2013 American Control Conference (ACC)
Washington, DC, USA, June 17-19, 2013

Optimization of Human Generated Trajectories for Safety Controller Synthesis

Andrew K. Winn and A. Agung Julius

***Abstract*—The aim of the optimal 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 while optimizing some function. Our previous work presented a method for using finitely many human generated trajectories to synthesize a non-optimal safety controller. We propose a formal method for optimizing the human generated trajectories used to synthesize the controller. Our method is based on the calculus of variations, but is different from other similar algorithms in that it uses a gradient descent based approach to directly solve the optimization problem without formulating the optimality conditions given by the Pontryagin Minimum Principle. This method provides a tool for improving the performance of a controller synthesized using the methods outlined in our previous work. We present an example of optimizing a human generated trajectory for a nonlinear system, specifically a quadrotor, and quantify the improvements it is able to generate.**

Keywords: hybrid systems, optimization, optimization algorithms.

I. Introduction

Safety controller synthesis in this paper refers to the problem of designing a controller that ensures that:
(A1) the execution trajectories of the closed-loop system do not enter a prescribed Unsafe set, and
(A2) these execution trajectories terminate in a prescribed Goal set.

This problem is thus tightly related to the concept of safety/reachability analysis. The results presented in this paper are related to the *trajectory-based* approach in safety controller synthesis. 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 [1], [2] or incremental stability [3], [4]. 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 [5], and has been used for trajectory based analysis of hybrid systems in [6], [7], [8].

Our approach in using approximate bisimulation in safety controller synthesis differs from others in the following sense. In [3], [4], [9], the notion of approximate bisimulation is established for incrementally stable systems. Then, it is possible to quantize the continuous state space, which results in a countable transition system approximation of the original dynamics. Later works by Tabuada *et al* relax the incremental stability assumption by achieving it using backstepping controller design [10], or replacing it with incremental forward completeness [11]. A recent work by Colombo and Del Vecchio [12] also uses similar approach in designing safety controller for differentially flat systems. In our approach, the controller is synthesized using finitely many valid human-generated trajectories [2], [13]. 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.

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

In the current paper, we extend our previous work [2], [13] by adding a performance objective to the safety control problem. We seek to optimize a cost function while maintaining the safety aspects, (A1) and (A2), above. We term this problem the *optimal safety control problem*. For hybrid systems, optimal control has been investigated by numerous researchers. For example, Hedlund and Rantzer extended the dynamic programming approach to hybrid systems [14], [15]. Xu and Antsaklis solved the problem of switching time optimization in hybrid systems by posing it as a finite-dimensional nonlinear optimization with the switching time as the variables [16], [17]. A recent work by Girard solved a time optimal control problem for switched systems with an abstraction generated by using approximate bisimulation [18]. Our approach differs from the existing ones in that we use the human generated trajectories as seeds in the optimization process. Subsequently, we morph the human generated trajectories using a gradient descent algorithm along a functional derivative of the cost function.

II. Problem Formulation

Consider a (possibly nonlinear) dynamical system

$$\dot{x} = f(x,u),\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m, \tag{1}$$

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, $\mathrm{Goal} \subset \mathbb{R}^n$, and a set of unsafe states $\mathrm{Unsafe} \subset \mathbb{R}^n$. A trajectory is deemed unsafe if it enters the unsafe set.

In [2], [13] we solve the safety controller synthesis problem by first finding a feedback controller that bounds the deviation between two system trajectories generated using the same input reference signal but initiated from two different states. This deviation defines a tube around a given trajectory that we shall denote as the *robustness tube*. The *robustness radius* denotes the largest tube around a trajectory for which the tube does not intersect Unsafe, and the final state for every trajectory within the tube terminates in Goal. We next have humans generate valid nominal trajectories such that the robustness tubes around the initial states of the trajectories cover Init. The resulting controller consists of determining the nominal trajectory for which the initial state is within the trajectories robustness tube, and applying the corresponding input, along with the feedback that guarantees trajectory robustness. In the ensuing analysis we aim to optimize these nominal trajectories such that the nominal robustness radius is maintained.

