IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 28, 2020, accepted August 17, 2020, date of publication September 18, 2020, date of current version October 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3025248

A Scalable Safety Critical Control Framework for Nonlinear Systems

THOMAS GURRIET[1], MARK MOTE[2], ANDREW SINGLETARY[1], PETTER NILSSON[1], (Associate Member, IEEE), ERIC FERON[3], AND AARON D. AMES[1], (Senior Member, IEEE)

[1]Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125, USA
[2]Department of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[3]Department of Electrical and Computer Engineering, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia

Corresponding author: Thomas Gurriet (tgurriet@caltech.edu)

This work was supported in part by Wandercraft; in part by the Caltech Big Ideas and ZEITLIN Funds; in part by the NASA JPL President's and Director's Fund; in part by the NSF Graduate Research Fellowship No. DGE-1745301; and in part by NSF under Award 1724464, Award 1544332, Award 1724457, and Award 1446758.

ABSTRACT There are two main approaches to safety-critical control. The first one relies on computation of control invariant sets and is presented in the first part of this work. The second approach draws from the topic of optimal control and relies on the ability to realize Model-Predictive-Controllers online to guarantee the safety of a system. In the second approach, safety is ensured at a planning stage by solving the control problem subject for some explicitly defined constraints on the state and control input. Both approaches have distinct advantages but also major drawbacks that hinder their practical effectiveness, namely scalability for the first one and computational complexity for the second. We therefore present an approach that draws from the advantages of both approaches to deliver efficient and scalable methods of ensuring safety for nonlinear dynamical systems. In particular, we show that identifying a backup control law that stabilizes the system is in fact sufficient to exploit some of the set-invariance conditions presented in the first part of this work. Indeed, one only needs to be able to numerically integrate the closed-loop dynamics of the system over a finite horizon under this backup law to compute all the information necessary for evaluating the regulation map and enforcing safety. The effect of relaxing the stabilization requirements of the backup law is also studied, and weaker but more practical safety guarantees are brought forward. We then explore the relationship between the optimality of the backup law and how conservative the resulting safety filter is. Finally, methods of selecting a safe input with varying levels of trade-off between conservatism and computational complexity are proposed and illustrated on multiple robotic systems, namely: a two-wheeled inverted pendulum (Segway), an industrial manipulator, a quadrotor, and a lower body exoskeleton.

INDEX TERMS Safety-critical control, nonlinear control, real-time optimization, optimal control, viability theory, barrier functions.

I. INTRODUCTION

Safety is arguably one of the most critical issues hindering the democratization of autonomous systems. Even 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.

The associate editor coordinating the review of this manuscript and approving it for publication was Wonhee Kim.

A system is commonly defined to be safe when 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 ensuring 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 it 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 can entirely explore while remaining capable of staying inside it for all times.

The largest control invariant subset of the safety set is called the *viability kernel*. Finding the viability kernel grants 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 degraded performances for the system. Nonetheless, given a control invariant set, safety of the system can then be easily guaranteed by continuously filtering the control signal in a minimally invasive way as proposed in [8] (see [9]–[11] for applications).

However, computing viable sets is only one possible approach for ensuring safe operation of a system. Another popular class of methods relies on predicting systems' trajectories to guarantee safety. A backup strategy is chosen, and the trajectory of the system under that backup control law is computed online at every instant. Guaranteeing safety of the system can then be achieved by switching between the nominal and backup controller intelligently based on the safety of the backup trajectory [12], [13]. Multiple backup strategies can also be chosen from on the fly with a same underlying switching strategy [14], [15]. Furthermore, the backup strategy can be determined on the fly so as to adapt to the situation and be as minimally invasive as possible [16]–[18]. This last methodology is at the heart of path planning and optimal control research, but even though it potentially yields the best system performances, its complexity made practical applications favor the simpler alternatives discussed before.

In this work, we propose to unify both set-based and trajectory-based approaches and show that they are really just two sides of the same coin. We present a safety critical control framework that combines the strengths of both approaches to deliver efficient and scalable methods of ensuring safety for complex dynamical systems. First, we present the safety-filtering methodology from a set-based perspective. We then show how it is possible to systematically define a control-invariant subset of the safety set, namely a Safe Backward Image (SBI) of the backup set. This set is defined implicitly from a backup control law and a backup set, and the implementation details of a safety filter in that context are then presented. We then show that one can relax the stability requirement on the backup control law and still get meaningful albeit weaker safety guarantees. In a following section, we explore the relation between optimality of the backup control law and size of the Safe Backward Image. In the case of linear systems we show that it is possible to implement and couple a Model Predictive Controller and a Safety Filter to obtain the largest possible Safe Backward Image. Finally, methods of selecting a safe input with varying levels of trade-off between conservativeness and computational complexity are proposed and illustrated on relevant systems and applications, namely: a two-wheeled inverted pendulum (Segway), an industrial manipulator, a quadrotor, and a lower body exoskeleton.

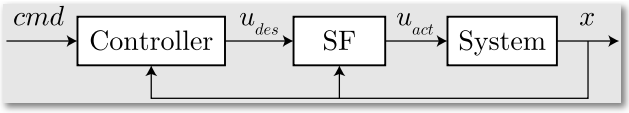


FIGURE 1. Safety filtering control structure.

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 to get as complexity of the controller increases. As an alternative paradigm, we consider the control structure depicted in Fig. 1. The idea here is to decouple performance from 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 [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. Realizing 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 control affine dynamical systems of the form

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

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 $u : \mathbb{R}^+ \times X \longrightarrow \mathbb{R}^n$ 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 $\forall t \in \mathbb{R}^+$, $u(t, x) \in U$.

Under these assumptions, system (1) is guaranteed to have a unique solution over a time interval $[0, T_X]$ for any initial condition $x(0) \in \text{Int}(X)$ and with $T_X > 0$ being the time when solutions to (1) leave X [19].

Let us denote the chosen safety set by $\overline{S} \subset X$ and require that it is compact. We will say that the system is **safe** as long as its state is in $\overline{S}$. The goal of Active Set Invariance Filtering is to ensure that the system remains safe for all time. But as discussed above, if $\overline{S}$ is chosen arbitrarily, it will contain states that cannot be visited without inexorably leading to

the system leaving the safety set—namely **dangerous states**. Arbitrary safety sets that have not been chosen with care are therefore fundamentally dangerous as they almost always contain dangerous states. Let us now assume that we have access to a non-empty set $S \subseteq \overline{S}$ and that S is control invariant for (1).

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

A control invariant set S is therefore very interesting as it does not contain any dangerous states. It is therefore possible to guarantee safety of the system by only ensuring the invariance of S. We will therefore call any control invariant subset of the safety set a **safe set**.

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

Having access to S thus makes the controlled set invariance task much easier. Firstly, it provides a sufficient condition for the initial state to not be dangerous—$x(t_0) \in S$. Secondly it allows us to use a local characterization of invariance originally provided by Nagumo in 1942 [20] and specialized here for our application [3]. Let $\mathcal{T}_S(x)$ denote the **contingent cone** to S at x [3], [20].

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

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

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

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

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

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

encodes capture the safety of the system at an input level. Indeed, if the control signal u takes values in $\overline{U_S}$ for all times, i.e. u is a **selection** of $\overline{U_S}$, then the **sub-tangentiality condition** (2) is trivially verified, guaranteeing the safety of the system.

B. SUB-REGULATION MAP

Depending on the type of set considered, there are different ways of expressing the contingent cone. Practical sets—as defined in [20]—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]:

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

[1] See [20, p. 103] for all conditions under which S is practical.

For such sets, the contingent cone can be expressed as

$$\mathcal{T}_S(x) = \left\{z \in \mathbb{R}^n \mid \forall i \in \mathrm{Act}(x),\ \nabla h_i(x) \cdot z \geq 0\right\}, \tag{5}$$

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

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

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

$$\overline{U_S}(x) = \{u \in U \mid \forall i \in \mathrm{Act}(x),\ TC_i(x,u) \geq 0\} \tag{7}$$

The sub-tangentiality condition is however not very practical as it only defines a non-trivial set of admissible inputs when the system is on the boundary of S, i.e. $\mathrm{Act}(x) = \emptyset$ if $x \notin \partial S$, which leads to a discontinuous regulation of the control input. Furthermore, it is not defined outside of S which makes any real implementation actually impossible. One solution to smooth out the regulated input and make the safety filter implementation possible is explored in [3]—further developed in [8]. It consists in considering a strengthening term in (6) and to impose a **barrier condition**:

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

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 **sub-regulation map** U_S of admissible inputs:

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

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 [3], [9], 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. 1) that can enforce safety of the system synergistically with the performance goals 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 as the following quadratic program:

Safety Filter QP

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

If S is control invariant and Assumptions 1 hold, $\overline{U_S}$ has non-empty compact convex values, but it is not necessarily the

case for U_S. As discussed before, this can be addressed by either choosing a control invariant set S and choosing the α_i functions accordingly so as to ensure that U_S is never empty, or by first choosing the α_i functions and then computing a set S over which U_S is never empty.

Alternatively, given a control invariant set S, barriers conditions can be used along with its smoothing action without having to carefully choose the α_i by using the following safety filter formulation:

Relaxed Safety Filter QP

$$\begin{aligned} u_{\text{act}}(t,x) = \ & \underset{u,\alpha}{\text{argmin}} \ \|u_{\text{des}}(t) - u\|^2 + M\alpha^2 \\ & \text{s.t. } L_f h_i(x) + L_g h_i(x) u + \alpha h_i(x) \geq 0 \\ & \alpha \geq 0 \\ & u \in U \end{aligned} \tag{11}$$

This way, the *slowing down* effect of a barrier condition can be chosen and followed if possible, but otherwise relaxed in way that still ensures that the resulting filter input is in $\overline{U_S}$.

Ideally, one would want to use the largest control invariant subset S of $\overline{S}$ (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. We will now show how it is possible have access to a **large** control invariant subset of $\overline{S}$ while only having to explicitly compute a **small** control invariant subset of $\overline{S}$—which is comparatively easy.

III. IMPLICIT SAFETY FILTERING

The key to our approach is to realise that, in practice, an explicit representation of a control invariant set S is not necessary. If S is practical and can be defined as in (4), then one only needs to be able to numerically evaluate $h_i(x)$ and $\nabla h_i(x)$ for any given state $x \in S$ quickly enough for the safety filter to run in real-time. We therefore propose a way to systematically define a control invariant subset of $\overline{S}$.

A. IMPLICIT CONTROL INVARIANT SET

Our approach for defining such a set is inspired by [21]. The idea is to start with a ''seed of safety'': the **backup set**, that is easy to compute explicitly and provide infinite time horizon guarantees, i.e it is control invariant. Ideally, this backup set would be big enough so that we can use it directly for safety filtering, but as discussed previously, explicit safe sets are hard to compute which leads to conservative results and poor performance for high-dimensional systems. Therefore, we chose to ''implicitly expand'' the backup set over an additional finite time horizon through the flow of the system under a carefully chosen **backup control law**. This way, if this implicit expansion is constructed properly, we get access to a larger control invariant subset of the safety set without going through the time-consuming process of computing an explicit representation of this large set. We will call this implicit expansion of the backup set the **safe backward image** of the backup set. Indeed, as we are about to see, the key for defining a set that is control invariant is to consider all the states that can safely reach the backup set.

Assumption 2: 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 all 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. [22]) for all $t \in [0, T_X]$. We will denote by $S_b \subseteq \overline{S}$ the non-empty compact **backup set** as depicted in Fig. 2. We furthermore assume that S_b is practical and can be represented as the super level set of a smooth function $h_b : \mathbb{R}^n \to \mathbb{R}$.

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:

$$R_T^{u_b} \triangleq \left\{ x \in X \mid \phi_T^{u_b}(x) \in S_b \right\} \tag{12}$$

is a set encoding the **reachability** of the backup set under the backup control law and:

$$\Omega_T^{u_b} \triangleq \left\{ x \in \overline{S} \mid \forall t \in [0,T], \ \phi_t^{u_b}(x) \in \overline{S} \right\} \tag{13}$$

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

Because $R_T^{u_b} = \left(\phi_T^{u_b}\right)^{-1}(S_b)$ and $\phi_T^{u_b}\left(R_T^{u_b}\right) = S_b$, the definition of forward invariance can be reformulated in terms of the flow. Indeed, $x(t_0) \in S \Rightarrow \forall t \geq t_0, \ x(t) \in S$ is equivalent to $\forall t \geq 0, \ \phi_t^{u_b}(S) \subseteq S$. Hence the following propositions.

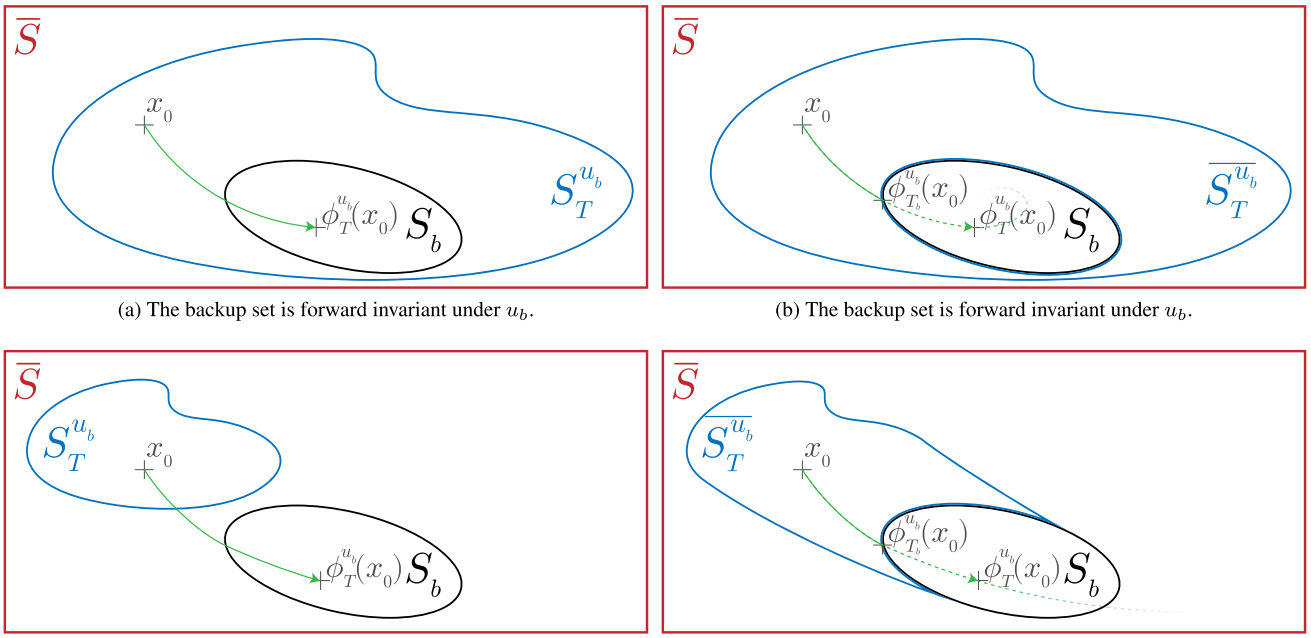
Proposition 2: If S_b is forward invariant under $u_b \in \mathcal{U}$, then for all $T \geq 0$, $R_T^{u_b}$ is forward invariant.

Proof: Let us reason by contraction and assume that $R_T^{u_b}$ is not forward invariant. That means there exist $x^* \in R_T^{u_b}$ and $t^* \geq 0$ such that $\phi_{t^*}^{u_b}(x^*) \notin R_T^{u_b}$. Let $\tilde{x} \triangleq \phi_{t^*}^{u_b}(x^*)$. By property of the flow, $\phi_T^{u_b}(\tilde{x}) = \phi_{t^*}^{u_b}\left(\phi_{T-t^*}^{u_b}(\tilde{x})\right)$. But $\phi_{T-t^*}^{u_b}(\tilde{x}) = \phi_T^{u_b}(x^*) \triangleq x_b \in S_b$. So $\phi_T^{u_b}(\tilde{x}) = \phi_{t^*}^{u_b}(x_b) \in S_b$ since S_b is forward invariant. This implies that $\tilde{x} \in R_T^{u_b}$, hence the contradiction that proves the proposition. □

Proposition 3: The set S_b is forward invariant under $u_b \in \mathcal{U}$ if and only if $S_b \subseteq R_T^{u_b}$ for all $T \geq 0$.

Proof: Let us first assume that for all $T \geq 0$, $S_b \subseteq R_T^{u_b}$. By definition of $R_T^{u_b}$ and because $\phi_T^{u_b}$ is a homeomorphism, $\forall T \geq 0, \ \phi_T^{u_b}\left(R_T^{u_b}\right) = S_b$. So for all $T \geq 0$, $\phi_T^{u_b}(S_b) \subseteq S_b$ which proves the necessity of forward invariance. The sufficiency of forward invariance follows directly from Prop. 2, as for all $T \geq 0$, $\phi_T^{u_b}\left(R_T^{u_b}\right) = S_b$ and $\phi_T^{u_b}\left(R_T^{u_b}\right) \subseteq R_T^{u_b}$ so $S_b \subseteq R_T^{u_b}$. □

These two propositions are actually fairly intuitive as they indicate that if the backup control law stabilizes the backup set, then the backward reachable set of the backup set is an



(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.

FIGURE 2. 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.

invariant set larger than the backup set. Let us now see how we can analytically describe this set.

