Control Input Synthesis for Hybrid Systems Using Informed Search

Klaus Schmidt, James Kapinski, and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
{klaussch|jpk3|krogh}@andrew.cmu.edu

***Abstract*— We present a method for synthesizing a sequence of robust control inputs for a class of hybrid systems. Our goal is the generation of a control sequence that drives the system from a given initial state set to a pre-specified goal set without violating constraints on the system state, under the assumption that the hybrid system is exposed to bounded disturbances. We use a technique that combines dynamic programming and informed search. The control sequence generated by our synthesis procedure is guaranteed to meet safety requirements. An extension to nonlinear systems is presented and computational time is compared to a mixed-integer programming approach for computing an optimal but non-robust solution to the problem.**

I. Introduction

We present a technique for synthesizing sequences of control inputs for discrete-time hybrid systems, which uses a combination of dynamic programming and informed search. At the mode-switching level, dynamic programming is used to determine the best mode-switching sequence that satisfies constraints on the system behavior. Within each mode, local vector field behavior within a mode is used to guide an informed search [13] to find a safe sequence of control inputs that attempts to follow the dynamic programming solution. Examples of systems that would benefit from this type of control sequence synthesis include air traffic control systems and chemical processes with safety constraints [14], [3].

Discrete-time synthesis techniques that formulate the problem as a mixed integer quadratic program (MIQP) have been examined [3]. These techniques use a numerical solution of the MIQP problem in a model predictive control (MPC) feedback loop. This technique does not accommodate uncertainties, however, and solving the MIQP is computationally expensive.

Dynamic programming has been applied to the controller synthesis problem [1]. The accuracy of the solution found by dynamic programming depends on the resolution that is used to partition the state space of the system. The partitioning of the state space is prohibitive for systems with a large number of state variables.

This research was supported in part by Ford, the US Defense Advance Projects Research Agency (DARPA) contract nos. F33615-00-C-1701 and F33615-02-C-4029, US Army Research Office (ARO) contract no. DAAD19-01-1-0485, the US National Science Foundation (NSF) contract no. CCR-0121547, and the Institute of Control Engineering and Automation at the University of Erlangen-Nuremberg.

A* search techniques, which are a form of informed search, have been applied to the synthesis problem [5], [6]. A* search techniques are often inefficient because they utilize knowledge of local behaviors that do not characterize the global behavior well.

We present a method that combines dynamic programming, which captures global information, and informed search, which uses local information to guide the system from mode to mode. To determine appropriate mode-switching behavior, we use a Bellman-Ford algorithm. The solution given by the Bellman-Ford algorithm is used to guide an informed search within each of the modes [5], [6]. The goal of the informed search in each mode is to find a path to the next mode, where the next mode is specified by the Bellman-Ford Solution.

Our informed search employs branch and bound ideas [12]. We use local vector field information in order to estimate the cost function for a best first search of the set of all control sequences. Branch and bound is applied to prune the search tree of failing sequences. We employ ellipsoidal reachability concepts to conservatively estimate the set of reachable states for a given input sequence [10].

We also present a method for applying our technique to nonlinear systems. The technique is performed on a piecewise affine approximation of the nonlinear dynamics, where the error incurred by the approximation is compensated for by adding an uncertain input term to the system dynamics.

II. Preliminaries

We consider the following class of switched-mode systems.

Definition 2.1: A discrete-time switched-mode system (DSS) is a tuple $\mathcal{S} = (I, \mathcal{X}, \mathcal{U}, \mathcal{D}, X_0)$, where:

- I is the finite set of *modes*;
- $\mathcal{X} = \{X_i\}_{i \in I}$ is a partition of the state space $\mathbb{R}^n$ (i.e., $\bigcup_{i \in I} X_i = \mathbb{R}^n$ and $X_i \bigcap X_j = \emptyset$ for $i \neq j$);
- $\mathcal{U} = \{U_j\}_{j \in J}$, is the collection of input disturbance sets for each mode, where each U_j is a compact set in $\mathbb{R}^m$;
- $\mathcal{D} = \{f_i\}_{i \in I}$ is the set of dynamics associated with each mode, where $f_i : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$; and
- $X_0 \subseteq \mathbb{R}^n$ is the set of initial conditions