The optimal safety control problem can be formulated as choosing the input function $u(t)$ to minimize the cost function given by

$$J(u) \triangleq G(x(T)) + \int_0^T g(x(\tau), u(\tau))\ d\tau, \tag{2}$$

subject to (1) and

$$x(0) = x_0 \in \text{Init}, \tag{3}$$
$$x(T) \in \text{Goal}, \tag{4}$$
$$x(t) \notin \text{Unsafe},\ \forall t \in [0, T] \tag{5}$$

Our goal is to synthesize a controller that solves this problem for every $x_0 \in$ Init. Any trajectory (not necessarily optimal) that satisfies the conditions (3) - (5) is called a *valid trajectory*.

Remark 1: Notice that although the optimal safety control problem as defined above is not formulated for hybrid systems, a similar problem for hybrid systems can be reduced to the above format by following the procedure that is outlined in our earlier papers [1], [2]. Specifically, given a safety controller for a hybrid system that uses a set of nominal trajectories, our method would then allow the designer to optimize each continuous state trajectory that drives the system from one discrete state to the next.

III. Technical Approach

For an initial condition $x_0 \in$ Init, suppose that we can generate a valid trajectory from a human subject. This trajectory corresponds to an input signal $u_0 : [0, T] \to \mathbb{R}^m$, which is generated by the human.

Notation 1: $\xi_\tau(u)$ is the state trajectory at time τ with initial state $x(0) = x_0$, under input signal $u(\cdot)$.

We term the trajectory $\xi_\cdot(u_0)$ the *nominal trajectory* for the initial condition x_0. Since we assume that this trajectory is valid, it satisfies the constraints (3) - (5). However, for the initial state x_0, the nominal trajectory in general will not be the optimal one w.r.t. the cost function (2). Subsequently, we alter the input signal to optimize the cost while maintaining the validity of the trajectory.

Remark 2: We are interested in improving a set of nominal trajectories that cover Init relative to some cost function $J(u)$. As such, we will not optimize the trajectories over the initial condition x_0, since this may compromise the coverage property of our solution. Furthermore, the optimized trajectory will need to maintain some minimum robustness radius to guarantee coverage; to this end, we bloat the unsafe set to constrain the resulting trajectory to meet this bound. A similar idea was used in [19]. This idea is illustrated in Figure 1.

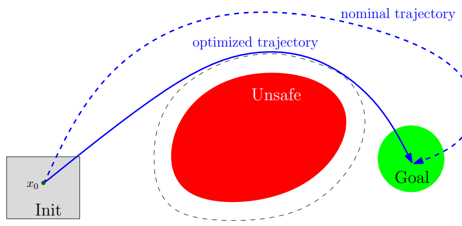

Fig. 1: The nominal trajectory is a valid trajectory that is obtained from a human subject. This is not necessarily optimal. We further optimize the trajectory using gradient descent method along the functional derivative of the cost function in the input signal space.

A. Computation of the Functional Derivative

Our optimization method is based on a gradient descent algorithm along the functional derivative of the cost function $J(u)$ in the input signal space. In this section, we will ignore the safety constraints in the optimal control problem. Later, in Section IV, we will discuss how we can handle the constraints by modifying the cost function.

Consider again the scalar cost function in (2), where we make use of Notation 1:

$$J(u) = G(\xi_T(u)) + \int_0^T g(\xi_\tau(u), u(\tau))\ d\tau.$$

We want to compute the functional derivative[1] (Frechet/Gateaux derivative) of J at a given input signal $u(t)$ in the direction $v(t)$ (see e.g. [20]).

$$dJ(u; v) \triangleq \lim_{\delta \to 0} \frac{J(u + \delta v) - J(u)}{\delta}. \tag{6}$$

This derivative can be written as a scalar valued linear functional of v, as follows:

$$dJ(u; v) \triangleq \langle q, v \rangle \triangleq \int_0^T q(\tau) v(\tau)\ d\tau, \tag{7}$$

where $q(\cdot)$ is an $\mathbb{R}^{1\times m}$-valued function that acts as the gradient of J in the function space of $u(\cdot)$.