Proposition 4: Given $u_b \in \mathcal{U}$ and a forward invariant set $S_b = \{x \in \mathbb{R}^n \mid h_b(x) \geq 0\}$, then:

$$R_T^{u_b} = \left\{x \in \mathbb{R}^n \mid h_b \circ \phi_T^{u_b}(x) \geq 0\right\}. \tag{14}$$

Proof: Consider $x \in R_T^{u_b}$, then $\phi_T^{u_b}(x) \in S_b$. So $h_b\left(\phi_T^{u_b}(x)\right) \geq 0$, hence $x \in \left\{x \in \mathbb{R}^n \mid h_b\left(\phi_T^{u_b}(x)\right) \geq 0\right\}$. Let us now consider $x \in \left\{x \in \mathbb{R}^n \mid h_b\left(\phi_T^{u_b}(x)\right) \geq 0\right\}$, then $\phi_T^{u_b}(x) \in S_b$, hence $x \in R_T^{u_b}$. □

The description for $\Omega_T^{u_b}$ is similar.

Proposition 5: Given $u_b \in \mathcal{U}$ and $\overline{S}$ described as in (4), then:

$$\Omega_T^{u_b} = \bigcap_{t\in[0,T]} \left\{x \in X \mid h_i \circ \phi_t^{u_b}(x) \geq 0,\ i \in \{1,\dots,N_s\}\right\}, \tag{15}$$

or equivalently:

$$\Omega_T^{u_b} = \left\{x \in X \mid h_{S_T^{u_b}}(x) \geq 0\right\}, \tag{16}$$

with:

$$h_{S_T^{u_b}}(x) \triangleq \min_{\substack{t\in[0,T]\\ i\in\{1,\dots,N_s\}}} h_i \circ \phi_t^{u_b}(x). \tag{17}$$

We can now state the core theoretical proposition of this work.

Proposition 6: If $S_b \subseteq \overline{S}$ is forward invariant under $u_b \in \mathcal{U}$, then for all $T \geq 0$, the safe backward image $S_T^{u_b}$ is a subset of S that is forward invariant under that control law.

Proof: The fact that $S_T^{u_b} \subseteq \overline{S}$ follows trivially from the definition of $\Omega_T^{u_b}$. Let us reason by contradiction and assume that $S_T^{u_b}$ is not forward invariant. This means that there exist $x^* \in S_T^{u_b}$ and $t^* \geq 0$ such that $\phi_{t^*}^{u_b}(x^*) \notin S_T^{u_b}$. But from Prop. 2, we know that $R_T^{u_b}$ is forward invariant so $\phi_{t^*}^{u_b}(x^*) \in R_T^{u_b}$ and $\phi_{t^*}^{u_b}(x^*) \notin \Omega_T^{u_b}$. This implies that there exists $t^\# \geq 0$ such that $\phi_{t^\#}^{u_b}(x^*) \notin S$. But $x^* \in \Omega_T^{u_b}$, so $t^\# > T$, i.e. there exists $t' > 0$ such that $\phi_{t'}^{u_b}\left(\phi_T^{u_b}(x^*)\right) \notin S$. But $x^* \in R_T^{u_b}$, so $\phi_T^{u_b}(x^*) \in S_b$ and because S_b is forward invariant, $\phi_{t'}^{u_b}\left(\phi_T^{u_b}(x^*)\right) \in S_b$, which contradicts $S_b \subseteq S$. □

It trivially follows from this last proposition that $S_T^{u_b}$ is a control invariant subset of $\overline{S}$, so it can be used to define a non-empty sub-regulation map $U_{S_T^{u_b}}$. The challenge now is to be able to evaluate this regulation map, as it only has an implicit expression inherited from the implicit nature of the construction of $S_T^{u_b}$. Let us now see how we can tackle this issue.

B. IMPLICIT SAFETY FILTER

1) IMPLICIT SUB-REGULATION MAP

In order to realize a safety filter, one must be able to evaluate the regulation map $U_S(x)$. This is easy if S has an explicit representation, which is why so much effort has been focused on finding such a representation. In our framework 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 $S_T^{u_b}$.

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

$$\begin{cases} \nabla h_b(x_T)\, D\phi_T^{u_b}(x_0) f_0(u) + \alpha_0\left(h_b(x_T)\right) \geq 0 \\ \nabla h_i\left(x_{t_b}\right) D\phi_{t_b}^{u_b}(x_0) f_0(u) + \alpha_i\left(h_i\left(x_{t_b}\right)\right) \geq 0, \end{cases} \tag{18}$$

for all $i \in \{1,\dots,N_s\}$, all $t_b \in [0,T]$, and with $x_{t_b} \triangleq \phi_{t_b}^{u_b}(x_0)$ and $f_0(u) \triangleq f(x_0) + g(x_0)u$.

Two issues arise at the sight of this expression of the sub-regulation map. First, the gradient of the flow—whose existence is guaranteed from the smoothness assumptions in 1 and 2—needs to be computed. Second, because t_b lives in the interval $[0, T]$, the images of $U_{S_T^{u_b}}$ are formed by an **uncountable** set of constraints, which elevates the safety filter's underlying optimization problem into the class of **robust optimization problems**, which is hard, if not impossible to solve in real-time. Let us therefore see how we can address these issues.

2) NUMERICAL APPROXIMATION OF THE SUB-REGULATION MAP

The practical solution to these issues is to recourse to numerical integration tools. By numerically integrating (1) forward in the interval $[0, T]$ under the backup law u_b, $\phi_{t_b}^{u_b}(x_0)$ can be numerically evaluated a discrete times $\{t_{b,0}, \ldots, t_{b,N_b}\}$. 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 becomes.

Then, to compute $D\phi_{t_b}^{u}(x_0)$, one only needs to integrate along with (1) a **sensitivity** matrix $Q(t_b, x_0)$. As explained in [23], the square matrix $Q(t_b, x_0)$ solution of the following differential equation:

$$\frac{dQ(t, x_0)}{dt_b} = Df_{cl}\left(\phi_{t_b}^{u_b}(x_0)\right) Q(t_b, x_0), \tag{19}$$

with $Q(0, x_0) = I$ and where $f_{cl}(x) \triangleq f(x) + g(x)u_b(x)$ is exactly the Jacobian of the flow $\phi_{t_b}^{u_b}$ at x_0:

$$Q(t_b, x_0) = D\phi_{t_b}^{u_b}(x_0). \tag{20}$$

Note that it is important for the backup control law used to smooth, which means that in the case of finite input bounds, a smooth saturation function has to be used for the expression of $u_b(x)$.

3) UNDER-APPROXIMATION OF THE SUB-REGULATION MAP

We now turn to the issue of having an infinite number of functions defining the set. In practice, we can only enforce positivity of a finite number of the functions in (18), and therefore propose a safety filter that enforces positivity of a finite subset of ϵ-tightened constraints evenly spaced in time:

$$\nabla h_b(x_T) D\phi_T^{u_b}(x_0) f_0(u) + \alpha_0(h_b(x_T)) \geq 0, \tag{21a}$$
$$\nabla h_i(x_{t_{b,k}}) D\phi_{t_{b,k}}^{u_b}(x_0) f_0(u) + \alpha_k(h_b(x_{t_{b,k}}) - \epsilon_i) \geq 0, \tag{21b}$$

for $i \in [\![1, N_s]\!]$ and $k \in [\![0, N_b]\!]$.

Although this just enforces positivity of a finite number of constraints, under some regularity conditions and appropriate margins ϵ_i, we expect that this should be sufficient to guarantee positivity of the whole family of functions. We make this more precise below via the following lemma.

Lemma 1: Let L_h be the Lipschitz constant of a function h with respect to the Euclidean norm and let:

$$L_\phi = \sup_{x \in \bar{S}} \|f(x) + g(x)u_b(x)\|_2, \tag{22}$$

be the maximal velocity of the backup vector field. Then:

$$\left|h \circ \phi_t^{u^B}(x) - h \circ \phi_s^{u^B}(x)\right| \leq L_h L_\phi |t - s|. \tag{23}$$

Proof: Assume WLOG that $t \geq s$ and let $y = \phi_s^{u^B}(x)$. Then:

$$\left|h \circ \phi_t^{u^B}(x) - h \circ \phi_s^{u^B}(x)\right| \leq L_h \left\|\phi_t^{u^B}(x) - \phi_s^{u^B}(x)\right\|_2 = L_h \left\|\phi_{t-s}^{u^B}(y) - y\right\|_2 \leq L_h L_\phi |t - s|,$$

since L_ϕ is the maximal velocity of the vector field. □

It follows that invariance of $S_T^{u_b}$ can be enforced via the finite subset of constraints in (21) provided that the times $t_{b,k}$ are spaced tightly enough.

Theorem 1: Let $\eta_i = \max_{k \in [\![1, N_i - 1]\!]} t_{b,k+1} - t_{b,k}$ be the granularity of the time discretization. If for all i it holds that $\epsilon_i \geq L_{h_i} L_\phi \frac{\eta_i}{2}$, then (21) enforce invariance of $S_T^{u_b}$.

Proof: By the same reasoning as before, the constraints (21) imply that $h_i(x_{t_{b,k}}) = h_j \circ \phi_{t_{b,k}}^{u_b}(x) \geq \epsilon$ for all $k \in [\![1, N_i]\!]$ for all times. Therefore, by Lemma 1, we can for each $t \in [0, T]$ find a k^* such that:

$$\left|h_i(x_t) - h_i(x_{t_{b,k^*}})\right| \leq L_{h_i} L_\phi \frac{\eta_i}{2}, \tag{24}$$

meaning that:

$$h_i(x_t) \geq \epsilon - L_{h_i} L_\phi \frac{\eta_i}{2} \geq 0. \tag{25}$$

Thus all the functions defining $S_T^{u_b}$ are positive, and hence $S_T^{u_b}$ is invariant. □

C. NUMERICAL EXAMPLE

We now illustrate these ideas on a simple example of nonlinear inverted pendulum. This system is defined by the state $x = \left[\theta, \dot{\theta}\right]^\top$ and the dynamics:

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \sin(\theta) + u \end{bmatrix} \tag{26}$$

with a saturated input $u \in [-u_{max}, u_{max}]$ with $u_{max} = 1.5$. This pendulum is upright at an unstable equilibrium when $\theta = 0$. The safety set is chosen to be a box centered at the origin and of edge size 2π (cf. Fig. 3).

The first step is to choose a backup control law and a backup set. In this example, we consider linear backup laws that stabilize the system to the origin:

$$u_b(x) = -K \cdot x \tag{27}$$

for some gain vector K. A backup set can then be carefully chosen as a level set of a quadratic Lyapunov function of the linearized dynamics of the system around the origin. As hinted at before, higher backup gains yield in general a larger SBI, as illustrated in Fig. 3. It is therefore important

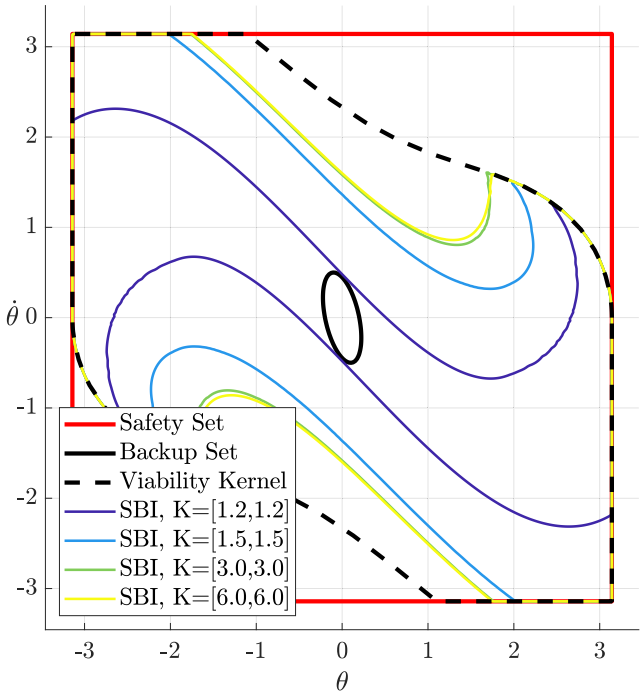


FIGURE 3. Plot of the SBI for different backup gains K with $T = 5$.

to choose a *good* backup law as we will discuss further in Sec. V.

Given a backup control law, a backup horizon T has to be chosen. As one can expect, the larger this time horizon is, the larger the resulting SBI also is, as illustrated in Fig. 4.

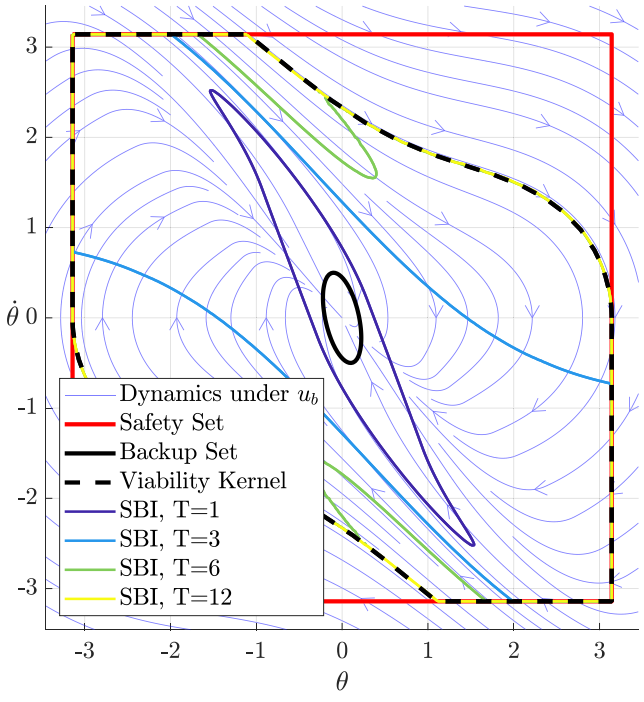


FIGURE 4. Plot of the SBI for different time horizons T.

Given a choice of backup law, set, and horizon, a safety filter can be implemented using numerical integration as explained in Sec. III-B2. At each safety filter iteration, the dynamics of the system are integrated under the backup law over a time horizon T. From the discrete values of the state and sensitivity matrix over this backup trajectory, a set of linear constraints approximating the regulation map can be constructed. Finally, a quadratic program can be solved to find the best input satisfying the safety constraint encoded by the regulation map. Some trajectories of the system under this safety filter can be seen in Fig. 5. In this illustration, the pendulum is started from various initial angles with zero velocity.

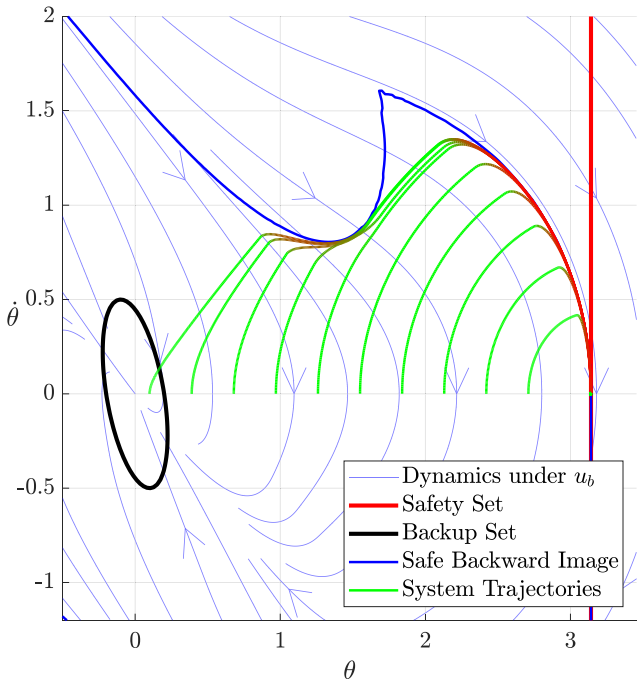


FIGURE 5. Trajectories of the system with $u_{des} = 0$, $T = 5$ and $K = [3, 3]$. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

The value of the backup horizon therefore has a crucial impact on the applicability of the method as one has to be able to do the numerical integration and the QP solving fast enough for the safety filter to run in real-time. Note also that the computational complexity of the backup law and the dynamics of the system have a non negligible impact on the speed of the safety filter computations as they have to be evaluated numerous times for the integration of the backup trajectory.

Finally, it is important to note that, although we compute the SBI explicitly in this example for the purpose of illustration, at no point it is required to do so for the safety filter to operate. Through the numerical scheme we propose, we can render the SBI forward invariant without having to find an explicit representation of it. This makes this framework applicable to nonlinear systems and does not require any particular

analytical structure for the dynamics of the system. As we will see in Sec. VII, not having to explicitly compute a control invariant set has important advantages in practice.

IV. FINITE TIME SAFETY GUARANTEES

So far, the backup policy and the backup set cannot be chosen arbitrarily as the former has to be invariant under the latter, which still hinders the scalability of the proposed approach. This comes from our desire for the safe backward image to be a control invariant set, i.e. a set in which the system can remain and evolve forever. In practice however, safety requirements often do not necessitate that the system be able to run forever, but only that the system be able to safely stop or terminate. In this section, we discuss an extension to the proposed approach that allows a safety filter to be used to enforce such safety requirements. As we will see, this extension allows for the backup controller and backup set to be chosen independently which makes our approach truly scalable in this case.

A. REFORMULATION OF THE REACHABILITY CONSTRAINT

The idea here is to relax the reachability constraint of $R_T^{u_b}$. For that, we will restrict ourselves to backup sets that can be represented as the upper level-set of a single function h_b **twice differentiable**. In that context we 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

$$T_b(x) = \min \left\{ t \geq 0 : h_b \left(\phi_t^{u_b}(x) \right) = 0 \right\}. \tag{28}$$

When $x \in S_b$, we choose $T_b(x) = 0$, and when $x \notin S_b$ and a solution to (28) does not exist, we choose $T_b(x) = +\infty$. Let us now consider the set $\overline{R_T^{u_b}}$ given by the closure:

$$\overline{R_T^{u_b}} \triangleq \overline{\{x \in X \mid 0 < T_b(x) \leq T\}}, \tag{29}$$

with $T > 0$ and $u_b \in \mathcal{U}$.

