Safety Controller Synthesis using Human Generated Trajectories: Nonlinear Dynamics with Feedback Linearization and Differential Flatness

A. Agung Julius and Andrew K. Winn

Abstract— **The aim of the safety controller synthesis problem is to synthesize a feedback controller that results in closed-loop trajectories that meet certain criteria, namely, the state or output trajectories terminate in a Goal set without entering an Unsafe set. We propose a formal method for synthesizing such a controller using finitely many human generated trajectories. The main theoretical idea behind our results is the concept of trajectory robustness, which is established using the theory of approximate bisimulation. Approximate bisimulation has been used to establish robustness (in the $\mathcal{L}_\infty$ sense) of execution trajectories of dynamical systems and hybrid systems, resulting in trajectory-based safety verification procedures.**

The work reported in this paper builds on our earlier work where the dynamics of the system is assumed to be affine linear. We extend the existing results to special classes of nonlinear dynamical systems, feedback linearizable and differentially flat systems. For both cases, we present some examples where it is possible to synthesize the controller using human generated trajectories, which are obtained through interactive computer programs with graphical interface (computer games).

Keywords: controller synthesis, trajectory based, feedback linearization, differential flatness.

I. Introduction

The problem of safety controller synthesis for hybrid systems has received a lot of attention from the controls community. This problem is concerned with the construction of control laws/algorithms for systems with input that result in safe executions. There have been several approaches developed to address this problem. Most of these approaches are based on the concept of safety/reachability analysis. For example, the optimal control method in [1], [2], [3] and the simulation based method in [4] 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 [5], [6]. The technique for discrete-time systems presented in [7] utilizes partitioning of the state space by polygonal approximation of the reachable set. For continuous dynamical systems, the theoretical results presented in [8] discuss some sufficient conditions for the existence of a controlled system trajectory that enters a prescribed Goal set.

The results that are presented in this paper follows a different approach from the above mentioned, i.e. trajectory-based approach. The key concept here is the assessment of safety/reachability based on the execution trajectories of the system, or the simulations thereof. To generalize the safety property of a simulated execution trajectory to a compact neighborhood around it, we use the concept of trajectory robustness [9], [10] or incremental stability [11], [12]. 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 [13], and has been used for trajectory based analysis of hybrid systems in [14], [15], [16].

The approach in [11], [12] and our approach differ in the way trajectory robustness is used in controller synthesis. In [11], [12], 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 [10]. Also, we do not require the open loop dynamics to have incremental stability property. Instead, a part of the controller synthesis procedure is devoted to establishing this property. In a similar spirit, a more recent work by Zamani and Tabuada [17] also drops the incremental stability requirement and aims to recover it by using back-stepping controller design.

In our previous work [10], we assume that the underlying dynamics of the hybrid system is affine linear. The contribution of the current paper is in the extension of the trajectory-based controller synthesis idea to a class of nonlinear dynamics, *feedback linearizable* and *differentially flat* systems [18], [19].

II. Controller Synthesis Using Human Generated Trajectories

In this section, we review the idea of using human generated trajectories in controller synthesis (see e.g. [10]). Consider a dynamical system with input

$$\Sigma_{\mathrm{inp}} : \frac{dx}{dt} = f(x,u), x \in \mathbb{R}^n, u \in \mathcal{U} \subset \mathbb{R}^m. \qquad (1)$$

where the function $f(x,u)$ is locally Lipschitz in x and continuous in u. Suppose that there is a given compact set of initial states $\mathrm{Init} \subset \mathbb{R}^n$, where the state is initiated at $t = 0$, i.e. $x(0) \in \mathrm{Init}$. Also, we assume that there is a set of goal states, Goal$\subset \mathbb{R}^n$, and a set of unsafe states Unsafe$\subset \mathbb{R}^n$. As usual, a trajectory is deemed unsafe if it enters the unsafe set.

The safety control problem can be formulated as follows [10]:

Problem 1 (State Safety): Design a feedback control law $u = k(x)$ such that for any initial state $x_0 \in \mathrm{Init}$, the trajectory of the closed loop system enters Goal before time $t = T_{\max}$, and remains safe until it enters Goal.

Agung Julius and Andrew Winn are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, Email: agung@ecse.rpi.edu,winna@rpi.edu.