For brevity, We use the following notations

$$\begin{array}{ll}
\hat{u} \triangleq u + \delta v, & \frac{\partial G(T)}{\partial x} \triangleq \frac{\partial G}{\partial x}\Big|_{\xi_T(u)} \\
f(\tau) \triangleq g(\xi_\tau(u), u(\tau)), & \frac{\partial g(\tau)}{\partial x} \triangleq \frac{\partial g}{\partial x}\Big|_{(\xi_\tau(u), u(\tau))} \in \mathbb{R}^{1\times n}, \\
\hat{g}(\tau) \triangleq g(\xi_\tau(\hat{u}), \hat{u}(\tau)), & \frac{\partial g(\tau)}{\partial u} \triangleq \frac{\partial g}{\partial u}\Big|_{(\xi_\tau(u), u(\tau))} \in \mathbb{R}^{1\times m}, \\
f(\tau) \triangleq f(\xi_\tau(u), u(\tau)), & \frac{\partial f(s)}{\partial x} \triangleq \frac{\partial f}{\partial x}\Big|_{(\xi_s(u), u(s))} \in \mathbb{R}^{n\times n}, \\
\hat{f}(\tau) \triangleq f(\xi_\tau(\hat{u}), \hat{u}(\tau)), & \frac{\partial f(s)}{\partial u} \triangleq \frac{\partial f}{\partial u}\Big|_{(\xi_s(u), u(s))} \in \mathbb{R}^{n\times m}.
\end{array}$$

The Taylor series expansion of $\hat{g}(\tau)$ at $g(\tau)$ and $G(\xi_T(\hat{u}))$ at $\xi_T(u)$ is given by:

$$\hat{g}(\tau) = g(\tau) + \frac{\partial g(\tau)}{\partial x} d\xi_\tau(u; v)\delta + \frac{\partial g(\tau)}{\partial u} v(\tau)\delta + o(\delta), \tag{8}$$

$$G(\xi_T(\hat{u})) = G(\xi_T(u)) + \delta \frac{\partial G(T)}{\partial x} d\xi_T(u; v) + o(\delta). \tag{9}$$

[1]We assume that the cost function and the dynamics are such that the functional derivative exists.

where $d\xi_\tau(u;v)$ is the functional derivative of $\xi_\tau(u)$ with respect to the input signal $u(t)$. Hence,

$$dJ(u;v) = \frac{\partial G(T)}{\partial x} d\xi_T(u;v) + \int_0^T \left(\frac{\partial g(\tau)}{\partial x} d\xi_\tau(u;v) + \frac{\partial g(\tau)}{\partial u} v(\tau) \right) d\tau. \quad (10)$$

Suppose that

$$d\xi_\tau(u;v) = \langle p_\tau, v\rangle = \int_0^\tau p_\tau(s)v(s)\, ds, \quad (11)$$

where $p_\tau(t)$ is an $\mathbb{R}^{n\times m}$-valued function, which is nonzero only if $\tau \geq t$. Then, we can obtain

$$dJ(u;v) = \int_0^T \frac{\partial G(T)}{\partial x} p_T(s)v(s)\, ds + \int_0^T \left(\int_s^T \frac{\partial g(\tau)}{\partial x} p_\tau(s)\, d\tau \right) v(s) ds + \int_0^T \frac{\partial g(\tau)}{\partial u} v(\tau)\, d\tau. \quad (12)$$

Therefore,

$$q(t) = \frac{\partial G(T)}{\partial x} p_T(t) + \frac{\partial g(t)}{\partial u} + \int_t^T \frac{\partial g(\tau)}{\partial x} p_\tau(t)\, d\tau. \quad (13)$$

Equation (13) suggests that we need to compute the functional derivative $p_\tau(t)$, in order to compute $q(t)$. Observe that

$$d\xi_\tau(u;v) = \lim_{\delta\to 0} \frac{1}{\delta} \int_0^\tau \left(\hat{f}(s) - f(s) \right) ds. \quad (14)$$

