on the particular Boolean expression or system under consideration. Note that, due to Remark V.4, for Boolean expressions only, the index calculation can be significantly simplified. In particular, given a Boolean expression $B[f_1, \cdots, f_k]$ with component expressions $B_1[f_1, \cdots, f_k]$, $B_2[f_1, \cdots, f_k]$, 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 locally encapsulating index function for B, because all of the Boolean expressions have the same component functions. Another noteworthy point is that Algorithm 1 requires a locally encapsulating index function for $\wedge/\vee$ functions, which is provided by Proposition III.3. The following example explicitly shows this calculation.

Example V.1: This example demonstrates a calculation of $I^l_{\wedge/\vee}(B(x'))$ in Algorithm 1. 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 locally encapsulating index function $I^l_{\wedge/\vee} : \mathbb{R}^n \to 2^{K_B}$ for the $\wedge$ or $\vee$ operation, according to Proposition III.3, evaluated at $B(x')$ is

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

for any fixed $\epsilon > 0$. •

VI. Experimental Results

This section presents experimental results for the framework developed in this article. Our experiment is inspired by 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 IV. The optimization program noted in Section IV synthesizes a controller that satisfies the objectives and the constraints, where the locally encapsulating index function is provided via Algorithm 1. For brevity, throughout the section, we drop the explicit dependence on time.

A. Experiment Formulation and Results

The formulation of the 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 state $x_i \in \mathbb{R}^2$, $i \in [N]$, and dynamics $\dot{x}_i = u_i$ (later, we utilize the method in [28] to map the single-integrator input onto the full nonlinear differential-drive dynamics). The ensemble state and input is written as $x \in \mathbb{R}^{2N}$ and $u \in \mathbb{R}^{2N}$, respectively.

This experiment requires that all robots avoid collisions, and this constraint may be encoded via the C^1 pairwise 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

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

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 ith and jth 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 $i = 1, 3, \cdots, N-1$, 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 ith component. For robots i and $i+1$ the objective that the robots visit p_i and p_{i+1} may be captured as

$$(V_{i,p_i} \wedge V_{i+1,p_{i+1}}) \vee (V_{i,p_{i+1}} \wedge V_{i+1,p_i}). \tag{13}$$

The abovementioned expression captures the specification that the robots must be at both points, but the order does not matter. The abovementioned expression is a candidate NLF; however, Proposition IV.6 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

$$(V_{i,p_i} \wedge V_{i+1,p_{i+1}}) \tag{14}$$

is a candidate NLF, and by Proposition IV.6, (13) 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 (14), V is also a candidate NLF.

Now that the system's constraints and objectives have been formulated, the locally encapsulating index functions for use with Algorithm 1 must be specified. This experiment utilizes the index function discussed in Proposition III.3, i.e.,

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

for some fixed $\epsilon > 0$. The function $I^l_{\wedge/\vee}$ is used as a locally encapsulating for every Boolean expression.

In the spirit of Proposition IV.10, the QP for this experiment is given by

$$\begin{aligned} u^\star(x) \in & \underset{u \in \mathbb{R}^m}{\operatorname{argmin}} \, u^\top u \\ \text{s.t. } & \nabla h_i(x)^\top (f(x) + g(x)u) \geq -\gamma h(x)^3, \forall i \in I^l_h(x) \\ & \nabla V_i(x)^\top (f(x) + g(x)u) \leq -\min\{c, V(x)\}, \forall i \in I^l_V(x) \end{aligned} \tag{15}$$

where $\gamma, c > 0$. Note that $V(x) \mapsto \min\{c, V(x)\}$ is the selected positive-definite function, as in Definition IV.3, and $h(x) \mapsto \gamma h(x)^3$ is the selected extended class-$\mathcal{K}$ function, as in Definition II.8.

For the experiment, we utilize Algorithm 1 to calculate a locally encapsulating index set for h and V. Then, combining h and V with these index functions into a QP, as in Proposition IV.10, yields a controller that ensures the robots visit the required locations and avoid collisions. For this experiment, we assume that the remaining hypotheses of Proposition IV.10 hold. Namely, that $u^\star$ exists and is measurable and locally bounded (we believe this property holds in general, albeit it has not been formally established). At each point, applications of Algorithm 1 calculate I^l_h and I^l_V, 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, techniques exist to ensure that this solution exists, such as the inclusion of slack variables (e.g., [3], [6]). For example, one may treat the safety constraints (i.e., those generated by barrier functions) as usual and associate slack variables with the objective constraints (i.e., those generated by Lyapunov functions). Note that this procedure increases feasibility of the QP at the cost of relaxing the satisfaction of the objective.

