Safety Controller Synthesis Using Human Generated Trajectories

Andrew Winn, *Student Member, IEEE,* A. Agung Julius, *Member, IEEE,*

Abstract

This paper focuses on the task of safety controller synthesis, that is, designing a controller that will take a system from any point within a compact set of initial states to a point inside a set of acceptable goal states, while never entering any state that is deemed unsafe. To do this we use a human-generated trajectory based approach. We introduce the control autobisimulation function, which is the analog of the control Lyapunov function for approximate bisimulation. We consider a class of hybrid systems and use this function to determine a set of admissible feedback control laws that guarantee trajectory robustness for underlying dynamics that are linear affine, feedback linearizable, and differentially flat. This property ensures that any trajectory of the closed-loop system that is initialized within some neighborhood of a nominal trajectory will stay within some tube of the nominal trajectory when given the same input. This feedback control and input can be used as the controller for some subset of the initial states. We demonstrate how to combine multiple trajectories into a synthesized controller that satisfies the safety problem.

Index Terms

Hybrid systems, feedback linearization, nonlinear systems.

A preliminary version of this work, dealing with safety controller synthesis for linear affine systems, was presented at the 49th IEEE Conference on Decision and Control, Atlanta, GA, 2010. Another preliminary version of this work, dealing with safety controller synthesis for feedback linearizable and differentially flat systems, was presented at the American Control Conference, 2012. This work was supported in part by the National Science Foundation (NSF) CAREER grant CNS-0953976 & CNS-1218109 and the Department of Defense SMART Scholarship

A. Winn and A. A. Julius are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, 12180 USA (email: winna@rpi.edu, agung@ecse.rpi.edu; mailing address: JEC 6th fl, 110 8th St, Troy NY 12180; fax: (518) 276-6261; preferred address of correspondence: winna@rpi.edu & agung@ecse.rpi.edu)

I. Introduction

The issue of safety/reachability is a concept that has been examined in great detail by the hybrid systems community. On one side of the issue we have *analysis*, which is a concept where a system is analyzed to see if it reaches a desired state or states while maintaining safety, wherein the system does not enter any state that is deemed unsafe. Safety/reachability has seen a lot of practical use in varied applications such as the safety analysis of air traffic systems [1], design verification for electronic circuits [2], design verification for synthetic biology (e.g. [3]), and model analysis for biochemical processes [4]. The other side of the issue is *synthesis*, wherein there is no preexisting controller and the desired safety/reachability properties are used to guide the controller design. For example, the optimal control method in [5], [6] and the simulation based method in [7] directly characterize the influence of the control input in the reachability formulation. The predicate abstraction technique for systems with piecewise affine dynamics in polytope sets leads to a control procedure based on the transversality of the vector field on the facets of the polytopes [8], [9]. The technique for discrete-time systems presented in [10] utilizes partitioning of the state space by polygonal approximation of the reachable set. For continuous dynamical systems, the theoretical results presented in [11] discuss some sufficient conditions for the existence of a controlled system trajectory that enters a prescribed Goal set.

Our approach is distinguished from these methods in that it is a *trajectory-based* technique. The synthesis paradigm uses execution trajectories of a system or simulations thereof to directly formulate a safe controller. The main idea is to guarantee that some compact set of initial states nearby a trajectory will produce trajectories that continue to stay nearby. This property can be ensured using trajectory robustness [12], [13] or incremental stability [14], [15]. Roughly speaking, these properties can provide us with a bound on the divergence of the trajectories (i.e. their relative distances in $\mathcal{L}_\infty$). The main conceptual tool that is used in this approach, the *approximate bisimulation*, was developed by Girard and Pappas [16], and has been used for trajectory based analysis of hybrid systems in [17], [18], [19].

The approach in [14], [15] and our approach differ in the way trajectory robustness is used in controller synthesis. In [14], [15], the notion of approximate bisimulation is used to establish a quantization of the continuous state space, which can result in a countable transition system approximation of the original dynamics. In our approach, the controller is synthesized using

finitely many valid human-generated trajectories [13]. Also, we do not require the open loop dynamics to be incrementally stable. Instead, a part of the controller synthesis procedure is devoted to establishing this property. In a similar spirit, more recent works by Zamani and Tabuada [20][21] also drop the incremental stability requirement and aim to recover it by using a backstepping controller design.

For our research we use the notion of a control autobisimulation function [12] to characterize a class of feedback laws that formally guarantee the trajectory robustness property used in our synthesis design. This can be thought of as an analogue of the control Lyapunov function [22] for autobisimulation. Approximate bisimulation [16] is a tool that quantifies the extent to which two different systems are similar. We use the term autobisimulation to emphasize that we are considering an approximate bisimulation between a system and itself.

The trajectories used in our controller synthesis technique can be generated by any means; however, our motivation is to have humans generate trajectories via computer simulations, i.e., video games. For complex systems where generating feasible controllers is too computationally intensive, the task can be "crowdsourced" to allow humans to generate the required trajectories. The idea behind doing this is that humans can employ heuristics that allow them to find a solution without searching the entire state space. Further, this would allow many humans working simultaneously on their own computers to solve the problem, which provides much more computational power than might be otherwise available. Since the generation of one trajectory does not depend on the generation of other trajectories, the task is highly parallelizable, and hence well suited for the crowdsourcing paradigm. A recent work by Langbort et al. investigated the use of a network of human players in a collaborative computer game [23]. In this case, an online ouija board game is introduced. This is a server-based game with the goal of driving a token across an alphabetical board and spelling as many words as possible in a given time by a team of agents. The highly parallelizable nature of the trajectory-based approach makes it ideal for this setup. Pioneering efforts in the area of exploiting online computer games to construct reliable human centered computation can be found in [24], [25], [26] and the references therein.

This paper provides a cohesive treatment of the results presented in [13] and [27], and extends the mathematical analysis to a class of hybrid systems. In this way we present a technique for synthesizing a controller that meets some given safety criteria using human generated trajectories for hybrid systems whose underlying continuous dynamics are linear affine, feedback linearizable

[28], or differentially flat [29], [30]. We present three example systems to illustrate the techniques and design methodologies presented in this paper. The first and third are taken from [13] and [27], and the second is a novel example of a hybrid system with feedback linearizable dynamics.

II. Problem Formulation

Consider a hybrid system defined by the tuple $(\mathcal{L}, \mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{U}, \mathcal{E}, \texttt{Inv}, \Sigma)$ as described in [31]. $\mathcal{L}$ is a finite set that represents the different discrete states or *locations* that determine the continuous dynamics of the system. $\mathcal{X} \subset \mathbb{R}^n$ represents the continuous state space for the hybrid system, and $\mathcal{Y} \subset \mathbb{R}^l$ represents the output space. $\mathcal{A}$ is a finite set of symbols which can be viewed as the values of the discrete input to the hybrid system. The discrete input of the system consists of a sequence of tuples $a[i] = (a^{(i)}, t^{(i)})$ for $i = 1, 2, \ldots, N$ where $a^{(i)} \in \mathcal{A}$ is the value and $t^{(i)}$ is the time at which the input is applied. $\mathcal{U}$ is the space of continuous external variables, which in our case is the space of input signals. $\mathcal{E}$ is a set of five-tuples $(\ell, a, \texttt{Guard}, \texttt{Reset}, \ell')$ known as transitions or *events*. For each tuple $(a^{(i)}, t^{(i)})$ in the input sequence, if there is an event such that $a = a^{(i)}$, the discrete state of the system at time $t^{(i)}$ is given by $\ell \in \mathcal{L}$, and the continuous state $x \in \mathcal{X}$ at time $t^{(i)}$ is within the set $\texttt{Guard} \subset \mathcal{X}$, then the event occurs, and the discrete state changes to state $\ell' \in \mathcal{L}$ and the continuous state x resets to a new state $x' \in \mathcal{X}$ given by the mapping $\texttt{Reset} : \mathcal{X} \to \mathcal{X}$. If two events have the same ℓ and a, then their $\texttt{Guards}$ must be mutually exclusive sets, so that only one event occurs at a given time. $\texttt{Inv}$ is the *location invariant* of location ℓ, and defines the set of states for which $x \in \texttt{Inv}(\ell) \subset \mathcal{X}$ whenever the system is at location ℓ, and Σ assigns to each location a set of continuous dynamics,

$$\Sigma(\ell) : \begin{cases} \frac{dx}{dt} = f_\ell(x, u),\ x \in \mathcal{X} = \mathbb{R}^n,\ u \in \mathcal{U} \subset \mathbb{R}^m, \\ y = h_\ell(x),\ y \in \mathcal{Y} = \mathbb{R}^l. \end{cases} \tag{1}$$

where $f_\ell(x, u)$ is locally Lipschitz in x and continuous in u.

Throughout this paper we shall use the term "trajectory" as defined in [31]. The trajectory of a hybrid system consists of a sequence of continuous trajectories and discrete inputs such that every transition between consecutive continuous trajectories corresponds to some event in $\mathcal{E}$.

Suppose that there is a given compact set of initial states $\texttt{Init} \subset \mathcal{X} \times \mathcal{L}$, where the state is initiated at $t = 0$, i.e. $(x(0), \ell_0) \in \texttt{Init}$. Also, we assume that there is a set of goal states, $\texttt{Goal} \subset \mathcal{Y} \times \mathcal{L}$, a set of unsafe states $\texttt{Unsafe} \subset \mathcal{Y} \times \mathcal{L}$, and a function $\texttt{Input} : \mathcal{L} \to 2^{\mathcal{U}}$ which

assigns a set of allowable inputs to each location. As usual, a trajectory is deemed unsafe if it enters the unsafe set. The safety control problem can be formulated as follows [13]:

Problem 1 (Safety Controller Synthesis)**.** *Design a discrete input $a[i] = (a^{(i)}, t^{(i)})$ and a feedback control law $u = k(t, \ell, x)$ such that for any initial state $(x_0, \ell_0) \in$ `Init`, the trajectory of the closed loop system enters* `Goal` *before time $T = T_{\max}$, and both remains safe and respects input constraints until it enters* `Goal`.

Definition 1 (Valid Trajectory)**.** *Any trajectory that satisfies the conditions in Problem 1 is called a valid trajectory.*

Remark 1. *Note that* `Goal` *and* `Unsafe` *in Problem 1 are formulated in terms of the output space $\mathcal{Y}$ for the system. This will be necessary when considering systems with zero dynamics in Section IV-B. To consider safety criteria with respect to the full state space, one may set the output functions $h_\ell(x)$ in (1) to be the identity function, and $\mathcal{Y} = \mathcal{X}$.*

III. Autobisimulation and Trajectory Robustness

We establish trajectory robustness through a Lyapunov-like function called a *control autobisimulation function (CAF)* [13].

Definition 2 (Control Autobisimulation Function)**.** *Consider a hybrid system with input whose continuous dynamics are as given in (1). A continuously differentiable function $\psi_\ell : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is a* ***control autobisimulation function*** *for location ℓ of (1) if it is a pseudometric and there exists a function $k_\ell : [0, T] \times \mathcal{X} \to \mathcal{U}$ such that*

$$\nabla_x \psi_\ell(x, x') f(x, k_\ell(t, x)) + \nabla_{x'} \psi_\ell(x, x') f(x', k_\ell(t, x')) \leq 0. \tag{2}$$

Note that a pseudometric, unlike a metric, does not require separate points to be distinguishable, i.e., $\psi_\ell(x, x')$ is allowed to be zero for $x \neq x'$.