Since

$$\hat{f}(s) - f(s) = \frac{\partial f(s)}{\partial x} d\xi_s(u;v)\cdot\delta + \frac{\partial f(s)}{\partial u} v(s)\cdot\delta + o(\delta), \quad (15)$$

it follows that

$$d\xi_\tau(u;v) = \int_0^\tau \left(\frac{\partial f(s)}{\partial x} d\xi_s(u;v) + \frac{\partial f(s)}{\partial u} v(s) \right) ds. \quad (16)$$

Combining (11) and (16), we obtain

$$p_\tau(t) = \frac{\partial f(t)}{\partial u} + \int_t^\tau \frac{\partial f(s)}{\partial x} p_s(t)\, ds. \quad (17)$$

We observe from (13) that $q(t)$ can be written as

$$q(t) = \frac{\partial G(T)}{\partial x} p_T(t) + \frac{\partial g(t)}{\partial u} + \hat{q}(t), \quad (18)$$

where

$$\hat{q}(t) \triangleq \int_t^T \frac{\partial g(\tau)}{\partial x} p_\tau(t)\, d\tau. \quad (19)$$

We shall see that both $\hat{q}(t)$ and $p_T(t)$ can be computed as state trajectories of a Linear Time Varying (LTV) system.

For each t, we define two new variables:

$$\xi_t(s) \triangleq \int_t^s \frac{\partial g(\tau)}{\partial x} p_\tau(t)\, d\tau \in \mathbb{R}^{1\times m}, \quad (20)$$

$$\eta_t(s) \triangleq p_s(t) \in \mathbb{R}^{n\times m}. \quad (21)$$

Observe that these two variables satisfy the boundary (or initial) conditions

$$\xi_t(t) = 0,\ \eta_t(t) = \frac{\partial f(t)}{\partial u}. \quad (22)$$

Moreover, we also have

$$\frac{d\xi_t(s)}{ds} = \frac{\partial g(s)}{\partial x} p_s(t) = \frac{\partial g(s)}{\partial x} \eta_t(s),$$
$$\frac{d\eta_t(s)}{ds} = \frac{\partial f(s)}{\partial x} p_s(t) = \frac{\partial f(s)}{\partial x} \eta_t(s).$$

Therefore, ξ_t and η_t form an LTV system

$$\frac{d}{ds}\begin{bmatrix} \xi_t \\ \eta_t \end{bmatrix} = \left[\begin{array}{c|c} 0 & \frac{\partial g(s)}{\partial x} \\ \hline 0 & \frac{\partial f(s)}{\partial x} \end{array}\right] \begin{bmatrix} \xi_t \\ \eta_t \end{bmatrix}. \quad (23)$$

Solving this LTV system with initial condition (22), we can obtain both $\hat{q}(t)$ and $p_T(t)$, which are given by

$$\hat{q}(t) = \xi_t(T),\ p_T(t) = \eta_t(T). \quad (24)$$

B. Gradient Descent Algorithm

The gradient descent algorithm that we use is the standard gradient descent algorithm with an adaptive step size. This method is not unlike the iterative techniques presented in [21]. Note that if we are not at a local minimum, the cost is guaranteed to decrease along the functional derivative for a small enough step size, and if we are at a local minimum, then the cost will not decrease, and we have converged. The initial iteration is seeded with the human generated input. The algorithm we used can be represented as

```
1: Generate initial trajectory $\xi_\tau(u_i)$
2: while $\lambda > \epsilon$ do
3:     Generate $q(\tau)$ from $\xi_\tau(u_i)$ and $u_i(\tau)$
4:     Generate next iteration's input $u_{i+1}(\tau) = u_i - \lambda q(\tau)$
5:     Generate next iteration's trajectory $\xi_\tau(u_{i+1})$
6:     Calculate cost $J(\xi_\tau(u_{i+1}), u_{i+1})$
7:     if $J(\xi_\tau(u_{i+1}), u_{i+1}) < J(\xi_\tau(u_{i+1}), u_i)$ then
8:         $\lambda = \alpha\lambda, \quad \alpha > 1$
9:     else
10:        $\lambda = \beta\lambda, \quad 0 < \beta < 1$
11:    end if
12:    $i = i + 1$
13: end while
```

C. Remarks About the Technical Approach

Trajectory optimization is an active research area. Earlier advances in this field were made in 1960's, with applications in the aeronautics and space flight (see the survey paper [22]). The approach that we outline in this section differs from some of the common ones. In the following, we present some (non-exhaustive) comparison.