To show the efficacy of these results on a real system, the controller $u^\star$ from (15) is deployed onto $N = 12$ differential drive robots in the Robotarium, a remotely accessible swarm-robotics testbed [8]. Fig. 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. Fig. 1 shows that all robots reach their designated locations while avoiding collisions. Specifically, Fig. 2 shows the value of h over the course of the experiment. The Boolean NBF h remains positive, indicating that all constraints are satisfied. Fig. 3 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. Note that, during the experiment, V increases marginally at a few times. These slight increases are due to disturbances present in real-world experiments, such as network latency or wheel slip. Both simulation and theory confirm that the value is strictly decreasing over time (simulation results are not given here for brevity). Over the course of the experiment, determining the locally encapsulating index functions was nearly instantaneous while solving the QP in (15) was on the order of milliseconds.

Regarding the assumed single-integrator dynamics, the robots presented in Fig. 1 show that the utilized robots are differential drive, a nonlinear system. Within the Robotarium, these robots are feedback linearized, meaning that they may be abstracted as single integrators. This article treats these systems as such for brevity, but note that these results are practically applied to a nonlinear system.

For brevity, this article does not model disturbances during controller synthesis, as their inclusion is not explicitly related to the results of this article. However, given the set-valued nature of differential inclusions, it falls within the same mathematical framework to model disturbances. Indeed, recent work [29] has considered similar scenarios.

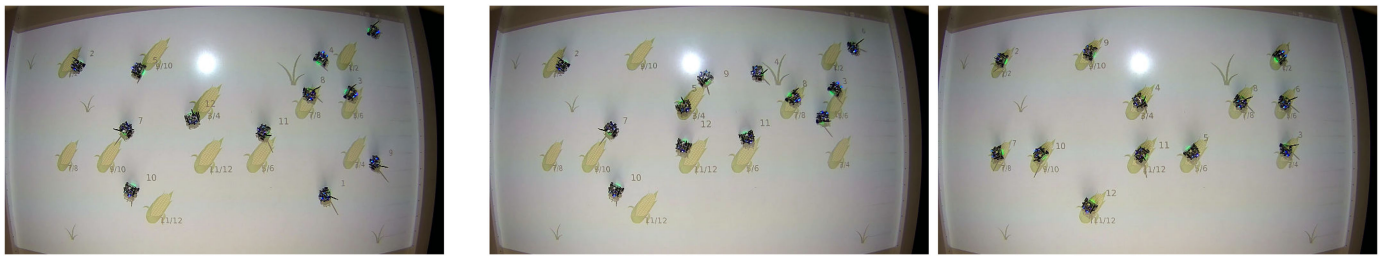

Fig. 1. Completion of the precision-agriculture experiment described in Section VI (initial position displayed on the left, final positions on the right). Each pair of robots (e.g., 1 and 2) must visit a pair of crop patches, which are labeled accordingly, while avoiding collisions. This figure shows that the robots successfully visit each crop patch and avoid collisions, completing the objectives and satisfying the constraints.

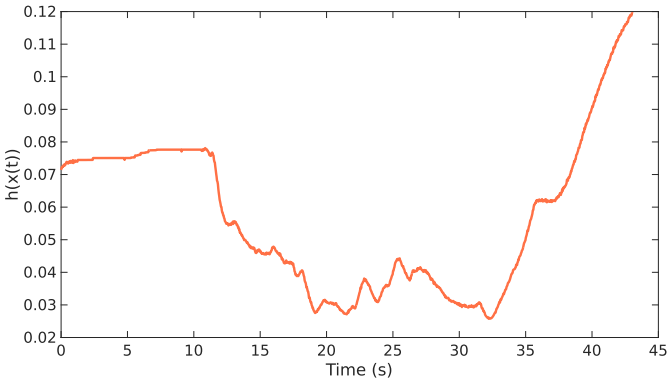

Fig. 2. Value of the NBF that encodes the collision-avoidance constraints for the experiment described in Section VI. The value of the NBF remains positive over the course of the experiment, showing that all of the constraints are satisfied, i.e., no robots collide.

