2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

A Scalable Controlled Set Invariance Framework with Practical Safety Guarantees

Thomas Gurriet[1], Mark Mote[2], Andrew Singletary[1], Eric Feron[2], and Aaron D. Ames[1]

***Abstract*—Most existing methods for guaranteeing safety within robotics require time-consuming set-based computations, which greatly limit their applicability to real-world systems. In recent work, the authors have proposed a novel controlled set invariance framework to tackle this limitation. The framework uses a classical barrier function formulation, but replaces the difficult task of computing large control invariant sets with the more tractable tasks of (i) finding a controller that stabilizes the system to a backup set, and (ii) verifying that this backup set is invariant under the stabilizing controller. In this paper, we build upon these results to show that the requirement of proving invariance of the backup set can be relaxed at the expense of providing weaker guarantees on the safety of the system. This trade-off is shown to be favorable in practice, as the theoretically weaker safety guarantees are sufficient in many practical applications. The end result is a framework with a computational complexity that scales quadratically. The effectiveness of the approach is demonstrated in simulation on a Segway.**

I. Introduction

Safety is arguably one of the most critical issues hindering the democratization of autonomous systems. Though safety is fairly well understood at a theoretical level, solutions that are both rigorous and practical have yet to be realized. In this paper, a framework is proposed that can handle complex high dimensional systems while still providing rigorous and practical safety guarantees.

A system is commonly defined to be safe if its state never leaves some chosen set, known as the *safety set*. This forms the basis of *Controlled Set Invariance* [1]: finding a control strategy that ensures that the system always remains in the safety set. When a system is simple, or enjoys some particular structure as in [2], analytical control strategies ensuring safety can be derived. In general, however, it is very difficult to directly find such a control strategy. This is due to the fact that inside an arbitrary safety set, some subsets cannot be visited by the system without eventually and inexorably leaving the safety set. However, finding a *safe* control law becomes much simpler if one is able to find a *control invariant* (also referred to as a viable) subset of the safety set [3]—i.e. a set that the system is capable of remaining within for all times, given that it started from any initial condition inside of it.

The largest control invariant subset of the safety set is called the *viability kernel*. Finding the viability kernel grants

[1]Thomas Gurriet, Andrew Singletary and Aaron D. Ames are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, 91125 USA.

[2]Mark Mote and Eric Feron are with the Department of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA.

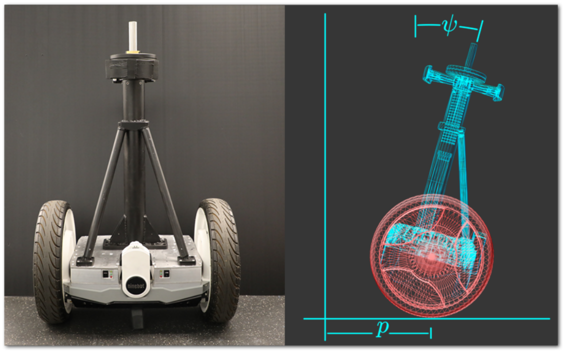


Fig. 1: Segway vehicle used for simulation.

maximum operational freedom to the system while ensuring that it can remain in the safety set. As a result, computing viability kernels has been the focus of a wide variety of research over the years. Approaches that have been proposed include: discretized solutions of Hamilton-Jacobi equations [4], SOS optimization [5], sampling [6] and many others [7]. Unfortunately, these algorithms take substantial time to run and can only handle high dimensional systems at the expense of conservative results, leading to small operational regions and poor performance for the system.

Given a control invariant set, safety of the system can then be guaranteed by continuously filtering the control signal in a minimally invasive way as proposed in [8] (see [9], [10], [11] for applications). In [12] the authors proposed a method to preserve the performance of this *Safety Filtering* methodology without the need to explicitly find a large control invariant set. This is achieved by considering a *backup controller* and a small forward invariant *backup set*. From these, it is possible to implicitly define a set for which the backup controller is effective at safely driving the system back to the backup set. This set is a control invariant subset of the safety set by construction. Thus, by performing sensitivity analysis around the backup trajectories, it is possible to use this implicitly defined set to safely regulate the system.

In the present paper, the work in [12] is first summarized in Sect. III and then extended by reformulating the *reachability requirement* as a *time based* condition in Sect. IV. This mitigates the need for the backup set to be forward invariant under the backup controller, and in turn decouples the choice of backup set and controller. In particular, this makes it possible to leverage an approximation of an optimal backup controller to provide near optimal safety filtering. This is explored in Sect. V, where a neural-network approximation of the optimal policy is implemented on a simulated Segway.

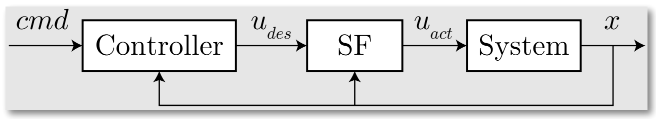


Fig. 2: Safety Filtering Control Structure.

II. Safety Filtering

A controller design task generally consists of finding a control policy that maximizes some performance criteria while ensuring safety of the system. However, finding a high performing policy that is safe by construction is difficult. System performance and safety are often conflicting goals, and guarantees on safety of the system generally become much more challenging as complexity of the controller increases. As an alternate paradigm, we consider the control structure depicted in Fig. 2. The idea here is to decouple performance from the task of enforcing hard safety constraints in such a way that prioritizes the latter over the former. Given a nominal controller that processes commands and focuses on performing the desired task, a *safety filter* can be used to preempt these desired inputs in a way that ensures safety of the system when necessary (see [10], [9], [11] for application examples). Ideally, the filter is minimally invasive to the desired input —i.e. u_{des} is left unmodified as long as the signal is not compromising to system safety. Designing such a filter is the goal of the proposed Controlled Set Invariance framework.

A. Sub-tangentiality Condition

In this paper, we will consider continuous-time affine control systems of the form

$$\dot{x} = f\left(x\right) + g\left(x\right) u. \tag{1}$$

Assumption 1. The functions f and g defined on a compact set $X \subset \mathbb{R}^n$ are continuously differentiable. The control policies are restricted to be functions $u : \mathbb{R}^+ \times X \longrightarrow \mathbb{R}^n$ that are Lipschitz continuous in state over X and piecewise continuous in time over $\mathbb{R}^+$. We furthermore define by $U \subset \mathbb{R}^m$ the compact and convex set of admissible inputs for this system, i.e. $\forall x \in X$ and $\forall t \in \mathbb{R}^+$, $u\left(t, x\right) \in U$.

