Overapproximating Reachable Sets by Hamilton-Jacobi Projections*

Ian M. Mitchell† and Claire J. Tomlin‡

September 23, 2002

Submitted to the Journal of Scientific Computing, Special Issue dedicated to Stanley Osher on the occasion of this 60th birthday.

Abstract

In earlier work, we showed that the set of states which can reach a target set of a continuous dynamic game is the zero sublevel set of the viscosity solution of a time dependent Hamilton-Jacobi-Isaacs (HJI) partial differential equation (PDE). We have developed a numerical tool—based on the level set methods of Osher and Sethian—for computing these sets, and we can accurately calculate them for a range of continuous and hybrid systems in which control inputs are pitted against disturbance inputs. The cost of our algorithm, like that of all convergent numerical schemes, increases exponentially with the dimension of the state space. In this paper, we devise and implement a method that projects the true reachable set of a high dimensional system into a collection of lower dimensional subspaces where computation is less expensive. We formulate a method to evolve the lower dimensional reachable sets such that they are each an overapproximation of the full reachable set, and thus their intersection will also be an overapproximation of the reachable set. The method uses a lower dimensional HJI PDE for each projection with a set of disturbance inputs augmented with the unmodeled dimensions of that projection's subspace. We illustrate our method on two examples in three dimensions using two dimensional projections, and we discuss issues related to the selection of appropriate projection subspaces.

keywords: reachability, Hamilton-Jacobi equation, projection, verification

*Research supported by DARPA under the Software Enabled Control Program (AFRL contract F33615-99-C-3014, and Boeing contract Z20040), and by ONR under MURI contract N00014-02-1-0720.

†Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, California, 94305; mitchell@sccm.stanford.edu

‡Department of Aeronautics and Astronautics, Stanford University, Stanford, California, 94305; tomlin@stanford.edu

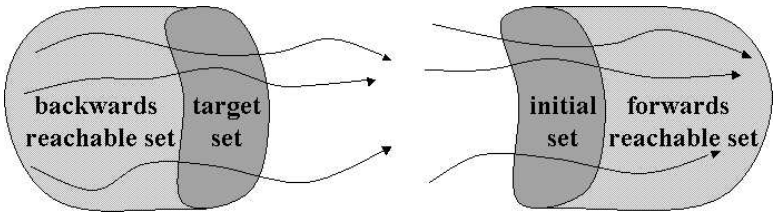

Figure 1: Difference between backwards and forwards reachability.

1 Introduction

Of fundamental importance in the safety verification of embedded control systems is the ability to compute reachable sets of states. If one can accurately determine the set of states from which a system may inadvertently step or be pushed into an unsafe configuration, then system safety can be verified by ensuring that the system state remains outside of this set. Consider, for example, the automatic control laws proposed for ensuring separation between aircraft in civilian air traffic control [1], in which unsafe configurations are those in which the distance between any pair of vehicles is less than a required minimum. We would like to guarantee that these control laws never allow two aircraft into a situation where loss of separation is inevitable, and reachable set analysis is an appropriate tool for this task.

There are two basic types of reachable sets, depending on whether an initial or a final condition is specified. For a forwards reachable set, we specify the initial conditions and seek to determine the set of all states that can be reached along trajectories that start in that set. Conversely, for a backwards reachable set we specify a final or target set of states and seek to determine the set of states from which trajectories start that can reach this target set (see figure 1). In this paper, we will primarily discuss backwards reachable sets. For autonomous systems with no inputs the two computations may be used interchangeably, but it is an as yet unresolved question how the computation of the two reachable sets compares for general continuous dynamical control systems.

Computing reachability for safety specifications has been studied in the control and computer aided verification communities for many years. While efficient algorithms have been designed for reachability computation in discrete state spaces [2], the computation of reachable sets for continuous systems whose state dimension exceeds four or five remains an open problem. In our

previous work [3, 4, 5], we have developed a method for computing backwards reachable sets based on level set techniques [6, 7, 8] and viscosity solutions [9, 10], using the ideas presented in [11, 12]. We allow for both control and disturbance inputs in our problem formulation, we represent a reachable set as the zero sublevel set of an appropriate function, and the boundary of this set is propagated under a nonlinear flow field using a validated numerical approximation of a time dependent Hamilton-Jacobi-Isaacs (HJI) partial differential equation (PDE). Our numerical methods compare favorably in efficiency and accuracy to other methods based on solutions to static Hamilton-Jacobi equations [13, 14] and to techniques from viability theory [15].