The control autobisimulation function is an analog of the control Lyapunov function (CLF)[22] for approximate bisimulation [32][18]. Whereas CLFs have been used to construct control laws that guarantee stability about an equilibrium point, [33], we shall use CAFs to construct control laws that guarantee stability about a nominal trajectory, a property that we call *trajectory robustness*.

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

The concept of approximate bisimulation was first introduced in the seminal work of Girard and Pappas [32]. It has been used for bounding the divergence of output trajectories of continuous and hybrid systems. For autonomous systems (i.e. systems without inputs/nondeterminism), approximate bisimulation is similar to the notion of contraction metric coined by Lohmiller and Slotine (see e.g. [34]). A minor difference between the two notions lies in the fact that approximate bisimulation allows for the use of pseudometric in the state space because it emphasizes on the distance between the outputs of the systems.

One can compare the CAF to a CLF for the product space of the system with itself, i.e., the system defined by

$$\frac{dx}{dt} = f_\ell(x,u),$$
$$\frac{dx'}{dt} = f_\ell(x',u').$$

However, whereas a CLF for this system could set $u = \tilde{k}_\ell(x,x')$ and $u' = \tilde{k}_\ell(x,x')$, that is, a function of both states, the CAF requires that $u = k_\ell(t,x)$ and $u' = k_\ell(t,x')$. In this way the requirement for a CAF is more restrictive than that for a CLF.

Definition 3 (Class of Admissible Feedback Laws for a Location)**.** *Consider location ℓ of a hybrid system with input whose continuous dynamics are as given in (1), and assign to ℓ a control autobisimulation function ψ_ℓ. The class of all feedback control laws $k_\ell(t,x)$ such that $\psi_\ell(\cdot,\cdot)$ and $k_\ell(\cdot,\cdot)$ satisfy (2) for each $\ell \in \mathcal{L}$ is called the* ***class of admissible feedback laws for location*** *ℓ, $\eta\left(\Sigma(\ell),\psi_\ell\right)$.*

In our discussion of trajectory robustness we make use of the following notations:

Notation 1. *Consider a hybrid system with input whose continuous dynamics are as given in (1) and a feedback control law $u = k(t,x)$, the evolution of the system dynamics from state x_0 within location ℓ_0 with no discrete input to induce an event is denoted by $\xi_k(t,\ell_0,x_0)$. Further, given a discrete input sequence $a[i]$, the closed loop trajectory initiated at $x(0) = x_0$ and location $\ell(0) = \ell_0$ is denoted by $\xi_{a,k}(t,\ell_0,x_0)$. For each event occurring at time $t^{(i)}$ we denote $\lim_{t\to t^{(i)}-} \xi_{a,k}(t,\ell_0,x_0)$, the trajectory's value immediately prior to the event occurring, by $\xi^-_{a,k}(t^{(i)},\ell_0,x_0)$.*

Notation 2. *For any $x \in \mathcal{X}$ and $\delta \geq 0$ we denote the set $\{x' \in \mathcal{X} \mid \psi_\ell(x,x') \leq \delta\}$ as $B_{\psi_\ell}(x,\delta)$.*

Proposition 2. *Consider a hybrid system with input whose continuous dynamics are as given in (1), with a CAF $\psi_\ell(\cdot,\cdot)$ assigned to location $\ell \in \mathcal{L}$ and a feedback law $k(\cdot,\ell,\cdot)$ in the class of admissible feedback laws for location ℓ. For any two initial states $x_0 \in \mathcal{X}$ and $x_0' \in \mathcal{X}$ at time t_0 of the closed loop system $\frac{dx}{dt} = f_\ell(x, k(t,\ell,x))$ and any time span $[t_0, t_1]$ during which no switching occurs, we have that*

$$\psi_\ell(\xi_k(t,\ell,x_0), \xi_k(t,\ell,x_0')) \leq \psi_\ell(x_0, x_0'), \quad \forall\, t \in [t_0, t_1]. \tag{3}$$

Proof. This situation is analogous to a continuous system with dynamics $\frac{dx}{dt} = \overline{f}(x, \overline{k}(t,x))$, where $\overline{f} = f_\ell(\cdot,\cdot)$ and $\overline{k}(\cdot,\cdot) = k(\cdot,\ell,\cdot)$, along with CAF $\overline{\psi}(\cdot,\cdot) = \psi_\ell(\cdot,\cdot)$. The result follows from the proof given for continuous systems in [18]. □

A closed loop system that satisfies (3) is said to be imbued with the property of **trajectory robustness for location** ℓ.

IV. Controller Synthesis

To solve Problem 1, we seek to synthesize a controller that will drive any state in `Init` to `Goal` in time $T < T_{\max}$ while both ensuring that the state never enters `Unsafe` and that the continuous input remains in $\texttt{Input}(\ell)$, which may be a bounded subset of $\mathbb{R}^m$.

To achieve this, we first have a human generate a discrete and a continuous input that yields a valid trajectory for some initial state $(x_0, \ell_0) \in \texttt{Init}$. For each location ℓ visited by the trajectory, we use this input to design a feedback controller in the class of admissible feedback laws for location ℓ (if not already done) using the techniques presented in Section IV-B. Using the trajectory robustness imbued by these feedback laws, we use the results presented in Section IV-A to find the robustness ball around the initial state. Section IV-C provides the computational tools for applying these results. This discrete input and feedback law provide a controller that solves Problem 1 for all states in the robustness ball around (x_0, ℓ_0). We can now repeat the process for a different state $(x_1, \ell_1) \in \texttt{Init} \setminus B_\psi(x_0, \delta_0)$. As a result, we have two different compact subsets of `Init` (not necessarily disjoint) that have safe control laws. One then repeats the above steps until `Init` is covered by a finite number of compact subsets that each have a corresponding safe control law.

Remark 3 (set coverage)**.** *In general, coverage assessment is nontrivial. One way to generate a coverage estimate is to generate random points uniformly over `Init` and determine the ratio*

of these points that are covered to the total number of generated points, which is an unbiased estimator of the actual coverage ratio. The confidence interval of this estimate depends only on the total number of sampled points and the coverage ratio, but not on the dimension of the state space.

The complete final controller operates by applying the following steps:

1) Determine initial states $x(0)$ and ℓ_0 (assumed to be in `Init`).
2) Determine the set of $B_{\psi_{\ell_0}}(x_i, \delta_i)$ to which $x(0)$ belongs. Since the $B_{\psi_{\ell_0}}(x_i, \delta_i)$ are a finite cover of `Init`, $x(0)$ is guaranteed to belong to at least one set. If $x(0)$ belongs to more than one set, then the controller can choose a set by any method, e.g., by choosing the set that minimizes $\|x(0) - x_i\|$.
3) Apply the control law corresponding to the $i^{\texttt{th}}$ trajectory.

A. Hybrid Trajectory Robustness

In this section we shall consider a hybrid system with input whose dynamics are as given in (1).

Definition 4 (Valid Control Law)**.** *The discrete input $a[i] = (a^{(i)}, t^{(i)})$ for $i = 1, 2, \ldots, N$ and the continuous feedback control law $k(t, \ell, x)$ for $t \in [0, T]$ along with a set of initial conditions $t_0, \ell_0,$ and x_0 are said to form a valid control law if*
(i) $\xi_{a,k}(t, \ell_0, x_0)$ is a valid trajectory beginning at time $t^{(0)}$ and ending at time $t^{(N+1)}$ with N events occurring at times $t^{(1)}, t^{(2)} \ldots, t^{(N)}$.
(ii) the function $k_{\ell^{(i)}}(\cdot, \cdot) = k(\cdot, \ell^{(i)}, \cdot)$ is in the class of admissible feedback laws for location $\ell^{(i)}$ for each $i = 0, 1, \ldots, N$., where $\ell^{(i)}$ denotes the location ℓ at time $t \in \left[t^{(i)}, t^{(i+1)}\right)$.

For brevity we denote this control law as (a, k, t_0, ℓ_0, x_0).

Note that for a valid control law (a, k, t_0, ℓ_0, x_0), $\xi_{a,k}(t, \ell_0, x_0)$ will consist of $N+1$ partitions. Our goal at this point is to use the robustness property imbued by the feedback control law to show that there is a neighborhood $B_{\psi_{\ell_0}}(x_0, \delta)$ of initial conditions for which $a[i]$ and $k(t, \ell, x)$ will produce valid trajectories. This will be done by finding the largest neighborhood about each trajectory partition such that

1) the neighborhood does not intersect `Unsafe`,

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

2) $k(t, \ell, x) \in \texttt{Input}(\ell)$ over the neighborhood,
3) and either the neighborhood around the end point is entirely within $\texttt{Goal}$ if it is the last partition, or the end point is entirely within both the guard of the switching event and within the robust neighborhood of the beginning point of the next partition.

To this end, we formalize the requirements on the neighborhood sizes as follows. For the last partition of $\xi_{a,k}(t, \ell_0, x_0)$, define

$$\delta^{(N)}_{\texttt{Goal}} \triangleq \inf_{x_{ng} \notin \texttt{Goal}} \psi_{\ell^{(N)}}(\xi_{a,k}(t^{(N+1)}, \ell_0, x_0), x_{ng}). \tag{4}$$

This defines the shortest distance with respect to the CAF $\psi_{\ell^{(N)}}$ between the end point of the trajectory and the boundary of $\texttt{Goal}$. Next we define the set of neighborhood sizes respecting the input bounds via

$$\Delta^{(i)}_{\texttt{Input}} \triangleq \left\{ \delta \,|\, k(t, \ell^{(i)}, x) \in \texttt{Input}(\ell^{(i)}),\ \forall x \in B_{\psi_{\ell^{(i)}}}(\xi_{a,k}(t, \ell_0, x_0), \delta),\ \forall t \in \left[t^{(i)}, t^{(i+1)}\right) \right\}. \tag{5}$$

For each partition i we define

$$\delta^{(i)}_{\texttt{Unsafe}} \triangleq \inf_{\substack{x' \in \texttt{Unsafe} \\ t^{(i)} \le t \le t^{(i+1)}}} \psi_{\ell^{(i)}}(\xi_{a,k}(t, \ell_0, x_0), x'), \tag{6}$$

$$\delta^{(i)}_{\texttt{Input}} \triangleq \sup \Delta^{(i)}_{\texttt{Input}}. \tag{7}$$

These represent the largest distances for each partition such that the corresponding neighborhoods do not violate the safety criteria or input bounds respectively. Let

$$\texttt{Reset}^{(i)^\dagger}(S) \triangleq \left\{ x \,|\, x \in \texttt{Guard}^{(i)}, \texttt{Reset}^{(i)}(x) \in S, S \subseteq \mathcal{X} \right\}, \tag{8}$$

where $\texttt{Guard}^{(i)}$ and $\texttt{Reset}^{(i)}$ are the $\texttt{Guard}$ and $\texttt{Reset}$ map associated with the i^{th} event.