Under these assumptions, system (1) is guaranteed to have a unique solution over a time interval $[0, T_X]$ for any initial condition $x\left(0\right) \in \mathrm{Int}\left(X\right)$ and with $T_X > 0$ being the time when solutions to (1) leave X [13].

Let's denote the chosen set of states that we allow our system to visit as the safety set $\widetilde{S} \subset X$ and require that it is compact. **The goal of Active Set Invariance Filtering is to ensure that the system will remain in $\widetilde{S}$ for all time.** But as discussed above, if $\widetilde{S}$ is chosen arbitrarily, then it will likely contain states that cannot be visited without inexorably leading to the system leaving the safety set at some future time. Such states are *unsafe* in the sense that they do not permit the system to *remain* inside of the safety set. Arbitrary safety sets that have not been chosen with care are therefore fundamentally unsafe as they almost always contain these unsafe states. Let us now assume that we have access to a non-empty set $S \subseteq \widetilde{S}$ and that S is control invariant for (1).

Definition 1. A closed set S is **control invariant** for system (1) if there exist a control policy u satisfying Assumptions *1* and an associate solution x for (1) such that $x\left(t_0\right) \in S \implies \forall t \geq t_0,\ x(t) \in S$.

Such a set is very interesting as it does not contain any unsafe states. It is therefore possible to guarantee safety of the system by ensuring the invariance of S.

Definition 2. A closed set S is **invariant** for system (1) under a policy u satisfying Assumptions *1* if the solution to (1) satisfies that $x\left(t_0\right) \in S \implies \forall t \geq t_0,\ x(t) \in S$.

Having access to S thus makes the controlled set invariance task much easier. First, it provides a sufficient condition for the initial state to be safe—$x\left(t_0\right) \in S$. Next, it allows us to use a local characterization of invariance originally provided by Nagumo in 1942 [14] and specialized here for our application [3]. *Let* $\mathcal{T}_S(x)$ denote the contingent cone to S at x [3], [14].

Proposition 1. Given (1) subject to Assumptions 1, if for almost all $t \geq 0$ and for all $x \in S$:

$$f(x) + g(x)u\left(t, x\right) \in \mathcal{T}_S(x), \tag{2}$$

then for all $x\left(0\right) \in S$ *there exist a solution to* (1) *that remains in* S *for all times* $t > 0$*, i.e.* S *is weakly invariant under* u*.*

Proof. The proof is a direct application of [3, Thm. 11.7.1] to the case of differential equations. ∎

Remark 1. Because u has been assumed to be Lipschitz continuous with respect to its second argument, this solution is unique and (1) extends to the "strong" invariance of S.

From this proposition, it naturally follows that the *regulation map* $\overline{U_S} : X \rightrightarrows 2^U$ defined by

$$\overline{U_S}\left(x\right) \triangleq \left\{u \in U \mid f(x) + g(x)u \in \mathcal{T}_S(x)\right\} \tag{3}$$

is sufficient to encode the safety of a control action. Indeed, if the control signal u takes values in $\overline{U_S}$ for all times, i.e. if u is a *selection* of $\overline{U_S}$, then the *sub-tangentiality condition* (2) if trivially verified, guaranteeing the safety of the system. Hence, $\overline{U_S}(x)$ encodes the set of inputs that are both admissible and safe with respect to the set S.

B. Regulation Map

Depending on the type of set considered, there are different ways of expressing the contingent cone. Practical sets—as defined in [14]—are suitable for most realistic cases and makes it convenient to express the contingent cone. To describe such sets, ones only needs to consider N_s continuously differentiable functions $h_i : \mathbb{R}^n \to \mathbb{R}$ such that [1]:

$$\begin{aligned} S &= \left\{x \in \mathbb{R}^n \mid \forall i \in \{1, \ldots, r\},\ h_i\left(x\right) \geq 0\right\} \\ \partial S &= \left\{x \in S \mid \exists i \in \{1, \ldots, r\},\ h_i\left(x\right) = 0\right\}. \end{aligned} \tag{4}$$

[1]See [14, p. 103] for all conditions under which S is practical.

For such sets, the contingent cone can be expressed as

$$\mathcal{T}_S(x) = \{z \in \mathbb{R}^n \mid \forall i \in Act(x),\ \nabla h_i(x).z \geq 0\}, \quad (5)$$

with $Act(x) \triangleq \{i \in \{1,\dots,N_s\} \mid h_i(x) = 0\}$. In that case, the sub-tangentiality condition (2) can be written as

$$TC_i(x,u) \triangleq L_f h_i(x) + L_g h_i(x) u(x) \geq 0, \quad (6)$$

for all $x \in \partial S$, and $i \in Act(x)$. Here, $L_f h$ and $L_g h$ denote the Lie derivatives of h along f and g respectively. The regulation map then becomes:

$$\overline{U_S}(x) = \bigcap_{i=1}^{N_s} \begin{cases} \{u \in U \mid TC_i(x,u) \geq 0\}, & \text{if } x \in \partial S \\ U, & \text{otherwise} \end{cases} \quad (7)$$

The sub-tangentiality condition is however not very usable in practice as it only defines a non-trivial set of admissible inputs when the system is on the boundary of the safety set (hence the notation), which leads to a discontinuous regulation of the control input. One solution to smooth out the regulated input that is explored in [3]—further developed in [8]—is to consider a strengthening term in (6) and to impose a new *barrier condition*:

$$BC_i(x,u) \triangleq L_f h_i(x) + L_g h_i(x) u(x) + \alpha_i(h_i(x)) \geq 0, \quad (8)$$

for all $x \in S$, $i \in \{1,\dots,N_s\}$ and with the *strengthening* extended class $\mathcal{K}$ functions $\alpha_i : \mathbb{R} \to \mathbb{R}$. This barrier condition defines a new *sub-regulation map* U_S of admissible inputs:

$$U_S(x) \triangleq \{u \in U \mid \forall i \in \{1,\dots,N_s\}, BC_i(x,u) \geq 0\} \quad (9)$$

and because for all $x \in S$, $U_S(x) \subseteq \overline{U_S}(x)$, the condition also implies forward invariance of S. Note that because in most cases $U_S(x) \subset \overline{U_S}(x)$ for some $x \in S$, finding a control invariant set is not necessarily sufficient to ensure that U_S does not take empty values. One has to be careful and choose the strengthening functions α_i accordingly, as discussed in [9], [3], which is always possible under the present assumptions.