Unfortunately, while these methods can find accurate approximations to the reachable set for systems with complicated nonlinear dynamics, their computational cost scales exponentially with the system's state space dimension. A number of more efficient methods for computing reachable sets have been developed [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26], but to achieve their efficiency they are forced to specialize in the types of dynamics and/or shapes of reachable sets with which they can operate.

In this paper, we present our own attempt to tackle the "curse of dimensionality". Instead of computing the true reachable set in the system's full state space, we work in a collection of lower dimensional subspaces to compute an overapproximation. We present the technique in the context of our time dependent Hamilton-Jacobi formulation of reachability, but it could be applied to any of the reachability methods mentioned above that can handle systems with disturbance inputs. The basic idea is simple: in each projection we calculate the reachable set assuming that the projection's unmodeled dimensions are added to the collection of disturbance inputs. We conjecture that each lower dimensional reachable set is provably an overapproximation of the projection of the true reachable set, so that the intersection of the back projections of the lower dimensional sets will also be an overapproximation. As such, we gain significant computational savings for high dimensional systems, at the expense of overapproximation. However, because we are interested in verifying system safety, computing an overapproximation of the set of states which evolve into an unsafe set and then proving that the system never enters that overapproximation is sufficient.

Our work is inspired by the ideas of [27, 28] for continuous systems and those of [29, 30, 31] for discrete systems, as well as research which uses intersections and projections to treat curves [32] and geometric optics [33, 34].

The outline of this paper is as follows. In section 2, we define the reachability problem and its solution, and we present an example derived from aircraft collision avoidance. In section 3, we describe our reachability technique based on projections; for clarity, we explain the method with a simple linear rotation example in three dimensions using projections into two dimensional subspaces. The method is then demonstrated on the aircraft collision avoidance example, for which significant computation time savings is demonstrated. We conclude the paper with a discussion of issues that remain to be investigated.

2 Reachable Sets

In this section we define backwards reachable sets, explain how they can be represented as the solution of an HJI PDE, and review computational techniques for their approximation. Our methods will be applicable to linear and nonlinear systems, whose dynamics are modeled by differential equations depending on control and disturbance parameters. For the kinds of practical systems in which we are interested, the controls represent parameters which may be manipulated to force the system to satisfy a property or achieve a goal, while the disturbances represent uncertainties in the system, environmental disturbances or unknown actions of a component or subsystem. We will be interested in formulating control strategies which will achieve the goal, despite the worst possible disturbance action. Hence, we use the framework of optimal control and dynamic games. We will first introduce an example of a system for which we would like to compute a reachable set, in order to make the subsequent informal discussion more concrete. A formal presentation of reachable set computation can be found in [3].

Throughout the discussion that follows, we use the notation x_i to refer to the i^{th} component of the vector x.

2.1 Collision Avoidance Example

As our demonstration example we will adopt a classical pursuit evasion game involving two identical vehicles moving in the plane (see [35, 36] for more details). If the vehicles get too close together, a collision occurs. One of the vehicles (the *pursuer*) wants to cause a collision, while the other (the *evader*) wants to avoid one. Each vehicle has a three dimensional state consisting of a location in the plane and a heading. The model for an individual vehicle