There are many trajectory optimization methods that are based on the idea of converting the infinite-dimensional optimization problem into a finite-dimensional nonlinear optimization problem ([22], [23]). This can be done, for example by approximating the continuous-time model with a discrete-time one. Another approach is to constrain the (input or state or output) trajectories in a finite dimensional space, which can be represented with finitely many parameters. For example, splines or other basis functions can be used. This is an attractive option, especially if the output or state trajectories can be arbitrarily chosen in this space, such as the case of differentially flat systems [24], [25], [26], [27].

Another class of methods is based on (random) sampling of the state space to seek the optimal solution. Examples of this family are the Rapidly exploring Random Trees (RRT) based methods (see e.g. [28], [29], [30]).

For discrete-time hybrid systems, optimal control has also been addressed in the context of model predictive control [31], [32], [33].

The most common approach in directly solving the optimal control problem as infinite-dimensional optimization problem is by using the calculus of variation [20], [21]. Necessary (local) condition for optimality is given the Pontryagin Minimum Principle (PMP) [22], [25]. In many cases, the PMP results in a two point boundary value problem (TPBVP). To find the optimal solution, we need to solve the TPBVP, for which there are various numerical techniques (see survey in [22]).

Our approach is also based on the calculus of variation. However, instead of formulating the optimality condition using Lagrange multipliers (i.e. PMP), we directly substitute the constraint into the optimization problem, and apply a gradient descent algorithm on the resulting unconstrained problem. As is the case with any PMP based method, upon convergence of the gradient descent algorithm, we can only guarantee local optimality. However, because of the nature of the gradient descent approach, we can guarantee that any solution to which we converge is better than the (human generated) nominal trajectory[2]. In fact, the same holds for any interim solution prior to convergence.

IV. Illustrative Example

A. Setup

For this example, we generate the nominal trajectories using a joystick on a simulation of the system. As such, the input signal to be a stepwise function of time, i.e., $u(t) = u[k]$ for $kT_s \le t < (k+1)T_s$. This transforms (11) to be

$$d\xi_{kT_s}(u;v) = \sum_{j=0}^{k-1} \left(\int_{jT_s}^{(j+1)T_s} p_{kT_s}(s)ds \right) v[j]. \tag{25}$$

In light of this, we define our cost function to be one that only applies to sampled states with a sampling time T_s, that is,

$$J(u,v) = G(\xi_{NT_s}(u)) + \sum_{k=0}^{N} g(\xi_{kT_s}(u), u[k]) \tag{26}$$

$$dJ(u;v) = \frac{\partial G(NT_s)}{\partial x} d\xi_{NT_s}(u;v) + \sum_{k=0}^{N} \left(\frac{\partial g(kT_s)}{\partial x} d\xi_{kT_s}(u;v) + \frac{\partial g(kT_s)}{\partial u} v[k]) \right), \tag{27}$$

where the index NT_s represents the terminal time.