C. Safety Filter as a Quadratic Program

In light of these results, it is now possible to construct a safety filter (cf. Fig. 2) that can guarantee safety of the system. Given a desired input u_{des} provided by a nominal controller, enforcing safety in a *minimally invasive* way can be naturally formulated with the following quadratic program:

Safety Filter QP

$$\begin{aligned} u_{\text{act}}(t,x) = \underset{u}{\text{argmin}} \quad & \|u_{\text{des}}(t) - u\|^2 \\ \text{s.t.} \quad & u \in U_S(x) \end{aligned} \quad (10)$$

For any $x \in X$ and $t \in \mathbb{R}^+$, (10) has a unique solution, so u_{act} is a well defined single-valued function. Assuming that u_{des} is piecewise continuous, the Lipschitz continuity in state of the resulting control policy u_{act} can be verified. For more information on the matter we refer the reader to [15], [3], [16] but note that it is of little concern in practice. Finally, because S is control invariant and Assumptions 1 hold, U_S has non-empty compact convex values, (10) is indeed a convex program which is always feasible, which makes its implementation realizable.

Ideally, one would want to use the largest control invariant subset S of $\widetilde{S}$ (i.e. the *viability kernel*) to maximize the operational freedom of the system. Finding an explicit representation of the *viability kernel* is however notoriously hard—just as hard as finding an optimal closed-loop controller [4]. Consequently, this approach has been restricted to low dimensional systems in practice (except in the unrealistic scenario where $U = \mathbb{R}^m$). The following section demonstrates an approach for accessing a *large* control invariant subset of $\widetilde{S}$, while only requiring the explicit computation of a *small* control invariant subset of $\widetilde{S}$—which is comparatively easy.

III. Implicit Safety Filtering

The key idea of our approach it to propose a way for systematically and implicitly defining a control invariant subset of $\widetilde{S}$. The *implicit* set will be defined as the collection of states such that a *backup strategy* can safely steer the system to a *backup set*. This idea is not new, but to our knowledge, it has only been developed in a context where switching between the backup and nominal policy is performed (cf. [17], [18]). In this section we summarize the results in [12] and show how this idea can be used to make the Active Set Invariance Filtering Framework of Sec. II more tractable.

A. Implicit Control Invariant Set

Let $\mathcal{U}$ be the set of all continuously differentiable backup control laws taking values in the set of admissible inputs: $u_b : \mathbb{R}^n \to U$. Under Assumptions 1, we know that for all $u_b \in \mathcal{U}$ there exists a solution to (1) that is unique and defined for all times when solutions to (1) stay in X. Therefore, one can define $\phi^{u_b} : [0,T_X] \times X \to \mathbb{R}^n$ to be the *flow* of (1) under the control law u_b. Under these assumptions, the map $\phi_t^{u_b} : X \to \mathbb{R}^n$ defined by $\phi_t^u(x) \triangleq \phi^u(t,x)$ is a homeomorphism of X (cf. [19]) for all $t \in [0,T_X]$. We will denote by $S_b \subseteq \widetilde{S}$ the non-empty compact *backup set* as depicted in Fig. 3. We furthermore assume that S_b is practical and can be represented as the super level set of smooth functions $h_{b,j} : \mathbb{R}^n \to \mathbb{R}$, $j \in \{1,\dots,N_b\}$.

We can now define the implicit set more precisely.

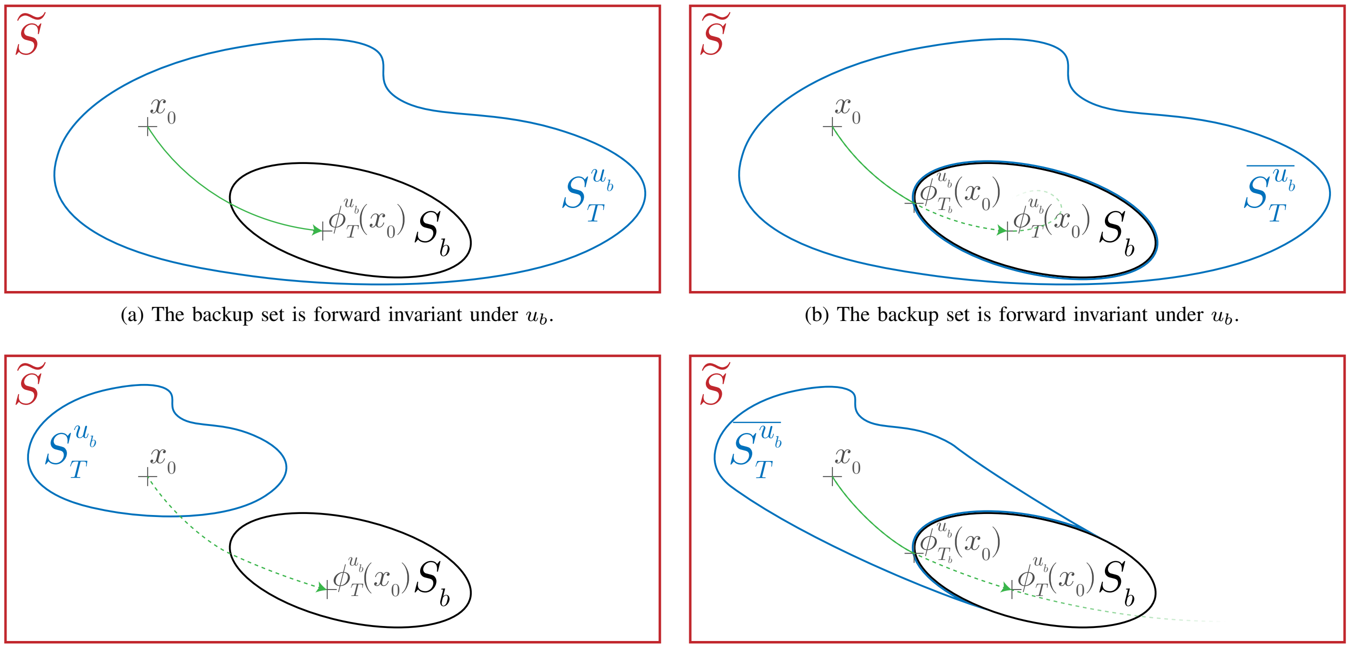
Definition 3. We define the **safe backward image** of the backup set to be the set $S_T^{u_b} \triangleq R_T^{u_b} \cap \Omega_T^{u_b}$, where

$$R_T^{u_b} \triangleq \{x \in X \mid \phi_T^{u_b}(x) \in S_b\}, \quad (11)$$

is a set encoding the *reachability* of the backup set under the backup trajectory and

$$\Omega_T^{u_b} \triangleq \left\{x \in \widetilde{S} \mid \forall t \in [0,T],\ \phi_t^{u_b}(x) \in \widetilde{S}\right\}, \quad (12)$$

is a set encoding the safety of the corresponding backup trajectory.