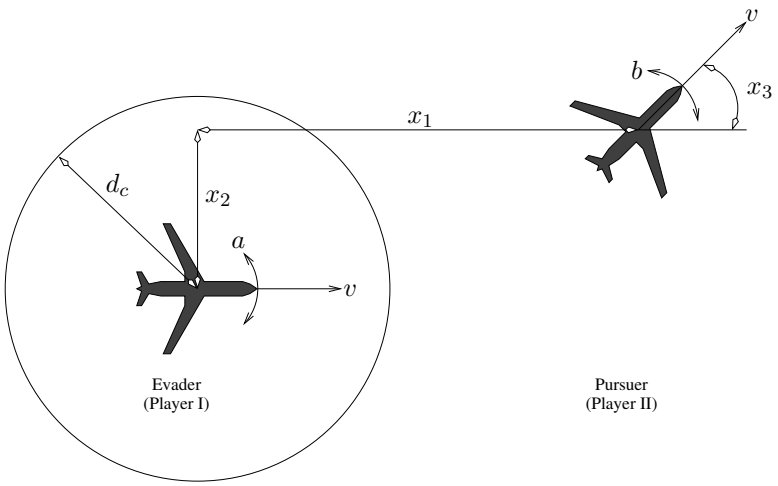

Figure 2: Coordinate system for the collision avoidance example.

is

$$\frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} v\cos z_3 \\ v\sin z_3 \\ \omega \end{bmatrix}, \tag{1}$$

where $\begin{bmatrix} z_1 & z_2 \end{bmatrix}^T \in \mathbb{R}^2$ is the vehicle's location in the plane, $z_3 \in [0, 2\pi]$ is its heading, $v \geq 0$ is its linear velocity and ω is its angular velocity. We draw our vehicles as airplanes, although as a simple kinematic model, (1) is equally applicable to a car or even a unicycle. We have used the solution to this example as inspiration for verifying two-aircraft tactical conflict avoidance strategies in air traffic control, in which the logic within each aircraft may be uncertain about the possible actions of the other aircraft [1].

In order for the two vehicles to pursue their respective goals, they must be able to affect their vehicles' dynamic evolution in some manner. We use the term *inputs* to refer to the free parameters in a system's ODE that can be modified to achieve some goal. For this particular game we allow each player to choose an angular velocity $\omega \in [-1, +1]$. To distinguish between the two players' inputs, we replace the variable ω by $a \in \mathcal{A} = [-1, +1]$ for the evader's input angular velocity and by $b \in \mathcal{B} = [-1, +1]$ for the pursuer's input angular velocity. The remaining parameters are fixed; for the visualizations shown below their values are linear velocity $v = 5$ and collision distance $d_c = 5$.

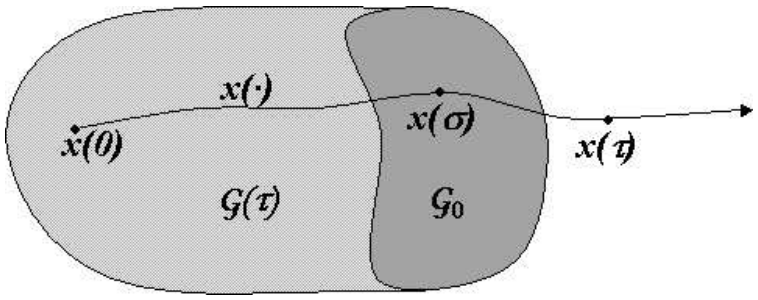

Figure 3: Notation for system trajectories and the backwards reachable set.

Because we are not interested in the absolute location of a collision, but only in whether or not one will occur, we study the problem in relative coordinates (see figure 2). Fixing the evader at the planar origin and facing to the right, the model becomes:

$$\dot{x} = \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -v + v\cos x_3 + ax_2 \\ v\sin x_3 - ax_1 \\ b - a \end{bmatrix} = f(x, a, b), \tag{2}$$

where the three state dimensions are relative planar location $\begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2$ and relative heading $x_3 \in [0, 2\pi]$. A collision occurs if $\sqrt{x_1^2 + x_2^2} \leq d_c$ for any value of x_3—in $\mathbb{R}^3$ this collision set is a cylinder of radius d_c centered on the x_3 axis. To solve this pursuit evasion game, we would like to determine the set of initial states from which the pursuer can cause a collision despite the best efforts of the evader.