Hereafter, any trajectory that satisfies the conditions above is called a *valid trajectory*. Notice that although the safety control problem as defined above is not formulated for hybrid systems, we have shown that a similar problem for hybrid systems can be reduced to the above format (see [9], [10]).

We establish trajectory robustness through a Lyapunov like function called control autobisimulation function (CAF) [10]. Without repeating the presentation in [10], we can summarize CAF as follows. A CAF is a nonnegative function $\psi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+$ such that there exists a feedback control law

$$u = k(x) \tag{2}$$

that provides trajectory robustness to the closed loop system (1) - (2). That is, if we denote the solution trajectory of (1) - (2) with initial condition $x(0) = x_0$ as $x_k(t; x_0)$, then $\psi\left(x_k(t; x_0), x_k(t; x_0')\right)$ is a nonincreasing function of time, for any pair $x_0, x_0' \in \mathbb{R}^n$. Any feedback law in the form of (2) that satisfies the condition above is called an *admissible feedback law*.

In [10], we have shown that for systems with linear affine dynamics, the feedback law can assume the following form:

$$u(t) = Kx(t) + v(t), \tag{3}$$

where K is a constant gain and $v(t)$ can be any (integrable) trajectory. The design procedure for K follows the classical feedback control design for stabilization of linear systems.

For any given initial condition, we use a computer game that simulates the system's dynamics to obtain $v(t)$ that results in a valid execution trajectory. The trajectory robustness conferred by CAF enables us to to guarantee formally the validity of the control law for a neighborhood around that initial condition. By repeating this procedure for a finite set of initial conditions, we can cover a compact set of initial conditions. This approach can thus be regarded as a highly parallelizable and lightweight (no quantization of state space is required) complement to the more formal approaches, such as [11], [12].

III. Feedback Linearizable Systems

We extend the idea of safety controller synthesis using human generated trajectories to systems with nonlinear dynamics. Consider a dynamical system with input and output

$$\Sigma_{\text{io}} : \begin{cases} \frac{dx}{dt} = f(x) + g(x)u, \ x \in \mathbb{R}^n, \ u \in \mathbb{R}^m, \\ y = h(x), \ y \in \mathbb{R}^m. \end{cases} \tag{4}$$

We assume that $f(\cdot)$ is Lipschitz and $h(\cdot)$ is a smooth function. 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.

We modify the State Safety control problem in Section II to introduce the Output Safety control problem. First, we define a set of goal outputs, GoalOutput$\subset \mathbb{R}^m$, and a set of unsafe outputs UnsafeOutput$\subset \mathbb{R}^m$. An output trajectory is deemed unsafe if it enters UnsafeOutput. Then, the problem can be formulated as:

Problem 2 (Output Safety): Design a feedback control law $u = k(x)$ such that for any initial state $x_0 \in$ Init, the output trajectory of the closed loop system enters GoalOutput before time $t = T_{\max}$, and remains safe until it enters GoalOutput.

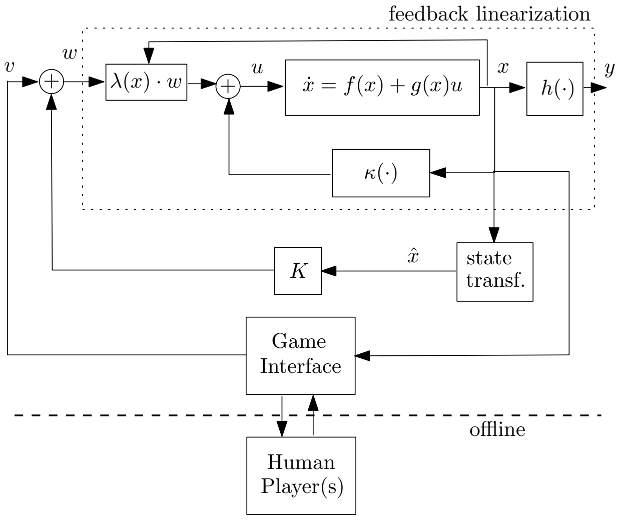

Fig. 1: A block diagram illustrating the concept of controller synthesis using human generated trajectories. The nonlinear dynamics is first transformed to a linear one via feedback linearization. Then, results from [10] can be applied.