(a) The backup set is forward invariant under u_b.

(b) The backup set is forward invariant under u_b.

(c) The backup set is **not** forward invariant under u_b.

(d) The backup set is **not** forward invariant under u_b.

Fig. 3: Safety Set in red, Backup Set in black, and Implicit Control Invariant set in blue. The backup trajectory under the backup control law u_b is in green.

The following propositions are stated without proofs as they follow directly from the ones in [12].

Proposition 2. *If S_b is invariant under $u_b \in \mathcal{U}$ then for all $T \in \mathbb{R}^+$, $S_T^{u_b}$ is a subset of $\widetilde{S}$ invariant under u_b.*

Remark 2. Note that in this case, $S_T^{u_b}$ is never empty as it always contains S_b.

Proposition 3. *If S_b is invariant under $u_b \in \mathcal{U}$ then for all $T \in \mathbb{R}^+$:*

$$\Omega_T^{u_b} = \bigcap_{t\in[0,T]} \left\{x \in X \mid h_i \circ \phi_t^{u_b}(x) \geq 0,\ i \in \{1,\dots,N_s\}\right\},$$

and

$$R_T^{u_b} = \left\{x \in X \mid h_{b,j} \circ \phi_T^{u_b}(x) \geq 0,\ j \in \{1,\dots,N_b\}\right\}.$$

It trivially follows from these 2 propositions that $S_T^{u_b}$ is a control invariant subset of $\widetilde{S}$, and so it can be used to define a non-empty sub-regulation map $U_{S_T^{u_b}}$. In [12] it is finally shown that the *size* of $S_T^{u_b}$ is—loosely speaking—increasing with both T and the *performance* of u_b in driving the system back to S_b. Therefore, it is now clear how we have constructed large control invariant sets $S_T^{u_b}$ from a small control invariant set S_b as depicted in Fig. (3a).

B. Implicit Safety Filter

In order to use a Safety Filter, one must be able to evaluate $U_S(x)$. This is easy if S has an explicit representation, which is why so much effort has been focused on finding such a representation. In this case however, $S_T^{u_b}$ is defined as a function of $\phi_t^{u_b}$ for which we do not have an explicit representation, hence the implicit nature of $S_T^{u_b}$. This is however not an issue as in practice, numerical methods have to be used to solve the optimization problem (10) at each sampled $(t_k, x(t_k))$ pair anyway. Because it is possible to use numerical methods to evaluate $U_{S_T^{u_b}}$ at each sampled state, this approach turns out to be satisfactory in practice.

The sub-regulation map $U_{S_T^{u_b}}$ evaluated at a given state x_0 is equal to the set of $u \in U$ such that

$$\begin{cases} \nabla h_i(x_{t_b})\, D\phi_{t_b}^{u_b}(x_0)\, \tilde{f}(x_0,u) + \alpha_i(h_i(x_{t_b})) \geq 0 \\ \nabla h_{b,j}(x_T)\, D\phi_T^{u_b}(x_0)\, \tilde{f}(x_0,u) + \beta_j(h_{b,j}(x_T)) \geq 0 \end{cases}$$

with $i \in \{1,\dots,N_s\}$, $j \in \{1,\dots,N_b\}$, $t_b \in [0,T]$, $x_t \triangleq \phi_t^{u_b}(x_0)$ and $\tilde{f}(x_0,u) \triangleq f(x_0) + g(x_0)u$.

One caveat here is that the image of $U_{S_T^{u_b}}$ is formed by an *uncountable* set of linear constraints, which elevates the Safety Filter into the class of *robust optimization problems*. However, by integrating (1) forward in the interval $[0,T]$, $\phi_{t_b}^{u_b}(x_0)$ can be numerically evaluated at discrete times $\{t_{b,0},\dots,t_{b,N_t}\}$. It is therefore possible to *approximate* $U_{S_T^{u_b}}(x_0)$ by considering the countable set of constraints at these different times $t_{b,k}$—the more points being considered the tighter the approximation. Interior point solvers are good at handling a large number of constraints which makes this type of approximation indeed tractable. To compute the Jacobian $D\phi_{t_b}^{u_b}(x_0)$, one only needs to integrate a *sensitivity* matrix of size n^2 along the backup trajectory of (1) (cf. [20], [12] for more details).

IV. Finite Time Safety Guarantees

Thus far, the backup policy and the backup set cannot be chosen independently, as the former has to be invariant under the latter. This fact inhibits the scalability of the proposed approach. In this section, we present an extension to this approach that allows for independent selection of the backup controller and backup set. As will be discussed, this comes at the expense of weaker safety guarantees.

A. Reformulation of the Reachability Constraint

The main idea presented here is to relax the reachability constraint for $R_T^{u_b}$. For simplicity, we will restrict ourselves to backup sets that can be represented as the upper level-set of a single function h_b twice differentiable. This restriction can be lifted by considering a Safety Filter that is a mixed-integer QP. Let us now define the notion of **time to safety.**

Definition 4. Given a backup control law $u_b \in \mathcal{U}$, the **time to safety** $T_b : X \to \mathbb{R}$ is given by

$$T_b(x) = \min\left\{t \geq 0 : h_b\left(\phi_t^{u_b}(x)\right) = 0\right\}. \tag{13}$$

When $x \in S_b$, we choose $T_b(x) = 0$, and when $x \notin S_b$ and a solution to (13) does not exist, we choose $T_b(x) = +\infty$.