The interest of considering such a set becomes clear when realizing that if S_b is not invariant under u_b, S_b is not a subset of $R_T^{u_b}$ (cd. Fig. (2c)). This makes the set $R_T^{u_b}$ unusable as it is not even guaranteed that $R_T^{u_b} \cap \overline{S} \neq \{\emptyset\}$ 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 to grow monotonically with T (cf. Fig. (2d)).

B. AUGMENTED REGULATION MAP

Similarly to Sec. 187252, we would like to regulate safe solutions using the sub-regulation map U_S while utilizing this new 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

$$-\nabla T_b(x_0) \tilde{f}(x_0, u) + \alpha (T - T_b(x_0)) \geq 0, \tag{30}$$

where $\tilde{f}(x_0, u) \triangleq f(x_0) + g(x_0)u$. As demonstrated in [24]:

$$\nabla T_b(x_0) = -\frac{\nabla h_b(\overline{x}) \cdot D\overline{x}(x_0)}{\nabla h_b(\overline{x}) \cdot \tilde{f}(\overline{x}, u_b(\overline{x}))}, \tag{31}$$

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 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—the safe backward reachable set of the backup set:

$$\overline{S_T^{u_b}} = \overline{R_T^{u_b}} \cap \Omega_T^{u_b} \cap C_T^{u_b}, \tag{32}$$

with

$$C_T^{u_b} \triangleq \left\{ x_0 \in \overline{R_T^{u_b}} \mid \widetilde{cos}(x_0) \geq \varepsilon_b \right\}, \tag{33}$$

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

$$\widetilde{cos}(x_0) \triangleq \frac{\nabla h_b(\overline{x}) \tilde{f}(\overline{x}, u_b(\overline{x}))}{\|\nabla h_b(\overline{x})\| \left\| \tilde{f}(\overline{x}, u_b(\overline{x})) \right\|}. \tag{34}$$

Proposition 7: If $\overline{R_T^{u_b}}$ is not empty, then for all $x_0 \in \overline{S_T^{u_b}} \setminus S_b$ their exist strengthening functions α_i such that $U_{\overline{S_T^{u_b}}}(x_0) \neq \{\emptyset\}$.

Proof: First, note that ∇T_b is not defined on ∂S_b, hence considering only states in $\overline{S_T^{u_b}} \setminus S_b$. Then, $U_{\overline{S_T^{u_b}}}$ 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

$$\begin{cases} \nabla h_i \left(x_{t_b} \right) D\phi_{t_b}^{u_b}(x_0) \tilde{f}(x_0, u) + \alpha_i \left(h_i \left(x_{t_b} \right) \right) \geq 0 \\ -\nabla T_b(x_0) \tilde{f}(x_0, u) + \beta (T - T_b(x_0)) \geq 0 \\ \nabla \widetilde{cos}(x_0) \tilde{f}(x_0, u) + \gamma (\widetilde{cos}(x_0) - \varepsilon_b) \geq 0 \end{cases} \tag{35}$$

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

$$\nabla T_b(x_0) \tilde{f}(x_0, u_b(x_0)) = -1. \tag{36}$$

This is fairly intuitive as (36) is the time derivative of the time to safety when the system is evolving along the backup trajectory. Secondly, because for all $t_b \in [0, T_b(x_0)]$, $\overline{x}\left(\phi_{t_b}^{u_b}(x_0)\right) = \overline{x}(x_0)$,

$$\nabla \widetilde{cos}(x_0) \tilde{f}(x_0, u_b(x_0)) = 0. \tag{37}$$

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

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

So all conditions in (35) 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 1: The extended class $\mathcal{K}$ functions β and γ can be chosen to be arbitrarily.

Remark 2: Computing $\nabla \widetilde{cos}(x_0)$ requires the evaluation of the hessian of h_b at x_0, but that 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 $\bar{x}$ but the details of these numerical issues are outside the scope of this paper.

C. WEAKER BUT PRACTICAL SAFETY GUARANTEES

Let us now study what guarantees we get when regulating the system in $\overline{S_T^{u_b}}$ using $U_{\overline{S_T^{u_b}}}$.

Theorem 2: 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 $T_s < +\infty$, $\phi_{T_s}^{u(t,x(t))}(x) \in S_b$.

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 $\overline{S}$ and within *reach* of the backup set S_b within a finite time T, or reach the backup set in finite time (cf. Fig. 2d). These guarantees may seem weak compared to the ones in (III), but they are actually 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 requested (cf. [25] for more details in the case of UAVs). For commercial aviation and 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 under some mild assumptions, 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. 2b). 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 $\tilde{S}$ for all times. Hence the soundness of this approach that 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. 2).

D. NUMERICAL EXAMPLE

We now illustrate this approach on the nonlinear inverted pendulum of Sec. III-C. In this case, we assume that the pendulum shall not go past $\theta = -\frac{\pi}{2}$ on one side, but that their is a hard stop on the other side at $\theta = \frac{\pi}{2}$ that the system can run into to safely stop. The backup set is therefore chosen to be a narrow band around $\theta = \frac{\pi}{2}$, and the safety set is a rectangle with edges $\theta = -\frac{\pi}{2}$ and $\dot{\theta} = \pm\frac{\pi}{3}$, so opened towards the edge $\theta = \frac{\pi}{2}$ (cf. Fig. 6). The backup policy is chosen to be $u_b(x) = 10 * (\frac{\pi}{10} - \dot{\theta})$ such as to drive the system back towards the hard stop. In this case, the backup set is certainly not invariant under the backup policy and so isn't the safe backward image of the backup set as illustrated in Fig. 6. The safe backward reachable set on the other end is well suited for this scenario where only finite time safety is required. Indeed, as can be seen in Fig. 6, the safe backward reachable set fully captures the set of safe states, and allows for safe operations. As illustrated in Fig. 7, if the pendulum is left to fall towards negative θs, the associated safety filter slows it down so that it stops before $\theta = -\frac{\pi}{2}$, but if it is left to fall towards positive θs, the filter makes sure the system safely reaches the backup set.

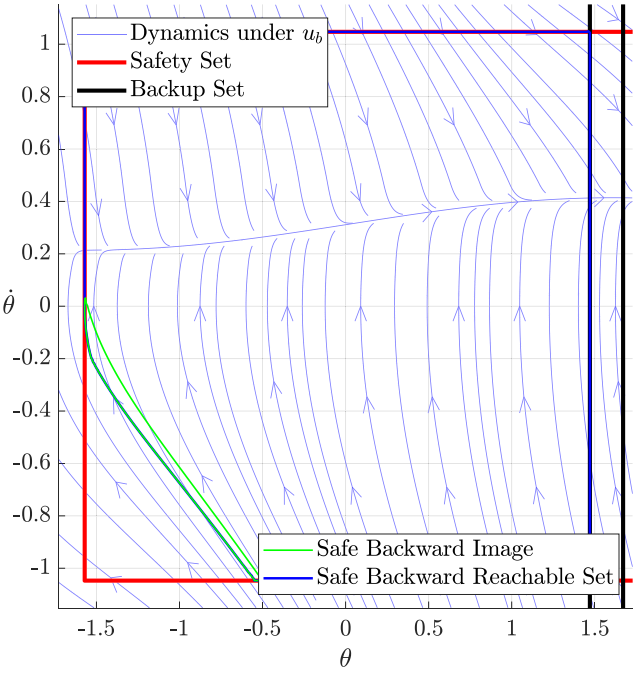


FIGURE 6. Comparison between safe backward image and safe backward reachable set for $T = 11$ and $u_b(x) = 10 * (\frac{\pi}{10} - \dot{\theta})$.

V. OPTIMALITY OF THE BACKUP CONTROLLER

A. MODEL PREDICTIVE BACKUP CONTROLLER

In light of the results illustrated in Fig. 3, a natural question that arises is "what is the *best* backup control law to choose", that is, a control law that maximizes the size of the safe backward image for a given backup set and backup horizon.

To address this question, let us consider the control law u^* given by

$$u^*(x_0) = u^*_{x_0}(0) \tag{38}$$

with $u^*_{x_0}$ being the control policy solution to the following optimal control problem:

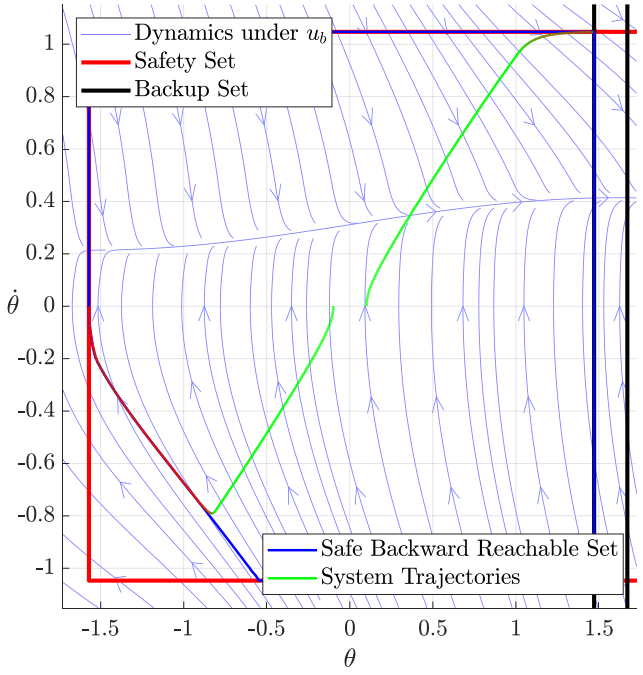


FIGURE 7. Trajectories of the system with $u_{des} = 0$, $T = 11$ and $u_b(x) = 10(\frac{\pi}{10} - \dot{\theta})$. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

Backup MPC

$$\begin{aligned} u^*_{x_0} \triangleq \ & \underset{u \in \mathcal{U}_T}{\operatorname{argmax}}\ J\left(x(t), u(t)\right) \\ & \text{s.t. } \dot{x} = f(x) + g(x)u \\ & \quad x(0) = x_0 \\ & \quad u(t) \in U, \quad \forall t \in [0, T] \\ & \quad x(t) \in \overline{S}, \quad \forall t \in [0, T] \\ & \quad h_b\left(x(T)\right) \geq 0 \end{aligned} \tag{39}$$

for some cost function J and where $\mathcal{U}_T$ denotes the set of all piecewise continuous *control policies*.

Remark 3: The time in (39) is shifted so that $x_0 = x(t_0) = x(0)$ as only time-invariant control systems are considered in this section.

Remark 4: Solutions to (39) might not be unique for a given x_0, in which case the right-hand side of (39) is chosen to be any element in the set of solutions.

It immediately follows that

$$S_T^{u^*} = \left\{ x_0 \in X \mid \exists u \in \mathcal{U}_T, \text{ s.t. } \begin{matrix} x_{x_0}(t) \in \overline{S}, \ \forall t \in [0, T] \\ \text{and } x_{x_0}(T) \in S_b \end{matrix} \right\}, \tag{40}$$

where $x_{x_0}(t)$ is the solution to (1) starting at x_0 at $t = 0$ under the time based control policy $u(t)$.

Therefore, for all $u_b \in \mathcal{U}$, $S_T^{u_b} \subseteq S_T^{u_*}$. We can therefore conclude that u_* is an optimal backup law in the sense that it yields an upper bound on the largest possible safe backward image for a given backup set, as illustrated in Fig. 8. However, u_* is not well suited for the current framework.

A couple of points are important to address if one wants to use the u^* law in the proposed framework. First, the cost function J has to be chosen preferably so as to stabilise S_b (see [21] for such conditions). Secondly, u^* must be smooth (or at least continuous if one uses the filtering approach of Sec. VI), which is rarely the case in general. Finally, if one is able to make u^* smooth, solving a nonlinear MPC online is not an easy task, and is even harder when one has to compute the gradient of the flow under u^* along with the optimal trajectory.

In some cases, these issues can be successfully addressed and an MPC backup controller can be used directly as showcased in [26]. Nevertheless, in the general case, it is not possible to use an MPC backup controller online, and we have to default to a more offline approach. In particular, it is possible to get near-optimal safe backward images by finding a smooth explicit approximation of the optimal backup control law.

B. NEURAL NETWORK APPROXIMATION OF THE MPC

In practice, one can use any functional basis of choice to fit the optimal backup policy, provided that it is fast enough to numerically compute along with its gradient. A smooth functional basis for approximating complex functions can be found with feedforward neural networks, whose recent popularity has made the associated tools very efficient. In particular, evaluating gradients of feedforward neural network is computationally easy.

Indeed, for simple neural networks with recursion law:

$$y_{i+1} = f\left(w_i y_i + b_i\right), \tag{41}$$

the gradient of the entire neural network can be computed in the same forward pass with the following recursion law:

$$dy_{i+1} = \operatorname{diag}\left(\frac{df(y_{i+1})}{dx}\right) w_i dy_i, \tag{42}$$

with dy_0 being the identity matrix of size n (state dimension).

Referring back to the nonlinear inverted pendulum of Sec. III-C, we can see in Fig. 8 that the optimal safe backward image is larger than the largest SBI we found using linear feedback for the backup controller (cf. Fig. 3).

We first solve the OCP (39) with the cost function:

$$J\left(x(t), u(t)\right) = \int_0^T \left(0.1u(t)^2 + 10\theta(t)^2 + \dot{\theta}(t)^2\right) dt \tag{43}$$

over a grid of size 200×200 using GPOPS-II [27]. We then fit a neural network with 2 hidden layers, each of size 35, and with a hyperbolic tangent activation function over the generated data. The resulting safe backward image of this approximately optimal policy, as can be seen in Fig. 8, captures most of the optimally of the OCP, but, is actually usable in the proposed framework. This method is applied on a larger dimensional system in Sec. VII and similarly strong results are observed.

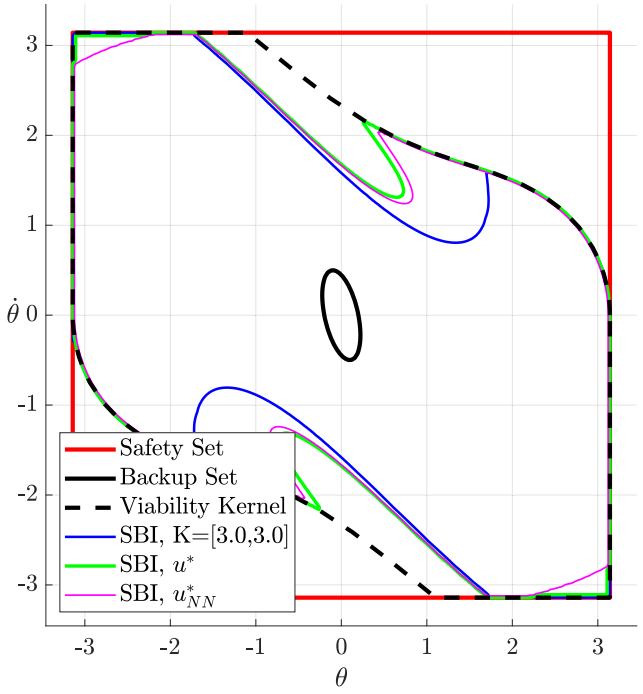


FIGURE 8. Comparison between the safe backward images of a linear backup control law (27), the optimal control law (38), and a Neural Network approximation of it, with $T = 5$.

VI. SCALABLE SAFETY FILTER

So far, we have been able to use an optimization-based safety filter by relying on our ability to numerically evaluate the sub-regulation map at any given state. Although this approach is optimal in terms of filtering, it comes at the cost of having to compute the gradient of the flow along the backup trajectories. The dimension of the system to integrate is therefore $n + n^2$, which can become a computational bottleneck for higher dimensional systems. It is however, possible to sacrifice optimally of the safety filter for better scalability. The key is that **by construction**, the backup law evaluated at the current state is always an element of the image of the regulation map for that state. In other words, if the backup policy is followed for the initial conditions inside $S_T^{u_b}$, the system will remain in the safety set for all time.

A. SMOOTH SWITCHING TO THE BACKUP CONTROL LAW

This idea is at the core of a lot of alternative approaches to safety filtering [12]–[18]. In most of these methods though, the safety filter just operates a simple switch between nominal and backup controller until the system has reached the backup set, which in practice is fairly intrusive.

The natural evolution of this idea is to implement a smooth transition between desired and backup inputs. To that end, let us look at the following proposition.

Proposition 8: Given a nonlinear control system (1) with a corresponding backup controller $u_b \in \mathcal{U}$ and a continuous function $\alpha : \mathbb{R}^+ \times \mathbb{R}^m \times X \times \mathbb{R} \to U$, the control law defined by

$$u_f(t, x) \triangleq \alpha\left(t, u_{des}(t), x, h_{S_T^{u_b}}(x)\right) \tag{44}$$

is a continuous selection of $\overline{U_{S_T^{u_b}}}$ if for all $t \geq 0$ and $x \in S_T^{u_b}$, u_{des} is continuous and:

$$\alpha\left(t, u_{des}(t), x, 0\right) = u_b(x). \tag{45}$$

Proof: This follows trivially from the fact that for all $x \in S_T^{u_b}$, $u_b(x) \in \overline{U_{S_T^{u_b}}}$ and that $\overline{U_{S_T^{u_b}}}$ is non-trivial only when $h_{S_T^{u_b}}(x) = 0$. □

This means that by choosing a **switching function** α appropriately, it is possible to *mimic* the behavior of a QP-based safety filter without the added computational complexity. Note, however, that this approach is fundamentally more conservative than with a QP-based filter as when $h_{S_T^{u_b}}(x) = 0$, it enforces $u_{act}(x) = u_b(x)$ whereas with a QP-based filter, $u_{act}(x) \in U_{S_T^{u_b}}$, which is in general larger than the singleton $u_b(x)$. Nonetheless, it is possible with a proper choice of filtering function to get good performances in practice. Especially since on a significant part of the boundary of the viability kernel of $\overline{S}$, $U_{S_T^{u_b}}$ is actually reduced to a singleton [4].