Thus, in this problem definition, the validity of an execution trajectory is determined by the output trajectory, and not by state trajectory as in [10] (see Section II).

A. Design Procedure: Theory

Feedback linearization is a classical controller design technique for nonlinear systems (see e.g. [20], [21]). A special case of feedback linearization that is applicable in the Output Safety problem above is the *input-output linearization* [20]. The idea is to introduce a new control input $w(t)$ and design a (nonlinear) feedback law

$$u(t) = \kappa(x) + \lambda(x)w(t), \ w(t) \in \mathbb{R}^m, \tag{5}$$

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 in Nonlinear Control textbooks, such as [21].

By using the feedback law (5), we obtain a linear input-output system in the chain integrator form [20].

$$\Sigma_{\text{lin}} : \begin{bmatrix} \frac{d^{r_1}}{dt^{r_1}} y_1 \\ \vdots \\ \frac{d^{r_m}}{dt^{r_m}} y_m \end{bmatrix} = \begin{bmatrix} w_1(t) \\ \vdots \\ w_m(t) \end{bmatrix}, \tag{6}$$

where $r_1, r_2, \cdots, r_m$ are the *relative degrees* of the system. Furthermore, a state space realization of Σ_{lin} can be obtained through some state transformation from x.

Since Σ_{lin} is linear, we can further apply our earlier results on the controller synthesis technique for linear systems, which was reported in [10]. The overall block diagram is shown in Figure 1.

Remark 1 (zero dynamics): In general, we will always have that [21]

$$r_1 + r_2 + \cdots + r_m \leq n. \tag{7}$$

When this inequality is strict, there is a part of the dynamics of Σ_{io} that is rendered unobservable by the linearizing

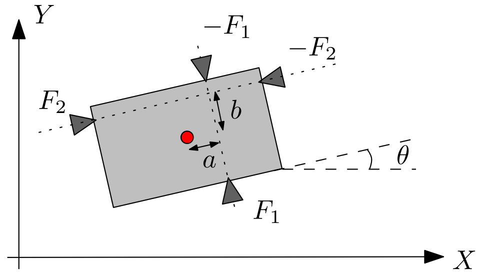

Fig. 2: The rigid object with two jet thrusters. The circle represents the object's center of mass. The control inputs are represented by F_1 and F_2.

feedback. This is called the *zero dynamics* of the system. In this paper, we ignore the possibility of unstable zero dynamics since it does not appear on the system output, and hence is irrelevant to the safety control criteria.

B. Example: Planar Manipulation of a Rigid Object

Consider the control problem related to manipulating a rigid object on a plane using a pair of jet thrusters as control inputs. The axis of the jet thrusts are off-centered (i.e. they do not pass through the object's center of mass). See Figure 2 for an illustration. Therefore, the control inputs will induce both translational and rotational motions on the object. Moreover, we also assume that the object is subject to an external force field that essentially renders the dynamics unstable. The system's dynamics is given by Equations (8) - (10).

$$\ddot{X} = -F_1 \sin\theta + F_2 \cos\theta + 1 - x + 0.1\dot{X}, \quad (8)$$
$$\ddot{Y} = F_1 \cos\theta + F_2 \sin\theta + 1 - y + 0.1\dot{Y}, \quad (9)$$
$$\ddot{\theta} = F_1 \frac{a}{I} - F_2 \frac{b}{I}. \quad (10)$$

Here I denotes the object's moment of inertia. We define the XY coordinate of the object's center of mass as the system's output. In state space form, the dynamics is given by

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \underbrace{\begin{bmatrix} x_2 \\ 1-x_1+0.1x_2 \\ x_4 \\ 1-x_3+0.1x_4 \\ x_6 \\ 0 \end{bmatrix}}_{\triangleq f(x)} + \underbrace{\begin{bmatrix} 0 & 0 \\ -\sin x_5 & \cos x_5 \\ 0 & 0 \\ \cos x_5 & \sin x_5 \\ 0 & 0 \\ \frac{a}{I} & -\frac{b}{I} \end{bmatrix}}_{\triangleq g(x)} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \quad (11)$$

$$y = h(x) \triangleq \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}. \quad (12)$$

Observing that

$$\begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} 1-x_1+0.1x_2 \\ 1-x_3+0.1x_4 \end{bmatrix} + \begin{bmatrix} -\sin x_5 & \cos x_5 \\ \cos x_5 & \sin x_5 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}, \quad (13)$$

we conclude that the vector relative degree of the system given by (11) - (12) is $\{2, 2\}$.