Consider now the set $\overline{R_T^{u_b}}$ given by the closure:

$$\overline{R_T^{u_b}} \triangleq \overline{\{x \in X \mid 0 < T_b(x) \leq T\}}, \tag{14}$$

with $T > 0$ and $u_b \in \mathcal{U}$. The interest of considering such a set becomes clear when one realizes that if S_b is not invariant under u_b, then S_b is not a subset of $R_T^{u_b}$ (cf. [12, Prop. 2] and Fig. (3c)). This makes the set $R_T^{u_b}$ unusable as it is not guaranteed that $R_T^{u_b} \cap \widetilde{S} \neq \emptyset$ and that $S_T^{u_b}$ is not empty. The set $\overline{R_T^{u_b}}$ on the other hand will at least contain part of the boundary S_b (provided the backup set is not completely repulsive) and grow monotonically with T (cf. Fig. (3d)).

B. Augmented Regulation Map

As in Sec. 3, we would like to regulate safe solutions using the sub-regulation map U_S; now however utilizing the set $\overline{R_T^{u_b}}$. The barrier condition (8) for set $\overline{R_T^{u_b}}$ evaluated at a state $x_0 \in \overline{R_T^{u_b}}$ for a given backup control law $u_b \in \mathcal{U}$ is given by

$$-\nabla T_b(x_0)\tilde{f}(x_0, u) + \alpha\left(T - T_b(x_0)\right) \geq 0, \tag{15}$$

where $\tilde{f}(x_0, u) \triangleq f(x_0) + g(x_0)u$. As demonstrated in [21]:

$$\nabla T_b(x_0) = -\frac{\nabla h_b(\overline{x}) \cdot D\overline{x}(x_0)}{\nabla h_b(\overline{x}) \cdot \tilde{f}(\overline{x}, u_b(\overline{x}))}, \tag{16}$$

with $\overline{x} \triangleq \phi_{T_b(x_0)}^{u_b}$. One will immediately notice that this gradient is only defined when $\nabla h_b(\overline{x}) \cdot \tilde{f}(\overline{x}, u_b(\overline{x}))$ does not vanish. States for which this happens should therefore be avoided to allow for the regulation of safe solutions using U_S with $\overline{R_T^{u_b}}$. Let us therefore consider the following extension of the safe backward image of the backup set

$$\overline{S_T^{u_b}} = \overline{R_T^{u_b}} \cap \Omega_T^{u_b} \cap C_T^{u_b}, \tag{17}$$

with

$$C_T^{u_b} \triangleq \left\{x_0 \in \overline{R_T^{u_b}} \mid \widetilde{cos}(x_0) \geq \varepsilon_b\right\}, \tag{18}$$

where $\varepsilon_b > 0$ is a *small* constant, and

$$\widetilde{cos}(x_0) \triangleq \frac{\nabla h_b(\overline{x})\tilde{f}(\overline{x}, u_b(\overline{x}))}{\|\nabla h_b(\overline{x})\| \left\|\tilde{f}(\overline{x}, u_b(\overline{x}))\right\|}. \tag{19}$$

Proposition 4. *If $\overline{R_T^{u_b}} \neq \emptyset$, then for all $x_0 \in \overline{S_T^{u_b}} \setminus S_b$, there exist strengthening functions α_i such that $U_{\overline{S_T^{u_b}}}(x_0) \neq \emptyset$.*

Proof. Note that since ∇T_b is not defined on ∂S_b, we need only consider states in $\overline{S_T^{u_b}} \setminus S_b$. The map $U_{\overline{S_T^{u_b}}}$ evaluated at a given state $x_0 \in \overline{S_T^{u_b}} \setminus S_b$ is equal to the set of $u \in U$ such that

$$\begin{cases} \nabla h_i(x_{t_b}) D\phi_{t_b}^{u_b}(x_0)\tilde{f}(x_0, u) + \alpha_i\left(h_i(x_{t_b})\right) \geq 0 \\ -\nabla T_b(x_0)\tilde{f}(x_0, u) + \beta\left(T - T_b(x_0)\right) \geq 0 \\ \nabla\widetilde{cos}(x_0)\tilde{f}(x_0, u) + \gamma\left(\widetilde{cos}(x_0)\right) \geq 0 \end{cases} \tag{20}$$

with $i \in \{1, \ldots, N_s\}$, $t_b \in [0, T_b(x_0)]$, $x_t \triangleq \phi_t^{u_b}(x_0)$ and $\tilde{f}(x_0, u) \triangleq f(x_0) + g(x_0)u$. First, as $D\overline{x}(x_0)\tilde{f}(x_0, u_b(x_0)) = \tilde{f}(\overline{x}, u_b(\overline{x}))$, we have

$$\nabla T_b(x_0)\tilde{f}(x_0, u_b(x_0)) = -1. \tag{21}$$

This is fairly intuitive as (21) is the time derivative of the time to safety when the system is evolving along the backup trajectory. Second, because for all $t_b \in [0, T_b(x_0)]$, $\overline{x}\left(\phi_{t_b}^{u_b}(x_0)\right) = \overline{x}(x_0)$,

$$\nabla\widetilde{cos}(x_0)\tilde{f}(x_0, u_b(x_0)) = 0. \tag{22}$$

Finally, because $x_0 \in \overline{S_T^{u_b}} \supset \Omega_T^{u_b}$, for all $t_b \in [0, T_b(x_0)]$, $\phi_{t_b}^{u_b}(x_0) \in \widetilde{S}$. By continuity of all of the functions involved, α_i can be chosen (cf. [9], [8]) such that for all $x_0 \in \overline{S_T^{u_b}}$:

$$\nabla h_i(x_{t_b}) D\phi_{t_b}^{u_b}(x_0)\tilde{f}(x_0, u_b) + \alpha_i\left(h_i(x_{t_b})\right) \geq 0.$$

Thus all conditions in (20) can be simultaneously satisfied in $\overline{S_T^{u_b}}$ by choosing $u(x_0) = u_b(x_0)$. Hence $U_{\overline{S_T^{u_b}}}$ is well defined and non-empty over all of $\overline{S_T^{u_b}}$. ■

Remark 3. The functions β and γ can be chosen as arbitrary extended class $\mathcal{K}$ functions.

Remark 4. Computing $\nabla\widetilde{cos}(x_0)$ requires the evaluation of the Hessian of h_b at x_0, otherwise $U_{\overline{S_T^{u_b}}}$ can be evaluated using the same technique as in Sec. III-B without any change in complexity of the algorithms. Also note that this new formulation of the sub-regulation map creates additional algorithmic challenges in reliably finding $\overline{x}$. The details of these numerical issues are outside the scope of this paper.