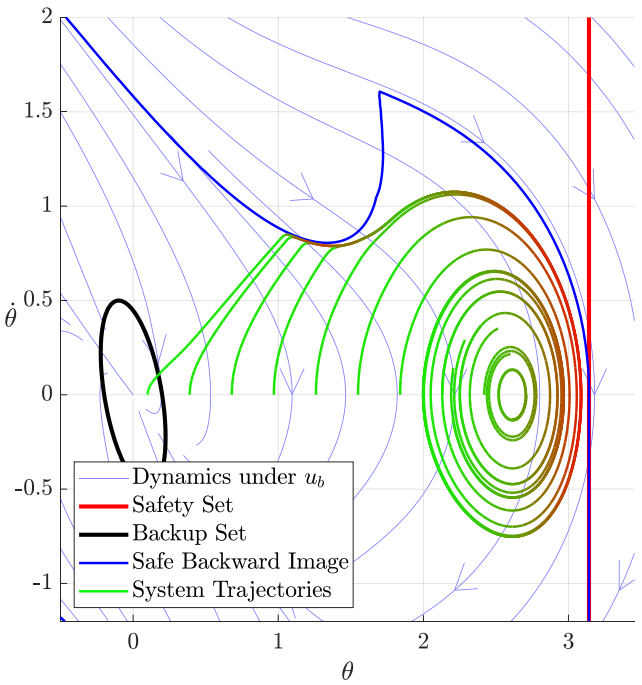


FIGURE 9. Trajectories of the system under a scalable implicit safety filter with $u_{des} = 0$, $T = 5$ and $K = [3, 3]$. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

B. NUMERICAL EXAMPLE

We now illustrate this approach on the nonlinear inverted pendulum of Sec. III-C. The safety specifications are the same as in Sec. III-C. The switching function is first chosen to be a basic ramp up of the backup input near the boundary

of the safe backward image:

$$\alpha\left(t, x, h_{S_T^{u_b}}(x)\right) = \left(1 - h_{S_T^{u_b}}(x)\right)^6 u_b(x). \quad (46)$$

Some trajectories of the system under this scalable safety filter can be seen in Fig. 9. As expected, this safety filter is more conservative than the QP-based one of Fig. 5, which often translates into an oscillatory behavior near the boundary of the safe backward image. However, it is possible to mitigate this behavior with a better choice of filtering function. For example, let us consider the switching function defined by:

$$\alpha\left(t, x, h_{S_T^{u_b}}(x)\right) = \sigma_0^1\left(\lambda(x) + (1-\lambda(x))\lambda_d(t)\right) u_b(x) \quad (47)$$

with

$$\lambda(x) \triangleq \left(1 - h_{S_T^{u_b}}(x)\right)^6, \quad (48)$$

$$\lambda_d(t) \triangleq -\zeta \frac{dh_{S_T^{u_b}}(x(t))}{dt}, \quad (49)$$

and σ_0^1 the saturation function between 0 and 1. As illustrated in Fig. 10, adding such a damping term can help reduce the oscillatory nature of such scalable filters. A further approach for enhancing the performance and applicability of these scalable safety filters will be presented in Sec. VII-D.

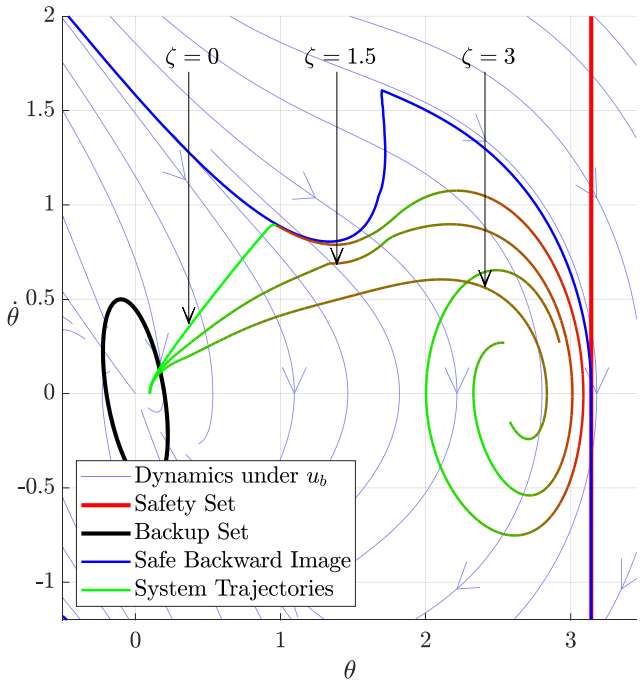


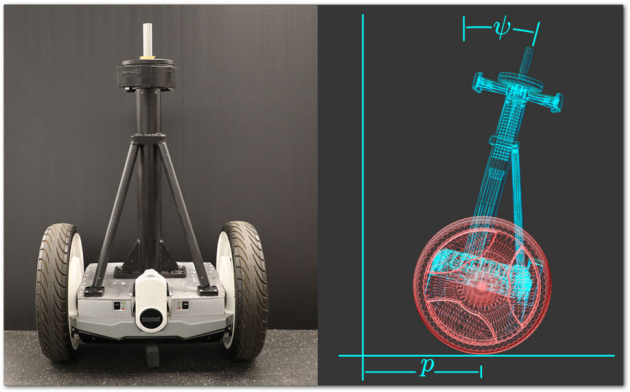
FIGURE 10. Trajectories of the system under a scalable implicit safety filter with $u_{des} = 0$, $T = 5$, $K = [3, 3]$ and different switching functions. The color of the trajectories indicate the magnitude of u_{act}, green corresponding to $u_{act} = u_{des} = 0$ and red to $|u_{act}| = u_{max} = 1.5$.

VII. APPLICATIONS

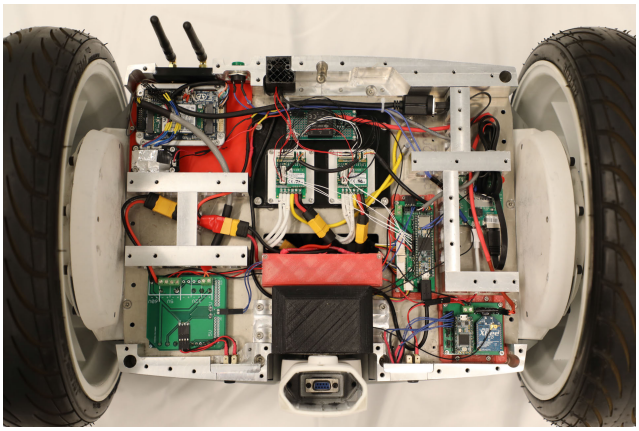
A. TWO-WHEELED INVERTED PENDULUM (SEGWAY)

1) HARDWARE SETUP

The Segway platform used here is a modified a Ninebot E+ with custom electronics and control hardware and software.



(a) General view of the hardware and parameterization of the state



(b) Custom made electronics

FIGURE 11. Segway vehicle used for experiments.

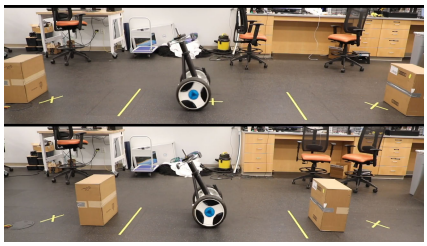
The onboard sensing is performed using wheel incremental encoders and a VectorNav VN-100 IMU. The main onboard computer is a Jetson TX2, which computes the control action that is sent to the motor controllers. The TX2 runs standard Linux and the ERIKA3 real-time operating system concurrently through the Jailhouse hypervisor. The Linux OS runs ROS, which allows external communication and logs all of the necessary data. The real-time operating system handles the low-level communication and the computation of the control actions. These two operating systems are able to share information through a shared memory interface. All of the code running on the Segway is written in C++.

2) SEGWAY TEST

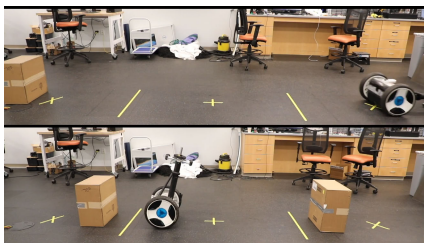
In order to test the implicit safety filter on the Segway, a model of the dynamics is required. The equations of motion are derived via Newton-Euler method, treating the Segway as a two-wheeled inverted pendulum with torque inputs at each wheel. For this experiment, the planar model is used, consisting of four states: position (p), velocity ($\dot{p}$), pitch angle (ψ), and angular rate ($\dot{\psi}$) (cf. Fig. 11a). Since the motor controllers command current, the motor torque constant is estimated via system identification. The other necessary parameters, including the mass and inertia properties of the Segway frame

(a) $t = 3$

(b) $t = 9$



(c) $t = 12.5$



(d) $t = 15$

FIGURE 12. Pictures of the implicit filtering Segway experiment.

and wheels, were measured directly using various custom-made testbeds.

The next step is to define the safety set for the test. This is simply defined as bounds on all of the states, with $p \in [-1, 1]$ m, and $\psi \in [-\frac{\pi}{6}, \frac{\pi}{6}]$ rad. The input bounds are $u \in [-20, 20]$A.

After identifying the system dynamics and determining a safety set, a backup set and backup controller must be generated. The backup set must be a subset of the safety set that is invariant under the backup controller. Thus, we choose a backup controller that stabilizes the Segway to the origin at its equilibrium angle. This is achieved with a simple LQR controller. To obtain the backup set, we compute a small region of attraction for this controller about the origin in the form of a level set of a quadratic Lyapunov function of the linearized dynamics of the system.

To implement the safety filter, a C++ implementation of the proposed safety filter was developed. The library requires an expression for the dynamics, the backup controller, the gradient of the closed-loop dynamics, the backup set, the safety set, and the gradient of the safety set. The library integrates the dynamics using an Euler scheme. The resulting quadratic program is solved using a modified version of OSQP [28] that can be compiled on the real-time operating system.

To showcase the effectiveness of the proposed approach, a simple scenario is executed on the Segway with and without an implicit safety filter. The nominal controller is a simple LQR that can be commanded a desired position. A sequence of desired positions outside of the safety set are commanded, and as can be seen in Fig. 13 and Fig. 12, without safety filter, the system blithely breaches the safety set to the point where it falls to the ground when the command is too aggressive. On the other hand, with the proposed implicit safety filter, the system stably remains inside the safety set despite the unsafe desired inputs. Note that for this experiment, the filter ran at 800Hz on the embedded hardware with a backup horizon $T = 1$s and an integration time-step of 0.01s. A video of this experiment can be found in [29].

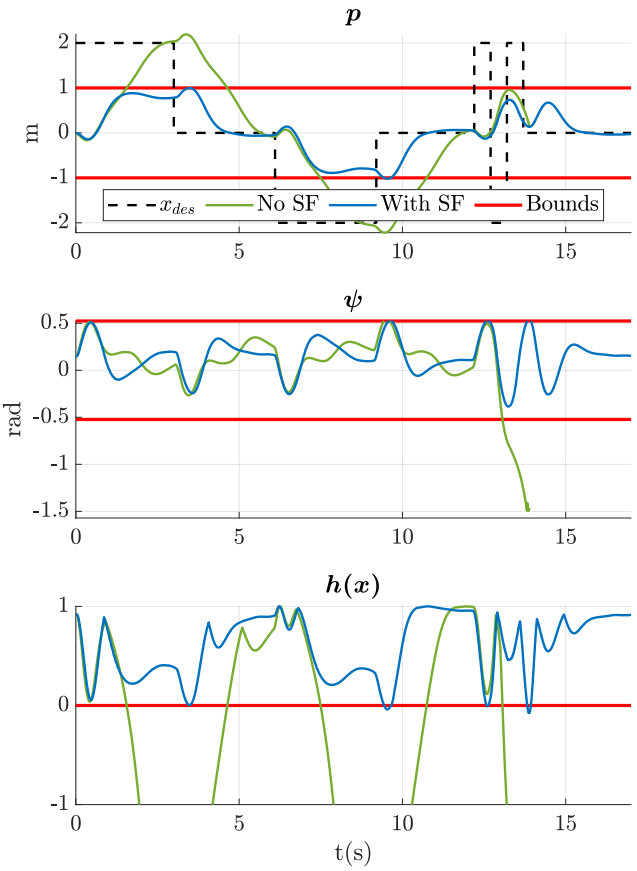


FIGURE 13. Results of the implicit filtering Segway experiments with and without safety filter. The nominal controller is an LQR driven by a desired position x_{des}.

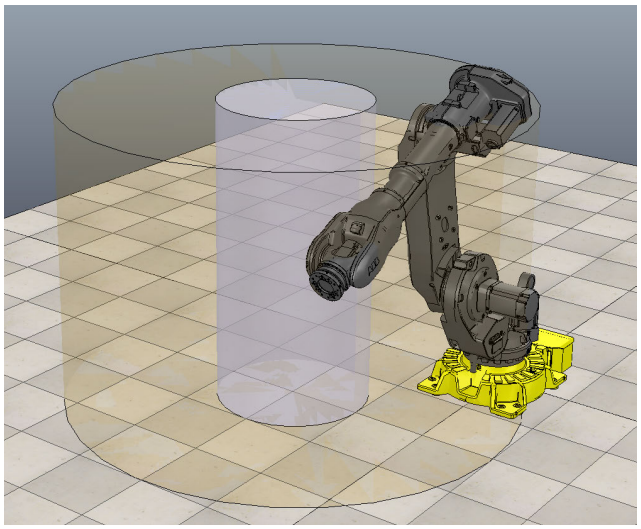


FIGURE 14. The IRB 6640 robotic arm along with the reachable sets for a human worker at $t = 0s$ in purple and $t = 1s$ in yellow.

B. INDUSTRIAL MANIPULATOR IN TIME VARYING ENVIRONMENT

Next, we apply the infinite-time implicit safety method described in Section III to the problem of collision avoidance in a dynamic environment. The global industrial robot market has more than doubled in the past five years, and the International Federation of Robotics expected almost two million new robot installations in factories by 2020 [30]. However, concern for the safety of their human counterparts grows along with the density of robots in factories. As a result, in heavy manufacturing, machines and humans are mostly separated. This makes the process rigid: it becomes spatially constrained and manual intervention in the vicinity of a robot may require halting the process altogether. To reduce down-time and allow for more human-robot interaction, we would like to be able to ensure that these robots cannot collide with human operators under any circumstances while also avoiding having to stop the robot altogether when a worker is in its vicinity. More information about this work can be found in [31].

1) PROBLEM FORMULATION

This problem is complex because human workers can move around the robot in a somewhat unpredictable manner. In order to guarantee that no collision can occur with such dynamic and uncertain targets, there is no choice but to assume all possible movements of the workers and avoid all of them. In other words, the safety set in this scenario has to be the complement of the forward reachable set of the human workers. Such a set spans space **and** time, so the state of the system has to be augmented to include this time dependency. Furthermore, the forward reachable set, and so the safety set, has to be periodically updated to account for the actual movement of the worker and avoid very punitive conservativeness.

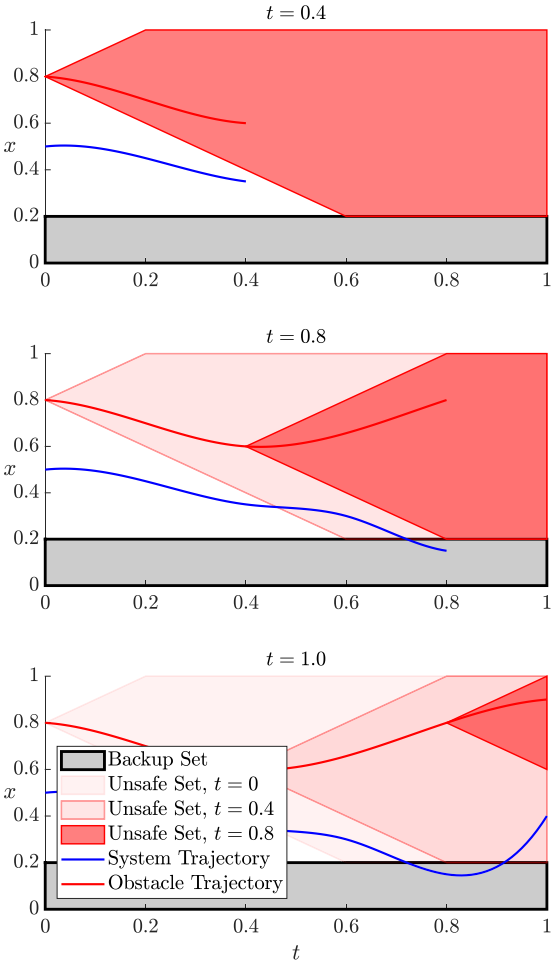


FIGURE 15. Illustration of time varying uncertain environment. The obstacle moves with time (red line) but its position is only measured at specific times (0.4,0.8,1.0). The safety set is the complement of the unsafe set, which is the forward reachable set of the obstacle minus the backup set. The system has to remain outside of the unsafe set, which shrinks for each obstacle measurement.