The safety control that we want to solve is a nonlinear version of the example presented in [10]. It can formulated as follows. Suppose that we are to steer the system from the initial set:

$$\text{Init} = \left\{ x \in \mathbb{R}^6 \mid |x_1| \le 0.1, |x_3| \le 0.1, \begin{bmatrix} x_2 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \right\}.$$

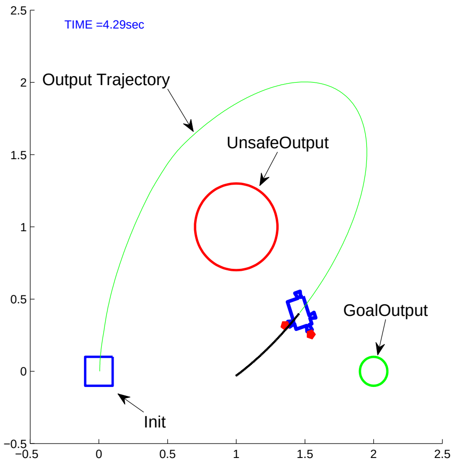

Fig. 3: Screen capture from a game play showing the manipulated object and all sets of interest.

The sets of unsafe and goal outputs are given by

$$\text{UnsafeOutput} = \{ y \in \mathbb{R}^2 \mid \left\| y - [1, 1]^T \right\| \le 0.3 \},$$
$$\text{GoalOutput} = \{ y \in \mathbb{R}^2 \mid \left\| y - [2, 0]^T \right\| \le 0.1 \}.$$

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

$$\left\| \begin{bmatrix} F_1(t) & F_2(t) \end{bmatrix} \right\| \le 2. \quad (14)$$

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

$$\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} \triangleq \begin{bmatrix} -\sin x_5 & \cos x_5 \\ \cos x_5 & \sin x_5 \end{bmatrix} \begin{bmatrix} F_1(t) \\ F_2(t) \end{bmatrix}. \quad (15)$$

Note that this formulation is a little bit different from the outlined procedure in the previous subsection. We can observe that the constraint in (14) is equivalent to

$$\left\| \begin{bmatrix} w_1(t) & w_2(t) \end{bmatrix} \right\| \le 2. \quad (16)$$

Using the linearization approach above, we transform the nonlinear system into a linear affine one

$$\frac{d}{dt}\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix} = \begin{bmatrix} \hat{x}_2 \\ 1-\hat{x}_1+0.1\hat{x}_2 \\ \hat{x}_4 \\ 1-\hat{x}_3+0.1\hat{x}_4 \end{bmatrix} + \begin{bmatrix} 0 \\ w_1 \\ 0 \\ w_2 \end{bmatrix}, \quad (17)$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_3 \end{bmatrix}. \quad (18)$$

We can then proceed to design a controller for the linear system using the method explained in detail in our earlier publication [10], which cannot be included here due to space constraint.

IV. Differentially Flat Systems

Differential flatness is a major tool in nonlinear controller design [18], [19]. The concept was first coined by Fliess et al. in [22], 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 Output Safety control problem for differentially flat nonlinear systems. Our development in this paper follows the work of van Nieuwstadt and Murray [23], who used differential flatness for trajectory generation in motion planning for constrained mechanical systems.

A. Design Procedure: Theory

Any nonlinear system

$$\dot{x} = f(x) + g(x)u,\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,$$

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

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

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,

$$x = \Xi(y, \dot{y}, \ldots, y^{(\ell)}), \tag{19}$$

$$u = \Upsilon(y, \dot{y}, \ldots, y^{(\ell+1)}), \tag{20}$$

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 (4) 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 [18], [23].

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:

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

where $\eta \in \mathbb{R}^m$, $\omega \in \mathbb{R}^m$ is the input, and $\hat{y} \in \mathbb{R}^m$ is the output. Observe that, by construction, any output trajectory of Σ_{flat} is at least $(\ell+1)$ times differentiable. Therefore, any $\hat{y}(t)$ 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 (19) - (20).