Although our implementation becomes a finite-dimensional nonlinear program, it is distinguished from the methods presented in [22] in that we do not approximate the system dynamics with a discrete time model. For brevity we define

$$\rho_k[j] \triangleq \int_{jT_s}^{(j+1)T_s} p_{kT_s}(s)ds. \tag{28}$$

From (27) and (25) we find that the functional derivative of the cost is given by $dJ(u;v) = \langle q, v \rangle$ where

$$q[j] = \tfrac{\partial G(NT_s)}{\partial x} \rho_N[j] + \textstyle\sum_{k=j}^{N} \tfrac{\partial g(kT_s)}{\partial x} \rho_k[j] + \tfrac{\partial g(jT_s)}{\partial u}. \tag{29}$$

We still require a method to calculate $\rho_k[j]$. To this end, we define a new function,

$$\beta_t(\tau) = \int_t^{\tau} p_{\tau}(s)ds. \tag{30}$$

If we differentiate both sides with respect to τ and combine it with (17) and (22) we find that

$$\frac{\partial \beta_t(\tau)}{\partial \tau} = \frac{\partial f(\tau)}{\partial u} + \frac{\partial f(\tau)}{\partial x} \beta_t(\tau). \tag{31}$$

Since this is a linear time-varying system, we can represent β at the sample times explicitly as

$$\beta_{jT_s}((k+1)T_s) = \Phi((k+1)T_s, kT_s)\beta_{jT_s}(kT_s) + \Gamma((k+1)T_s, kT_s)), \tag{32}$$

where $\Phi(\cdot,\cdot)$ can be interpreted as a state transition matrix,

$$\frac{\partial \Phi(\tau, kT_s)}{\partial \tau} = \frac{\partial f(\tau)}{\partial x} \Phi(\tau, kT_s), \tag{33}$$

$$\frac{\partial \Gamma(\tau, kT_s)}{\partial \tau} = \frac{\partial f(\tau)}{\partial u} + \frac{\partial f(\tau)}{\partial x} \Gamma(\tau, kT_s), \tag{34}$$

with initial conditions

$$\Phi(kT_s, kT_s) = I, \qquad \Gamma(kT_s, kT_s) = \mathbf{0}. \tag{35}$$

Noting from (30) that $\beta_t(t) = 0$ and using (32), we find that

$$\rho_{j+1}[j] = \Gamma(((j+1)T_s, jT_s,) \tag{36}$$

$$\rho_k[j] = \Phi((k+1)T_s, kT_s)\rho_{k-1}[j]. \tag{37}$$

Thus, to find the gradient to use in our update law we use an ODE solver to simultaneously simulate the dynamics of the system as well as the two first order initial value problems defined by (33)–(35) over each sample time period for the input signal of the current iteration. The next iterations input signal is then generated by

$$u_{i+1}[j] = u_i[j] - \lambda q[j], \tag{38}$$

[2]We assume that the human generated trajectory is not optimal to begin with.

since we now have all of the required values from (29).

Remark 3: If we denote the number of states in our system as n_x and the number of inputs as n_u, then we see that in general for each iteration we will need to solve n_x (the dynamics) + $n_x \times n_x$ (Φ) + $n_x \times n_u$ (Γ) first order ODEs. In practice, many of the elements in Φ and Γ will have a derivative of zero, and can be removed from the computation. For example, in Section IV-B $n_x = 6$ and $n_u = 2$, yet we only need to solve 33 first order ODEs, rather than 54. The computation time will subsequently depend on the choice of ODE solver (we used MATLAB's `ode23`) and the number of iterations, which is influenced by the method of gradient descent and the terminating condition.

We consider the problem of finding an input initialized to a human generated input that generates the trajectory with minimal terminal time that maintains a distance d from Unsafe and whose endpoint is in the interior of Goal and is at least a distance d from the boundary of Goal. Further, we require the input to be both upper and lower bounded by some value. Without loss of generality we can inflate the boundary of Unsafe by the distance $d+\epsilon_d$, shrink the boundary of Goal by the distance $d+\epsilon_g$, shrink the input bounds by ϵ_i and discuss trajectories that avoid the modified Unsafe and terminate in the modified Goal while meeting the modified input bounds.

To this end, we introduce three cost functions. The first is a cost applied to the end point of the trajectory that is zero inside Goal and increases monotonically with the distance away from Goal. The second cost is integrated along the trajectory, whose value is zero outside and on the boundary of Unsafe and whose value increases inside Unsafe monotonically as a function of distance from the boundary. Finally, we introduce a cost on the input, whose value is zero between the lower and upper bound and whose value increases monotonically as a function of the distance from the closer bound. If the algorithm converges to some ϵ close to zero that is determined by ϵ_u, ϵ_g, and ϵ_i, then the converged trajectory is guaranteed to be outside of the original Unsafe by d, the trajectory terminates inside the original Goal by d, and the optimal input meets the input constraints.