2.2 Defining the Reachable Set

As was discussed previously, the backwards reachable set is the set of initial conditions giving rise to trajectories that lead to some target set. More formally, let $\mathcal{G}_0$ be the target set, $\mathcal{G}(\tau)$ be the backwards reachable set over finite horizon $\tau < \infty$, $x(\cdot)$ denote a trajectory of the system, and $x(\tau)$ be the state of that trajectory at time τ. Then $\mathcal{G}(\tau)$ is the set of $x(0)$ such that $x(\sigma) \in \mathcal{G}_0$ for some $\sigma \in [0, \tau]$ (see figure 3).

We partition any input parameters in the system's dynamics into two subsets: those inputs trying to reach the target set, and those inputs trying to avoid it. The names we give these two subsets are based on the fact that in

our examples the target set is usually a set of dangerous states. The *control* inputs are those whose values we can choose to drive the system away from the target set, while the *disturbance* inputs are those we conservatively assume will take on a worst case value and drive the system toward the target set. Control inputs will be designated by a (such as the evader's input) and disturbance inputs will be designated by b (such as the pursuer's input).

The choice of input values over time influences how a trajectory $x(\cdot)$ evolves. For systems with inputs, the backwards reachable set $\mathcal{G}(\tau)$ is the set of $x(0)$ such that for every possible control input a there exists a disturbance input b that results in $x(\sigma) \in \mathcal{G}_0$ for some $\sigma \in [0, \tau]$[1]

The solution to the pursuit evasion game described in section 2.1 is a backwards reachable set. Let the target set be the collision set

$$\mathcal{G}_0 = \left\{ x \in \mathbb{R}^3 | \sqrt{x_1^2 + x_2^2} \leq d_c \right\}. \tag{3}$$

Then $\mathcal{G}(\tau)$ is the set of initial configurations such that for any possible control input chosen by the evader, the pursuer can generate a disturbance input that leads to a collision within τ time units.

2.3 Computing the Reachable Set

Analytic determination of a reachable set is only possible in rare instances; consequently, we have developed a numerical method to find these sets. We have chosen to use the very general implicit surface function representation for our reachable sets. To demonstrate this representation, consider the cylindrical target set (3) for the collision avoidance example. We represent this set as the zero sublevel set of a scalar function $\phi_0(x)$ defined over the state space

$$\phi_0(x) = \sqrt{x_1^2 + x_2^2} - d_c,$$
$$\mathcal{G}_0 = \left\{ x \in \mathbb{R}^3 | \phi_0(x) \leq 0 \right\}.$$

In words, a point x is inside $\mathcal{G}_0$ if $\phi_0(x)$ is negative, outside $\mathcal{G}_0$ if $\phi_0(x)$ is positive, and on the boundary of $\mathcal{G}_0$ if $\phi_0(x) = 0$.

Let the backwards reachable set $\mathcal{G}(\tau) = S_\tau(\mathcal{G}(0)) = S_\tau(\mathcal{G}_0)$, where the $S_\tau(\cdot)$ operator computes the backwards reachable set of its set valued ar-

[1] The control and disturbance inputs are technically signals over time, but here we refer interchangeably to the signal over time and its instantaneous value. A formal discussion of the admissible non-anticipative input sets and strategies is provided in [3].

gument over time τ. In [3] we proved that $S_\tau(\cdot)$ can be accomplished on sets represented by an implicit surface function by solving the modified HJI PDE

$$\frac{\partial \phi(x,t)}{\partial t} + \min\left[0, H(x, \nabla\phi(x,t))\right] = 0, \tag{4}$$

with $t = -\tau$, Hamiltonian

$$H(x,p) = \max_{a\in\mathcal{A}} \min_{b\in\mathcal{B}} p \cdot f(x,a,b), \tag{5}$$

and terminal conditions

$$\phi(x,0) = \phi_0(x). \tag{6}$$

If $\mathcal{G}_0$ is the zero sublevel set of $\phi_0(x)$, then the zero sublevel set of the viscosity solution $\phi(x,t)$ to (4)–(6) specifies the backwards reachable set as

$$\mathcal{G}(\tau) = \left\{x \in \mathbb{R}^3 | \phi(x, -\tau) \leq 0\right\}. \tag{7}$$

Notice that (4) is solved from time $t = 0$ backwards to some $t = -\tau \leq 0$.