A major advantage of our implicit safety filtering over explicit safety filtering is that the safety set can be changed on the fly without any additional computations, whereas with an explicit safe set approach, if the safety set changes, a new safe set has to be computed, which is time consuming and represent a real bottleneck that hinder the capability of these explicit approaches to handle practical scenarios such as this one.

Let us consider the 6-link IRB 6640 manipulator from ABB depicted in Figure 14 that has six degrees of freedom. The dynamics of this robotic arm can be written in a classic manipulator equations form:

$$M(q)\ddot{q} + C\,(q, \dot{q})\,\dot{q} + G(q) = \tau, \tag{50}$$

where q describes the joint angles and τ is a vector of applied torques.

For manipulators with many degrees of freedom, the explicit expressions for $M(q)$, $C(q, \dot{q})$ and $G(q)$ are very complicated. As an alternative, they can be evaluated at given points via the Articulated Body Algorithm (ABA) that steps

over links of the manipulator in a recursive fashion [32]. Only having "black-box" access to the equations of motion would pose a problem for most methods for finding invariant sets, but the implicit method proposed in this paper only requires access to the numerical values of the dynamics and its derivatives. As discussed above, the state of the system is extended to contain time which makes the overall system's state 13-dimensional:

$$x = [q, \dot{q}, t]^T. \tag{51}$$

A point in the Cartesian and time space will be denoted by:

$$k = [a, b, c, t]^T. \tag{52}$$

2) SAFETY AND BACKUP SETS

The backup set is considered to be a vertical tube around the robot. In practice, this would be a small closed-off area that is inaccessible to the human. For this implementation, it is described by angle constraints on the second and third joints (the joints being enumerated from the base to the end effector):

$$S_b = \left\{ x \in \mathbb{R}^{13} \;\middle|\; \begin{array}{l} q_2 \in \left[-\frac{\pi}{12}, \frac{\pi}{12}\right] \\ q_3 \in \left[-\frac{7\pi}{12}, -\frac{5\pi}{12}\right] \end{array} \right\}. \tag{53}$$

The safety set is then simply the union of the backup set and complement of the reachable set of the human in space-time over the duration of the backup maneuver. For the purpose of this demonstration, the human is modeled as a single integrator with a maximum velocity $v_{\max}$, meaning that the size of its reachable set grows linearly in time. By adding time as a state, we prevent the filter from being overly conservative, which would be the result if we only used the reachable set of the human over the time horizon of the backup controller. Note that more complex models of human mobility could be used in order to further reduce the conservativeness of the safety set.

If (a_0, b_0) is the current horizontal Cartesian position of the human, the reachable set of the human can be simply expressed as an n-cylinder [33] centered at $(a_0, b_0, H/2)$ in Cartesian space, where H is the height of the human. We can then write this set as the lower-level set of a time-dependent differentiable function $h_r : \mathbb{R}^4 \to \mathbb{R}$ defined by:

$$\begin{aligned} h_r(k) = {} & (a - a_0)^2 + (b - b_0)^2 \\ & + \frac{(c - \sqrt{H})^2 (r_0 + v_{\max} t)^2}{H} \\ & - (r_0 + v_{\max} t)^2. \end{aligned} \tag{54}$$

Thus, for the robot to not come in contact with the human, $h_r(k)$ must be positive for all physical points k along the robot. However, because the dynamics of the robot are defined in joint space and the safety set is defined in Cartesian space, one must be careful when using the h_r. Let us denote our forward kinematics function that takes a point from joint and time space to Cartesian and time space, by $K : \mathbb{R}^{13} \to \mathbb{R}^4$. The gradient of h_r with respect to the states is:

$$\frac{\partial h_r(k)}{\partial x} = \frac{\partial h_r(K(x))}{\partial x} = \frac{\partial h_r(K(x))}{\partial k} \frac{\partial K(x)}{\partial x}, \tag{55}$$

where:

$$\frac{\partial K(x)}{\partial x} = \begin{bmatrix} \frac{\partial K}{\partial q} & \frac{\partial K}{\partial \dot{q}} & \frac{\partial K}{\partial t} \end{bmatrix} = \begin{bmatrix} J & \vec{0} & J\dot{q} \\ \vec{0} & \vec{0} & 1 \end{bmatrix}, \tag{56}$$

with the kinematic Jacobian J being computed numerically.

The safe set h is then defined as the union of h_r and the backup set, S_b, in order to avoid issues with the reachable set of the human intersecting with the backup set, which is not possible in reality.

Finally, as the system evolves, the reachable set of the worker gets larger in the Cartesian space. It is therefore important to **update** this reachable set when a new measurement (a_0, b_0) is available so as to avoid the reachable set filling the entire work envelope of the robot. A fundamental constraint for this update to be possible is that the resulting safety set must be larger after the update than before. In our case, this is guaranteed by the fact that any new position of the human will be contained in its reachable set (cf. Fig. 15), so the safety set grows in the full state space, even though it does not when only looking at Cartesian space.

3) BACKUP CONTROLLER

For the backup controller, we leverage the power of the Recursive Newton-Euler Algorithm (RNEA) [34], which provides the necessary joint torques to generate desired joint accelerations. The flexibility of this method is again showcased by the fact that we do not need an analytic expression for the backup controller, as long as we know its gradient.

There are only two joints that require actuation to reach the backup set. A simple PD controller is used to obtain desired joint accelerations for these joints, which is fed into the RNEA that generates the control inputs, as well as their gradient. The controller is of the form:

$$\begin{aligned} a_{\text{des}}(q, \dot{q}) &= -k_p(q - q_d) - k_d(\dot{q}), \\ u_{\text{b}}(q, \dot{q}) &= \text{RNEA}(q, \dot{q}, a_{\text{des}}(q, \dot{q})). \end{aligned} \tag{57}$$

The gradient of this backup controller, which is required to evaluate the sub-regulation map, is described by:

$$\begin{aligned} \frac{\partial u_b}{\partial q} &= \frac{\partial \text{RNEA}}{\partial q} + \frac{\partial \text{RNEA}}{\partial a_{\text{des}}} \frac{\partial a_{\text{des}}}{\partial q} = \frac{\partial \text{RNEA}}{\partial q} - k_p \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}, \\ \frac{\partial u_b}{\partial \dot{q}} &= \frac{\partial \text{RNEA}}{\partial \dot{q}} + \frac{\partial \text{RNEA}}{\partial a_{\text{des}}} \frac{\partial a_{\text{des}}}{\partial \dot{q}} = \frac{\partial \text{RNEA}}{\partial \dot{q}} - k_d \frac{\partial \text{RNEA}}{\partial a_{\text{des}}}, \\ \frac{\partial u_b}{dt} &= 0. \end{aligned}$$

Since the RNEA provides the exact torques needed to achieve desired joint accelerations, the forward invariance of the backup controller is guaranteed under the proper choice of desired joint accelerations.

4) SIMULATION

The rigid body algorithm library used for this simulation is Pinocchio [35]. This C++ library has been shown to be the fastest of its kind, with the Table 1 showcasing the average computation times of each necessary expression for the robot.

TABLE 1. Computation time for IRB 6640 in Pinocchio.

Expression	Time (μs)
Affine forward dynamics ($f(x)$ and $g(x)$)	4
Gradient of closed-loop forward dynamics	42
Backup controller	5
Gradient of backup controller	31

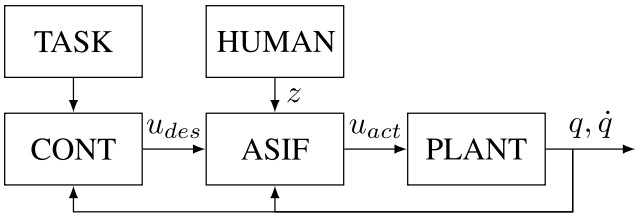


FIGURE 16. Block diagram of the control architecture used in the manipulator simulations.

A ROS environment was created to simulate the system, with V-REP used as a visualizer. The simulation consisted of five component: the robotic arm (PLANT), the task giver (TASK), a nominal controller (CONT), the human (HUMAN), and the safety filter (ASIF), connected as shown in Figure 16. Each component of the system ran at 200 Hz on a desktop PC with an Intel 8700k processor. The dynamics were integrated in the plant node via the ODEint C++ library, with the `runge_kutta_dopri5` scheme over a timestep of 5 ms.

The controller node tracked a sequence of desired end-effector positions given to it by the task giver node. Once the system reached the desired position, the task giver would send a new desired location to the system to mimic a typical operational cycle for such robot. The RNEA approach is also used for this tracking controller. The human node allowed the user to joystick a human, modeled as a single integrator, around the factory floor.

Lastly, the safety filter node handles safety for the system. It takes in the state from the robot and the desired inputs from the controller, and outputs the actual inputs that is used for integration by the plant.

The ASIF uses an adaptive-step RK4 scheme for integration under the backup controller, and the resulting quadratic program is solved by the OSQP library [28].

Figure 17 shows the value of the ASIF when a human attempts to pass through the working area of the arm. This image well illustrates the minimally invasive property of the ASIF, as the filter keeps the value of $h(x)$ just barely above zero. For a video demonstration of the filter's capabilities, please see [36].

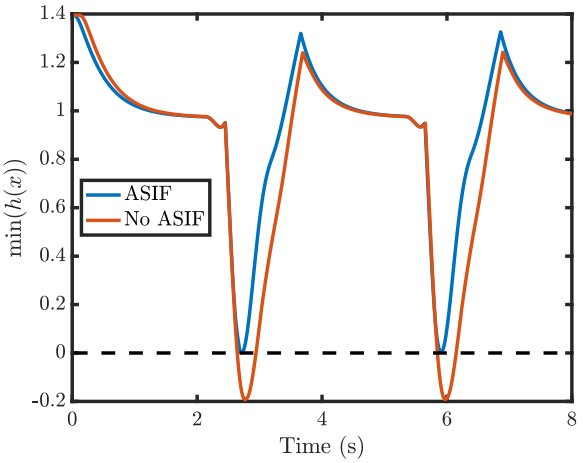


FIGURE 17. Value of the Barrier Function with and without ASIF engaged.

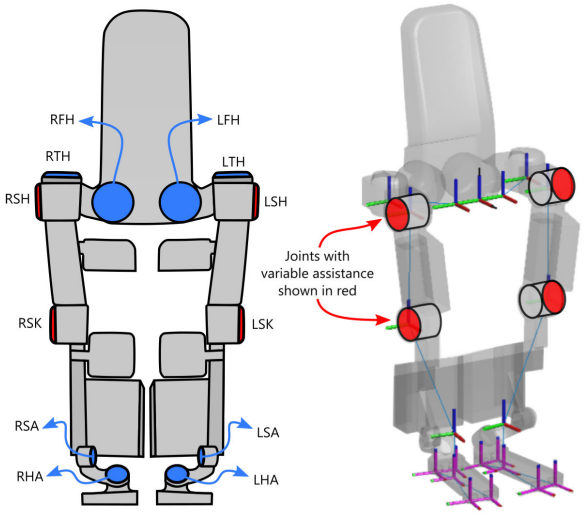


FIGURE 18. Schematic representation of the Atalante exoskeleton. In red are the joints that will be used for variable assistance.

C. VARIABLE ASSISTANCE FOR LOWER BODY EXOSKELETONS

Next, we apply this framework, and more specifically the approach of Sec. VI, to the problem of variable assistance for lower body exoskeletons. Contrary to most common applications of this framework, this particular one is not about safety. The main focus of this application is exoskeleton technology aimed at restoring locomotion for people with a leg pathology [37]–[41]. Recently, dynamically stable crutch-less exoskeleton walking has been demonstrated for patients with paraplegia by leveraging the full nonlinear dynamics of the system and generating dynamically stable gaits [42]. While this approach enables crutch-less exoskeleton walking, it is no longer optimal when exoskeleton technology is extended to patients who are recovering muscle functionality. A previous study showed that permitting partial assistance and variability during step training enhanced stepping recovery after a complete spinal cord transection in adult mice [43]. As we are about to see, the framework presented in this

paper can be used to enable assist-as-needed strategies while guaranteeing coherence of the walking pattern. The method presented here allows users to control their own motions when they are performing well, but intervene when they are not, so as to maintain a functional walking pattern. More details about this method can be found in [44].

The exoskeleton used for this work, named Atalante, was developed by the French startup company Wandercraft. As shown in Fig. 18, this lower-body exoskeleton has 12 actuated joints. The position and velocity of each actuated joint is measured using a digital encoder. Additionally, the exoskeleton has four Inertial Measurement Units (IMUs) that are used to provide additional information about the attitude of the robot with respect to the world. To detect ground contact, four 3-axis force sensors are attached to the bottom of each foot. All of the actuators and sensors are controlled by an embedded computing unit running a real-time operating system.

The main advantage of the proposed framework is its ability to not require offline computation given the system dynamic. In this scenario, it means that the framework can be applied to multiple patients without any overhead. Indeed, in order to accommodate each patient, the exoskeleton is equipped with thigh and shank length adjustments to change the dimensions of the exoskeleton. A combined human-exoskeleton model for each patient is generated to account for each person's unique physical characteristics. In particular, this combined human-exoskeleton system can be mathematically represented as a rigid body system. A floating-base generalized coordinate system is constructed as $q = (p, \phi, q_b) \in Q \in \mathbb{R}^{18}$, where $p \in \mathbb{R}^3$ and $\phi \in \mathbb{SO}^3$ denote the position and orientation of the exoskeleton's base frame with respect to the world frame. The relative angles of the actuated joints are denoted by $q_b \in \mathbb{R}^{12}$. In total, the system has 18 degrees of freedom, and is fully actuated when one foot is flatly in contact with the ground.

1) VARIABLE ASSISTANCE FRAMEWORK

As discussed in [43], the correct muscle activation pattern is an important criterion for the spinal learning process. To that end, we utilise the proposed set invariance framework to precisely control how much freedom is granted to the user, as the better the motricity of the patient is, the more he or she can be relied on to execute a stable walking pattern. First, we choose joints that we want to let the user control: the assisted joints. All the other joints will be rigidly controlled. In this work, we choose to only assist the sagittal hip and sagittal knee of the swing leg (cf. Fig. 18).

The architecture of the variable assistance framework, as shown in Fig. 19, contains four main components. First, a nominal gait is obtained from a neural network based library built from PHZD trajectories (cf. [42], [45]–[49]). This trajectory is modulated by a deadbeat mechanism. This deadbeat mechanism is critical in this case because the nominal joint trajectory will not be followed very accurately when the user is in control of the assisted joints.

The filtered trajectory $q_{des}(\cdot)$ is then fed into two separate controllers. One is the **baseline controller** that plays back the trajectory and generates position and velocity targets $q_{des}(t - t_i)$ and $q'_{des}(t - t_i)$ for the PID controllers that in turn generate tracking torques $u_t(t)$. The flatfoot ankle controller separately computes targets for the swing leg ankle that are then substituted in place of the nominal ones.

The other controller is the **variable assistance controller**. This controller is the heart of this variable assistance approach and leverages the proposed controlled invariance framework. The variable assistance controller has three subcomponents: joint idealization, feedforward assistance, and virtual guide filter.

The **joint idealization** component computes the torques required to compensate for gravity and friction in the assisted joints. The goal is to make these joints as transparent as possible such that when there is no assistance, the user does not feel any resistance that would impede his ability to walk freely. This joint idealization component is, however, not sufficient to make the exoskeleton fully transparent as the inertia of the exoskeleton is not compensated for, which makes the user's legs harder to move. The **feedforward assistance** component therefore provides feedforward torques $u_f(t)$ – calculated during the PHZD gait generation process [42] – to obtain a first-order level of compensation for the inertia of the assisted joints. This does not truly compensate for inertia, but at least provides enough assistance for the user to move the exoskeleton legs along the desired trajectory. The intensity of both idealization and feedforward components can be adjusted to produce varying levels of user effort.

The **virtual guide filter** computes the joint torques $u_v(t)$ required to limit the discrepancy between the actual and desired trajectory of the assisted joints. To that end, we will explore two approaches. First, the discrepancy limit is described by a tube around the desired trajectory: a virtual guide. The shapes and sizes of the virtual guides can be chosen almost arbitrarily. In a second time, the discrepancy will be characterised by the position of the swing foot with respect to a nominal trajectory.

Finally, an impact detection block also records which leg of the exoskeleton is in stance or swing, and generates an ''assisted joints selection matrix'' that controls which joints are being assisted at a given instant. Only these joints are assigned the assistive torques. The remaining joints are assigned the baseline tracking torques. The merging of these torques comprises the final joint torques $u(t)$ that are commanded to the exoskeleton.