We assume the continuous dynamics of the DSSs are linear; that is, for each $i \in I$, there are matrices A_i, B_i

such that $f_i(x,u) = A_i x + B_i u$. The input set $\mathcal{U}$ is used to represent a discrete set of inputs with bounded additive noise, that is, if $u \in \mathbb{R}^m$ is a discrete input and $V \subset \mathbb{R}^m$ is a bounded input set, then $U \in \mathcal{U}$, where

$$U = \bigcup_{v \in V} \{u + v\}.$$

Note that the class of DSS systems include piecewise linearizations of nonlinear systems.

Definition 2.2: A sequence $(x_0 u_0 x_1 u_1 x_2, \ldots)$ is a *run* of a DSS $\mathcal{S}$ if for all $k \geq 0$, if $x_k \in X_i$, then $x_{k+1} = f_i(x_k, u_k)$.

Given a DSS $\mathcal{S}$, a set $X \subset \mathbb{R}^n$, and a sequence $\pi = (j_0 j_1 \ldots j_k)$, where each $j_i \in J$, $Reach(X, \pi)$ is given by

$$\begin{aligned} Reach(X,\pi) \quad = \quad & \{x | x = x_{k+1} \text{ for some run of } \mathcal{S} \\ & (x_0 u_0 \ldots x_{k+1}), \text{ where} \\ & x_0 \in X, \text{ and each } u_i \in U_{j_i} \\ & \text{for } 0 \leq i \leq k\}. \end{aligned}$$

We are interested in solving the following problem.

Definition 2.3: Given a DSS $\mathcal{S}$, a fail set $F \subset \mathbb{R}^n$, and a goal set $G \subset \mathbb{R}^n$, the *safe synthesis problem* is to compute an input sequence π such that the following holds:

- $Reach(X_0, \pi) \subseteq G$;
- for every prefix $\hat{\pi}$ of π, $Reach(X_0, \hat{\pi}) \cap F = \emptyset$.

In applications, the set F represents some region of the state space that the system must never enter, such as a temperature limit in a chemical process or a collision condition in an air traffic control system. The set G is a region of the state space that the system must reach, such as a safe shutdown condition in a nuclear reactor.

We use the following notation in our procedure to compute the reachable states. For $X \subseteq \mathbb{R}^n$, $U \subseteq \mathbb{R}^m$, and $i \in I$,

$$\begin{aligned} Post_i(X,U) \quad \triangleq \quad & \{x' | x' = f_i(x,u) \text{ for some } x \in X \\ & \text{and } u \in U\}. \end{aligned}$$

Note that the $Post_i$ operator applies the dynamics for mode i to all states in X and inputs in U, even if these sets include states and inputs that are not defined for mode i. This freedom will be used to compute over-approximations to the reachable sets for the DSS.

An ellipsoid $\mathcal{E}(x_c, Q) \subset \mathbb{R}^n$ is defined as

$$\mathcal{E}(x_c, Q) \quad = \quad \{x | (x - x_c)^T Q^{-1} (x - x_c) \leq 1\},$$

where $x_c \in \mathbb{R}^n$ and $Q \in \mathbb{R}^n \times \mathbb{R}^n$ is a symmetric, positive definite matrix. Since $\mathcal{E}(x_c, Q)$ is a closed convex set, it can be described by its support function

$$\begin{aligned} \rho(l | \mathcal{E}(x_c, Q)) \quad &= \quad \sup_{x \in \mathcal{E}(x_c, Q)} l^T x \\ &= \quad l^T x_c + \sqrt{l^T Q l} \end{aligned}$$

III. Safe synthesis procedure

Our method for solving the safe synthesis problem uses informed search guided by local vector field information. Since local vector field behavior in one mode is not indicative of the vector field behavior in other modes, a second technique, dynamic programming, is used to establish a desired mode switching sequence. This two level approach is described as follows:

- Construct a graph that represents the approximate time needed to go from mode to mode with each control, and find the best mode switching path using dynamic programming;
- In each mode, starting from the mode that contains the initial condition set, use an informed search to compute a path to the next mode, where the next mode is given by the dynamic programming solution.