Now define

$$\delta^{(i)} \triangleq \min(\delta^{(i)}_{\texttt{Unsafe}}, \delta^{(i)}_{\texttt{Input}}, \delta^{(i)}_{\texttt{Goal}}), \tag{9}$$

where for $i < N$, $\delta^{(i)}_{\texttt{Goal}}$ is the largest value of δ such that

$$B_{\psi_{\ell^{(i)}}}(\xi^{-}_{a,k}(t^{(i+1)}, \ell_0, x_0), \delta) \subset \texttt{Reset}^{(i+1)^\dagger}\left(B_{\psi_{\ell^{(i+1)}}}(\xi_{a,k}(t^{(i+1)}, \ell_0, x_0), \delta^{(i+1)}) \right). \tag{10}$$

This specifies the largest neighborhood about each partition that maintains validity with respect to Problem 1. Note that (9) and (10) are used to iteratively determine $\delta^{(i)}$ from $i = N$ to $i = 0$. We shall now use these distances to prove the following lemma.

Lemma 4. *Consider a hybrid system with input whose continuous dynamics are as given in (1) along with the valid control law* (a, k, t_0, ℓ_0, x_0)*. Further consider a trajectory initiated at* x_0' *generated by the same inputs* $a[i]$ *and* $k(t, \ell, x)$*. If* $\xi_{a,k}(t^{(i)}, \ell_0, x_0') \in B_{\psi_{\ell^{(i)}}}(\xi_{a,k}(t^{(i)}, \ell_0, x_0), \delta^{(i)})$ *for some* $0 \leq i < N$*, then for all* $t \in \left[t^{(i)}, t^{(i+1)}\right)$*,*

$$\xi_{a,k}(t, \ell_0, x_0') \notin \texttt{Unsafe}, \tag{11}$$

$$k(t, \ell^{(i)}, \xi_{a,k}(t, \ell_0, x_0')) \in \texttt{Input}(\ell^{(i)}), \tag{12}$$

$$\xi_{a,k}(t^{(i+1)}, \ell_0, x_0') \in B_{\psi_{\ell^{(i+1)}}}(\xi_{a,k}(t^{(i+1)}, \ell_0, x_0), \delta^{(i+1)}) \tag{13}$$

Proof. By Proposition 2 and Notation 2 we have

$$\psi_{\ell^{(i)}}(\xi_{a,k}(t, \ell_0, x_0), \xi_{a,k}(t, \ell_0, x_0')) \leq \psi_{\ell^{(i)}}(\xi_{a,k}(t^{(i)}, \ell_0, x_0), \xi_{a,k}(t^{(i)}, \ell_0, x_0')) < \delta^{(i)}. \tag{14}$$

From (9) we see that $\delta^{(i)} \leq \delta^{(i)}_{\texttt{Unsafe}}$. Combining this fact with (14) and (6) we note that for $t \in \left[t^{(i)}, t^{(i+1)}\right)$,

$$\psi_{\ell^{(i)}}(\xi_{a,k}(t, \ell_0, x_0), \xi_{a,k}(t, \ell_0, x_0')) < \psi_{\ell^{(i)}}(\xi_{a,k}(t, \ell_0, x_0), x_u), \quad \forall x_u \in \texttt{Unsafe}.$$

If $\xi_{a,k}(t, \ell_0, x_0) \in \texttt{Unsafe}$ for some $t = \bar{t}$, then we could set $x_u = \xi_{a,k}(\bar{t}, \ell_0, x_0)$ to yield

$$\psi_{\ell^{(i)}}(\xi_{a,k}(\bar{t}, \ell_0, x_0), \xi_{a,k}(\bar{t}, \ell_0, x_0')) < \psi_{\ell^{(i)}}(\xi_{a,k}(\bar{t}, \ell_0, x_0), \xi_{a,k}(\bar{t}, \ell_0, x_0')),$$

a contradiction. Thus (11) must hold.

From (9) we also see that $\delta^{(i)} \leq \delta^{(i)}_{\texttt{Input}}$. By using this fact with (14) and applying Notation 2 we see that $\xi_{a,k}(t, \ell_0, x_0') \in B_{\psi_{\ell^{(i)}}}(\xi_{a,k}(t, \ell_0, x_0), \delta^{(i)}_{\texttt{Input}})$. Combining this fact with (5) and (7) yields (12).

By the same logic we have

$$\xi^-_{a,k}(t^{(i+1)}, \ell_0, x_0') \in B_{\psi_{\ell^{(i)}}}(\xi^-_{a,k}(t^{(i+1)}, \ell_0, x_0), \delta^{(i)}) \tag{15}$$

By (10) we have

$$\xi^-_{a,k}(t^{(i+1)}, \ell_0, x_0') \in \texttt{Reset}^{(i+1)^\dagger}\left(B_{\psi_{\ell^{(i+1)}}}(\xi_{a,k}(t^{(i+1)}, \ell_0, x_0), \delta^{(i+1)})\right). \tag{16}$$

By (8), we see that $\xi^-_{a,k}(t^{(i+1)}, \ell_0, x_0')$ will be in the guard of the i^{th} event, and will correctly transition to the next discrete state. Further, we can see that the trajectory will be reset into $B_{\psi_{\ell^{(i+1)}}}(\xi_{a,k}(t^{(i+1)}, \ell_0, x_0), \delta^{(i+1)})$, yielding (13), as desired. □

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

Definition 5 (Radius of Robustness)**.** *Consider a hybrid system with input whose continuous dynamics are as given in (1) along with the valid control law* (a, k, t_0, ℓ_0, x_0)*. The* radius of robustness *for the trajectory* $\xi_{a,k}(t, \ell_0, x_0)$ *is given by* $\delta = \delta^{(0)}$*, where* $\delta^{(0)}$ *is determined via (4)–(10).*

Now we are in a position to present the main result of this section.

Proposition 5. *Consider a hybrid system with input whose continuous dynamics are as given in (1) along with the valid control law* (a, k, t_0, ℓ_0, x_0)*. For all* $x_0' \in B_{\psi_{\ell^{(0)}}}(x_0, \delta)$*, where* δ *is the radius of robustness,* $\xi_{a,k}(t, \ell_0, x_0')$ *is a valid trajectory.*

Proof. By Lemma 4, we have that for all $t \in \left[t^{(0)}, t^{(1)}\right)$, $\xi_{a,k}(t, \ell_0, x_0')$ is safe and respects the inputs bounds. Further, we have that $\xi_{a,k}(t^{(1)}, \ell_0, x_0') \in B_{\psi_{\ell^{(1)}}}(\xi_{a,k}(t^{(1)}, \ell_0, x_0), \delta^{(1)})$, From which we can again apply Lemma 4. By repeated application of Lemma 4, we see that the entire trajectory is safe and respects the input bounds. We also see that

$$\xi_{a,k}(t^{(N)}, \ell_0, x_0') \in B_{\psi_{\ell^{(N)}}}(\xi_{a,k}(t^{(N)}, \ell_0, x_0), \delta^{(N)}) \tag{17}$$

It only remains to prove that the trajectory terminates inside of `Goal`. By Proposition 2 and Notation 2 we have

$$\psi_{\ell^{(N)}}(\xi_{a,k}(t^{(N+1)}, \ell_0, x_0), \xi_{a,k}(t^{(N+1)}, \ell_0, x_0')) \leq \psi_{\ell^{(N)}}(\xi_{a,k}(t^{(N)}, \ell_0, x_0), \xi_{a,k}(t^{(N)}, \ell_0, x_0')) < \delta^{(N)}. \tag{18}$$

From (9) we see that $\delta^{(N)} \leq \delta^{(N)}_{\texttt{Goal}}$. Combining this fact with (18) and (4) we have

$$\psi_{\ell^{(N)}}(\xi_{a,k}(t^{(N+1)}, \ell_0, x_0), \xi_{a,k}(t^{(N+1)}, \ell_0, x_0')) < \psi_{\ell^{(N)}}(\xi_{a,k}(t^{(N+1)}, \ell_0, x_0), x_{ng}), \quad \forall x_g \notin \texttt{Goal}. \tag{19}$$

If $\xi_{a,k}(t^{(N+1)}, \ell_0, x_0'))$ is not in `Goal`, then we can set x_{ng} to be this state in (19), yielding a contradiction. Thus, the final state of the trajectory must be in `Goal`, and validity of the trajectory is proven. □

Remark 6. *If any of the* `Guard`*s associated with the hybrid switching of a trajectory have measure zero, such as a two-dimensional plane in a three-dimensional state space, then the radius of robustness will necessarily be zero. As such, the method outlined in this paper is only useful for trajectories that pass through* `Guard`*s with non-zero measure.*

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

B. Generating Feedback Laws for Different Classes of Systems

In order to apply the results given in Section IV-A, one needs to be able to find a valid control law (a, k, t_0, ℓ_0, x_0). This can be achieved via the following method.

1) Have a human find a valid open-loop trajectory with initial condition (x_0, ℓ_0) by controlling the system or a simulation thereof. Call the discrete and continuous inputs to this trajectory $a[i]$ and $u_0(t)$ respectively.
2) For each location ℓ that the trajectory visits, find a CAF ψ_ℓ and a $k_\ell(t, x)$ in the class of admissible feedback laws for location ℓ such that $k_\ell(t, \xi_{a,k}(t, \ell_0, x_0)) = u_0(t)$.
3) Define $k(\cdot, \ell, \cdot) = k_\ell(\cdot, \cdot)$ for each location ℓ visited by the trajectory.

A method for performing steps 2–3 above for three different classes of nonlinear dynamics are given in Sections IV-B1–IV-B3. Since this method applies to an individual location, we will drop the indexes that identify the location. For example, $\Sigma, \delta_{\texttt{Unsafe}}$, and $\delta_{\texttt{Input}}$ will be used in place of $\Sigma(\ell^{(i)}), \delta^{(i)}_{\texttt{Unsafe}}$, and $\delta^{(i)}_{\texttt{Input}}$.

1) Linear Affine Systems: The results in this section have been adapted from [13]. Consider a system with linear affine dynamics, that is,

$$\Sigma_{\text{lin}} : \frac{dx}{dt} = Ax + Bu + c, \quad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m, \tag{20}$$

where $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$, and $c \in \mathbb{R}^n$. For such systems we propose to construct a CAF using quadratic functions that satisfies (2). Specifically, we take the CAF to be

$$\psi(x, x') = [(x - x')^T P(x - x')]^{\frac{1}{2}}, \tag{21}$$

where $P \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix. We see from (2) that in order to be a CAF this function must satisfy

$$\frac{(x - x')P(Ax + Bk(t, x) + c)}{[(x - x')^T P(x - x')]^{-\frac{1}{2}}} - \frac{(x - x')P(Ax' + Bk(t, x') + c)}{[(x - x')^T P(x - x')]^{-\frac{1}{2}}} \leq 0, \quad \forall x, x' \in \mathbb{R}^n. \tag{22}$$

By multiplying through by the denominator and combining terms, we can represent this condition as

$$(x - x')^T P(A(x - x') + B(k(t, x) - k(t, x'))) \leq 0, \quad \forall x, x' \in \mathbb{R}^n. \tag{23}$$

We propose to construct a feedback law of the form

$$u(t) = k(t, x) = Kx + v(t), \tag{24}$$

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

where $K \in \mathbb{R}^{m\times n}$ and $v(t) \in \mathbb{R}^m$. We take $v(t)$ to be

$$v(t) = u_0(t) - K\xi_k(t, x_0), \tag{25}$$

where $u_0(t)$ is the nominal input generated by a human (or some other external entity) and $\xi_k(t, x_0)$ is the corresponding state trajectory. This guarantees that $u(t) = u_0(t)$ when $x(0) = x_0$. Further, note that $k(t, x) - k(t, x') = K(x - x')$ does not depend on $v(t)$, and thus K may be chosen to satisfy (23) without prior knowledge of $v(t)$. Combining (23) with (24),

$$(x - x')^T P(A + BK)(x - x') \leq 0, \quad \forall x, x' \in \mathbb{R}^n. \tag{26}$$

Finding a K that satisfies inequality (26) is equivalent to finding a K such that $(A + BK)$ is Hurwitz. Such a K will exist if and only if (A, B) is stabilizable. We can determine this K using the following method. We first recognize that $(A + BK)$ is Hurwitz if and only if $(A + BK)^T$ is Hurwitz. If this is true, then there exists a solution to the Lyapunov equation

$$(A + BK)\widetilde{P} + \widetilde{P}(A + BK)^T \preceq 0, \quad \widetilde{P} = \widetilde{P}^T \succ 0.$$

Carrying out some algebraic manipulations allows us to write

$$A\widetilde{P} + BD + \widetilde{P}A^T + D^T B^T \preceq 0, \tag{27}$$

where we define $D = K\widetilde{P}$. We can now use standard tools [35] to solve the above linear matrix inequality for D and $\widetilde{P}$, and then solve for K and P using

$$K = D\widetilde{P}^{-1}, \quad P = \widetilde{P}^{-1}. \tag{28}$$

Since $\widetilde{P}$ is positive definite, it is invertible, and (28) is a valid expression.