There are several interesting points to make about the HJI PDE (4)–(6). First, the $\min[0, H]$ formulation in (4) ensures that the reachable set only grows as τ increases; thus, states labeled unsafe cannot become safe at some later time. Second, the "max min" operation in the Hamiltonian (5) may give a slight advantage to the disturbance, since it chooses a value second and hence may observe the action of the control. In the examples presented here the inputs are independent, but this choice of order is conservative in those cases where order matters. Third, we show in [3] that the viscosity solution is the correct weak solution of (4)–(6) to generate an implicit surface representation of the reachable set. Therefore, we can draw on the well developed numerical schemes of the level set literature to compute accurate approximations of $\phi(x,t)$. Finally, the solution $\phi(x,t)$ can be used to create a *weakest safe controller*: if the state is outside the reachable set, any control policy for input a is safe, but on the boundary the control must choose the optimal a from (5) to ensure that the system remains outside the reachable set and hence to guarantee safety. In practice, we gradually introduce input constraints as the system approaches the boundary to avoid a chattering controller, using the distance and gradient information in $\phi(x,t)$.

To compute numerical approximations of the viscosity solution to (4)–(6), we have developed a C++ implementation based on high resolution level set methods (an excellent introduction to these schemes can be found in [8]). To approximate $\nabla\phi(x,t)$ we rely primarily on a fifth order accurate weighted,

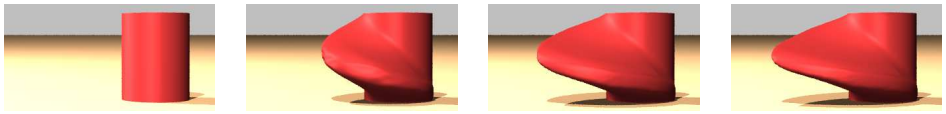

Figure 4: Growth of the reachable set (animation at [42]).

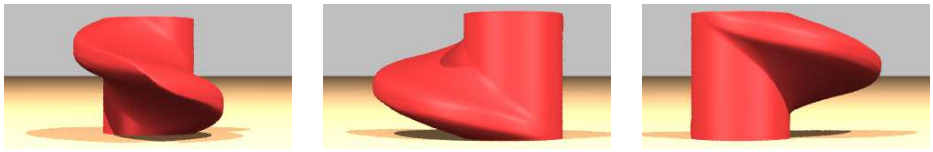

Figure 5: Other views of the reachable set (animation at [42].)

essentially non-oscillatory (WENO) stencil [37, 7], although we have also implemented a basic first order scheme for speed [6, 38]. While upwinding would be the least dissipative way to numerically approximate the Hamiltonian (5), the optimizations over a and b make it difficult to implement. Instead, we use the well studied Lax-Friedrichs (LF) approximation [39, 40]. We have considered the Local Lax-Friedrichs and Roe with entropy fix numerical approximations of the Hamiltonian [7], but neither demonstrated a significant reduction in dissipation for our problems. We suspect that regular reinitialization of the level set function and the switching nature of the optimal inputs a and b in (5) effectively reduces these more involved approximations to the basic LF approximation in those regions where dissipation must be introduced for stability (near shocks), while away from these regions none of the schemes introduce significant dissipation. Finally, we treat the time derivative in (4) with the method of lines and a second order total variation diminishing (TVD) Runge-Kutta scheme [41]. Although TVD schemes of higher order are available, we found this one to be sufficiently accurate for our purposes.

2.4 Collision Avoidance Example Results

We can apply this code to the collision avoidance problem. In figure 4, the collision cylinder/target set $\mathcal{G}_0$ for the example appears on the far left; the remaining images show how $\mathcal{G}(\tau)$ grows as τ increases from zero. For the parameters chosen in section 2.1, the reachable set converges to a fixed point

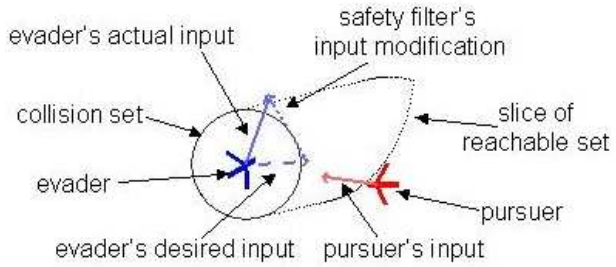

Figure 6: Annotated frame from collision avoidance example animation.