A. The dynamic programming step

The dynamic programming step constructs a graph, where the vertices represent the modes and the edges are labelled with estimates of the amount of time it takes to travel from mode to mode. While the safe synthesis problem has no optimization concept, the minimum time criterion is used so that the dynamic program finds a useful solution.

Definition 3.1: Given a DSS $\mathcal{S}$, a fail set $F \subset \mathbb{R}^n$, and a goal set $G \subset \mathbb{R}^n$, a *dynamic programming graph* (DPG) is a tuple $\mathcal{G} = (M, \delta)$, where

- $M = \{M_i\}_{i \in I}$: a set of vertices, one for each mode in $\mathcal{S}$;
- $\delta : M \times J \to M \times \mathbb{R}$: if input set U_j takes the system from state a to state b with an estimated cost of c, we write $\delta(M_a, j) = (M_b, c)$.

Costs for each mode-to-mode transition are computed by performing simulations from the center points of the modes, once for each input. The first mode that the simulation enters determines the destination of the arc in the DPG and the number of simulation steps it takes determines the weight. This is similar to techniques used in the dynamic programming literature for discretizing continuous optimization problems [11].

Once the DPG has been constructed, a Bellman-Ford algorithm is performed on it to produce a function $s : M \to M$, where $s(m) = m'$ means that m' is the mode reachable from mode m with the lowest estimated cost-to-go.

B. The informed search step

Starting from the mode that contains the initial condition set, the s function provides a *subgoal* to a Search Procedure, where the subgoal is an intermediate goal that is to be reached before the overall goal is reached. In each mode, the subgoal is the optimal next mode as given by the function s.

Definition 3.2: Given a DSS $\mathcal{S}$, a *DSS automaton* (DSSA) for $\mathcal{S}$ is a tuple,
$\mathcal{D} = (Q, \Sigma, E, \lambda, \rho_X, \rho_U)$, where

- Q - a finite set of states;
- Σ - a finite set of symbols;
- $E \subseteq Q \times Q$ - a set of transitions;
- $\lambda : E \rightarrow \Sigma^*$ - a labelling function that assigns an input string to each transition;
- $\rho_X : Q \rightarrow 2^{C_{\mathbb{R}^n}}$ - a function that assigns regions in $\mathbb{R}^n$ to each state in Q. $C_{\mathbb{R}^n}$ denotes the set of compact, connected subsets of $\mathbb{R}^n$;
- $\rho_U : \Sigma \rightarrow C_{\mathbb{R}^m}$ - a function that assigns a region in $\mathbb{R}^m$ to each symbol in Σ.

During the Search Procedure, a DSS automaton will be constructed whose purpose is to represent the portions of the state space that have been explored and the input sequences by which they were reached.

The Search Procedure is shown in Fig. 1. The procedure assumes that a DSS $\mathcal{S}$, a fail region $F \subset \mathbb{R}^n$ and a goal region $G \subset \mathbb{R}^n$ are given, and a DPG $\mathcal{G}$ and a function s are provided. Also, we assume that the initial condition set X_0 is an ellipsoid $\mathcal{E}_0$.

```
/* Search Procedure */
Q := q_0
Σ := J
E := ∅
ρ_X(q_0) := E_0
For all j ∈ J
   ρ_U(j) := U_j
queue := q_0
stillworking := 1
While stillworking
   /* Find best candidate */
   For all q ∈ queue
      best := 0
      mode := determinemode(D, q)
      subgoal := s(mode)
      If merit(q, D, subgoal, G) > best
         best := merit(q, D, subgoal, G)
         bestq := q
   /* Expand best q and remove it from the queue */
   Remove bestq from queue
   For all j ∈ Σ
      Add q̂_j to queue
      Add q̂_j to Q
      Add (q, q̂_j) to E
      (ρ_X(q̂_j), π) := prop(ρ_X(q), j)
      δ((q, q̂_j)) := π
      (G, s) := updateDPS(D, G)
         If ρ_X(q̂_j) ⊆ G
            stillworking := 0
```

Fig. 1. Informed Search Procedure.