By applying the feedback control law (24) to the system (20), we obtain the closed loop system

$$\Sigma_{\text{cl}} : \frac{dx}{dt} = (A + BK)x + Bv + c, \quad x \in \mathcal{X},\ v \in \mathcal{U}. \tag{29}$$

2) Feedback Linearizable Systems: The results in this section have been adapted from [27]. Consider a dynamical system with input and output

$$\Sigma_{\text{io}} : \begin{cases} \frac{dx}{dt} = f(x) + g(x)u,\ x \in \mathcal{X},\ u \in \mathcal{U}, \\ y = h(x),\ y \in \mathbb{R}^m, \end{cases} \tag{30}$$

where $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$. Note that the system is affine with respect to its input, and that the dimensions of the input space and the output space are the same. We assume that $f(\cdot)$ is Lipschitz and that $f(\cdot)$, $g(\cdot)$, and $h(\cdot)$ are sufficiently smooth functions, in the sense that all partial derivatives appearing in the ensuing analysis exist. Hereafter, we use the notation $f_i(\cdot)$ and $h_i(\cdot)$ to denote the i-th element of the vector valued functions $f(\cdot)$ and $h(\cdot)$, respectively. For the matrix valued function $g(\cdot)$, we use $g_i(\cdot)$ to denote its i-th row.

Feedback linearization is a classical controller design technique for nonlinear systems (see e.g. [28], [36]). A special case of feedback linearization that is applicable in the output safety controller synthesis problem is the *input-output linearization* [28]. The idea is to introduce a new control input $w(t)$ and design a (nonlinear) feedback law

$$u(t) = \kappa(x) + \lambda(x)w(t),\ w(t) \in \mathbb{R}^m, \tag{31}$$

such that the new system, with input $w(t)$ and output $y(t)$, is a linear system. The design procedure for $\kappa(x)$ and $\lambda(x)$ is given as follows [36].

Notation 3 (Lie Derivatives)**.** *[36] For a smooth scalar valued function $h : \mathbb{R}^n \to \mathbb{R}$, we use the following standard notation for Lie derivatives:*

$$\begin{aligned} L_f^0(h) &\triangleq h(x), \\ L_f^{k+1}(h) &\triangleq \frac{\partial L_f^k(h)}{\partial x} f(x), \\ L_{g_j} L_f^k(h) &\triangleq \frac{\partial L_f^k(h)}{\partial x} g_j^T(x). \end{aligned}$$

Definition 6 (Relative Degree)**.** *[36] For the input-output system Σ_{io} in (30), we define the vector relative degree $\{r_1, r_2, \cdots, r_m\}$ at a point $x_0 \in \mathbb{R}^n$ as the largest integers such that:*
(i) For all $i \in \{1, \ldots, m\}$, $0 \leq j < r_i - 1$, $k \in \{1, \ldots, m\}$,

$$L_{g_k} L_f^j(h_i(x_0)) = 0. \tag{32}$$

(ii) The $m \times m$ matrix $\Gamma(x_0)$ defined such that the element in the ith row and jth column is given by

$$\Gamma_{i,j}(x_0) = L_{g_j} L_f^{r_i - 1}(h_i(x_0)) \tag{33}$$

is nonsingular.

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

If the system Σ_{io} in (30) has a uniform relative degree $\{r_1, r_2, \cdots, r_m\}$ in the entire state space[1], then by definition, we have

$$\begin{bmatrix} \frac{d^{r_1}}{dt^{r_1}} y_1 \\ \vdots \\ \frac{d^{r_m}}{dt^{r_m}} y_m \end{bmatrix} = \underbrace{\begin{bmatrix} L_f^{r_1} h_1(x) \\ \vdots \\ L_f^{r_m} h_m(x) \end{bmatrix}}_{\triangleq \Phi(x)} + \Gamma(x) u. \tag{34}$$

Therefore, by using the feedback law (31), with

$$\kappa(x) = -\Gamma^{-1}(x)\Phi(x), \tag{35}$$

$$\lambda(x) = \Gamma^{-1}(x), \tag{36}$$

we obtain a linear input-output system in the chain integrator form [28]

$$\Sigma_{\text{lin}} : \begin{bmatrix} \frac{d^{r_1}}{dt^{r_1}} y_1 \\ \vdots \\ \frac{d^{r_m}}{dt^{r_m}} y_m \end{bmatrix} = \begin{bmatrix} w_1(t) \\ \vdots \\ w_m(t) \end{bmatrix}. \tag{37}$$

Furthermore, a state space realization of Σ_{lin} can be obtained through some state transformation from x. For example, we can observe that for any $i \in \{1, \ldots, m\}$ and $j < r_i$,

$$y_i = h_i(x), \qquad \frac{d^j}{dt^j} y_i = L_f^j(h_i(x)). \tag{38}$$

Since Σ_{lin} is linear, we can apply the controller synthesis technique give in Section IV-B1 for linear affine systems.

Remark 7 (zero dynamics)**.** *In controller design for nonlinear systems, one typically wants to have asymptotically stable zero dynamics. In this paper, we ignore the possibility of unstable zero dynamics since they are not observable from the system output, and hence are irrelevant to the safety control criteria.*

3) Differentially Flat Systems: The results in this section have been adapted from [27]. Differential flatness is a major tool in nonlinear controller design [29][30]. The concept was first coined by Fliess et al. in [37], and since then there have been thousands of papers that use it in controller design. In this section, we outline a controller synthesis technique for the

[1]This can be relaxed to an invariant subset of interest of the state space.

Output Safety control problem for differentially flat nonlinear systems. Our development in this paper follows the work of van Nieuwstadt and Murray [38], who used differential flatness for trajectory generation in motion planning for constrained mechanical systems.

Any nonlinear system that is affine with respect to its input

$$\frac{dx}{dt} = f(x) + g(x)u, \quad x \in \mathcal{X} = \mathbb{R}^n,\ u \in \mathcal{U} \subset \mathbb{R}^m,$$

is said to be *differentially flat* if it has a set of flat outputs

$$y = h(x, u, \dot{u}, \cdots, u^p), \quad y \in \mathbb{R}^m,$$

for some integer p. The outputs $y = (y_1, \ldots, y_m)$ are *flat outputs* if x and u can be written as functions of y and its time derivatives,

$$x = \Xi(y, \dot{y}, \ldots, y^{(\ell)}), \tag{39}$$

$$u = \Upsilon(y, \dot{y}, \ldots, y^{(\ell+1)}), \tag{40}$$

for some integer ℓ, and $(y, \dot{y}, \ldots, y^{(\ell)})$ are not constrained to satisfy a differential equation by themselves. In other words, any sufficiently smooth trajectory y is admissible.

The concept of differential flatness is tightly related to feedback linearization. In fact, we can show that if Σ_{io} in (30) is feedback linearizable and it does not have any zero dynamics, then it is differentially flat and $y = (y_1, \ldots, y_m)$ are flat outputs [29], [38].

In the subsequent discussion, we assume that Σ_{io} is differentially flat, with y as the flat outputs. Consider the following $m\ell$-th order linear system:

$$\Sigma_{\text{flat}} = \begin{cases} \frac{d}{dt}\begin{bmatrix} \eta \\ \dot{\eta} \\ \vdots \\ \eta^{(\ell-1)} \\ \eta^{(\ell)} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & I \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} \eta \\ \dot{\eta} \\ \vdots \\ \eta^{(\ell-1)} \\ \eta^{(\ell)} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix} \omega, \\ \widehat{y} = \eta, \end{cases} \tag{41}$$

where $\eta \in \mathbb{R}^m$, $\omega \in \mathbb{R}^m$ is the input, and $\widehat{y} \in \mathbb{R}^m$ is the output. Observe that, by construction, any output trajectory of Σ_{flat} is at least $(\ell + 1)$ times differentiable. Therefore, any $\widehat{y}(t)$ that is an output trajectory of Σ_{flat} (regardless of the input $\omega(t)$) is also an output trajectory of Σ_{io}, and vice versa. Furthermore, the corresponding state and input trajectories of Σ_{io} can be computed from (39) - (40).

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

For brevity, let us rewrite the state equation of Σ_{flat} as

$$\frac{dq}{dt} = \widehat{A}q + \widehat{B}\omega,\ q \in \mathbb{R}^{m\ell},\ \omega \in \mathbb{R}^m, \tag{42}$$

$$\widehat{y} = \widehat{C}q,\ \widehat{y} \in \mathbb{R}^m. \tag{43}$$

Notation 4. *We denote the output trajectory of Σ_{flat} starting from an initial state $q_0 \in \mathbb{R}^{m\ell}$ under an input signal $\omega(t)$ by $\widehat{y}(t; q_0, \omega)$.*

Notation 5. *Given that $q = [\eta^T\ \dot{\eta}^T\ \cdots \eta^{(\ell)T}]^T$ and $\omega = \eta^{(\ell+1)}$, we introduce the following shorthand notation:*

$$\Xi(q) \triangleq \Xi(\eta, \dot{\eta}, \cdots, \eta^{(\ell)}),$$

$$\Upsilon(q, \omega) \triangleq \Upsilon(\eta, \dot{\eta}, \cdots, \eta^{(\ell+1)}).$$

Definition 7 (Valid Flat Trajectory)**.** *Consider Problem 1. An output trajectory of Σ_{flat}, $\widehat{y}(t; q_0, \omega)$, is said to be a valid flat trajectory for an initial state $x_0 \in$ `Init` if*
(i) $\Xi(q_0) = x_0$, and
(ii) the trajectory $\widehat{y}(t; q_0, \omega)$ enters `Goal` *before time $t = T_{\max}$, and remains safe (i.e. does not enter* `Unsafe`*) until it enters* `Goal`.