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

$$\frac{dq}{dt} = \hat{A}q + \hat{B}\omega,\ q \in \mathbb{R}^{m\ell},\ \omega \in \mathbb{R}^m,$$
$$\hat{y} = \hat{C}q,\ \hat{y} \in \mathbb{R}^m.$$

Notation 1: 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 $\hat{y}(t; q_0, \omega)$.

Notation 2: Given that $q = [\eta^T\ \dot{\eta}^T\ \cdots \eta^{(\ell)T}]^T$, we introduce the following shorthand notation:

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

Definition 1 (valid flat trajectory): Consider the Output Safety control problem as given in Section III. An output trajectory of Σ_{flat}, $\hat{y}(t; q_0, \omega)$, is said to be a valid flat trajectory for an initial state $x_0 \in \text{Init}$ if
(i) $\Xi(q_0) = x_0$, and
(ii) the trajectory $\hat{y}(t; q_0, \omega)$ enters GoalOutput before time $t = T_{\max}$, and remains safe (i.e. does not enter UnsafeOutput) until it enters GoalOutput.

Therefore, a valid flat trajectory for an initial state $x_0 \in \text{Init}$ is mapped by (19) 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 (20). 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 (20). 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 [10]. The procedure is further explained as follows.

Suppose that for an initial state $x_0 \in \text{Init}$ we obtain a valid trajectory for Σ_{io}, which we denote by $y^*(t)$. This can be obtained, for example, from a human playing a computer game that simulates Σ_{io}. We will demonstrate that by having the knowledge of $y^*(t)$, 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 $y^*(t)$ is also an output trajectory of Σ_{flat}, i.e. the one corresponding to the input signal

$$\omega^*(t) = \frac{d^{\ell+1}}{dt^{\ell+1}}y^*(t), \tag{21}$$

and initial states q_0, which are given by

$$\eta_0 = y^*(0);\ \dot{\eta}_0 = \dot{y}^*(0); \cdots;\ \eta_0^{(\ell)} = y^{*(\ell)}(0). \tag{22}$$

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

$$(\hat{A} + \hat{B}\hat{K})^T P + P(\hat{A} + \hat{B}\hat{K}) \preceq 0. \tag{23}$$

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

$$\omega = \omega^* + \hat{K}q. \tag{24}$$

The closed-loop system is then given by

$$\Sigma_{\text{cl}} : (\hat{A} + \hat{B}\hat{K})q + \hat{B}\omega^*. \tag{25}$$

Notation 3: 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 $\hat{y}(t; q_0)$.

Proposition 2: Define the quadratic function $\phi : \mathbb{R}^{m\ell} \times \mathbb{R}^{m\ell} \to \mathbb{R}_+$ as

$$\phi(q_1, q_2) \triangleq (q_1 - q_2)^T P(q_1 - q_2).$$

Then, for any $(q_1, q_2) \in \mathbb{R}^{m\ell} \times \mathbb{R}^{m\ell}$, $\phi(q(t; q_1), q(t; q_2))$ 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, ϕ defines a control autobisimulation function for the linear system Σ_{flat} (see [10]). ■

Proposition 2 establishes trajectory robustness for the state trajectories of Σ_{cl}. Its consequence on the output trajectories is given as follows.

Proposition 3: For any $(q_1, q_2) \in \mathbb{R}^{m\ell} \times \mathbb{R}^{m\ell}$,

$$(\hat{y}(t; q_1) - \hat{y}(t; q_2))^T \hat{C} P \hat{C}^T (\hat{y}(t; q_1) - \hat{y}(t; q_2)) \leq \phi(q_1, q_2). \tag{26}$$

Proof: Equation (26) can be rewritten as

$$\underbrace{(q(t; q_1) - q(t; q_2))^T \hat{C}^T \hat{C} P \hat{C}^T \hat{C} (q(t; q_1) - q(t; q_2))}_{\spadesuit} \leq \phi(q_1, q_2).$$