The Search Procedure requires four functions: $determinemode$, $merit$, $updateDPS$, and $prop$.

The function $determinemode(\mathcal{D}, q)$ returns an identifier for the mode that q occupies. This mode is then used to determine the next subgoal.

The $merit(q, \mathcal{D}, subgoal, \mathcal{G})$ function assigns a merit value to the state q given the subgoal associated with q. The merit value is based on an estimate of the minimum time it will take for the reachable set will enter the goal region, which is in turn based on a notion of distance between sets of points. The state with the highest merit value is expanded by the informed search.

The $updateDPS(\mathcal{D}, \mathcal{G})$ function takes the updated DSSA and performs an update of the DPG and the function s. Updates of the DPG and s are useful when a state that has just been added to the DSSA is inside a mode that has not been reached previously, in which case it is helpful to recompute the cost estimates from the new mode using a point inside the region associated with the new state q, since this cost estimate will be more accurate than the original one, which was computed using the center of the mode.

The $prop(\mathcal{N}, j)$ function computes a conservative estimate of the reachable set computed from the set of regions $\mathcal{N}$ using the input j. Fig. 2 presents the $prop$ procedure. The function first attempts to identify an input string π, where π is a string of j's of length k and $1 \leq k \leq iter_{max}$, such that the estimate of $Reach(\mathcal{E}, \pi)$, for each $\mathcal{E} \in \mathcal{N}$, is completely within one mode. If such a string π does not exist, then $prop$ returns a conservative estimate of the reachable set of states from the regions in $\mathcal{N}$ given any input from the set U_j.

The function $\hat{Post}_i(X, U)$ computes an estimate of $Post_i(X, U)$ such that $Post_i(X, U) \subseteq \hat{Post}_i(X, U)$. Assuming all regions are ellipsoidal, $\hat{Post}_i(X, U)$ can be computed using the ellipsoidal technique detailed by Kurzhanski [10]. The $bound(\hat{\mathcal{N}})$ function uses a numerical method, involving the solution of a linear matrix inequality [4], to produce one ellipsoid that contains all of the ellipsoids in $\hat{\mathcal{N}}$.

```
/* Procedure prop */
work := 0
N̂ := N
π := ∅
For iter = 1 : iter_max
   π := π · j
   For all Ê ∈ N̂
      For all i such that Ê ∩ X_i ≠ ∅
         N̂ := N̂ ∪ P̂ost_i(Ê, U)
   If Ê ⊆ X_i ∀Ê ∈ N̂ for some i
      N̂ := bound(N̂)
      work := 1
      break
If ¬work
   N̂ := ∅
   π := j
   For all E ∈ N
      For all i such that E ∩ X_i ≠ ∅
         N̂ := N̂ ∪ P̂ost_i(E, U)
Return N̂ and π
```

Fig. 2. Procedure to compute propagation of regions.

IV. Switched-mode approximations of nonlinear systems

A DSS can be used to approximate a continuous-time, nonlinear system such that the behaviors of the former contain the behaviors of the latter. Our synthesis technique can then be performed on the approximate system to compute safe control sequences for nonlinear systems with state constraints.

The approximation is performed by first partitioning the nonlinear system's state space. Then an affine approximation of each mode is computed by taking the first two terms of a Taylor series approximation of the nonlinear vector field about the center point of each partition element.

Consider a nonlinear vector field $g(x(t), u(t))$ and a partition of the state space $\mathcal{X} = \{x_i\}_i$. Let $x_i \in \mathbb{R}^n$ be the center point of partition element i, and $\hat{f}_i(x(t), u(t))$ be the affine approximation of the nonlinear vector field around the point x_i. The affine approximation of the vector field is converted to a discrete-time, affine update equation $x_{k+1} = f_i(x_k, u_k)$ using an Euler approximation.