Therefore, a valid flat trajectory for an initial state $x_0 \in$ `Init` is mapped by (39) to a valid trajectory of Σ_{io}, whose states originate at x_0. Further, the control input $u(t)$ that achieves this valid trajectory can be found by using (40). These facts are crucial in our controller synthesis technique, since they essentially allow us to design a feedback control law that establishes output trajectory robustness for Σ_{flat} and then translate the result to Σ_{io} via (40). Since Σ_{flat} is a controllable linear system, establishing output trajectory robustness is straightforward, and for that we can use linear feedback gain according to, for example, the results reported in [13]. The procedure is further explained as follows.

Suppose that for an initial state $x_0 \in$ `Init` we obtain a valid nominal trajectory for Σ_{io}, which we denote by $\widetilde{y}(t)$. This can be obtained, for example, from a human playing a computer game that simulates Σ_{io}. We will demonstrate that by having knowledge of $\widetilde{y}(t)$ we can find the appropriate control input $u(t; x_0')$ that results in a valid trajectory for any initial state x_0' in the neighborhood of x_0.

First of all, we notice that $\widetilde{y}(t)$ is also an output trajectory of Σ_{flat}, i.e. the one corresponding to the input signal

$$\widetilde{\omega}(t) = \frac{d^{\ell+1}}{dt^{\ell+1}}\widetilde{y}(t), \tag{44}$$

and initial states q_0, which are given by

$$\eta_0 = \widetilde{y}(0); \quad \dot{\eta}_0 = \dot{\widetilde{y}}(0); \quad \cdots \quad ; \quad \eta_0^{(\ell)} = \widetilde{y}^{(\ell)}(0). \tag{45}$$

Further, observing that $(\widehat{A}, \widehat{B})$ is in controller canonical form, we infer the existence of:
(i) a feedback gain $\widehat{K} \in \mathbb{R}^{m \times m\ell}$ such that $(\widehat{A} + \widehat{B}\widehat{K})$ is Hurwitz, and
(ii) a symmetric positive definite matrix $P \in \mathbb{R}^{m\ell \times m\ell}$ that satisfies the Lyapunov equation

$$(\widehat{A} + \widehat{B}\widehat{K})^T P + P(\widehat{A} + \widehat{B}\widehat{K}) \preceq 0. \tag{46}$$

We then form a linear feedback loop around Σ_{flat} by defining

$$\omega = \widehat{K} q + \nu, \tag{47}$$

where

$$\nu = \widetilde{\omega} - \widehat{K}\,\widetilde{q}, \tag{48}$$

which is analogous to (24)–(25) from Section IV-B1. The closed-loop system is then given by

$$\Sigma_{\text{cl}} : (\widehat{A} + \widehat{B}\widehat{K})q + \widehat{B}\nu. \tag{49}$$

Notation 6. *We denote the state trajectory of Σ_{cl} starting from the initial state q_0 by $q(t; q_0)$. The corresponding output trajectory is denoted by $\widehat{y}(t; q_0)$.*

Proposition 8. *Define the quadratic function $\psi : \mathbb{R}^{m\ell} \times \mathbb{R}^{m\ell} \to \mathbb{R}_+$ as*

$$\psi(q_1, q_2) \triangleq [(q_1 - q_2)^T P (q_1 - q_2)]^{\frac{1}{2}}.$$

Then, for any $(q_1, q_2) \in \mathbb{R}^{m\ell} \times \mathbb{R}^{m\ell}$, $\psi(q(t; q_1), q(t; q_2))$ is monotonically nonincreasing with time.

Proof. This is straightforward from the fact that P defines a quadratic Lyapunov function for the closed-loop system Σ_{cl}. Equivalently, ψ defines a control autobisimulation function for the linear system Σ_{flat} (see Section IV-B1). □

Proposition 8 establishes trajectory robustness for the state trajectories of Σ_{cl}. Its consequence on the output trajectories is given as follows.

Proposition 9. *Let $\psi(q_1, q_2)$ be as defined in Proposition 8 and let the matrix P be partitioned as*

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{12}^T & P_{22} \end{bmatrix}$$

where

$$P_{11} \in \mathbb{R}^{m\times m}, \quad P_{12} \in \mathbb{R}^{m\times(m-1)\ell}, \quad P_{22} \in \mathbb{R}^{(m-1)\ell\times(m-1)\ell}.$$

Let $S = P_{11} - P_{12}P_{22}^{-1}P_{12}^T$ denote the Schur complement of P_{22} in P. For any $(q_1, q_2) \in \mathbb{R}^{m\ell}\times\mathbb{R}^{m\ell}$,

$$(\Delta\widehat{y})^T S\,(\Delta\widehat{y}) \leq \psi^2(q_1, q_2), \tag{50}$$

where

$$\Delta\widehat{y} \triangleq \widehat{y}(t; q_1) - \widehat{y}(t; q_2),$$
$$\Delta q \triangleq q(t; q_1) - q(t; q_2).$$

Proof. Let $z \in \mathbb{R}^{(m-1)\ell}$. A result given in an appendix of [39] concerning linear algebra states that

$$(\Delta\widehat{y})^T S\,(\Delta\widehat{y}) = \inf_z \begin{bmatrix} (\Delta\widehat{y})^T & z^T \end{bmatrix} P \begin{bmatrix} \Delta\widehat{y} \\ z \end{bmatrix}. \tag{51}$$

If we choose $z = \hat{z}$ such that $[\Delta\hat{y}^T, \hat{z}^T]^T = \Delta q$, then from (51) we have

$$(\Delta\widehat{y})^T S\,(\Delta\widehat{y}) \leq (\Delta q)^T P\,(\Delta q)\,. \tag{52}$$

Substituting the right hand side of (52) using the definition of $\psi(q_1, q_2)$ yields (50), as desired. □

Intuitively, S defines the ellipsoid that is the projection from the state space of Σ_{flat} to the space of flat outputs of the robustness ball defined by P. Observing that S is a symmetric positive definite matrix, we can define a norm in $\mathbb{R}^m$ as follows.

$$\|y\|_\eta \triangleq \sqrt{y^T S y}. \tag{53}$$

In the following, we establish the trajectory robustness for Σ_{io}. Recall that $\widetilde{y}(t)$ is a valid nominal output trajectory corresponding to a state trajectory of Σ_{io} with initial state $x_0 \in \texttt{Init}$.

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

Theorem 10. *Denote the output trajectory of Σ_{io} starting from an initial state $x_0 \in \mathbb{R}^n$ under an input signal $u(t)$ by $y(t; x_0, u)$. Suppose that there exists a $\delta_1 > 0$ such that any output trajectory $y(t)$ satisfying*

$$\sup_t \|y(t) - \widetilde{y}(t)\|_\eta < \delta_1 \tag{54}$$

is also a valid output trajectory. Also, suppose that there exists a $\delta_2 > 0$ such that the following two conditions are satisfied.

(C1) *$\Xi(\cdot)$ is continuously differentiable in $B_\psi(q_0, \delta_2)$, i.e. the $\|\cdot\|_\psi$ ball of radius δ_2 around q_0 as defined in (45).*

(C2) *The Jacobian $\frac{\partial \Xi}{\partial q}$ has full row rank (equals n) in $B_\psi(q_0, \delta_2)$.*

Then, there exists a neighborhood around x_0, $\mathcal{N}(x_0)$, such that for any $\xi \in \mathcal{N}(x_0)$, the following are true:

(i) There exists a $q_\xi \in \mathbb{R}^{m\ell}$ such that $\Xi(q_\xi) = \xi$.

(ii) $y(t; \xi, u_\xi)$ is a valid output trajectory, with the input signal u_ξ defined by

$$u_\xi = \Upsilon(q(t; q_\xi), \widetilde{\omega}(t)), \tag{55}$$

where $\widetilde{\omega}(t)$ is given by (44).

Proof. Define $\delta \triangleq \min(\delta_1, \delta_2)$. Then:

(i) According to the Implicit Function Theorem, conditions C1 and C2 imply the existence of a neighborhood around x_0, $\mathcal{N}(x_0)$, such that for any $\xi \in \mathcal{N}(x_0)$, there is a $q_\xi \in B_\psi(q_0, \delta)$ satisfying $\Xi(q_\xi) = \xi$.

(ii) By definition of differential flatness, applying the input signal $\widetilde{u}_\xi$ to Σ_{io} with initial state $\xi \in \mathcal{N}(x_0)$ yields the output trajectory $y(t; q_\xi)$. Further, from Proposition 9 we can see that

$$\|y(t; q_\xi) - \widetilde{y}(t)\|_\eta = \|y(t; q_\xi) - y(t; q_0)\|_\eta \leq \psi(q_\xi, q_0) \leq \delta \leq \delta_1.$$

Therefore $y(t; q_\xi)$ is a valid output trajectory. □

C. Calculating the Radius of Robustness

In this section we examine how to calculate the various δ's defined in Section IV-A using the CAFs generated by the methods described in Section IV-B.

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

These methods rely on transforming the states in order to find a quadratic CAF $\psi_\ell(x,x') = \sqrt{(x-x')P(x-x')}$ with a linear feedback law. These CAFs generate level sets B_{ψ_ℓ} that are ellipsoids. If the states are transformed using $z = P^{1/2}x$, then $\bar{\psi}_\ell$, the CAF given in terms of the z-coordinates, is

$$\bar{\psi}_\ell(z,z') = \sqrt{(z-z')^T(z-z')}, \tag{56}$$

which shows that $\bar{\psi}_\ell$ is just the Euclidean distance in z-space. Thus, to find $\delta^{(i)}_{\texttt{Unsafe}}$ one needs only to find the minimum distance between the i^{th} trajectory partition and `Unsafe`. Similarly, to find $\delta^{(N)}_{\texttt{Goal}}$ for a trajectory with N events, one needs to find the minimum distance between the terminal z-coordinate of the trajectory and `Goal`. To find $\delta^{(i)}_{\texttt{Goal}}$ for $i < N$, one needs to find the minimum distance between the z-coordinate of the trajectory at time $t^{(i+1)}$ and $\texttt{Reset}^{(i+1)^\dagger}\left(B_{\psi_{\ell^{(i+1)}}}(\xi_{a,k}(t^{(i+1)},\ell_0,x_0),\delta^{(i+1)}).\right)$

Handling $\delta^{(i)}_{\texttt{Input}}$ is a little trickier, as the bound will depend on the underlying dynamics. In the following analysis we will use input bounds of the form $\|u(t)\| \le M$.

Consider the linear affine case presented in Section IV-B1. If we combine (24) with (25), we find that

$$u(t) = K(x - \xi_{a,k}(t^{(i)},\ell_0,x_0)) + u_0(t). \tag{57}$$

Keeping in mind Notation 2, we see from (5) and (7) that $\delta^{(i)}_{\texttt{Input}}$ is defined such that

$$\max_{\substack{t\in\left[t^{(i)},t^{(i+1)}\right)\\ x\in B_{\psi_{\ell^{(i)}}}(\xi_{a,k}(t,\ell_0,x_0),\delta^{(i)}_{\texttt{input}(\ell)})}} \|K(x-\xi_{a,k}(t^{(i)},\ell_0,x_0)) + u_0(t)\| \le M.$$

Further, we note that equality is obtained as otherwise there would exist a $\delta > \delta^{(i)}_{\texttt{Input}}$ satisfying (5) and (7), a contradiction. Applying the triangle inequality and performing the transformation of variables $z = P^{1/2}(x - \xi_{a,k}(t^{(i)},\ell_0,x_0))/\delta^{(i)}_{\texttt{Input}}$ we see that

$$\max_{\|z\|\le 1} \delta^{(i)}_{\texttt{Input}}\|KP^{-\frac{1}{2}}z\| + \|u_0(t)\| = M \quad \Rightarrow \quad \delta^{(i)}_{\texttt{Input}} = \frac{M - \|u_0(t)\|}{\|KP^{-\frac{1}{2}}\|}, \tag{58}$$

where we have applied the definition of a matrix norm.