2) JOINT-BASED VIRTUAL GUIDE FILTER

Formulation: In this first approach, each joint is idealized so that it can be handled independently of the rest of the system. We therefore consider the following dynamics for each joint:

$$J\ddot{q} = u_v + u_f(t - t_i) + u_{ext}, \tag{58}$$

where J is the inertia at the joint, u_v is the torque the virtual guide filter can apply, $u_f(t)$ the feedfoward torque applied to

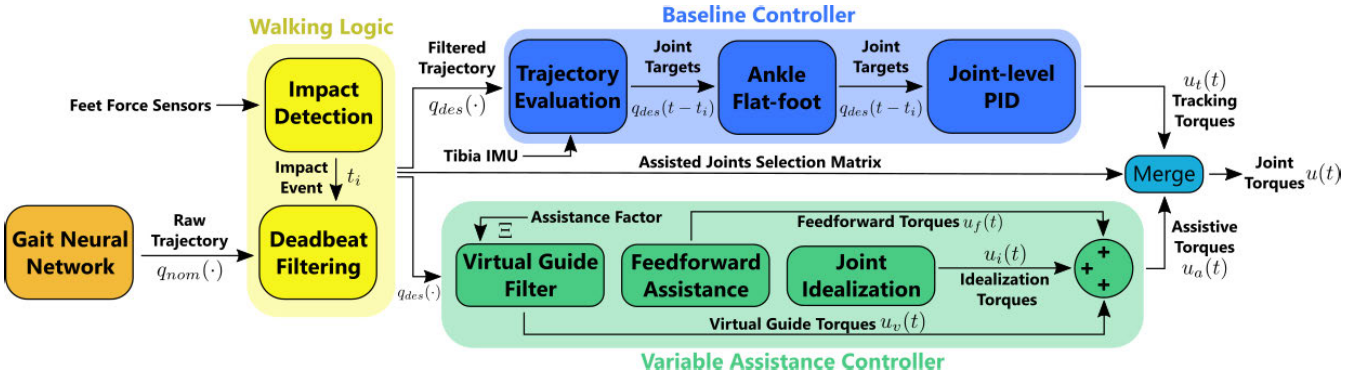


FIGURE 19. Architecture of the variable assistance framework.

the joint, and u_{ext} the torque applied by the exoskeleton user on the joint. The state of the system is therefore $x = [q, \dot{q}]^\top$.

The virtual guide $\overline{S}$ we want to constrain the joint to stay in is characterized by:

$$h(t, x) = 1 - \left(\frac{q_{des}(t - t_i) - q(t)}{q_{bound}(t - t_i)} \right)^2 \tag{59}$$

for some properly chosen q_{bound} to achieve the desired shape of the guide (cf. Fig. 20 for examples of shapes). Note that this is a time varying set, so the propositions of Sec. VI have to be modified as done in [44].

Because u_{ext} is not known ahead of time, a robust version of the method presented in VI has to be used. The key is that system (58) is monotone [50]. In this case, the safe backward image is characterized by:

$$h_T^{\Omega}(t, x) = \min_{\substack{\tau \in [0, T-t] \\ u_{ext} \in \{u_{ext}^{min},\ u_{ext}^{max}\}}} h \circ \phi_\tau^{u_b, u_{ext}}(q)\,, \tag{60}$$

where u_{ext}^{min} and u_{ext}^{max} are the extreme values of the disturbance the user can generate. So in order to evaluate $h_T^{\Omega}(t, q)$, the numerical integration of the dynamics only has to be performed twice each time assuming the extremal values of the disturbance. The backup policy is chosen to be:

$$u_b(t, x) = K_p(q_{des}(t - t_i) - q) + K_d(\dot{q}_{des}(t - t_i) - \dot{q}) \tag{61}$$

for some properly chosen gains K_p and K_d. For this work, these gains were chosen to be the same as the one used for the PIDs of the baseline controller.

Finally, the filtering law is given by:

$$u_v(t, x) = (\lambda(t, q) + (1 - \lambda(t, q))\, \lambda_d(t, q))\, u_b(t, x), \tag{62}$$

where $\lambda(t, q) = \left(1 - h_T^{\Omega}(t, q)\right)^3$ and $\lambda_d(t, q) = \zeta \frac{dh_T^{\Omega}(t, q(t))}{dt}$ for some derivative gain ζ. The usage of this derivative term helps dampen the behavior of the safety filter.

Experiments: The validation experiments were performed on the empty exoskeleton as it hung in the air in an effort to show the behavior of the filter without user perturbations and without feedforward torque. The plots of the experimental results, shown in Fig. 20, illustrate the actual joint angles over 30 steps with each step overlaid on top of each other. It can be seen that for all tube shapes, the actual joint angles remained inside of the bounds and the filter only acts when necessary.

Then, the variable assistance framework was tested with a subject inside the exoskeleton and walking on a treadmill. The required assistive torque as well as the trajectory tracking are presented in Fig. 21. It can be observed that when the subject is passive under partial assistance, the joint trajectories tend to group near the virtual guides as expected. Alternatively, when the subject is active under partial assistance, the actual joint trajectories tend to span more of the virtual guide as the subject is actively trying to avoid hitting the bounds of the guide. In all cases, the trajectories stay contained within the virtual guides. For a video of these experiments please see [51].

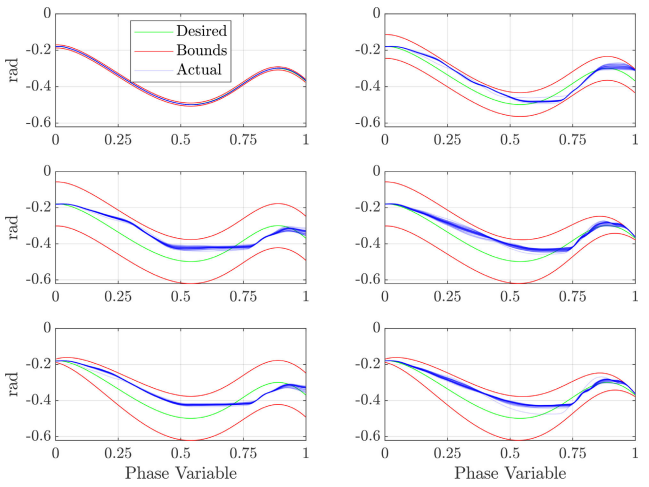


FIGURE 20. Left hip positions joint angles for the Exoskeleton empty and hanging in the air. The plots corresponds to 30 right steps and each subplot correspond to a different virtual guide shape.

3) FEATURE-BASED VIRTUAL GUIDE FILTER

One of the issues with this first approach is that the subject has to follow the nominal trajectory which in practice is not particularly anthropomorphic. This makes it difficult for the

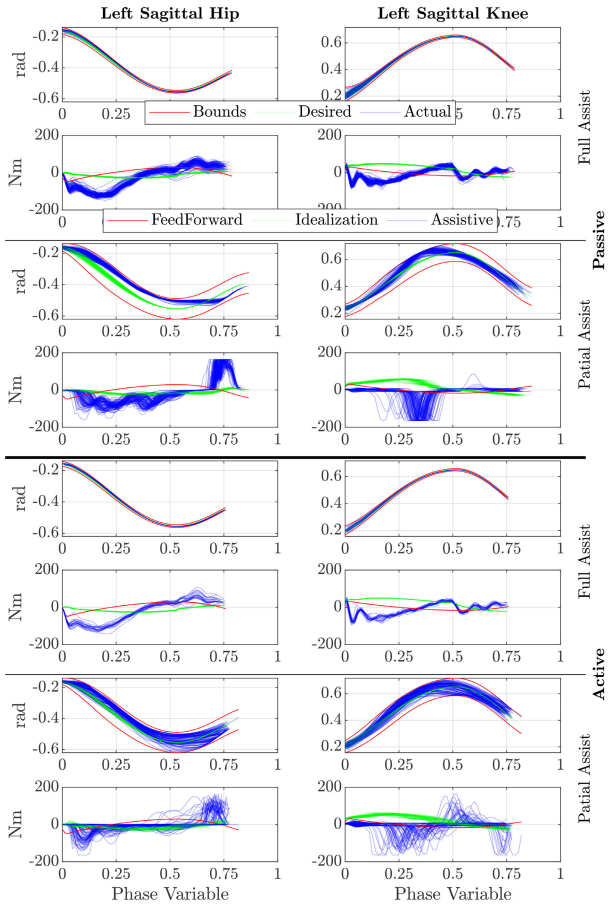


FIGURE 21. Comparison at the joint level between a subject actively trying to walk and being passive under both full and partial assistance of the exoskeleton. Because of early striking, most steps ended before the phase variable reached 1, unlike in Fig. 20 where the exoskeleton was in the air.

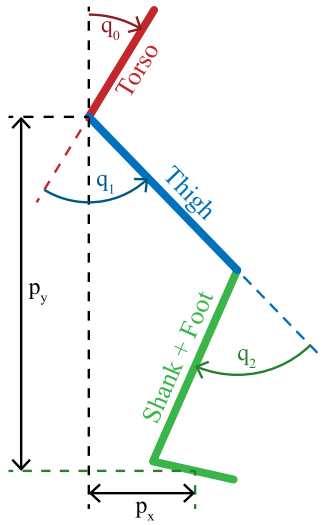


FIGURE 22. Simplified model of the exoskeleton and its swing leg.

subject to perform well. So to make the variable assistance more permissive while still ensuring coherent walking if the subject is not performing well, we propose to extend the previous approach to a feature-based constraint instead of a joint-based one. Here, we propose to constrain the trajectory of the swing foot to ensure a correct ground clearance and forward motion. This way, the user has more freedom on individual joints trajectories which allows him to perform more natural steps.

For this approach, the controller architecture remains the same as in Fig. 19. Only the formulation of the virtual guide filter is changed from the previous section.

Formulation: This time, we consider a simplified model of the swing leg as a whole. In particular, we model the swing leg as a double pendulum whose joints corresponds the sagittal hip and knee, the ankle joints being assumed locked (cf. Fig. 22). The state of the system is now 4-dimensional: $x = [q, \dot{q}]^\top$ with $q = [q_1, q_2]^\top$, and both hip and knee joint torques are being filtered concurrently. For simplicity, the torso pitch angle q_0 is fixed at the desired angle chosen to generate the nominal trajectory. The dynamics can be found using classical Euler-Lagrange formalism and is of the form:

$$M(q)\ddot{q} = u - C(q, \dot{q}) - g(q), \tag{63}$$

where $u = [u_1, u_2]^\top$ is the vector of joint torques.

The virtual guide is now characterized by the forward position of the swing foot **with respect to the nominal trajectory** (cf. Fig. 22):

$$h(t, x) = p_x\left(q(t)\right) - p_x\left(q_{des}(t - t_i)\right) + \epsilon_x. \tag{64}$$

In other words, the swing foot is constrained to move forward at least as fast as it does in the nominal trajectory.

The backup policy is chosen to be the same as for the joint-based virtual guides—PID tracking of the nominal trajectory—and similarly with the filtering law. The constant ϵ_x is chosen so as to allow the backup law to steer the system away from the boundary of the virtual guide.

Experiments: The validation experiments were performed on the empty exoskeleton as it hung in the air. The plots of the experimental results, shown in Fig. 23, illustrate the foot position and constraints over several steps. It can be seen that in the absence of user effort, the proposed formulation results in the actual foot trajectory following closely the nominal one.

Then, this approach was tested with a subject inside the exoskeleton and walking on a treadmill. The results are presented in Fig. 24 and 25. It can be observed that in this case, the subject can freely execute a gait with longer step length if he desires (cf. Fig. 25), while still guaranteeing that a minimum step stride is respected if the subject is not able to perform longer steps (cf. Fig. 24).

D. SAFE EXPLORATION OF UNKNOWN ENVIRONMENTS WITH A QUADCOPTER UAV

Finally, we apply this controlled set invariance framework to the problem of safe exploration of an unknown environment (note that more information about this work can be found in [52]). This task is particularly relevant for drones whose usage is becoming prevalent for tasks such as autonomous deliveries, aerial surveillance, or disaster relief. Most of these

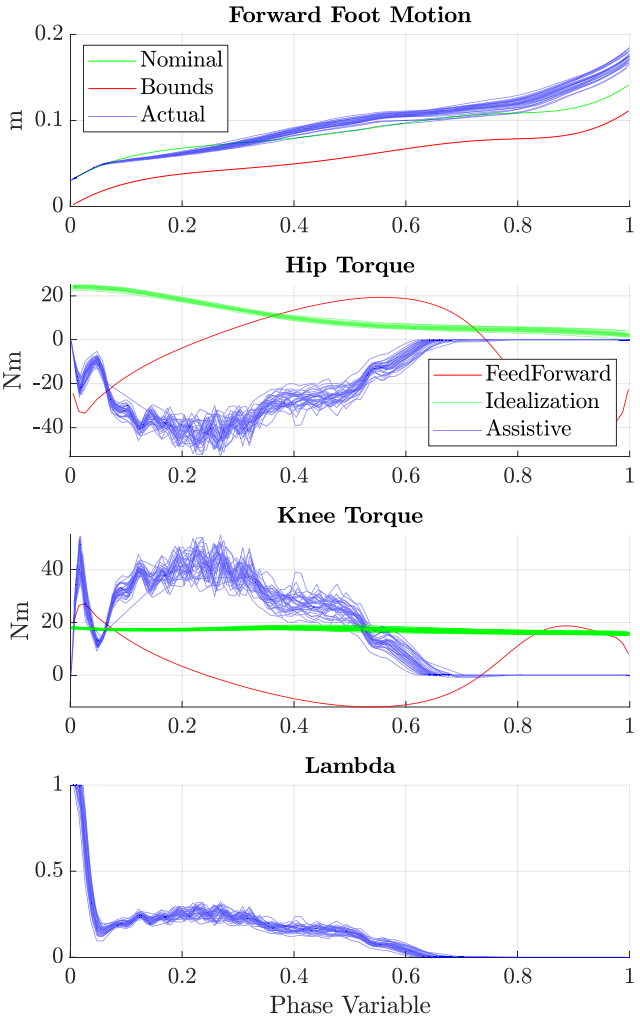


FIGURE 23. Foot level variable assistance for the Exoskeleton empty and hanging in the air.

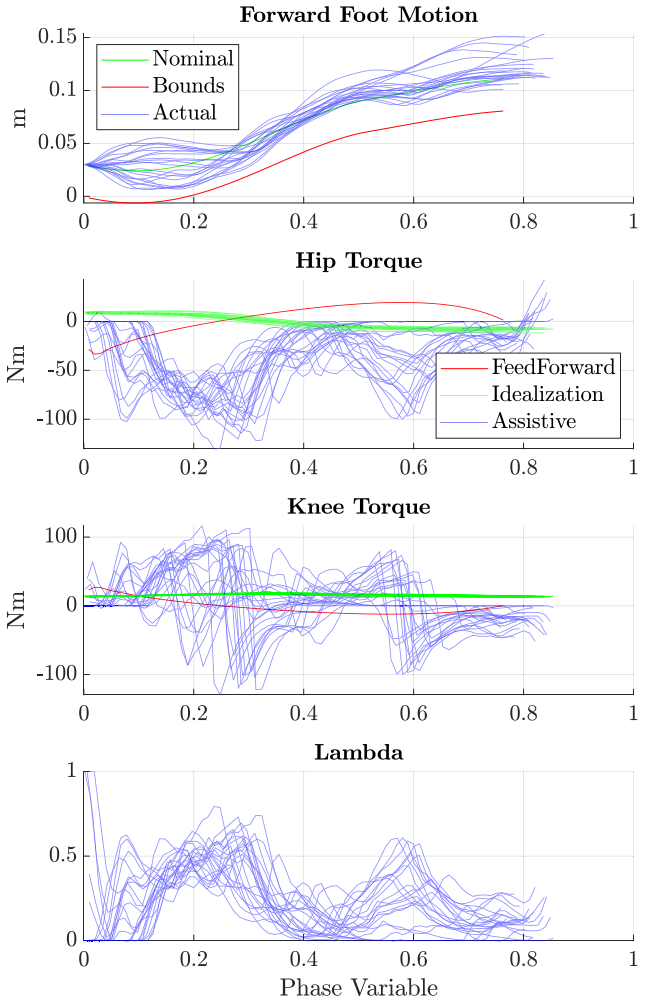


FIGURE 24. Foot Level Assistance for an able-bodied subject being passive.

missions involve navigating through unknown or uncertain environments. Due to the altitude of the vehicles and their often exposed propellers, collisions are catastrophic for the drone and might also be dangerous for its surroundings. For this reason, collision avoidance techniques are crucial to further the use of these systems in everyday life.

In typical drone flight, collision avoidance is the topic of navigating safely through an environment. This usually translates into creating and tracking trajectories that take the drone through the surrounding free space and that avoid occupied or uncertain space. While this approach to collision avoidance can be effective in practice, as evidenced in [18], [53], [54], its computational complexity necessitates simplified abstractions of the model and obstacles. Couple to that the inherent uncertainty associated with mapping an environment and it is easy to see why such approaches are typically conservative, which can lead to slow mobility, while still lacking guarantees of collision-free tracking of the trajectories.

The authors therefore believe that trajectory planning is not the most effective layer in which to enforce safety. Planner updates are too infrequent, and there is too much uncertainty stemming from the aforementioned hurdles to be able to provide rigorous guarantees of safety with such a method. Instead, we propose an approach leveraging the proposed set invariance framework to ensure collision avoidance enforced at a control level while relying solely on local sensing information (i.e. no mapping is required). This approach can be applied in conjunction with any planning algorithm (or even a human operator), which means that it allows for planning algorithms that are more aggressive, since they do not need to guarantee collision avoidance or dynamic feasibility.