Remark 4: Note that for this example, the cost function is zero for all trajectories that have the desired robustness radius, and positive otherwise. Thus, the optimization strategy is used to change infeasible trajectories into feasible ones. To find the minimum time, we use a bisection search over all terminal times for the smallest one for which the truncated trajectory can be made feasible.

B. Simulation for 2D Control of a Quadrotor

Consider the task of controlling a quadrotor over a hill into some goal region on the other side constrained to fly below some ceiling and with its vertical thrust upper bounded by 14.9 N and lower bounded by 0 N. This example is examined in our previous paper [13] and is inspired by a similar one used in [34]. The dynamics of the system are given by

$$\begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \mu(w_X - \dot{X}) - \frac{u_1}{m}\sin\theta \\ \mu(w_Y - \dot{Y}) + \frac{u_1}{m}\cos\theta \\ u_2 \end{bmatrix} \tag{39}$$

where $[w_X, w_Y]^T$ is the velocity vector for the wind, μ is a friction coefficient, $[X,Y]^T$ is the position of the quadrotor, and θ is the pitch of the quadrotor.

For this system, we take the cost function associated with the input to be a quadratic function of the form

$$g_{\text{input}}(u) = \begin{cases} u_1^2 & u_1 < 0 \\ 0 & 0 \le u_1 < 14.9 \\ (u_1 - 14.9)^2 & u_1 \ge 14.9 \end{cases}. \tag{40}$$

We choose for this example to have the trajectory maintain a robustness radius of 1.0. We define ModGoal to be the goal region shrunk by a distance of 1.0. For the cost associated with the end point, the cost function was taken to be

$$G(x(T)) = \begin{cases} 0 & x(T) \in \text{ModGoal} \\ r^2 & x(T) \notin \text{ModGoal} \end{cases} \tag{41}$$

where r is the shortest distance between $x(T)$ and ModGoal.

Regarding the unsafe regions, note that in this case there are two of them, Unsafe_1 and Unsafe_2. The cost for a given point on the trajectory with respect to Unsafe_i is given by

$$g_{\text{Unsafe}_i}(x(\tau)) = \begin{cases} 0 & x(\tau) \notin \text{Unsafe}_i,\ r_i > d \\ -\log(r_i - d) + \frac{r_i - d}{d} & x(\tau) \notin \text{Unsafe}_i,\ r_i \le d \\ \infty & x(\tau) \in \text{Unsafe}_i \end{cases} \tag{42}$$

where r_i is the shortest distance between $x(\tau)$ and Unsafe_i.

The results for a given human generated input are shown in Figure 2. Although this initial trajectory is feasible, it appears to overshoot the hill and work its way back to the goal set at the end. This trajectory also maintains an undesirably small buffer with the unsafe set—a distance of 0.22 in the scale of

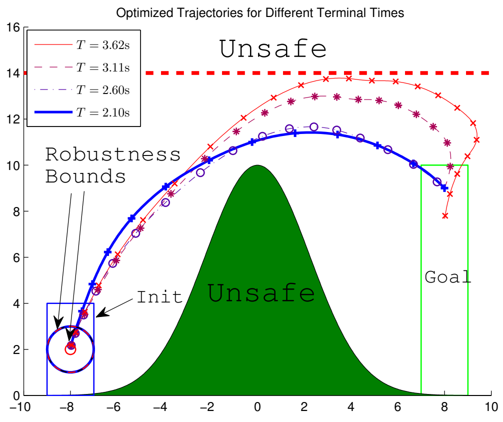

Fig. 2: Initial, optimal, and intermediate feasible trajectories generated by the bisection method. The rectangle to the left represents the initial states to be covered, and the results are shown for only one of the nominal trajectories. The circles about the initial point show the robustness bounds, and the markers along the trajectories are evenly spaced to give a sense of velocity along each trajectory.

the figure. The optimization algorithm is run on this trajectory, and resultant trajectories at several intermediate iterations of the bisection method where are shown. The optimal trajectory is 42% faster, satisfies input constraints, and maintains the desired distances from the unsafe regions and the boundary of the goal region.