Consider the feedback linearizable case presented in Section IV-B2. Using the same method as was used to derive (57), we see that

$$w(t) = K(x - \xi_{a,k}(t^{(i)},\ell_0,x_0)) + w_0(t), \tag{59}$$

where $w_0(t)$ is the nominal linearized input. This implies

$$\|u(t)\| = \|\kappa(x) + \lambda(x)(K(x - \xi_{a,k}(t^{(i)}, \ell_0, x_0)) + w_0(t)\| \leq M. \tag{60}$$

We again apply a transformation of variables to $z = P^{1/2}(x - \xi_{a,k}(t^{(i)}, \ell_0, x_0)$ so that $\|z\| = \psi_{\ell^{(i)}}(x, \xi_{a,k}(t^{(i)}, \ell_0, x_0))$, yielding

$$\|\kappa(\theta) + \lambda(\theta)(K(P^{-1/2}z) + w_0(t))\| \leq M, \tag{61}$$

where we define $\theta = P^{-1/2}z + \xi_{a,k}(t^{(i)}, \ell_0, x_0)$. If we can manipulate (61) into the form $\|z\| \leq \delta$, then we see that (12) holds for δ, making it a lower bound for $\delta^{(i)}_{\texttt{Input}}$, which we can then use as a conservative estimate for the controller synthesis procedure.

Consider the differentially flat system given in Section IV-B3. $\delta^{(i)}_{\texttt{Input}}$ is determined by first finding the largest value δ such that

$$\left\| \Upsilon((q - \widetilde{q}) + \widetilde{q},\ \widehat{K}(q - \widetilde{q}) + \widetilde{\omega}) \right\| \leq M, \quad \forall \|q - \widetilde{q}\| \leq \delta, \tag{62}$$

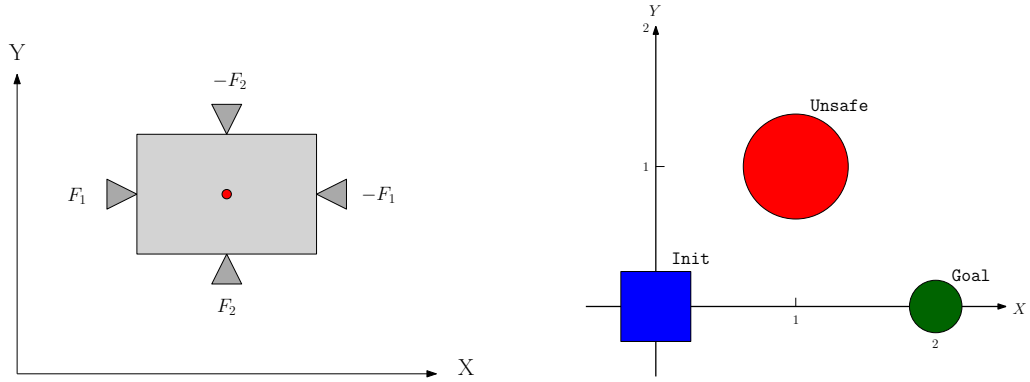
and then setting

$$\delta^{(i)}_{\texttt{Input}} = \delta \sqrt{\lambda_{\min}(P)}. \tag{63}$$

Note that finding the value δ in (62) is nontrivial, and depends heavily on the form of $\Upsilon(\cdot, \cdot)$.

Remark 11. *If the nominal input generated by the human approaches the boundary of* $\texttt{Input}(\ell)$ *during the segment* $t \in [t^{(i)}, t^{(i+1)})$*, then* $\delta^{(i)}_{\texttt{Input}}$ *in most cases will go to zero. Rather than making the human responsible for ensuring that their generated inputs are bounded away from* $\texttt{Input}(\ell)$*'s boundary, the control system designer may wish to impose on the simulation a conservative set of input bounds. This imposes a trade-off wherein a slightly conservative set of bounds may lead to very small* $\delta^{(i)}_{\texttt{Input}}$*'s, but a very conservative set of bounds may make it exceedingly difficult for the human to find a valid trajectory. A simple method for handling this trade-off is to try to find a set of bounds where* $\delta^{(i)}_{\texttt{Input}}$ *is the same size as* $\min(\delta^{(i)}_{\texttt{Unsafe}}, \delta^{(i)}_{\texttt{Goal}})$*, balancing the trade-off. For example, if the input bounds are given by* $\|u(t)\| \leq M$*, then search over* Δ *(say, by a bisection search) where the input bound* $\|u(t)\| \leq M - \Delta$ *is imposed on the human.*

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST



(a) Vehicle to be controlled in the linear affine case.

(b) Safety problem specification.

Fig. 1. Linear affine example.

V. Examples

In order to demonstrate the utility of the methods described in Sections IV-B, three examples were devised to illustrate the applicability of these techniques to nonlinear systems. We start with a relatively simple linear affine system (Sec. V-A). We next consider an input-output linearizable system which is a more general class of nonlinear system in that every linear affine system is trivially input-output linearizable (Sec. V-B). Finally, we consider a differentially flat system, which again is a more general class of nonlinear systems in that all feedback linearizable systems are differentially flat, if they are controllable, observable and have no zero dynamics [40].

A. Control of a Linear Affine Planar Vehicle

Consider the task of controlling the vehicle presented in Figure 1a. Suppose that the vehicle starts in some compact set around the origin and we wish to drive it to some region of the point $(2, 0)$. Further, the environment is such that there is an object at $(1, 1)$ the vehicle must avoid that provides a constant attractive force. Finally, the system has an anti-friction term which causes the autonomous system to be unstable. The orientation of these regions are shown in Figure 1b. This system can be modeled with the following dynamics,

$$\Sigma : \frac{dx}{dt} = Ax + Bu + c, \tag{64}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0.1 & 0 \\ 0 & -1 & 0 & 0.1 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad c = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{65}$$

For our example, we take $\mathcal{X} = \mathbb{R}^4$ and define `Init`, `Unsafe` and `Goal` to be,

$$\texttt{Init} = \{x \mid |x_1| \leq 0.2, |x_2| \leq 0.2, x_3 = 0, x_4 = 1\}, \tag{66}$$

$$\texttt{Unsafe} = \left\{x \;\middle|\; \sqrt{(x_1-1)^2 + (x_2-1)^2} \leq 0.3\right\}, \tag{67}$$

$$\texttt{Goal} = \left\{x \;\middle|\; \sqrt{(x_1-2)^2 + x_2^2} \leq 0.1\right\}. \tag{68}$$

Further, we wish to impose the constraint that the vehicle enter the goal set within $t \in [0, 5]$ seconds and with the input constraints $\|u(t)\| \leq 1$.

We use the convex optimization solver `cvx`[41] to solve (27) for a feedback gain K which will guarantee the existence of a quadratic CAF. The gain matrix (up to three decimal places) is calculated to be

$$K = \begin{bmatrix} 0.000 & 0.000 & -0.100 & 0.000 \\ 0.000 & 0.000 & 0.000 & -0.100 \end{bmatrix}. \tag{69}$$

With this feedback gain we see that

$$A + BK = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}. \tag{70}$$

The control autobisimulation function for this system is given by any function

$$\phi(x, x') = [(x - x')^T P (x - x')]^{\frac{1}{2}}$$

such that P satisfies (26). We can see that if P is the identity matrix, then the quadratic form in (26) will be zero for all $(x - x')$, and the condition will be met. Note that for this P our CAF becomes the Euclidean norm, which is especially convenient and intuitive. Now that we have designed a feedback control law that guarantees trajectory robustness, we need to design the open loop trajectories that will cover `Init`. To this end we developed a video game in MATLAB that simulates the closed loop system,

$$\Sigma_{\text{cl}} : \frac{dx}{dt} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}. \tag{71}$$

We discretize these dynamics with a constant time step δ_s and assume the open loop input is constant between time intervals, that is, the input to the system is a zero order hold of a discrete control sequence. For the affine dynamics that we consider in this example, the discretization can be made exact by integrating the closed loop dynamics (71) analytically. We obtain

$$x(t + \delta_s) = \exp(A_{\text{cl}}\delta_s) + A_{\text{cl}}^{-1}(\exp(A_{\text{cl}}\delta_s - I)(Bv + c),$$

$$A_{\text{cl}} := A + BK.$$

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

In order to make sure our input bound is met, we need to choose a desired $\delta_{\texttt{Input}}$, an upper bound on the radius of robustness. Note that for this system, the radius of `Goal` is 0.1, which means that the largest $\delta_{\texttt{Goal}}$ that we can generate is 0.1. For any value of $\delta_{\texttt{Input}}$ larger than this, we would be limiting the input without any gain in the robustness radius, so let's choose $\delta_{\texttt{Input}} = 0.1$. We can see from (58) that our open loop control must be bounded by

$$\|v\| \leq M - \delta_{\texttt{Input}} \left\| KP^{-1/2} \right\| = 2 - 0.1(0.1) = 1.99. \tag{72}$$

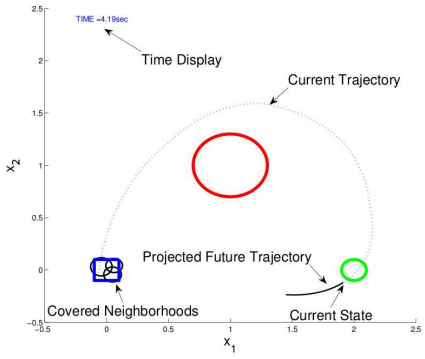
This is a minor change in the upper bound and it guarantees robustness, so we have little motivation to choose a smaller $\delta_{\texttt{Input}}$ to allow for large inputs. Note that in general there will be a trade-off between these two values.

The input to the video game is generated by a human using a joystick. One of the benefits of using a simulation to generate the controller is that the system can be simulated at a slower rate than the actual dynamics, allowing a human to generate feasible trajectories for a system that would otherwise exceed the bandwidth of a human's sensory capabilities. In this case we simulate the system at one-third of the speed of the system's dynamics.

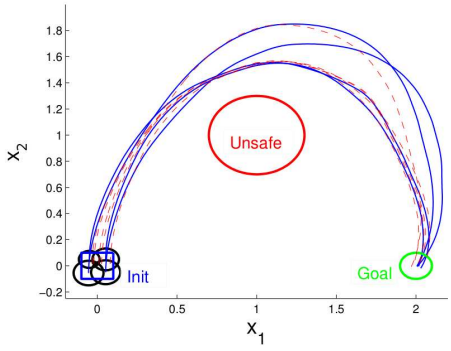
Figure 2a shows the graphical user interfaced for the MATLAB video game. The user chooses an initial condition in `Init` from which to start the simulation. The position coordinates (x_1 and x_2) are displayed on the screen. When the simulation begins, a point begins to follow a trajectory determined by the dynamics and the user input. There are many possible ways that the video game designer can aid the player in generating feasible trajectories. In our program, we include a timer in the upper left to help players determine the efficacy of their current input strategy. At each point in the simulation we determine the trajectory that will be generated if the input is held constant up to some finite horizon, and plot this graphically for the user to use to adjust their input. The results from one session with the video game are given in Figure 2b. The solid lines show the trajectories generated by the human player. The union of the robustness bounds around the initial conditions cover `Init`, and thus the controller that uses the feedback law in (69) and the open loop input given by one of those four nominal trajectories (to be determined by the initial condition) solve the safety controller synthesis problem.