The planner used for this work is designed to work in tandem with the Octomap mapping library [55] to represent the environment map. The path planning is a basic implementation of the A* algorithm that searches for a path to a nearby frontier cluster. The search algorithm runs directly on the octomap, which is possible via implementation of algorithms that enumerate neighbors in a 3D octree [56]. Special heuris-

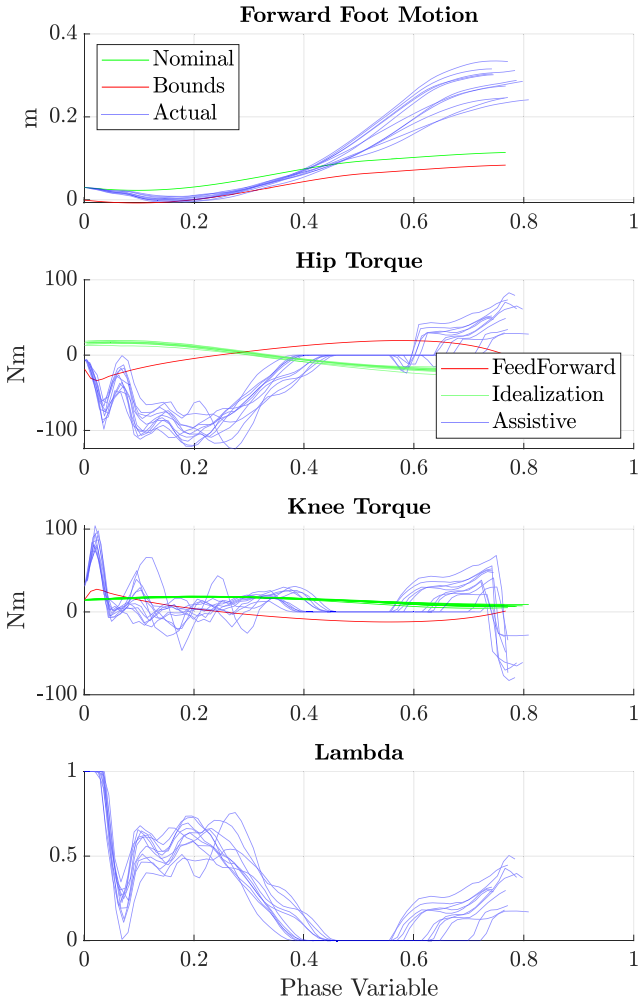


FIGURE 25. Foot Level Assistance for an able-bodied subject actively trying to take long steps.

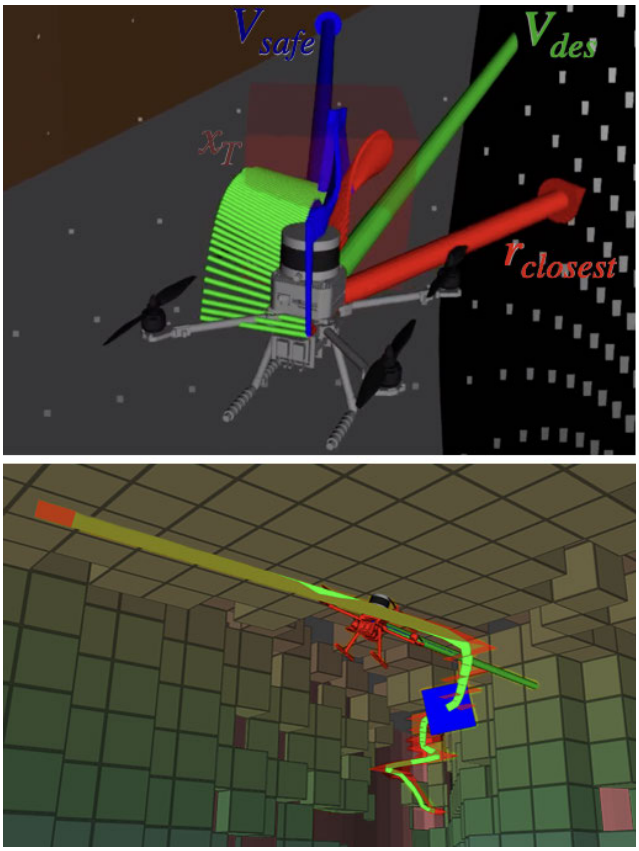


FIGURE 26. Simulation environment. The top shows the desired and filtered velocity commands based on the closest point in the point cloud. The bottom shows the drone navigating through the cave.

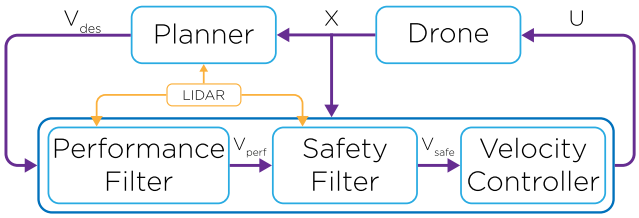


FIGURE 27. Safety filtering control structure for UAV exploration.

tics encourages the planner to visit large clusters that are close to the current position of the drone. After a global plan is achieved, the local planner creates a spline and continuously updates a target point on this spline in front of the UAV. The position of this target point in the UAV frame is then used as the desired velocity V_{des} for the performance filter (cf. Fig. 27).

1) COLLISION AVOIDANCE FRAMEWORK

The goal of this Collision Avoidance Framework is to allow the vehicle to safely navigate around any unknown environment. As discussed above, we want to only rely on local sensing information, i.e. no mapping will be performed. It is therefore fundamental that the sensing method used allows full 360 degrees coverage of the vehicle surroundings. The goal is to define the safety set as an envelope around the UAV in which we know there is no obstacles.

For this scenario, the environment is assumed to be static, and we will use a point-cloud representation of the environment. We will assume that this point-cloud is obtained from a ray-based method of sensing such that all segments between the vehicle and the points of the point-clouds corresponds to unoccupied space (i.e. does not go through any obstacles). Points therefore correspond to the boundary between free space and either physical obstacles or unknown space (cf. Fig. 31). The safety set will hence be described as in (4) with:

$$h_i = \|p - p_i\|^2 - \Delta_h^2, \tag{65}$$

where p is the position of the UAV in cartesian space, p_i are the points of the point-cloud, and Δ_h a *hard margin* introduced to account for the size of the vehicle (cf. Fig. 28).

Most ray-tracing sensors available to date do not enjoy the same high update rate as the sensors necessary for low level control (i.e. IMU, visual odometry, etc…). Therefore,

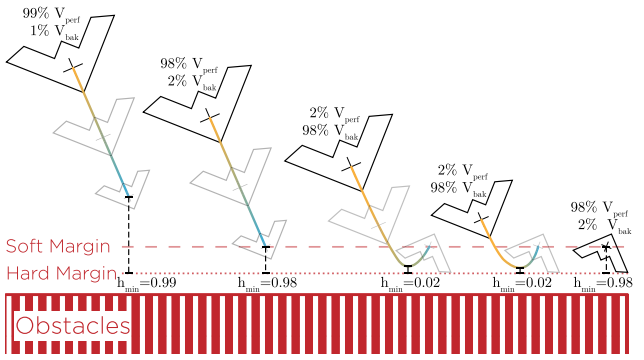


FIGURE 28. Illustration of the safety filter. From left to right is a depiction of the evolution of the drone as it gets close to the obstacles. In yellow to blue are the backup trajectories. In grey are the hypothetical positions of the drone if it were to follow the backup trajectory. The size of the drone icon corresponds to its velocity.

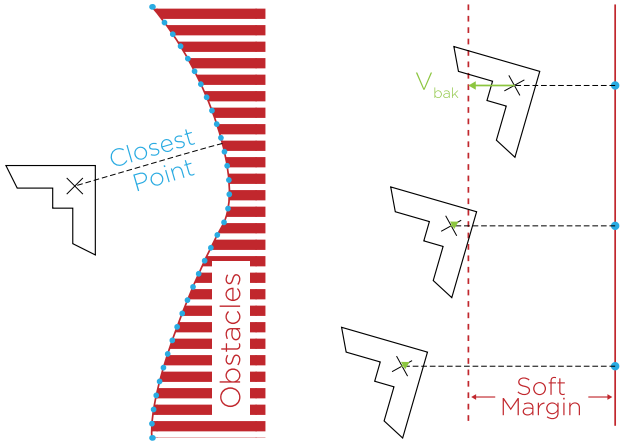


FIGURE 29. Illustration of the UAV backup law.

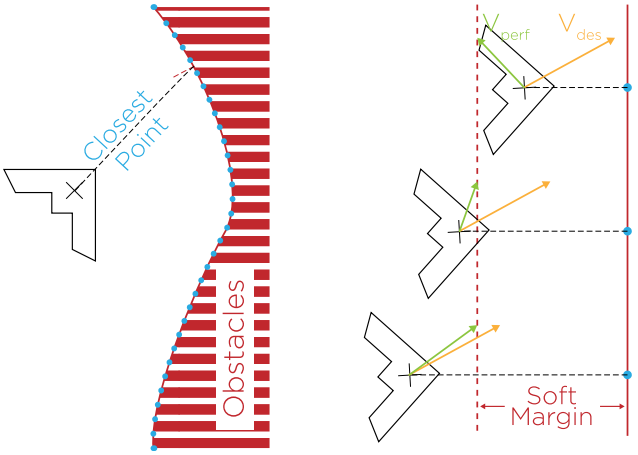


FIGURE 30. Illustration of the UAV performance filter.

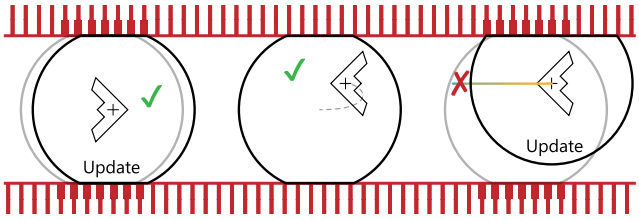


FIGURE 31. Illustration of the UAV recursive feasibility issue.

the position of the UAV only has to be tracked between each environment sensor update. This is one of the main advantages of the proposed framework over the ones requiring global planning and positioning. Only local positioning over a duration corresponding to the update period of the environment sensor is required.

Similarly to Sec. VII-B, the safety set is continuously changing as the environment gets explored. However, contrary to Sec. VII-B, the time dependency of the safety set is not known ahead of time and so the safety set does not necessarily always grow inside the state space (cf. Fig. 31). Furthermore, it is not trivial to find an efficient backup policy for all the possible shapes of safety set that can arise during exploration. Finally, embedded systems for UAVs do not quite have the computational capabilities to run a QP-based safety filter as for Sec. VII-B, especially at the control rates necessary for such agile vehicles.

To tackle this problem, we will therefore take the same approach as in Sec. VII-C and use the scalable safety filter of Sec. VI. The filtering law is chosen to be:

$$v_{safe}(x) = \lambda(x) v_{perf}(x) + (1 - \lambda(x))\, v_b(x), \tag{66}$$

with

$$\lambda(x) = 1 - e^{-3h_{S_T^{u_b}}(x)/(\Delta_s - \Delta_h)}, \tag{67}$$

where Δ_s is the *soft margin* (cf. Fig. 28).

A key difference with the work presented so far though is that the safety filter is placed **before** a velocity controller (cf. Fig. 27). This velocity controller closes the loop around desired velocities in the world frame. In this work, it is a simple Velocity-Attitude-Rates cascade PID controller, but any controller that can track a desired velocity would work in this framework. This has two effects: first it makes the overall system more robust to model uncertainty. Secondly, it allows us to think about backup policies in a space that is simpler to grasp than the actuator space (in that case a 3D velocity space). Therefore, efficient backup policies can be constructed more easily.

The backup policy chosen for this application is inspired by previous work on a geofencing for civilian UAVs [57]. The idea is to slow down to a halt while steering the vehicle away when getting close to obstacles. This can be achieved by commanding $V_{bak} = 0$ unless the vehicle is between the soft and hard margins, in which case the V_{bak} vector is pointed away from the nearest obstacles as depicted in Fig. 29.

The effect of that backup policy in conjunction with the filtering law is depicted in Fig. 28. However, even though this approach guarantees safety of the system, it yields poor performances in some circumstances. In particular, it does not allow the system to *smoothly glide* along obstacles as the

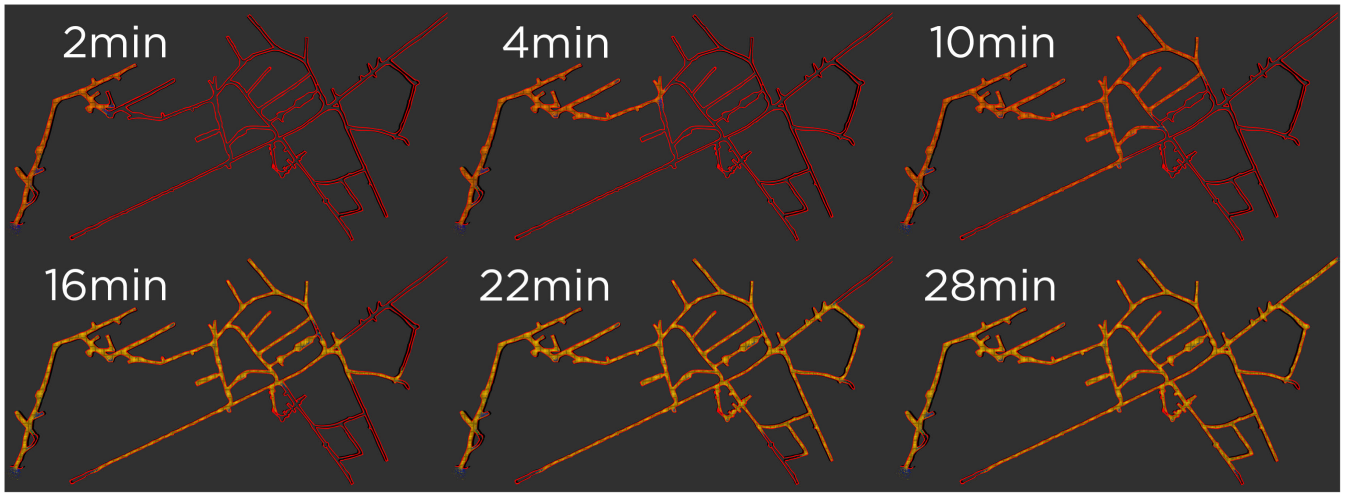


FIGURE 32. Pictures of the cave (in red) and the octomap (in yellow) being built throughout the 28 minutes it takes for the drone to completely explore the cave.

backup policy is intermittently switched to and from, leading to very oscillatory behaviors. Therefore, in order to increase the overall performance of the system, another component is used to filter the velocity inputs commanded by the planner in order to minimize the interventions from the safety filter: the performance filter.

The base of this performance filter is the same as the one of the backup policy (cf. [57]). The desired velocity is altered in a way that slows down convergence to the obstacles when getting close to it, and incentivizes divergence when past the soft margin. However, the closest point considered for the filtering of V_{des} is not based on the distance between the current drone position and the obstacles, but the shortest distance between the drone position **along the backup trajectory** and the obstacles (cf. Fig. 30). Furthermore, the distance to this closest point is the shortest distance between the drone position along the backup trajectory and the obstacles. This way, the performance filter is able to better anticipate incoming obstacles which leads to less intrusions of the backup trajectory into the soft margin, which in turn leads to less interventions of the safety filter. In the end, these 3 components work together to provide a filter with minimal conservativeness and with guaranteed collision avoidance (cf. Fig. 27).

2) RECURSIVE FEASIBILITY

As discussed before, at each update of the environment sensors, a new safety set is redefined. This means that after an update, the system could end up outside of $S_T^{u_b}$ as illustrated in Fig. 31. To address this use, there are two approaches.

A first approach it to check for each update whether or not the system would end up inside of $S_T^{u_b}$ with this new safety set. If it does, then the update can be carried out, otherwise this new safety set is discarded and the current one continues to be used until the next update. This approach carries a non-negligible computational weight as the safety filter has to be ran twice when an update is not successful. One must also be careful because when safety set updates are skipped, the system only relies on localisation data whose drift can become substantial.

A second approach is to rely on the backup policy to bring the system to a stop if the system ends up outside of $S_T^{u_b}$ after a safety set update. Indeed, even though the system is outside of $S_T^{u_b}$, it is not before the update, which means that the backup policy can safely bring the system to a stop. As the slowing down occurs, the safety set can continue to be updated in hope to regain feasibility of the safety filter. In practice, feasibility is regained quickly and this is the approach we take for the following simulations.

3) SIMULATION

The simulation environment is a ROS-based C++ environment. The point-cloud data is obtained from a modified Velodyne LIDAR sensor inside of the Gazebo simulator at a frequency of 10 hz. The simulation, including visualization in Gazebo and RVIZ, was able to run at a frequency of 300 Hz on a modern laptop computer.

The cave environment to explore was a large 240m by 460m structure with one entrance and one exit (cf. Fig. 32 and Fig. 26). The cave height is constant at roughly 3m, but the width is constantly changing, and gets as small as 0.5m with several protruding areas.

The quadrotor was able to explore the entire 240m by 460m cave in just under 28 minutes (cf. Fig. 32). The maximum allowable speed from the planner was 5 m/s, which the drone reached during open areas of the cave. The average desired speed sent from the planner was 4.09 m/s, and the average speed of the drone after the safety filter was 3.28 m/s.

A positive value of the barrier functions was maintained throughout, meaning the quadrotor never went closer than the minimum allowed distance from a point in the point-cloud, which was set at $\Delta_h = 0.2m$ meters. For a video of these simulations please see [58].

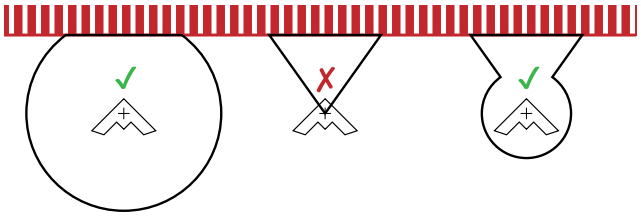


FIGURE 33. Illustration of the issue when using a mapping sensor with restricted field of view. An assumption about the safety of the vehicle has to be made at the initialization of the mapping in the form of a *bubble* of free space around the vehicle. The black line around the UAV represent the boundary between free and either unknown or occupied space.