The maximum error incurred in approximating the nonlinear dynamics is accounted for by adding an uncertainty term to the update equation. If $\phi(x_0, u(t), T)$ is the solution to the differential equation $\dot{x}(t) = g(x(t), u(t))$ with initial condition x_0 and $u(t) = u_k$ for $0 \leq t \leq T$, $f_i(x_0, u)$ is the update equation for partition element i, x_0 is in element i, and $\|\phi(x_0, u(t), T) - f_i(x_0, u_k)\| \leq e^*$, then $\phi(x_0, u(t), T) \subseteq f_i(x_0, u) + V$, where $V = \mathcal{E}(0, e^{*2} I)$.

From the Fundamental Inequality Theorem [9], a bound on $\|\phi(x_0, u(t), T) - f_i(x_0, u_k)\|$ is given by

$$\|\phi(x_0, u(t), T) - f_i(x_0, u_k)\| \leq \frac{e_f}{L}(e^{LT} - 1), \quad (1)$$

where L is the Lipschitz constant of g in the region of interest, and e_f is given by

$$e_f = \max_{i \in I} \max_{x \in \hat{X}_i, u \in \mathcal{U}} \|g(x, u) - \hat{f}_i(x, u)\|,$$

where

$$\hat{X}_i = \{x | x = \phi(x', u(\tau), t), 0 \leq t \leq T, x' \in X_i, u(\tau) \in \mathcal{U} \; \forall \; 0 \leq \tau \leq t\}.$$

V. Examples

The following example illustrates the reason that the dynamic programming stage of the technique is useful. Consider a DSS $\mathcal{D}$, where $n = 2$ and the state space is partitioned as shown in Fig. 3. The target mode is labelled with a G and the dynamics for each mode are given by a through d, where

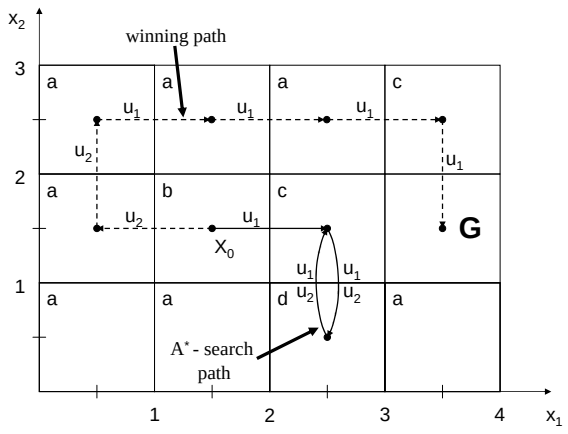


Fig. 3. Example illustrating the need for mode switching guidance.

$$\begin{aligned} A_a = A_b = A_c = A_d &= I \\ B_a &= I \\ B_b &= \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \\ B_c &= \begin{bmatrix} 0 & 0 \\ -1 & -1 \end{bmatrix} \\ B_d &= \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}. \end{aligned}$$

The inputs are $\mathcal{U} = \{u_1, u_2\}$, where $u_1 = (1, 0)$ and $u_2 = (0, 1)$, and the initial condition set is $X_0 = \{(1.5, 1.5)\}$.

From X_0, u_1 takes the system to $(2.5, 1.5)$ while u_2 takes the system to $(0.5, 1.5)$. The point $(2.5, 1.5)$ is selected to be expanded because it has a high merit value since it is heading directly towards the target mode. The point $(0.5, 1.5)$ has a low merit value because it is moving directly away from the goal mode. From $(2.5, 1.5)$ both inputs take the system to $(2.5, 0.5)$, which has a higher merit value than $(0.5, 1.5)$ since the former is moving away from the goal region at a slower rate than the latter. From $(2.5, 0.5)$ both inputs take the system back to $(2.5, 1.5)$. The search then continues to oscillate between $(2.5, 1.5)$ and $(2.5, 0.5)$ indefinitely.