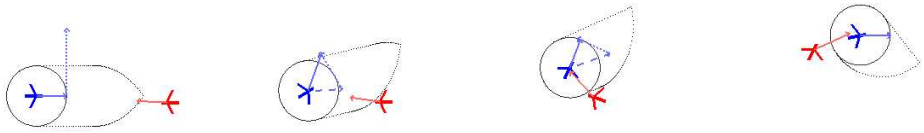

Figure 7: Evader keeps pursuer from entering reachable set, and hence avoids collision (animation at [42]).

for $\tau \gtrsim 2.6$. Figure 5 shows several views of this fixed point. Should the pursuer start anywhere within this reachable set, it can cause a collision by choosing an appropriate input b, no matter what input a the evader might choose. Conversely, if the pursuer starts outside this reachable set, then there exists an input a that the evader can choose that will avoid a collision no matter what input b the pursuer might choose.

We can build some intuition for the shape of the reachable set in figure 5 by considering a few horizontal slices through it. The relative heading coordinate x_3 is the vertical coordinate in these images. The largest horizontal slice of the reachable set lies at the vertical midpoint $x_3 = \pi$, which is when the two aircraft are flying in opposite directions. The horizontal slice at top $x_3 = 0$ or bottom $x_3 = 2\pi$ (which are equivalent) represents the case in which the aircraft are flying in the same direction; this slice is nothing more than the initial collision set.

Figure 6 shows an annotated frame from an animation of the collision system, and a series of frames from that animation are shown in figure 7. The evader starts on the left surrounded by the solid collision circle, while the pursuer starts on the right. The dotted shape surrounding the evader is the

Figure 8: Pursuer starts within the reachable set, and can thus cause a collision despite the best efforts of the evader (animation at [42]).

slice of the reachable set for the current relative heading of the two vehicles; for example, in the leftmost figure the vehicles have relative heading $x_3 \approx \pi$ and so the horizontal midplane slice of the reachable set is shown. By choosing an appropriate input, the evader keeps the pursuer from entering the reachable set and thus from causing a collision as time progresses from left to right. Figure 8 shows a sequence in which the pursuer starts within the reachable set and causes a collision.

The numerical techniques described here can be applied to general asymmetric versions of this game; for example, cases in which the two vehicles' linear velocities and/or their range of angular velocities are not identical. For the special case of identical vehicles examined above, we can find an analytic solution for points on the boundary of the reachable set. We have used this solution to show convergence of our numerical approximation as the computational grid is refined, and thus validate our implementation. For more details, see [36].

3 Reachability Computation in Projections

The Hamilton-Jacobi-Isaacs formulation and level set solution described in the previous section provide a computationally elegant method to determine the set of reachable states of a continuous dynamic game. The main problem with this method is the expense of computing the full reachable set. To reduce this cost, we wish to represent a high dimensional reachable set as the intersection of a collection of lower dimensional reachable sets. If we can formulate a way to evolve the lower dimensional reachable sets—called the projections—such that they are each an overapproximation of the full reachable set, then their intersection will also be an overapproximation. The key is to evolve the projections without referring explicitly to the full dimension reachable set. It turns out that the HJI formulation provides this

for free: in any projection, we simply augment the space of disturbances with the unmodeled dimensions and form a new HJI PDE in a lower dimensional space.

Throughout the remainder of the paper, we will consider for clarity the specific case in which the true reachable set is of dimension three, and we will work with a set of projections in two dimensional spaces spanned by subsets of the coordinate axes. The generalization both to higher dimension, as well as to projections of different dimension, is not theoretically difficult, yet issues regarding the selection of projective subspaces are important, and will be discussed following the presentation of an example.