C. Weaker but Practical Safety Guarantees

Let us now consider the guarantees that can be attained when regulating the system in $\overline{S_T^{u_b}}$ using $U_{\overline{S_T^{u_b}}}$.

Theorem 1. *If $\overline{R_T^{u_b}}$ is not empty, $x(0) \in \overline{S_T^{u_b}}$, and for almost all $t \in \mathbb{R}^+$, $u(t, x(t)) \in U_{\overline{S_T^{u_b}}}$, then there exist $T_s \in [0, +\infty]$ such that for all $t \in [0, T_s)$, $\phi_t^{u(t,x(t))}(x) \in \overline{S_T^{u_b}}$. Furthermore, if $T_s < +\infty$, $\phi_{T_s}^{u(t,x(t))}(x) \in S_b$.*

Proof. The proof follows from [3, Prop. 4.3.7]. Indeed, from Prop. 4 we know that $U_{\overline{S_T^{u_b}}}$ is non-empty on $\overline{S_T^{u_b}} \setminus S_b$ so the sub-tangentiality condition is satisfied on the boundary of that set. Hence the system will never cross the boundary of $\overline{S_T^{u_b}} \setminus S_b$. So if the system ever leaves the compact set $\overline{S_T^{u_b}}$ it will be through $\overline{S_T^{u_b}} \cap S_b$ which is never empty. ■

Concretely, this means that regulating the inputs using $U_{\overline{S_T^{u_b}}}$ (with safety filter (10) for example) will guarantee that, if the system starts in $\overline{S_T^{u_b}}$ but outside of S_b, it will either stay in $\widetilde{S}$ and within *reach* the backup set S_b within a finite time T, or reach the backup set in finite time (cf. Fig. 3d). These guarantees may seem weak compared to the ones in (III), but they are indeed very relevant in practice.

For autonomous systems for example, the priority is (almost) always given to the avoidance of human casualty over the integrity of the system. Being able to safely *terminate* the system is often all that is required (cf. [22] for more details in the case of UAVs). For commercial transoceanic flights, being able to safely reach an airfield within a set amount of time is the safety criterion used by the Federal Aviation Administration (cf. ETOPS). In this case, the modality of landing the plane—what happens once the system has reached the backup set—can be handled separately.

Finally, it will not be proven here but it is easy to verify that $\left(\overline{R_T^{u_b}} \cup S_b\right) \supseteq R_T^{u_b}$, and that when S_b is forward invariant under u_b, $\left(\overline{R_T^{u_b}} \cup S_b\right) = R_T^{u_b}$ (cf. Fig. 3b). Therefore, when S_b is forward invariant (and ε_b small enough), the present approach yields the original safety guarantees of Sec. (III), i.e. the system remains in $\widetilde{S}$ for all times. Hence the soundness of this approach provides *weak but practical* safety guarantees without the challenge of having to verify the forward invariance of the backup set under u_b, but also provides *strong* safety guarantees when the backup set is actually forward invariant under u_b (cf. Fig. 3).

V. Implementation on a Segway

The performance of the safety filter—in the sense of least restrictive filtering—is directly correlated to the size of the safe backward image (SBI) of the backup set: the larger this region is, the less intrusive the safety filter will be to the nominal control input u_{des}. As discussed in [12], backup controllers that solve an optimal control problem (OCP) whose cost is related to reaching the backup set *quickly* are good at maximizing the size of the backward image of the backup set. While there exist software to solve this problem programmatically, doing so introduces a heavy computational burden, often making the formulation intractable online.

An attractive solution to this issue is to solve the OCP offline and to learn an approximate policy over the safety set using a neural network (NN). At the expense of introducing some conservatism in the filter compared to a pure online implementation of the OCP, this function approximation is much more computationally efficient. However, verifying the forward invariance of the backup set for such controller parameterization is difficult. So by using the reformulation of the framework proposed in this paper, it is possible to leverage the performance benefit of having a near optimal backup policy without having to solve the OCP online—or verify that the NN backup controller yield forward invariance of the backup set. In this section, we first demonstrate the effectiveness of using a NN to learn a *near-optimal* backup policy, and then present an simulated implementation of the framework using this NN.

A. Neural Network Approximation of the Optimal Backup Controller

Consider a planar model of the two wheeled inverted pendulum (commonly referred to as a Segway) in Fig. 1. The state vector is composed of the following variables: position (p), velocity ($\dot{p}$), pitch angle (ψ) and pitch rate ($\dot{\psi}$), and the input (u) is taken as the voltage applied to the motors. The equations are derived using classical Lagrangian mechanics assuming no-slip between the ground and the wheels and a first order model of a DC motor. The resulting formulation is control affine.

Our controller is derived from solutions to the following optimal control problem, stated in continuous time:

Optimal Control Problem (OCP)

$$\begin{aligned} u^*(x) = \underset{u \in U}{\text{argmin}} \quad & t_f + \int_0^{t_f} x^T Q x + R u^2 dt \\ \text{s.t.} \quad & \dot{x} = f(x) + g(x)u \\ & x(0) = x_0,\ x(t_f) = 0 \\ & x \in S \end{aligned} \tag{23}$$

where the state and input cost matrices are $Q = 0.02 I_4$ and $R = 0.01$ respectively, and the input is bounded to the range $U = [-20, 20]$. The goal of this controller is to drive the vehicle to the origin while remaining in the safety set,

$$S = \{x \in \mathbb{R}^4 : |p| \leq 3, |\dot{p}| \leq 3, |\psi| \leq \tfrac{\pi}{4}, |\dot{\psi}| \leq \pi\}. \tag{24}$$

Note that penalizing the final time t_f conditions the solution to reach its goal under shorter time horizon. However, the numerical solution to a purely time optimal formulation tends to result in highly non-smooth output signals that border constraint boundaries. As such, state and input penalties are added to the cost in order to make the solutions more amenable to functional approximation.

The OCP (23) is solved using GPOPS-II [23] software with the "hp-PattersonRao" mesh method at a tolerance of 0.01. Initial states for each trajectory are generated randomly within the bounds of $\widetilde{S}$. A total of 816 trajectories are considered, comprising approximately 49000 trajectory points. The trajectory and control data are to train a network $\mathcal{N}(x) : \mathbb{R}^4 \mapsto \mathbb{R}$ consisting of a single hidden layer with 100 nodes. Hyperbolic tangent sigmoid (tansig) activation functions are used in this layer. The training process consists of 280 epochs with the "Levenberg–Marquardt" training algorithm, resulting in a mean squared error (MSE) of 2.96.