B. Control of a Hybrid Input-Output Linearizable Rotating Planar Vehicle

Now consider a vehicle that is similar to that presented in Section IV-B1, but with one difference. Assume that the axis of the thrusters on the vehicle are not in line with the center

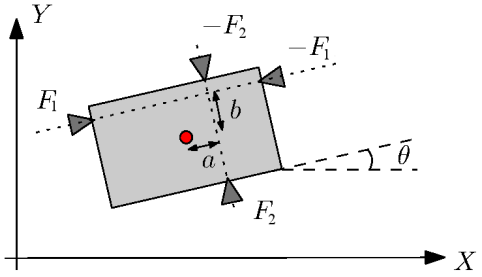


(a) Interface for the MATLAB simulation.

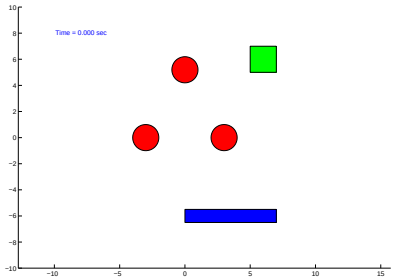


(b) Four trajectories (solid) that form the controller along with several trajectories (dashed) generated by the controller.

Fig. 2. Screenshots for linear affine example.



(a) The rigid object with two off-center jet thrusters.



(b) Safety problem for hybrid system.

Fig. 3. Hybrid linear affine example.

of mass, but rather they are offset by some fixed amount. See Figure 3a for an illustration. Therefore, the control inputs will induce both translational and rotational motions on the object.

The safety problem for this example consists of three unsafe regions; please refer to Figure 3b. Each region exerts a force along each coordinate that is linear with respect to the distance from that region along the corresponding coordinate. The user has control over whether this is a repulsive (positive) force or an attractive (negative) force. Thus, this is a hybrid system where the discrete input determines the signs on these terms in the dynamics. Since there are three regions with two possible states each, there are $2^3 = 8$ locations. The dynamics for each location

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

are given by

$$\Sigma(\ell) = \begin{cases} \ddot{x} = F_1 \cos\theta - F_2 \sin\theta + 0.1\dot{x} + \sum_{k=1}^{3} s_k(\ell)(x - x_k), \\ \ddot{y} = F_1 \sin\theta + F_2 \cos\theta + 0.1\dot{y} + \sum_{k=1}^{3} s_k(\ell)(y - y_k), \\ \ddot{\theta} = F_2 \frac{a}{I} - F_1 \frac{b}{I}. \end{cases} \tag{73}$$

where I denotes the object's moment of inertia, x_k and y_k are the x and y coordinates of the center of the k^{th} unsafe regions and $s_k(\ell) = \pm 1$ determines the sign of the force induced by the k^{th} unsafe region in location ℓ.

We define the (x, y) coordinate of the object's center of mass as the system's output. In state space form, the dynamics for location ℓ are given by

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \underbrace{\begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ 0.1x_4 + \sum_{k=1}^{3} s_k(\ell)(x_1 - x_k) \\ 0.1x_5 + \sum_{k=1}^{3} s_k(\ell)(x_2 - y_k) \\ 0 \end{bmatrix}}_{\triangleq f(x)} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \cos x_5 & -\sin x_5 \\ \sin x_5 & \cos x_5 \\ -\frac{b}{I} & \frac{a}{I} \end{bmatrix}}_{\triangleq g(x)} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \tag{74}$$

$$y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \triangleq h(x). \tag{75}$$

Observing that

$$\begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} 0.1x_4 + \sum_{k=1}^{3} s_k(\ell)(x_1 - x_k) \\ 0.1x_5 + \sum_{k=1}^{3} s_k(\ell)(x_2 - y_k) \end{bmatrix} + \begin{bmatrix} \cos x_5 & -\sin x_5 \\ \sin x_5 & \cos x_5 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \tag{76}$$

we conclude that the vector relative degree of the system given by (74) – (75) is $\{2, 2\}$.

The safety control that we want to solve is identical to Section IV-B1, except that the sets are defined to include all angles and angular velocities in our new higher-dimensional state space. It can be formulated as follows. Suppose that we are to steer the system from the initial set in $\mathcal{X} = \mathbb{R}^6$:

$$\texttt{Init} =$$

$$\{x \mid |x_1 - 3.5| \leq 3.5, |x_2 + 6| \leq 0.5, x_3 = x_4 = x_5 = x_6 = 0\}.$$

The sets of unsafe and goal outputs are given by

$$\begin{aligned} \texttt{Unsafe} = \{y \in \mathbb{R}^2 \mid \left\|y - [-3, 0]^T\right\|_2 \leq 1\} \\ \cup \{y \in \mathbb{R}^2 \mid \left\|y - [3, 0]^T\right\|_2 \leq 1\} \\ \cup \{y \in \mathbb{R}^2 \mid \left\|y - [0, \sqrt{3}]^T\right\|_2 \leq 1\}, \\ \texttt{Goal} = \{y \in \mathbb{R}^2 \mid \left\|y - [6, 6]^T\right\|_\infty \leq 1\}. \end{aligned}$$

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

Additionally, we also require that the control inputs are bounded in magnitude, i.e.

$$\left\| \begin{bmatrix} F_1(t) \\ F_2(t) \end{bmatrix} \right\| \leq 2. \tag{77}$$

To solve this problem, we can define a linearizing input transformation as follows:

$$\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} \triangleq \begin{bmatrix} \cos x_5 & -\sin x_5 \\ \sin x_5 & \cos x_5 \end{bmatrix} \begin{bmatrix} F_1(t) \\ F_2(t) \end{bmatrix}. \tag{78}$$

This yields a linear affine system for each location,

$$\frac{d}{dt} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix} = \begin{bmatrix} \hat{x}_3 \\ \hat{x}_4 \\ 0.1\hat{x}_3 + \sum_{k=1}^{3} s_k(\ell)(\hat{x}_1 - x_k) \\ 0.1\hat{x}_4 + \sum_{k=1}^{3} s_k(\ell)(\hat{x}_2 - y_k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w_1 \\ w_2 \end{bmatrix}, \tag{79}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}. \tag{80}$$

We can now find the a feedback law for each of the eight locations using the techniques presented in Section IV-B2 to guarantee trajectory robustness. As in the last section, we are able to find gains that provide trajectory robustness for a quadratic CAF where $P = I$.

Observe that the linear system has a lower order than the original nonlinear systems. This is because the rotational dynamics has now become unobservable. We see that for this linearizing input $\kappa(x) = 0$ and $\lambda(x)$ is the rotation matrix. Since $\|\lambda(x)\| = 1$ for all x we see that

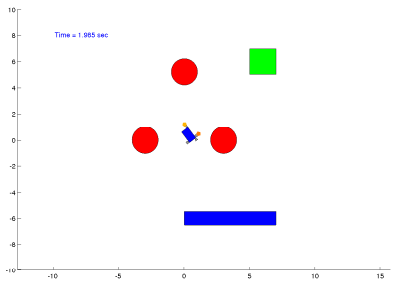
$$\|\widetilde{w}(t)\| \leq M - \max_{x \in B_\phi(\widetilde{x}, \delta_{\texttt{Input}})} \|K(x - \widetilde{x})\|$$

Since we can use the same quadratic control autobisimulation function as the previous section, we see that this bound becomes

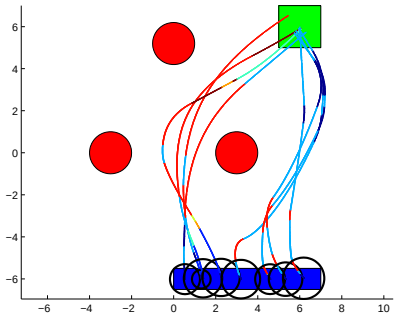
$$\|\widetilde{w}(t)\| \leq M - \delta_{\texttt{Input}} \|KP^{-1/2}\|$$

Again, choosing $\delta_{\texttt{Input}} = 0.1$ will yield a bound on the nominal linearized input of $\|\widetilde{w}(t)\| \leq 1.99$.

We augment the video game used previously to also simulate the rotational dynamics. The graphic user interface was augmented to show an image of the vehicle traversing the screen to give the user information on the orientation of the vehicle. The thrusters on the vehicle also light up to a color that is proportional to the magnitude of the input. Valid trajectories can then be obtained from a human player. The game interface is shown in Figure 4a. In Figure 4b, we can also see that a single valid trajectory can be generalized to cover a robust neighborhood around its initial condition.

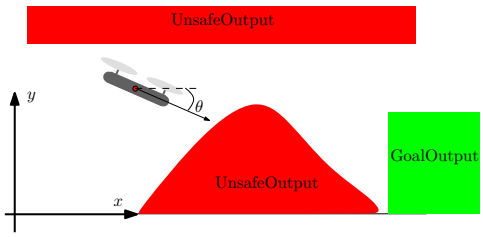


(a) Trajectory simulation

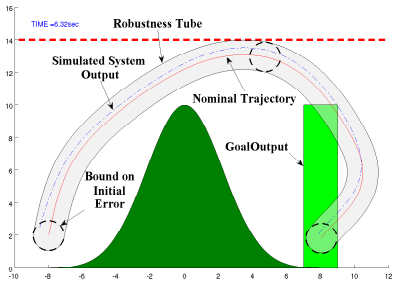


(b) Covering set of valid post-simulation trajectories. The colors along the trajectory represent the discrete state at that time.

Fig. 4. Annotated screen capture of the video game.



(a) 2D control of a quadrotor.



(b) Simulation of quadrotor system.

Fig. 5. Setup and results for quadrotor system.

C. 2D Control of a Differentially Flat Quadrotor

We consider the control problem related to the motion of a quadrotor on a vertical plane, as shown in Figure 5a. This example is inspired by a similar one used in [42]. This two-dimensional quadrotor is idealized as having two propellers, one on the front of the body, the other on the rear. These propellers are able to induce a positive force along the propeller axis, which we shall choose as our first controller input u_1, and a rotational motion in the vertical plane, which we shall choose as our second controller input u_2. In this system gravity is acting along the negative y-axis, and a force is induced by a constant wind $w = [w_1, w_2]^T$ with friction coefficient μ.