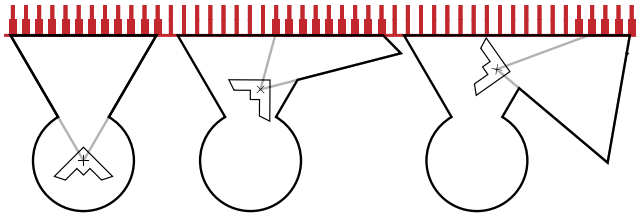


FIGURE 34. Illustration of the mapping process. The black line around the UAV represent the boundary between free and either unknown or occupied space.

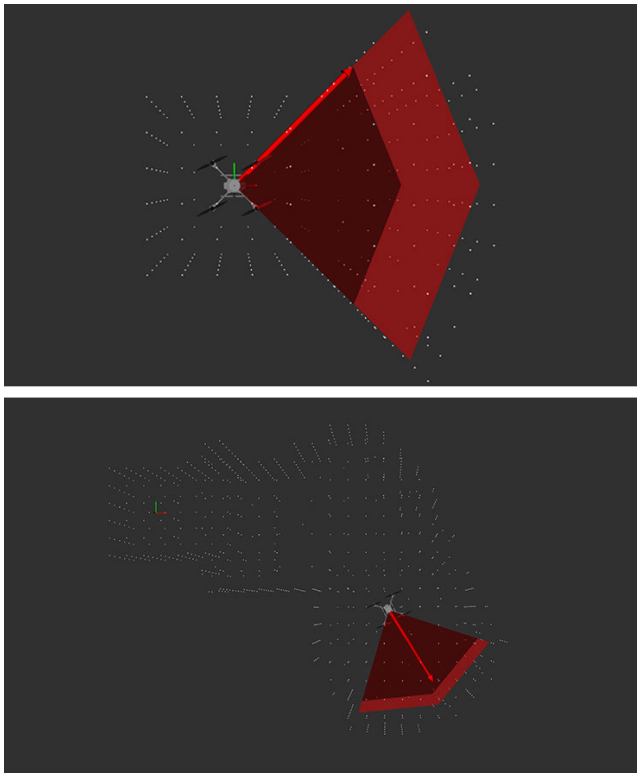


FIGURE 35. Image of the mapping process in the simulation environment. The free space boundary is represented by a point-cloud.

4) RESTRICTED FIELD OF VIEW

So far, we required that the vehicle be equipped with sensors that cover all of the UAV surroundings. Even though this is possible with available technology, such a sensing capability would be fairly expensive and would require a large enough vehicle to carry the sensors as well as an embedded computer capable of processing that much data. Therefore, we propose to extend the proposed approach for vehicle equipped with only partial sensing of the vehicle surrounding. This is for example the case of most UAVs that are equipped with a single forward-facing depth-sensing camera.

In that case, each individual sensor point-cloud is not sufficient to create a safety set that envelops the vehicle as illustrated in Fig. 33. It is therefore necessary to implement a mapping strategy to create a safety set envelope that the vehicle can evolve in. Note, however, that for the first iteration of the algorithm, the vehicle is not inside the safety set, so an assumption must be made that the immediate surroundings of the vehicle are safe for this 1st iteration (cf. Fig. 33). During subsequent iterations, mapping can be performed based only on new sensor data and a safety set can be progressively built as illustrated in Fig. 34.

To implement this approach in a way that leverages the precision and efficiency of a point-cloud representation of the safety set, we propose an algorithm that combines point-cloud and voxel representations of the environment. A naive approach would be to just fusion the point-clouds given by the sensor, however this would only define the boundary between free and occupied space, but not between free and unknown space. One could therefore rely on a voxel-based representation of the environment, but this would come at the cost of conservatism on the location of the obstacle. One way to address this issues is therefore to combine both approaches.

Point-cloud data is therefore used to generate a voxel map of the environment, but point-cloud data is also conserved to refine the position of the frontier between free and occupied space (cf. Fig, 35). Successive point-clouds are merged together and the resulting point-cloud is down-sampled using the generated voxel map such that each voxel contains only one point. The position of this point is then associated with the corresponding voxel. A point-cloud of the safety set envelope is then generated by using the down-sampled point-cloud augmented by another point-cloud of the centers of all voxels on the frontier between free and unknown spaces. The rest of the safety filtering algorithm is the same as in the previous section.

The simulated cave environment is explored again using an Intel Realsense as the only source of environmental data. This time, the yaw of the vehicle is controlled so as to point forward with respect to the path. As expected, the UAV is able to explore the cave safely, but does so more slowly overall.

VIII. CONCLUSION AND FUTURE WORK

In this work, a Scalable Safety Critical Control Framework is presented. This framework makes it possible to enforce safety for high dimensional nonlinear systems in a minimally invasive way. The trade off between computational complexity and conservativeness is analysed and approaches with varying levels of scalability are proposed. The idea of composing backup controllers though functional approximation of

optimal policies is explored as a potential method of combining the advantages of scalability seen in simpler controllers with the permissiveness characterising optimal controllers. Finally, the effectiveness of the framework is illustrated with multiple relevant applications. In particular, we show how this framework both makes it possible to guarantee safety in time varying and uncertain environments, and enables fast and safe exploration of unknown environments with UAVs. We showcase the effectiveness of the framework on hardware though the safe control of a two-wheeled inverted pendulum (Segway), and with the assistive control of a lower body exoskeleton.

As touched upon in the work of Sec. VII-D, the extension of this work to handle dynamics and sensing uncertainty is key to providing meaningful guarantees for real world safety critical applications.

Another challenge with safety critical control frameworks in general is the issue of representing the environment in such a way that provides meaningful guarantees when used in such framework. In Sec. VII-D, a discrete point-cloud representation is used that does not actually provide guarantees of avoidance with obstacles. Indeed, with this representation, avoidance with only the parts of the obstacles represented by the points of the point-cloud will be avoided, which can be an issue if the point-cloud is not dense enough. Furthermore, in this same work, a map of the environment is used while it is generated from perfect estimation of the vehicle position, which is obviously not realistic. Developing strategies for mapping the environment that are dense and truly safe, i.e. guarantee that no obstacles are within the free space while accounting for all sources of uncertainties, and are not too computationally expensive is a real challenge that deserves attention.

ACKNOWLEDGMENT

The authors would like to thank Laurent Ciarletta for his pivotal role in making this work possible. The authors would also like to thank Maegan Tucker for her valuable help with the Variable Assistance part of this work, and Terry Suh for seeding the Industrial Manipulator part of this work.

REFERENCES

[1] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, Nov. 1999.

[2] R. Ghaemi and D. Del Vecchio, "Control for safety specifications of systems with imperfect information on a partial order," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 982–995, Apr. 2014.

[3] J.-P. Aubin, *Viability Theory*. Basel, Switzerland: Birkhäuser, 2009. [Online]. Available: https://www.springer.com/gp/book/9780817649104

[4] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005.

[5] X. Xu, J. W. Grizzle, P. Tabuada, and A. D. Ames, "Correctness guarantees for the composition of lane keeping and adaptive cruise control," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1216–1229, Jul. 2018.

[6] J. H. Gillula, S. Kaynama, and C. J. Tomlin, "Sampling-based approximation of the viability kernel for high-dimensional linear sampled-data systems," in *Proc. 17th Int. Conf. Hybrid Syst., Comput. Control - HSCC*, 2014, pp. 173–182.

[7] I. M. Mitchell, "A summary of recent progress on efficient parametric approximations of viability and discriminating kernels," in *Proc. SNR@ CAV*, 2005, pp. 23–31.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[9] T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. Ames, "Towards a framework for realizable safety critical control through active set invariance," in *Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, Apr. 2018, pp. 98–106.

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

[11] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2460–2465.

[12] N. Ho, G. G. Sadler, L. C. Hoffmann, K. Zemlicka, J. Lyons, W. Fergueson, C. Richardson, A. Cacanindin, S. Cals, and M. Wilkins, "A longitudinal field study of auto-GCAS acceptance and trust: First-year results and implications," *J. Cognit. Eng. Decis. Making*, vol. 11, no. 3, pp. 239–251, Sep. 2017.

[13] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The system-level simplex architecture for improved real-time embedded system safety," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp.*, Apr. 2009, pp. 99–107.

[14] B. T. Lopez and J. P. How, "Aggressive 3-D collision avoidance for high-speed navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5759–5765.

[15] C. Munoz, A. Narkawicz, and J. Chamberlain, "A TCAS-II resolution advisory detection algorithm," in *Proc. AIAA Guid., Navigat., Control (GNC) Conf.*, Aug. 2013, p. 4622.

[16] T. Schouwenaars, "Safe trajectory planning of autonomous vehicles," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2005.

[17] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. Amer. Control Conf.*, May 2002, pp. 1936–1941.

[18] S. Hrabar, "3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 807–814.

[19] A. F. Filippov, *Differential Equations With Discontinuous Righthand Sides: Control Systems*, vol. 18. Berlin, Germany: Springer, 2013.

[20] F. Blanchini and S. Miani, *Set-Theoretic Methods Control*. Basel, Switzerland: Birkhäuser, 2008.

[21] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding-horizon control of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 46, no. 5, pp. 776–783, May 2001.

[22] J. M. Lee, "Smooth manifolds," in *Introduction to Smooth Manifolds*. New York, NY, USA: Springer, 2003, pp. 1–29.

[23] H. Seywald and R. R. Kumar, "Desensitized optimal trajectories," *Adv. Astron. Sci.*, vol. 93, no. 1, pp. 103–116, 1996.

[24] I. A. Hiskens and M. A. Pai, "Trajectory sensitivity analysis of hybrid systems," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 47, no. 2, pp. 204–220, 2000.

[25] J.-P. Afman, L. Ciarletta, E. Feron, J. Franklin, T. Gurriet, and E. N. Johnson, "Towards a new paradigm of UAV safety," 2018, *arXiv:1803.09026*. [Online]. Available: http://arxiv.org/abs/1803.09026

[26] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 3592–3599.

[27] M. A. Patterson and A. 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 Trans. Math. Softw.*, vol. 41, no. 1, pp. 1–37, Oct. 2014.

[28] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," 2017, *arXiv:1711.08013*. [Online]. Available: http://arxiv.org/abs/1711.08013

[29] *Experimental Results for the Segway*. Accessed: Oct. 13, 2020. [Online]. Available: youtu.be/7wZJksFMAvk

[30] *International Federation of Robotics*. Accessed: Oct. 13, 2020. [Online]. Available: https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf

[31] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames, "Online active safety for robotic manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 173–178.

[32] R. Featherstone, "A divide-and-conquer articulated-body algorithm for parallel O(log(n)) calculation of rigid-body dynamics—Part 1: Basic algorithm," *Int. J. Robot. Res.*, vol. 18, no. 9, pp. 867–875, Sep. 1999.
[33] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1985, pp. 500–505.
[34] W. Khalil, "Dynamic modeling of robots using recursive Newton–Euler techniques," in *Proc. 7th Int. Conf. Inform. Control, Automat. Robot. (ICINCO)*, vol. 1, Jun. 2010, pp. 19–31.
[35] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2019, pp. 614–619.
[36] *Simulation of the Robotic Arm*. Accessed: Oct. 13, 2020. [Online]. Available: vimeo.com/320906655
[37] R. Jiménez-Fabián and O. Verlinden, "Review of control algorithms for robotic ankle systems in lower-limb orthoses, prostheses, and exoskeletons," *Med. Eng. Phys.*, vol. 34, no. 4, pp. 397–408, May 2012.
[38] M. R. Tucker, J. Olivier, A. Pagel, H. Bleuler, M. Bouri, O. Lambercy, J. R. del Millán, R. Riener, H. Vallery, and R. Gassert, "Control strategies for active lower extremity prosthetics and orthotics: A review," *J. Neuroeng. Rehabil.*, vol. 12, no. 1, p. 1, 2015.
[39] K. Anam and A. A. Al-Jumaily, "Active exoskeleton control systems: State of the art," *Procedia Eng.*, vol. 41, pp. 988–994, Jan. 2012.
[40] R. Drillis, R. Contini, and M. Bluestein, *Body Segment Parameters*. New York, NY, USA: Univ., School of Engineering and Science Research Division, 1966.
[41] D. A. Winter, *Biomechanics Motor Control Human Movement*. Hoboken, NJ, USA: Wiley, 2009.
[42] T. Gurriet, S. Finet, G. Boeris, A. Duburcq, A. Hereid, O. Harib, M. Masselin, J. Grizzle, and A. D. Ames, "Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2804–2811.
[43] L. L. Cai, A. J. Fong, C. K. Otoshi, Y. Liang, J. W. Burdick, R. R. Roy, and V. R. Edgerton, "Implications of assist-as-needed robotic step training after a complete spinal cord injury on intrinsic strategies of motor learning," *J. Neurosci.*, vol. 26, no. 41, pp. 10564–10568, Oct. 2006.
[44] T. Gurriet, M. Tucker, A. Duburcq, G. Boeris, and A. D. Ames, "Towards variable assistance for lower body exoskeletons," *IEEE Robot. Autom. Lett.*, vol. 5, no. 1, pp. 266–273, Jan. 2020.
[45] A. D. Ames, "Human-inspired control of bipedal walking robots," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1115–1130, May 2014.
[46] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1447–1454.
[47] A. Agrawal, O. Harib, A. Hereid, S. Finet, M. Masselin, L. Praly, A. D. Ames, K. Sreenath, and J. W. Grizzle, "First steps towards translating HZD control of bipedal robots to decentralized control of exoskeletons," *IEEE Access*, vol. 5, pp. 9919–9934, 2017.
[48] O. Harib, A. Hereid, A. Agrawal, T. Gurriet, S. Finet, G. Boeris, A. Duburcq, M. E. Mungai, M. Masselin, A. D. Ames, K. Sreenath, and J. W. Grizzle, "Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking," *IEEE Control Syst.*, vol. 38, no. 6, pp. 61–87, Dec. 2018.
[49] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot locomotion*. Boca Raton, FL, USA: CRC Press, 2018.
[50] D. Angeli and E. D. Sontag, "Monotone control systems," *IEEE Trans. Autom. Control*, vol. 48, no. 10, pp. 1684–1698, Oct. 2003.
[51] *Experimental Results for the Exoskeleton*. Accessed: Oct. 13, 2020. [Online]. Available: youtu.be/UJC5j4BFxyo
[52] A. Singletary, T. Gurriet, P. Nilsson, and A. D. Ames, "Enabling rapid aerial exploration of unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 10270–10276.
[53] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017.
[54] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3179–3192, Nov. 2017.
[55] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auto. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.
[56] H. Samet, "Neighbor finding in images represented by octrees," *Comput. Vis., Graph., Image Process.*, vol. 46, no. 3, pp. 367–386, Jun. 1989.
[57] T. Gurriet and L. Ciarletta, "Towards a generic and modular geofencing strategy for civilian UAVs," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 540–549.
[58] *Simulation of the UAV*. Accessed: Oct. 13, 2020. [Online]. Available: youtu.be/MdOKY-ykCSw

THOMAS GURRIET received the M.S. degree in mechanical engineering from the Arts et Métiers ParisTech and Georgia Institute of Technology, in 2015. He is currently pursuing the Ph.D. degree with the Department of Mechanical and Civil Engineering, California Institute of Technology. His current research interests are in safety-critical control.

MARK MOTE received the B.S. and M.S. degrees in aerospace engineering from the Georgia Institute of Technology, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree in robotics with the School of Aerospace Engineering. His current research interests are in trajectory planning and runtime-assurance for safety-critical systems.

ANDREW SINGLETARY received the B.S. degree in mechanical engineering and nuclear and radiological engineering from the Georgia Institute of Technology, in 2017, and the M.S. degree in mechanical engineering from the California Institute of Technology, in 2019, where he is currently pursuing the Ph.D. degree in mechanical engineering. His current research interests include safety-critical control and planning for multi-agent systems.

PETTER NILSSON (Associate Member, IEEE) received the B.S. degree in engineering physics and the M.S. degree in optimization and systems theory from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2011 and 2013, respectively, the Ph.D. degree in electrical engineering from the University of Michigan, in 2017, and the B.S. degree in business and economics from the Stockholm School of Economics. He is currently a Postdoctoral Scholar with the California Institute of Technology, where he conducts research on specification-driven control and autonomy for safety-critical cyber-physical systems, with applications in autonomous driving, space exploration, and multi-agent coordination.

ERIC FERON received the B.S., M.S., and Ph.D. degrees from Ecole Polytechnique and Ecole Normale Supérieure, France, and Stanford University, USA. He is currently a Professor of Electrical and Computer Engineering with the King Abdullah University of Science and Technology, Saudi Arabia. He is on leave from the Georgia Institute of Technology, USA. He uses his knowledge of control systems, computer science, and operations research to support innovative aerospace engineering applications.

AARON D. AMES (Senior Member, IEEE) received the B.S. degree in mechanical engineering and the B.A. degree in mathematics from the University of St. Thomas, in 2001, and the M.A. degree in mathematics and the Ph.D. degree in electrical engineering and computer sciences from UC Berkeley, in 2006. He is currently the Bren Professor of Mechanical and Civil Engineering and Control and Dynamical Systems at Caltech. Prior to joining Caltech, in 2017, he was an Associate Professor at the Woodruff School of Mechanical Engineering and the School of Electrical and Computer Engineering, Georgia Tech. He has served as a Postdoctoral Scholar in Control and Dynamical Systems with Caltech, from 2006 to 2008, and began his faculty career at Texas A&M University, in 2008. At UC Berkeley, he was a recipient of the 2005 Leon O. Chua Award for achievement in nonlinear science and the 2006 Bernard Friedman Memorial Prize in Applied Mathematics. He received the NSF CAREER award, in 2010, and the 2015 Donald P. Eckman Award.