Sampled initial conditions are taken to characterize a level slice of the SBI for (i) the computed solution to the OCP (23),

(ii) the NN approximation $\mathcal{N}$, and (iii) an LQR controller with state and input cost matrices equal to those defined in (23) (see Fig. 4). In this scenario, the initial position and angle of the vehicle are fixed at zero $x, \psi = 0$, and a set of initial velocity perturbations $\dot{x_0}, \dot{\psi}_0$ are given in the bounds of S. The backup set is defined by,

$$h_b \triangleq 0.1^2 - (p/3)^2 - (\dot{p}/3)^2 - (4\psi/\pi)^2 - (\dot{\psi}/\pi)^2, \quad (25)$$

and each controller is integrated over a horizon of $T = 5s$.

The SBI for the LQR and NN controllers are estimated by considering a 250×250 grid of initial velocity perturbations. The SBI boundaries indicated in Fig. 4 separate the sets of sampled initial conditions for which a safe backup policies could always be found (interior) from the sets of sampled initial conditions where safe backup policies were never found (exterior). For the LQR controller, **21.0%** of the samples in fall into the SBI. The NN controller extends this to cover **67.3%** of the sample points. The increased performance of the NN can be attributed to both its suitability to the nonlinear dynamics, and the explicit consideration of state constraints in the training process.

We may also compare the performance of the approximated controller to the returned solution to the OCP directly. Fig. 4 shows the result of a number of GPOPS-II evaluations corresponding to a set of uniformly distributed initial conditions in this domain. We see that many feasible points extend beyond the SBI of $\mathcal{N}$, in theory indicating that this region could be extended with more data and a larger network. Infeasible outputs arise when either the program fails to return a solution, or experiences numerical issues resulting in a solution that violates one of the constraints. In contrast to the controllers previously considered, the numerical solution to the OCP does not appear to have a clear boundary for its SBI. Solutions may fail in any subset of the domain. The NN approximation excludes these failed trials, which effectively regularizes the output of the solver. Hence in addition to providing a more scalable approach, the learned control strategy is seen to demonstrate a comparative improvement in numerical reliability over its operational domain.

B. Simulation Results

The proposed framework has been implemented in C++ and tested on a simulator of the Segway depicted in Fig. 1. For the integration of the backup trajectory, the ODEINT library [24] was used with the *Dormand-Prince 5* adaptive stepper scheme. For solving the resulting Quadratic Program, the OSQP solver [25] was used.

The simulation consists of a Segway starting near the origin of the state space. The desired input is set to be a constant $u_{\text{des}} = 15V$, which if left unaltered, will quickly drive the system out of the safe set. The maximum time to safety is chosen to be $T = 3s$ at first, and is then changed to $T = 6s$ after $2s$ of simulation. As can be seen in Fig. 5, the system never leaves the safety set, but also does not come back to the backup set—it stabilizes at an equilibrium point near the boundary of $\overline{S_T^{u_b}}$. In Fig. 6, we can see u_{des} being

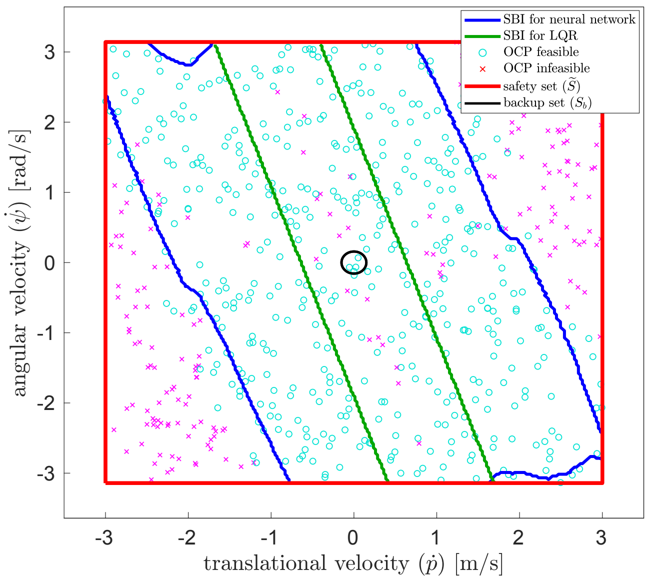


Fig. 4: Comparison of the safe backward images (SBI) of S_b when using an LQR, solving the optimal control problem (OCP), and using a NN regression of the OCP.

filtered within the set of admissible inputs, as well as the time to safety and the safety of the backup trajectory trading place as the active constraint in the SF.

For reference, when $T = 6s$ and the integration step for the backup trajectory is $0.01s$, the average computation time of each SF sampling iteration is around $600us$ on a laptop with an i7-6820HQ processor at 2.7GHz.

VI. Conclusions and Future Work

In this paper, a novel approach is presented for guaranteeing the safety of a system via minimally invasive filtering of a nominal control output. In contrast to existing approaches, this method does not require the difficult task of explicitly computing a control invariant safety set. Instead, the safe region of operation is implicitly defined by the ability of an arbitrary backup control strategy to safely reach a chosen backup set—from which all the computation necessary to run a Safety Filter can be performed online. The practicality of this is framework is demonstrated through a simulated case study, where a neural network based backup controller is used to ensure safety of a Segway. The authors are now focusing on the hardware implementation of the proposed method.

Acknowledgment

This work is supported by NSF award #1724457, #1446758 and #1555332. The authors would like to thank Claudia Kann for her valuable help in proofreading the present paper.

References

[1] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[2] Reza Ghaemi and Domitilla Del Vecchio. Control for safety specifications of systems with imperfect information on a partial order. *IEEE Transactions on Automatic Control*, 59(4):982–995, 2014.