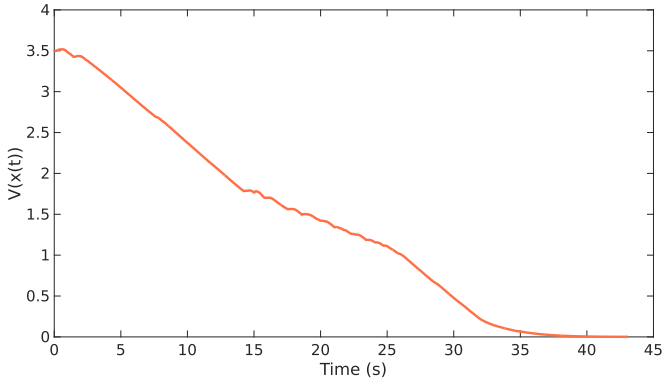

Fig. 3. Value of objective-encoding NLF in Section VI. The value of the NLF goes to zero as time increases, indicating that the objective has been completed for all robots, that is, all crop patches have been visited.

B. Discussion of Parameters

Here, we discuss the role of the parameters in the experiment of Section VI-A. The values that we have employed are

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

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 locally encapsulating index function and controls how many indices are included at each point. For example, for h, the locally encapsulating index function is

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

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 (15)]. 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^l.

Theoretically speaking, as long as $\epsilon > 0$, I_h^l is indeed a locally 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. A similar line of reasoning holds for V and I_V^l.

VII. Conclusion

This article has built on the current capabilities of barrier and Lyapunov functions to represent constraints and stability objectives for controlled dynamical systems, respectively. We have presented a new class of nonsmooth barrier functions and nonsmooth Lyapunov functions using the theory of piecewise smooth (PC^r) functions, and we have shown that Boolean combinations of barrier and Lyapunov functions fall into the class of PC^r functions. The notion of PC^r function depends heavily on its corresponding index functions, and by utilizing a particular class of them, we have proved that one may efficiently synthesize controllers that are discontinuous yet, nonetheless, provably guarantee the validity of the barrier and Lyapunov functions. The experimental results have illustrated how our theoretical contributions can be used to generate safe controllers that also accomplish an objective for a swarm of physical robots

in a precision-agriculture scenario. Future work would explore the expansion of the proposed Boolean-composition framework for NLFs and the generalization of the controller-synthesis procedure beyond quadratic programs.

REFERENCES

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

[2] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, R. Alur and G. J. Pappas, eds. Berlin, Germany: Springer, 2004, pp. 477–492.

[3] A. D. Ames *et al.*, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. 53rd IEEE Conf. Decis. Control*, Dec. 2014, pp. 6271–6278.

[4] U. Borrmann *et al.*, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[5] L. Wang *et al.*, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 2659–2664.

[6] X. Xiangru *et al.*, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[7] L. Wang *et al.*, "Safety barrier certificates for heterogeneous multi-robot systems," in *Proc. Amer. Control Conf.*, 2016, pp. 5213–5218.

[8] D. Pickem *et al.*, "The Robotarium: A remotely accessible swarm robotics research testbed," in *Proc. IEEE Int. Conf. Robot. Autom.* , May 2017, pp. 1699–1706.

[9] F. Clarke, Yu.S. Ledyaev, and R.J. Stern, "Asymptotic stability and smooth lyapunov functions," *J. Differ. Equ.*, vol. 149, no. 1, pp. 69–114, 1998.

[10] D. Shevitz and B. Paden, "Lyapunov stability theory of nonsmooth systems," *IEEE Trans. Autom. Control*, vol. 39, no. 9, pp. 1910–1914, Sep. 1994.

[11] 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 *Proc. 25th IEEE Conf. Decis. Control*, Dec. 1986, pp. 578–582.

[12] Y. Lin *et al.*, "A smooth converse Lyapunov theorem for robust stability," *SIAM J. Control Optim.*, vol. 34, no. 1, pp. 124–160, 1996.

[13] P. Glotfelter *et al.*, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE Control Syst. Lett.*, vol. 1, no. 2, pp. 310–315, Oct. 2017.

[14] P. Glotfelter *et al.*, "Boolean composability of constraints and control synthesis for multi-robot systems via nonsmooth control barrier functions," in *Proc. IEEE Conf. Control Technol. Appl.*, pages 897–902, Aug. 2018.

[15] Stefan Scholtes, *Introduction to Piecewise Differentiable Equations*. Berlin, Germany: Springer Science & Business Media, 2012.

[16] J. Cortés, "Discontinuous dynamical systems," *IEEE Control Syst. Mag.*, vol. 28, no. 3, pp. 36–73, Jun. 2008.