3.1 Subspaces and Projections

We consider as state space $\mathbb{R}^3$ spanned by its coordinate axes e_1, e_2 and e_3. Let $\mathbb{Y}_i$ be the subspace spanned by coordinate axis e_i, and $\mathbb{Y}_{ij}$ the subspace spanned by coordinate axes e_i and e_j. Note that $\mathbb{Y}_{123} = \mathbb{R}^3$.

Define the *projection operators*:

- $\mathfrak{p}_i[x]$, which projects a point $x \in \mathbb{R}^3$ into the subspace $\mathbb{Y}_i$, defined as:

$$\mathfrak{p}_i[x] = x_i.$$

- $\mathfrak{p}_{ij}[x]$, which projects a point $x \in \mathbb{R}^3$ into the subspace $\mathbb{Y}_{ij}$, defined as:

$$\mathfrak{p}_{ij}[x] = \begin{bmatrix} x_i \\ x_j \end{bmatrix}.$$

  We will sometimes write the pair $\begin{bmatrix} x_i & x_j \end{bmatrix}^T$ as x_{ij}.

- $\mathfrak{p}_{ij}^{-1}[y_{ij}]$, which represents the back projection of the point $y_{ij} \in \mathbb{Y}_{ij}$ into $\mathbb{R}^3$, defined as:

$$\mathfrak{p}_{ij}^{-1}[y_{ij}] = \{x \in \mathbb{R}^3 | \mathfrak{p}_{ij}[x] = y_{ij}\}.$$

  Note that $\mathfrak{p}_{ij}^{-1}[y_{ij}]$ is a subset of $\mathbb{R}^3$.

We will sometimes abuse notation by applying these operators to sets instead of points. For example, if $\mathcal{X} \subset \mathbb{R}^3$, then the projection of $\mathcal{X}$ into $\mathbb{Y}_{ij}$ is represented as

$$\mathfrak{p}_{ij}[\mathcal{X}] = \{y_{ij} \in \mathbb{Y}_{ij} | \exists x \in \mathcal{X} \text{ with } \mathfrak{p}_{ij}[x] = y_{ij}\}.$$

As defined in (7), we represent the true, full dimensional reachable set $\mathcal{G}(t)$ as the zero sublevel set of the scalar function $\phi(x,t)$. In subsequent discussions we will have reason to refer to sublevel sets other than the zero sublevel set. In those cases we use a superscript to denote the particular sublevel set in which we are interested—for some constant $d \in \mathbb{R}$,

$$\mathcal{G}^d(t) = \left\{x \in \mathbb{R}^3 | \phi(x,t) \leq d\right\}.$$

The projections' reachable sets are represented by implicit surface functions defined in their respective subspaces

$$\mathcal{Y}_{ij}(t) = \{y_{ij} \in \mathbb{Y}_{ij} | \phi_{ij}(y_{ij},t) \leq 0\},$$

where $\phi_{ij} : \mathbb{Y}_{ij} \times \mathbb{R} \to \mathbb{R}$. The intersection of the projections is given by

$$\begin{aligned}\mathcal{X}(t) &= \bigcap_{i=1}^{3} \bigcap_{j=i+1}^{3} \mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(t)\right] \\ &= \{x \in \mathbb{X} | \mathfrak{p}_{ij}[x] \in \mathcal{Y}_{ij}(t) \text{ for } i,j \in \{1,2,3\}, j > i\}, \\ &= \{x \in \mathbb{X} | \phi_{ij}(\mathfrak{p}_{ij}[x], t) \leq 0 \text{ for } i,j \in \{1,2,3\}, j > i\}.\end{aligned} \tag{8}$$

Notice that $\mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}(t)\right]$ will be a prism in $\mathbb{R}^3$ whose cross section is $\mathcal{Y}_{ij}(t)$; for example, $\mathfrak{p}_{12}^{-1}\left[\mathcal{Y}_{12}(t)\right]$ is a prism aligned with the e_3 axis whose cross section in the e_1-e_2 plane is $\mathcal{Y}_{12}(t)$. Therefore, $\mathcal{X}(t)$ from (8) is simply the intersection of three orthogonal prisms.