V. Conclusion and Discussion

We present a method for optimizing a human generated trajectory with respect to some cost function such that the aim of the underlying safety controller synthesis problem is still met. Our method uses calculus of variations, but is distinguished from other similar approaches in that it uses a gradient descent approach to directly solve the optimization problem without formulating the optimality conditions given by the Pontryagin Minimum Principle. This approach guarantees that a solution generated by our method will be an improvement over the human generated trajectory.

We present a method for implementing the algorithm on a continuous nonlinear system with discrete input signal, and provided an example of its application to a quadrotor system. The constraints and the desired robustness bound were included via appropriately chosen cost functions, and the corresponding unconstrained system was minimized. The trajectory of (locally) minimal terminal time was found by using a bisection method to determine where to truncate the nominal trajectory and determine if it is feasible.

VI. 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] 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.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] J. Camara, A. Girard, and G. Goessler, "Safety controller synthesis for switched systems using multi-scale symbolic models," in *Proc. IEEE Conf. Decision and Control*, Orlando, Florida, 2011.

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

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

[12] A. Colombo and D. Del Vecchio, "Supervisory control of differentially flat systems based on abstraction," in *Proc. IEEE Conf. Decision and Control*, Orlando, Florida., 2011.

[13] 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.

[14] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *Proc. IEEE Conf. Decision and Control*, 1999.

[15] S. H. and A. Rantzer, "Convex dynamic programming for hybrid systems," *IEEE Trans. Automatic Control*, vol. 47, no. 9, pp. 1536–1540, 2002.

[16] X. Xu and P. Antsaklis, "Optimal control of switched autonomous systems," in *Proc. IEEE Conf. Decision and Control*, 2002.

[17] X. X. and Panos Antsaklis, "Results and perspectives on computational methods for optimal control of switched systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2623. Springer, 2003, pp. 540–555.

[18] A. Girard, "Synthesis using approximately bisimilar abstractions: time-optimal control problems," in *Proc. IEEE Conf. Decision and Control*, Atlanta, Georgia., 2010, pp. 5893 – 5898.

[19] A. Girard, A. A. Julius, and G. J. Pappas, "Approximate simulation relations for hybrid systems," *Int. J. Discrete Event Dynamic Systems*, vol. 18, pp. 163–179, 2008.

[20] D. G. Luenberger, *Optimization by Vector Space Methods*. New York, NY: Wiley-Interscience, 1969.

[21] D. E. Kirk, *Optimal Control Theory*. Dover Publications Inc., 1970.

[22] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[23] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[24] 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.

[25] M. B. Milam, "Real-time optimal trajectory generation for constrained dynamical systems," Ph.D. dissertation, California Institute of Technology, 2003.

[26] I. M. Ross and F. Fahroo, "Pseudospectral methods for optimal motion planning of differentially flat systems," *IEEE Trans. Automatic Control*, vol. 49, no. 8, pp. 1410–1413, 2004.

[27] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE. Trans. Robotics*, vol. 21, pp. 1077–1091, 2005.

[28] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in *Proc. IEEE Conf. Decision and Control*, Hawaii, USA, 2003.

[29] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS, vol. 2993. Springer, 2004, pp. 142–256.

[30] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[31] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming the explicit solution," *Automatic Control, IEEE Transactions on*, vol. 47, no. 12, pp. 1974 – 1985, dec 2002.

[32] F. Borrelli, M. Baoti, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709 – 1721, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109805001524

[33] A. Bemporad and S. Di Cairano, "Model-predictive control of discrete hybrid stochastic automata," *Automatic Control, IEEE Transactions on*, vol. 56, no. 6, pp. 1307 –1321, june 2011.

[34] 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.