This search, which is performed without the benefit of the dynamic programming solution, does not discover that the target mode can be reached from point $(0.5, 1.5)$ by applying the input sequence $(u_2, u_1, u_1, u_1, u_1)$. Dynamic programming finds the mode switching sequence that drives DSS $\mathcal{D}$ from the initial mode to the target mode.

As an application, we introduce a conventionally steered vehicle as an example. Fig. 4 shows the configuration of the vehicle. The variables x_1 and x_2 describe the coordinates of the rear axis of the vehicle in an inertial frame (X_1, X_2) and θ is the heading angle measured with respect to the horizontal axis. The control inputs are given by the velocity v of the vehicle and by the steering angle φ. The constant

length of the car is L.

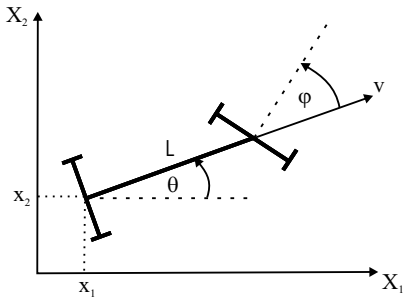


Fig. 4. Conventionally steered vehicle

The nonlinear continuous dynamics of the vehicle are

$$\begin{aligned} \dot{x}_1 &= v\,\cos\theta \\ \dot{x}_2 &= v\,\sin\theta \\ \dot{\theta} &= \frac{v\,\tan\theta}{L} \end{aligned} \tag{2}$$

Using the approximation technique described in Sec. IV, the switched-mode system representation of the conventionally steered vehicle is obtained by partitioning the 3-dimensional state space into boxes of side-length 0.4 in each direction and identifying each partition element with a discrete mode. The dynamics are converted to discrete-time using a sampling period of 0.1 seconds.

The input set is given by $\{(v,\varphi)|(v,\varphi) \in \{0.1, 0.6, 1.0\} \times \{-\frac{3}{8}\pi, -\frac{3}{4}\pi, 0, \frac{3}{4}\pi, \frac{3}{8}\pi\}\}$, and the maximum error due to the linearization is bound by $e^* = 0.0102$. The input added in order to account for the error is given by $V = \mathcal{E}((0,0), e^{*2} I)$.

In the first example we consider the situation shown in Fig. 5. The initial set is $X_0 = \mathcal{E}((0,0,0), 10^{-4} I)$ and the goal (G) and fail (F) regions are hyperboxes as shown in Figs. 5-(a) and 5-(c). The mode size is 0.4 in each direction and the origin of the the state space partition is $(-1,-1,-1)$. The region $[-1\ 1.8] \times [-1\ 1.8] \times [-1\ 2.2]$ is considered, which corresponds to 392 partition elements, resulting in a DPG $\mathcal{G}$ with 392 vertices.

Figs. 5-(a) and 5-(b) show the complete DSSA which was created during the Search Procedure until a good path to the goal region was found. The mode-switching sequence proposed by the DPS is indicated by the arrows. The first subgoal lies to the right of the $x_1 = 0.2$ plane. The search is guided successfully towards the first subgoal, and switches into the mode that lies on the other side of the $x_1 = 0.2$ plane (see Fig. 5-(a)). After that, the DPG proposes proceeding along the θ direction before entering the goal region by crossing the plane $x_2 = 0.2$.

The fact that no ellipsoid intersects the plane $x_1 = 0.2$ indicates that, the *prop* procedure (see Fig. 2) was successful in identifying an input sequence such that the reachable set landed completely inside the second mode of the given mode sequence. Note that it is not always possible to identify such an input sequence, in which case *prop* returns a set of ellipsoids that intersect more than one mode, as in the case shown in Fig. 5-(a) where some ellipsoids intersect modes on either side of the $x_2 = 0.2$ plane.