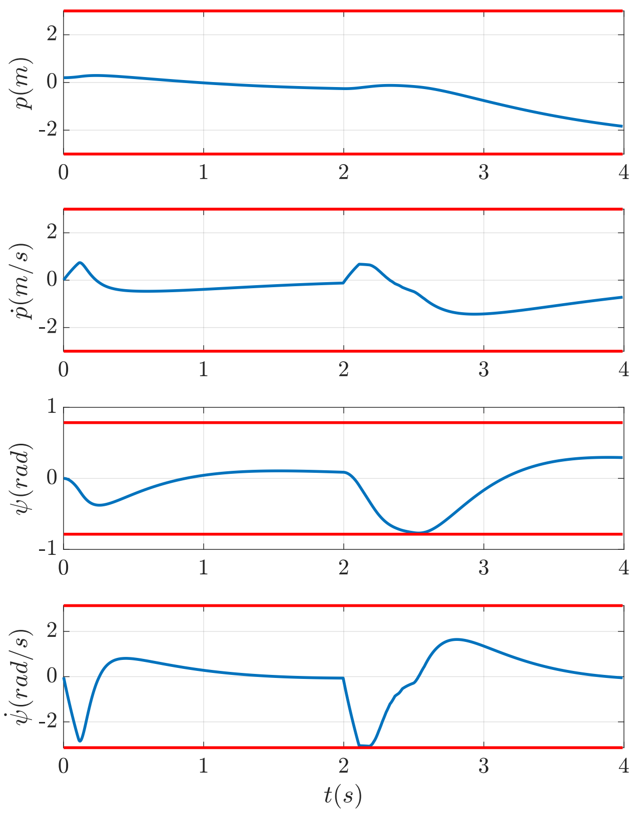


Fig. 5: States of the system during simulation in blue. In red are the safety set bounds (24).

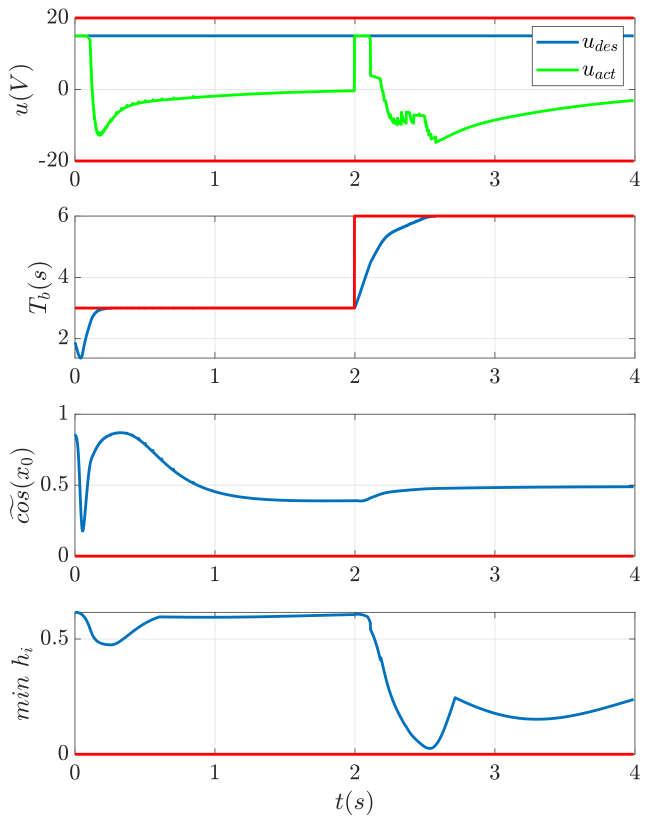


Fig. 6: Information about the SF during simulation. In red are the different constraints enforced by the SF.

[3] Jean-Pierre Aubin. *Viability theory*. Springer Science, 2009.

[4] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 2005.

[5] Xiangru Xu, Jessy W Grizzle, Paulo Tabuada, and Aaron D Ames. Correctness guarantees for the composition of lane keeping and adaptive cruise control. *IEEE Transactions on Automation Science and Engineering*, 2017.

[6] Jeremy H Gillula, Shahab Kaynama, and Claire J Tomlin. Sampling-based approximation of the viability kernel for high-dimensional linear sampled-data systems. In *Proceedings of the 17th international conference on Hybrid systems*, pages 173–182. ACM, 2014.

[7] Ian Mitchell. A summary of recent progress on efficient parametric approximations of viability and discriminating kernels. In *SNR@ CAV*, pages 23–31, 2015.

[8] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[9] Thomas Gurriet, Andrew Singletary, Jake Reher, Laurent Ciarletta, Eric Feron, and Aaron Ames. Towards a framework for realizable safety critical control through active set invariance. In *Proceedings of the 9th International Conference on Cyber-Physical Systems*. IEEE, 2018.

[10] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015.

[11] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[12] Thomas Gurriet, Mark Mote, Aaron D Ames, and Eric Feron. An online approach to active set invariance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3592–3599. IEEE, 2018.

[14] Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*. Springer, 2008.

[13] Aleksej Fedorovič Filippov. *Differential equations with discontinuous righthand sides: control systems*, volume 18. Springer Science & Business Media, 2013.

[15] Randy Freeman and Petar V Kokotovic. *Robust nonlinear control design: state-space and Lyapunov techniques*. Springer Science & Business Media, 2008.

[16] Benjamin J Morris, Matthew J Powell, and Aaron D Ames. Continuity and smoothness properties of nonlinear optimization-based feedback controllers. In *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pages 151–158. IEEE, 2015.

[17] Tom Schouwenaars. *Safe trajectory planning of autonomous vehicles*. PhD thesis, Massachusetts Institute of Technology, 2005.

[18] Stanley Bak, Deepti K Chivukula, Olugbemiga Adekunle, Mu Sun, Marco Caccamo, and Lui Sha. The system-level simplex architecture for improved real-time embedded system safety. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 99–107. IEEE, 2009.

[19] John M Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–29. Springer, 2003.

[20] Hans Seywald and Renjith R Kumar. Desensitized optimal trajectories. *Advances in the Astronautical Sciences*, 93(1):103–116, 1996.

[21] Ian A Hiskens and MA Pai. Trajectory sensitivity analysis of hybrid systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(2):204–220, 2000.

[22] Juan-Pablo Afman, Laurent Ciarletta, Eric Feron, John Franklin, Thomas Gurriet, and Eric N Johnson. Towards a new paradigm of uav safety. *arXiv preprint arXiv:1803.09026*, 2018.

[23] Michael A Patterson and Anil V Rao. Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[24] Karsten Ahnert and Mario Mulansky. Odeint–solving ordinary differential equations in c++. In *AIP Conference Proceedings*, volume 1389, pages 1586–1589. AIP, 2011.

[25] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*, November 2017.