Because of the structure of $\hat{C}$, we have

$$\hat{C}^T \hat{C} P \hat{C}^T \hat{C} \preceq P. \tag{27}$$

Therefore, using Proposition 2 we can derive

$$\spadesuit \leq \phi(q(t; q_1), q(t; q_2)) \leq \phi(q_1, q_2).$$

■

Observing that $\hat{C} P \hat{C}^T$ is a symmetric positive definite matrix, we can define a norm in $\mathbb{R}^m$ as follows.

$$\|y\|_\eta \triangleq \sqrt{y^T \hat{C} P \hat{C}^T y}. \tag{28}$$

Similarly, we define a norm in $\mathbb{R}^{m\ell}$ as follows.

$$\|q\|_\phi \triangleq \sqrt{q^T P q}. \tag{29}$$

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

Theorem 4: 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

$$\sup_t \|y(t) - y^*(t)\|_\eta < \delta_1 \tag{30}$$

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 $B_\phi(q_0, \delta_2)$, i.e. the $\|\cdot\|_\phi$ ball of radius δ_2 around q_0 as defined in (22).

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

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

$$u^*_\xi = \Upsilon(q(t; q_\xi), \dot{q}(t; q_\xi)). \tag{31}$$

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 $q_\xi \in B_\phi(q_0, \delta)$ satisfying $\Xi(q_\xi) = \xi$.
(ii) By definition of differential flatness, applying the input signal u^*_ξ to Σ_{io} with initial state $\xi \in \mathcal{N}(x_0)$ yields the output trajectory $y(t; q_\xi)$. Further, from Proposition 3 we can see that

$$\begin{aligned}\|y(t; q_\xi) - y^*(t)\|_\eta &= \|y(t; q_\xi) - y(t; q_0)\|_\eta, \\ &\leq \|q_\xi - q_0\| \leq \delta \leq \delta_1.\end{aligned}$$

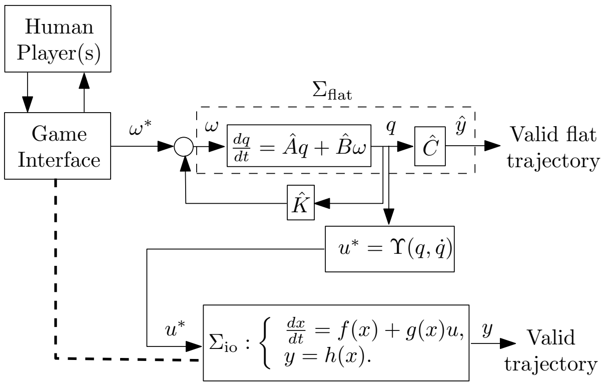
Human Player(s)
Game Interface
ω^*
Σ_{flat}
ω
$\frac{dq}{dt} = \hat{A}q + \hat{B}\omega$
q
$\hat{C}$
$\hat{y}$
Valid flat trajectory
$\hat{K}$
$u^* = \Upsilon(q, \dot{q})$
u^*
$\Sigma_{\text{io}} : \begin{cases} \frac{dx}{dt} = f(x) + g(x)u, \\ y = h(x). \end{cases}$
y
Valid trajectory

Fig. 4: A block diagram of the controller. The dash line connecting the game interface with Σ_{io} indicates that the game is a computer simulation of Σ_{io}.

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

The controller that confers trajectory robustness to Σ_{io} and therefore can be used in trajectory based controller synthesis is shown in Figure 4.

B. Example: 2D Control of a Quadrotor

We consider the control problem related to the motion of a quadrotor on a vertical plane. 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 $w = [w_X, w_Y]^T$ with friction coefficient μ.

We denote the horizontal and vertical positions of the quadrotor with X and Y, respectively. Its orientation w.r.t. the horizontal axis is represented by the angle θ. The system dynamics are given by Equations (32) - (34).

$$\ddot{X} = \mu(w_X - \dot{X}) - \frac{u_1}{m} \sin\theta \tag{32}$$

$$\ddot{Y} = \mu(w_Y - \dot{Y}) - g + \frac{u_1}{m} \cos\theta \tag{33}$$

$$\ddot{\theta} = u_2 \tag{34}$$

Here m denotes the object's mass. We define the XY 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

$$y = [\eta_1, \eta_2]^T = [X, Y]^T.$$

It can be verified that 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.