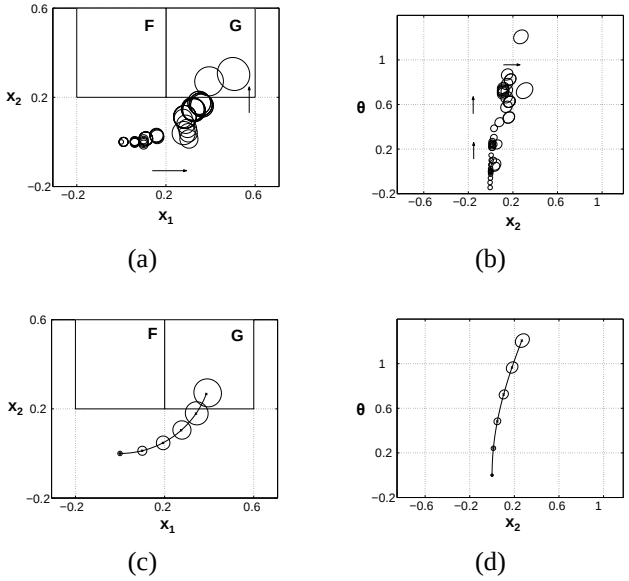


Fig. 5. First conventionally steered vehicle example synthesis procedure results: (a and b) all ellipsoidal regions that were searched projected onto (a) the first and second dimensions and (b) the second and third dimensions; (c and d) the reachable set of points along the winning path and a simulation trace of the winning path from the center of the initial condition set projected onto (c) the first and second dimensions and (d) the second and third dimensions.

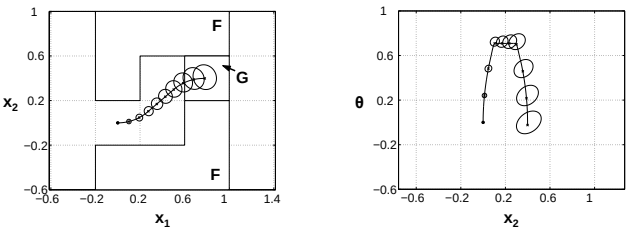
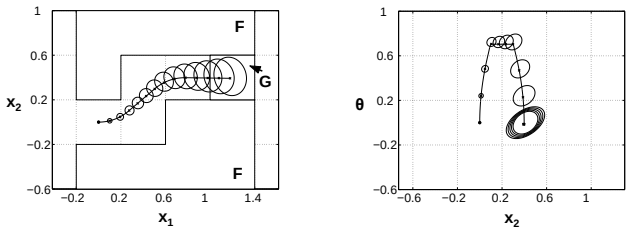


Fig. 6. Second conventionally steered vehicle example synthesis procedure results: (a and b) the reachable set of points along the winning path and a simulation trace of the winning path from the center of the initial condition set projected onto (a) the first and second dimensions and (b) the second and third dimensions.

Figs. 5-(c) and 5-(d) show the reachable set given by the input sequence found using the Search Procedure along with a simulation of the original continuous-time nonlinear system using the same input sequence starting from the center of the initial condition set. The points represent the state of the nonlinear system at the sample instants.

For the second example, shown in Fig. 6, the goal region is placed further away from the initial region and failure regions are placed such that the vehicle must move around them to reach the goal.

The third example, shown in Fig. 7, extends the second example in that the goal region is now even further away from the initial region.



(a) (b)

Fig. 7. Third conventionally steered vehicle example synthesis procedure results: (a and b) the reachable set of points along the winning path and a simulation trace of the winning path from the center of the initial condition set projected onto (a) the first and second dimensions and (b) the second and third dimensions.

Table I provides the results of all of the experiments. The experiments were performed on a Pentium 4, 2.8 GHz machine, with 512 Meg of RAM, running Windows XP. All times are given in seconds.

TABLE I

RESULTS OF SYNTHESIS FOR CAR EXAMPLE

convt. vehicle example no.	total ellipsoids searched	length of solution	DPG comp. time	search time	total comp. time
1	53	5	68.6	24.6	93.2
2	82	9	84.0	61.5	145.5
3	144	13	90.1	112.0	202.1

An alternative approach to the problem considered in this paper, developed by Morari et al., formulates the system as a mixed logical dynamic (MLD) system [3]. The MLD and a quadratic cost function are used to formulate an MIQP, which is then solved using numerical techniques.