[17] F. Clarke, ,*Optimization and Nonsmooth Analysis*. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 1990.

[18] A. Bacciotti and F. Ceragioli, "Stability and stabilization of discontinuous systems and nonsmooth Lyapunov functions," *ESAIM, Control, Opt. Calculus Variations*, vol. 4, pp. 361–376, 1999.

[19] S. L. Herbert *et al.*, "FaSTrack: A modular framework for fast and guaranteed safe motion planning," in *Proc.56th IEEE Conf. Decis. Control*, Dec. 2017, pp. 1517–1522.

[20] Ian Mitchell *et al.*, "Safety preserving control synthesis for sampled data systems," *Nonlinear Anal., Hybrid Syst.*, vol. 10, pp. 63–82, 2013.

[21] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

[22] M. Hoy *et al.*, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: A survey," *Robot.*, vol. 33, no. 3, pp. 463–497, 2015.

[23] V. Shapiro, "Semi-analytic geometry with r-functions," *Acta Numerica*, vol. 16, pp. 239–303, 2007.

[24] A. Balestrino *et al.*, "R-composition of Lyapunov functions," in *Proc. 17th Mediterranean Conf. Control Autom.*, June. 2009, pp. 126–131.

[25] D. Panagou *et al.*, "Distributed coordination control for multi-robot networks using Lyapunov-like barrier functions," *IEEE Trans. Autom. Control*, vol. 61, no. 3, pp. 617–632, Mar. 2016.

[26] M. Della Rossa *et al.*, "Max-min lyapunov functions for switching differential inclusions," in *Proc. IEEE Conf. Decis. Control*, Dec. 2018, pp. 5664–5669.

[27] H. K. Khalil, "Nonlinear systems," Prentice-Hall, 2002.

[28] J. Cortés *et al.*, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[29] Y. Emam *et al.*, "Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed," in *Proc. 58th IEEE Conf. Decis. Control*, 2019.

Paul Glotfelter received the B.S. degree in computer engineering from the York College of Pennsylvaniathe, York, PA, USA, in 2011, the M.S. degree in electrical and computer engineering and the Ph.D. degree in robotics from the Georgia Institute of Technology, Atlanta, GA, USA, in 2014 and 2019, respectively.

He is a member of the Georgia Robotics and Intelligent Systems Laboratory, where his research interests include composition of specifications and controller synthesis for robotic systems.

Jorge Cortés (Fellow, IEEE) received the Licenciatura degree in mathematics from Universidad de Zaragoza, Zaragoza, Spain, in 1997, and the Ph.D. degree in engineering mathematics from Universidad Carlos III de Madrid, Madrid, Spain, in 2001.

He is a Postdoctoral with the University of Twente, Twente, The Netherlands, and the University of Illinois at Urbana-Champaign, Urbana, IL, USA. He was an Assistant Professor with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA, USA, from 2004 to 2007. He is currently a Professor with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, USA. He has authored *Geometric, Control and Numerical Aspects of Nonholonomic Systems* (Springer-Verlag, 2002) and coauthored (together with F. Bullo and S. Martínez) of *Distributed Control of Robotic Networks* (Princeton University Press, 2009). His current research interests include distributed control and optimization, network science, resource-aware control, nonsmooth analysis, reasoning and decision making under uncertainty, network neuroscience, and multiagent coordination in robotic, power, and transportation networks.

Prof. Cortés is with the IEEE Control Systems Society, he has been a Distinguished Lecturer (2010–2014), and is currently its Director of Operations and an elected member (2018–2020) of its board of governors.

Magnus Egerstedt (Fellow, IEEE) received the B.A. degree in philosophy from Stockholm University, Stockholm, Sweden, in 1996, the M.S. degree in engineering physics and the Ph.D. degree in applied mathematics from the Royal Institute of Technology, Stockholm, Sweden, in 1996 and 2000, respectively.

He is the Steve W. Chaddick School Chair and Professor with the School of Electrical and Computer Engineering, the Georgia Institute of Technology. He was a Postdoctoral Scholar with Harvard University. His research interests include control theory and robotics, with particular focus on control and coordination of complex networks, such as multirobot systems, mobile sensor networks, and cyber-physical systems.

Dr. Egerstedt was the recipient of a number of teaching and research awards, including the Ragazzini Award from the American Automatic Control Council, the Outstanding Doctoral Advisor Award and the HKN Outstanding Teacher Award from Georgia Tech, and the Alumni of the Year Award from the Royal Institute of Technology.