Preprint submitted to IEEE Transactions on Automatic Control. Received: September 3, 2014 06:38:25 PST

The system's dynamics are given by Equations (81) – (83).

$$\ddot{x}_1 = \mu(w_1 - \dot{x}_1) - \frac{u_1}{m}\sin\theta \tag{81}$$

$$\ddot{y} = \mu(w_2 - \dot{y}) - g + \frac{u_1}{m}\cos\theta \tag{82}$$

$$\ddot{\theta} = u_2 \tag{83}$$

Here m denotes the object's mass. We define the (x_1, x_2) coordinate of the object's center of mass as the system's output. This system is differentially flat with a set of flat outputs given by

$$[\eta_1, \eta_2]^T = [x, y]^T. \tag{84}$$

The equations that satisfy Equations (39) – (40) are given below. For brevity they are given with respect to $A_1 = \ddot{\eta}_1 - \mu(w_x - \dot{\eta}_1)$, $A_2 = \ddot{\eta}_2 - \mu(w_y - \dot{\eta}_2) + g$, and their time derivatives.

$$x = \eta_1, \qquad \dot{x} = \dot{\eta}_1, \qquad y = \eta_2, \qquad \dot{y} = \dot{\eta}_2, \qquad \theta = -\arctan\left(\frac{A_1}{A_2}\right), \qquad \dot{\theta} = \frac{A_1\dot{A}_2 - A_2\dot{A}_1}{A_1^2 + A_2^2}$$

$$u_1 = m\sqrt{A_1^2 + A_2^2}, \qquad u_2 = \frac{(A_1^2 + A_2^2)(A_1\ddot{A}_2 - A_2\ddot{A}_1)}{(A_1^2 + A_2^2)^2} - \frac{(A_1\dot{A}_2 - A_2\dot{A}_1)(2A_1\dot{A}_1 + 2A_2\dot{A}_2)}{(A_1^2 + A_2^2)^2}$$

Thus, given a trajectory of class C^4 in the flat output space, a unique set of inputs can be determined to generate the trajectory for a system with the same set of initial conditions.

We now need to verify the two conditions of Theorem 10. For $x, y, \dot{x}, \dot{y}$, Ξ is clearly continuously differentiable. The functions for θ and $\dot{\theta}$ are continuously differentiable everywhere except for $A_1 = A_2 = 0$. The Jacobian is lower block diagonal and given by

$$\frac{\partial \Xi}{\partial q} = \begin{bmatrix} I_{4\times 4} & \mathbf{0}_{4,1} & \mathbf{0}_{4,1} & \mathbf{0}_{4,1} & \mathbf{0}_{4,1} \\ \cdot & \frac{-A_2}{A_1^2+A_2^2} & \frac{A_1}{A_1^2+A_2^2} & 0 & 0, \\ \cdot & \cdot & \cdot & \frac{-A_2}{A_1^2+A_2^2} & \frac{A_1}{A_1^2+A_2^2} \end{bmatrix}$$

This will lose full row rank only when $A_1 = A_2 = 0$. From the dynamics, we see that $A_1 = u_1 \sin\theta/m$ and $A_2 = u_2\cos\theta/m$. Both values can be zero only if $u_1 = 0$. Quadrotor helicopters always operate at some minimum idling speed for their propellers, and thus u_1 can never have a thrust of 0 while operating. Therefore the conditions are satisfied.

As seen in Figure 5b, the trajectory in our controller is generated by a human via the game interface. To ensure the property of trajectory robustness in Σ_{flat}, a feedback gain $\widehat{K}$ was chosen to be

$$\widehat{K} = \left[\begin{smallmatrix} 625 & 0 & 500 & 0 & 150 & 0 & 20 & 0 \\ 0 & 625 & 0 & 500 & 0 & 150 & 0 & 20 \end{smallmatrix}\right] \tag{85}$$

We compute the P that solves Equation (46) using the convex optimization solver CVX [41].

$$P = \begin{bmatrix} 0.05 & 0.00 & -0.06 & 0.00 & 0.03 & 0.00 & 0.02 & 0.00 \\ 0.00 & 0.05 & 0.00 & -0.06 & 0.00 & 0.03 & 0.00 & 0.02 \\ -0.06 & 0.00 & 0.10 & 0.00 & -0.07 & 0.00 & -0.06 & 0.00 \\ 0.00 & -0.06 & 0.00 & 0.10 & 0.00 & -0.07 & 0.00 & -0.06 \\ 0.03 & 0.00 & -0.07 & 0.00 & 0.11 & 0.00 & -0.01 & 0.00 \\ 0.00 & 0.03 & 0.00 & -0.07 & 0.00 & 0.11 & 0.00 & -0.01 \\ 0.02 & 0.00 & -0.06 & 0.00 & -0.01 & 0.00 & 2.00 & 0.00 \\ 0.00 & 0.02 & 0.00 & -0.06 & 0.00 & -0.01 & 0.00 & 2.00 \end{bmatrix} \tag{86}$$

As in the feedback linearization case, a MATLAB graphical user interface was created to obtain the nominal trajectory from a human. Figure 5b shows the simulated result of this controller. The size of the robustness tube was chosen such that it does not intersect with `UnsafeOutput`, and ends entirely within `GoalOutput`.

VI. Conclusion

In this paper we addressed the tasks of safety controller synthesis. We described the concepts of autobisimulation and trajectory robustness and introduced a general framework for synthesizing controllers using the concepts. We developed specific techniques for applying this framework to hybrid systems whose continuous dynamics are linear affine, feedback linearizable, and differentially flat. In addition, we discussed how to synthesize controllers that respect a given set of input bounds. For each class of dynamics, an example system was considered, and a feedback law that guaranteed trajectory robustness was found. For each example a MATLAB video game was constructed that would synthesize a controller by having the user provide nominal trajectories using a joystick.

The resulting controller from the presented method is guaranteed to be safe with respect to the given safety problem. The nominal trajectories may be generated by any means. If one were to have humans generate these trajectories, then one can crowdsource the task, which could be especially useful in high-dimensional or otherwise complex scenarios where classical techniques may fail to yield a valid controller. Techniques to locally improve the nominal trajectories with respect to some cost function were not considered here, but are being examined by the authors and will appear in a forthcoming publication.

References

[1] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1(4), pp. 199–220, 2000.

[2] T. Dang, A. Donze, and O. Maler, "Verification of analog and mixed-signal circuits using hybrid system techniques," in *Formal Methods in Computer-Aided Design*, ser. LNCS, vol. 3312. Springer, 2004, pp. 21–36.

[3] G. Batt, B. Yordanov, R. Weiss, and C. Belta, "Robustness analysis and tuning of synthetic gene networks," *Bioinformatics*, vol. 23, no. 18, pp. 2415–2422, 2007.

[4] D. Riley, X. Koutsoukos, and K. Riley, "Modeling and analysis of the sugar cataract development process using stochastic hybrid systems," *IET Systems Biology*, vol. 3(3), pp. 137–154, 2009.

[5] 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. Automatic Control*, vol. 50, pp. 947 – 957, 2005.

[6] A. B. Kurzhanski, I. M. Mitchell, and P. Varaiya, "Optimization techniques for state-constrained control and obstacle problems," *Journal of Optimization Theory and Applications*, vol. 128, no. 3, pp. 499–521, 2006.

[7] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg, "On systematic simulation of open continuous systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2623. Springer, 2003, pp. 283–297.

[8] C. Belta and L. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Trans. Automatic Control*, vol. 51, no. 11, pp. 1749–1759, 2006.

[9] L. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Trans. Automatic Control*, vol. 51, no. 6, pp. 938–948, 2006.

[10] G. Reissig, "Computation of discrete abstractions of arbitrary memory span for nonlinear sampled systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 5469. Springer, 2009, pp. 306–320.

[11] F. Clarke and P. R. Wolenski, "Control of systems to sets and their interiors," *Journal of Optimization Theory and Applications*, vol. 88, pp. 3–23, 1994.

[12] A. A. Julius, "Trajectory-based controller design for hybrid systems with affine continuous dynamics," in *Proc. IEEE Conf. Automation Science and Engineering*, Toronto, Canada, 2010, pp. 1007–1012.

[13] A. A. Julius and S. Afshari, "Using computer games for hybrid systems controller synthesis," in *Proc. 49th IEEE Conf. Decision and Control*, Atlanta, Georgia, 2010, pp. 5887–5892.

[14] P. Tabuada, "An approximate simulation approach to symbolic control," *IEEE Trans. Automatic Control*, vol. 53, no. 6, pp. 1406–1418, 2008.

[15] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[16] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Trans. Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.

[17] A. G. and G. J. Pappas, "Verification using simulation," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 3927. Springer Verlag, 2006, pp. 272–286.

[18] A. A. Julius, G. Fainekos, M. Anand, I. Lee, and G. J. Pappas, "Robust test generation and coverage for hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 4416. Springer Verlag, 2007, pp. 329–342.

[19] F. Lerda, J. Kapinski, E. M. Clarke, and B. H. Krogh, "Verification of supervisory control software using state proximity and merging," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 4981, 2008, pp. 344–357.

[20] M. Zamani and P. Tabuada, "Backstepping design for incremental stability," *IEEE Trans. Automatic Control*, vol. 56, no. 9, 2011.

[21] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *Automatic Control, IEEE Transactions on*, vol. 57, no. 7, pp. 1804–1809, 2012.

[22] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis*, vol. 15, no. 11, pp. 1163–1170, 1983.

[23] J. Barral, R. Wilson, and C. Langbort, "The online ouija board : a testbed for multi-party control of dynamical systems," in *Proc. American Control Conference*, St. Loius, MO., 2009, pp. 872–877.

[24] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "Captcha: Using hard AI problems for security," in *Advances in Cryptology*, ser. LNCS, vol. 2656. Springer, 2003, pp. 294–311.

[25] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*. New York: ACM, 2004, pp. 319–326.

[26] L. von Ahn and L. Dabbish, "General techniques for designing games with a purpose," *Communications of the ACM*, pp. 58–67, August 2008.

[27] A. A. Julius and A. K. Winn, "Safety controller synthesis using human generated trajectories: Nonlinear dynamics with feedback linearization and differential flatness," in *Proc. American Control Conference*, Montreal, Canada., 2012.

[28] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[29] H. Sira-Ramirez and S. Agrawal, *Differentially Flat Systems*. New York, NY: Marcel Dekker Inc., 2004.

[30] J. Levine, *Analysis and Control of Nonlinear Systems: a Flatness based approach*. New York, NY: Springer, 2009.

[31] A. J. van der Schaft and J. M. Schumacher, *An Introduction to Hybrid Dynamical Systems*. London: Springer, 2000.

[32] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," Dept. Computer and Information Science, University of Pennsylvania, Philadelphia, USA, MS-CIS-05-10, May 2005.

[33] E. D. Sontag, "A 'universal' construction of Artstein's theorem on nonlinear stabilization," *Systems and Control Letters*, vol. 13, no. 2, pp. 117–123, 1989.

[34] W. Lohmiller and J. J. E. Slotine, "On contraction analysis for nonlinear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[35] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in Systems and Control Theory*. Philadelphia: SIAM, 1994.

[36] A. Isidori, *Nonlinear Control Systems*, 3rd ed. New York, NY: Springer, 1994.

[37] M. Fliess, J. Levine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: introductory theory and examples," *International Journal of Control*, vol. 61, pp. 1327–1361, 1995.

[38] M. J. van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. Journal of Robust and Nonlinear Control*, vol. 8, no. 11, pp. 995 – 1020, 1998.

[39] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004, available online at www.stanford.edu/ boyd/cvxbook/.

[40] M. van Nieuwstadt, M. Rathinam, and R. Murray, "Differential flatness and absolute equivalence," in *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, vol. 1, dec 1994, pp. 326 –332 vol.1.

[41] S. Boyd and M. C. Grant, "cvx – MATLAB software for disciplined convex programming," 2005, http://www.stanford.edu/∼boyd/cvx/.

[42] A. Donze, B. Krogh, and A. Rajhans, "Parameter synthesis for hybrid systems with an application to Simulink models," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 5469. Springer, 2009, pp. 165–179.