The MIQP technique was implemented for the first case of the conventionally steered vehicle example (the case shown in Fig. 5). Memory limitations prevented using the full 392 mode model due to the number of variables and the number of constraints in the resulting MIQP problem. Instead the smaller state space region $[-0.2\ 1.4]\times[-0.2\ 1.4]\times[-0.2\ 1.4]$, which corresponds to 64 modes, was considered. In each mode, the piecewise affine approximation of the system was computed by linearizing about the center of the mode and the input point $u = (1, \frac{\pi}{4})$. The resulting MIQP problem had 2406 variables, 1242 of which were discrete, and 12885 constraints.

The CPLEX MIQP solver in the TOMLAB optimization environment was used to perform the optimization [7], [8]. An optimal solution was found in 282.7 seconds.

Our synthesis technique is able to compute a solution in less time (see Table I) due to the heuristics that we use, which exploit the nature of the problem (e.g., the smoothness of the vector field, the grid-like, switched-mode geometry of the system). It should be noted that there have been recent results in improving the efficiency of MIQP based techniques for solving hybrid system synthesis problems by taking advantage of the problem structure [2].

There are differences in the nature of the solutions found using our technique and the MIQP technique. The solution found using the MIQP technique enters the target region and avoids the fail region given one initial condition, while our solution is valid for a range of initial conditions. Furthermore, the MIQP solution does not account for the error incurred due to the approximation of the vector field. If applied to the original system, the input sequence found using the MIQP technique is not guaranteed to avoid the fail region or to drive the system into the target region. This is in contrast to our technique, which accounts for approximation error and which can account for uncertainty in the input.

VI. DISCUSSION

We have demonstrated a new technique for synthesizing safe input sequences for hybrid systems, which uses a combination of dynamic programming and informed search.

An issue that we are currently addressing deals with the $prop$ function. The complexity of the search procedure increases when the $prop$ function is unable to identify control sequences that propagate a reachable set of points completely across a switching surface. When this happens, the number of ellipsoids that must be propagated to continue the search increases. We are currently developing more efficient methods for propagating reachable regions across switching surfaces.

REFERENCES

[1] R. Bellman and R. Kalaba. *Dynamic Programming and Modern Control Theory*. Academic Press, 1965.

[2] A. Bemporad and N. Giorgetti. A sat-based hybrid solver for optimal control of hybrid systems. In *Hybrid Systems: Computation and Control: 7th International Workshop*, pages 126 – 141. Springer-Verlag Heidelberg, 2004.

[3] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, pages 407–427, 1999.

[4] S. Boyd, L.E. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *SIAM Studies in Applied Mathematics*. SIAM, 1994.

[5] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Jounral of the ACM*, 32(3):505–536, 1985.

[6] H. Farreny. A generalization for heuristically-ordered search: algorithms ρ, results about termination and admissibility. In *IEEE Conference on Systems, Man, and Cybernetics*, pages 1442–1447, 1996.

[7] K. Holmström. Practical optimization with the tomlab environment in matlab. In *42nd SIMS Conference*, Porsgrunn, Norway, 2001.

[8] K. Holmström, A. Göran, and M. Edvall. Tomlab /cplex v9.0 users guide. http://www.tomlab.biz/docs/TOMLAB_ CPLEX.pdf, 2004.

[9] J.H. Hubbard and B.H. West. *Differential Equations*. Springer-Verlag, 1991.

[10] A. B. Kurzhanski and I. Vályi. *Ellipsoidal Calculus for Estimation and Control*. Birkhä user, Boston, 1997.

[11] R. Larson. Dynamic programming with reduced computational requirements. *IEEE Transactions on Automatic Control*, 10(2):135–143, April 1965.

[12] E. Lawler and D. Wood. Branch-and-bound methods: a survey. *Operations Research*, pages 699–719, 1966.

[13] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2003.

[14] C. Tomlin, I. Mitchell, A. Bayen, and M. Oishi. Computational techniques for the verification and control of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, July 2003.