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

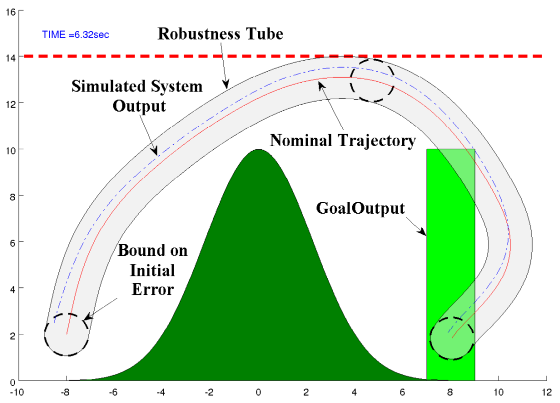

Fig. 5: Simulation of quadrotor system. All trajectories generated for a different initial position within the bounding circle will stay within the robustness tube.

V. Conclusion and Discussion

We present some theoretical results for trajectory based safety controller synthesis for two classes of nonlinear dynamical systems, i.e. feedback linearizable systems and differentially flat systems. These results are the generalization of our earlier results for affine linear systems, which is reported in [10]. We show that by using some appropriate transformations, we can apply the results for affine linear systems to the nonlinear systems.

The key step in trajectory based safety controller synthesis is achieving trajectory robustness, i.e. the ability to generalize the validity of a nominal execution trajectory to other trajectories in its compact neighborhood. With this property, it is possible to solve a safety controller synthesis problem using the knowledge of finitely many valid execution trajectories. In this paper, we demonstrate that we can obtain these valid trajectories from humans, through a computer game that simulates the dynamical system. Related to this approach, two issues are noteworthy.

First, computer games are not the only possible source of valid trajectories. For example, techniques from motion planning can be applicable, too. See, for example, the approach taken in [9]. In fact, motion planning has been extensively used for controlling differentially flat systems [18].

Second, for controller synthesis, the use of human inputs has been previously studied in the context of machine learning. For example, *apprenticeship learning* by Abbeel et al. (see [24] and related references) builds a controller that learns about task specifications from human demonstrations. Our approach is fundamentally different from this approach, because (i) we formally guarantee safety, and (ii) in our approach, it is possible to terminate the ”learning” process (i.e. obtaining human inputs) once the valid trajectories achieve complete coverage of the Init set. Beyond this point, more human generated trajectories are not required, as far as the safety control problem is concerned. Nevertheless, for future work, we hypothesize that our approach can benefit from the learning based approach. Machine learning can be applied on the human generated trajectories to further generate more valid trajectories. This mechanism can potentially speed up the coverage of the Init set.

Acknowledgements

The research reported in this paper is partially supported by the NSF through grant CAREER CNS-0953976 and the Department of Defense SMART Scholarship.

References

[1] 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.
[2] 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.
[3] K. Margellos and J. Lygeros, “Hamilton-Jacobi formulation for reach-avoid differential games,” *IEEE Trans. Automatic Control*, vol. 56, no. 8, pp. 1849–1861, 2011.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] P. Tabuada, “An approximate simulation approach to symbolic control,” *IEEE Trans. Automatic Control*, vol. 53, no. 6, pp. 1406–1418, 2008.
[12] G. Pola, A. Girard, and P. Tabuada, “Approximately bisimilar symbolic models for nonlinear control systems,” *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.
[13] 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.
[14] ——, “Verification using simulation,” in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 3927. Springer Verlag, 2006, pp. 272–286.
[15] 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.
[16] 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.
[17] M. Zamani and P. Tabuada, “Towards backstepping design for incremental stability,” *IEEE Trans. Automatic Control*, vol. 56, no. 9, 2011.
[18] H. Sira-Ramirez and S. Agrawal, *Differentially Flat Systems*. Marcel Dekker Inc., 2004.
[19] J. Levine, *Analysis and Control of Nonlinear Systems: a Flatness based approach*. Springer, 2009.
[20] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.
[21] A. Isidori, *Nonlinear Control Systems*, 3rd ed. Springer, 1994.
[22] 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.
[23] 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.
[24] P. Abbeel, A. Coates, and A. Y. Ng, “Autonomous helicopter aerobatics through apprenticeship learning,” *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608 – 1639, 2010.