We overload the projection operators to apply them to implicit surface functions. First, define the *depth* of a point $y_{ij} \in \mathbb{Y}_{ij}$ as

$$D(y_{ij},t) = \min_{x \in \mathfrak{p}_{ij}^{-1}[y_{ij}]} \phi(x,t).$$

There are a number of possible ways to define a projection of the full dimensional function $\phi(x,t)$, but we will use the depth operator:

$$\mathfrak{p}_{ij}[\phi] : \mathbb{Y}_{ij} \times \mathbb{R} \to \mathbb{R}, \qquad \mathfrak{p}_{ij}[\phi](y_{ij},t) = D(y_{ij},t). \tag{9}$$

With this definition,

$$\mathcal{G}(t) = \left\{x \in \mathbb{R}^3 | \phi(x,t) \leq 0\right\} \implies \mathfrak{p}_{ij}[\mathcal{G}(t)] = \left\{y_{ij} \in \mathbb{Y}_{ij} | \mathfrak{p}_{ij}[\phi](y_{ij},t) \leq 0\right\}.$$

The inverse projection for the implicit surface function of a subspace is easier to define

$$\mathfrak{p}_{ij}^{-1}[\phi_{ij}] : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}, \qquad \mathfrak{p}_{ij}^{-1}[\phi_{ij}](x,t) = \phi_{ij}(\mathfrak{p}_{ij}[x], t). \tag{10}$$

Under this definition, $\mathfrak{p}_{ij}^{-1}[\phi_{ij}](x,t)$ is an implicit surface function in $\mathbb{R}^3$ for the prism $\mathfrak{p}_{ij}^{-1}\left[\mathcal{Y}_{ij}\right]$ aligned normal to the e_i-e_j plane whose cross section is $\mathcal{Y}_{ij}(t)$.

3.2 The Linear Rotation Example

To illustrate these definitions and the projection evolution procedure, we use a simple example involving purely rotational dynamics (about the e_3 axis) and no inputs. The dynamics are given by the linear rigid body rotation

$$\dot{x} = Ax = f(x), \tag{11}$$

with $x \in \mathbb{R}^3$ and $A \in \mathbb{R}^{3\times 3}$

$$A = \pi \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For this example, we will compute the forward evolution of the initial set under the rotation rather than a forwards or backwards reachable set, because it is easier to visualize the progress of this evolution and its projections. The entire region swept out by this evolution would be the forwards reachable set. If the initial set is represented implicitly by $\phi_0(x)$, we can compute the evolution of this initial set by solving a regular HJI PDE forward in time (note that $t \geq 0$ in this case)

$$\begin{aligned} \frac{\partial \phi(x,t)}{\partial t} + H(x, \nabla\phi(x,t)) &= 0, \\ \phi(x,0) &= \phi_0(x), \\ H(x,p) &= p \cdot f(x). \end{aligned} \tag{12}$$

The projection based overapproximation method outlined below will assume that $S_t(\cdot)$ set evolution is accomplished with the forward time PDE (12). The method can be directly adapted to the computation of regular reachable sets by instead using (4)–(6) for $S_\tau(\cdot)$ set evolution.

For the purposes of this example, let $\mathcal{G}_0$ be our initial set and $\mathcal{G}(t)$ be the same set rotated under (11) for time t (in the future we will call $\mathcal{G}(t)$ a reachable set, even though it is only a forward time evolution in this particular example). The dynamics are scaled such that $\mathcal{G}(2) = \mathcal{G}(0) = \mathcal{G}_0$. Ideally, we would like $\mathcal{G}_0$ to be a sphere of radius $r = 0.30$ centered at the point $c = \begin{bmatrix} 0.00 & 0.55 & 0.00 \end{bmatrix}^T$. Solving for the viscosity solution $\phi(x,t)$ of (12) with $f(x)$ from (11) and

$$\phi_0(x) = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + (x_3 - c_3)^2} - r \tag{13}$$

would generate an implicit surface representation of $\mathcal{G}(t)$, but would require solving (12) over three spatial dimensions. To reduce computational costs,

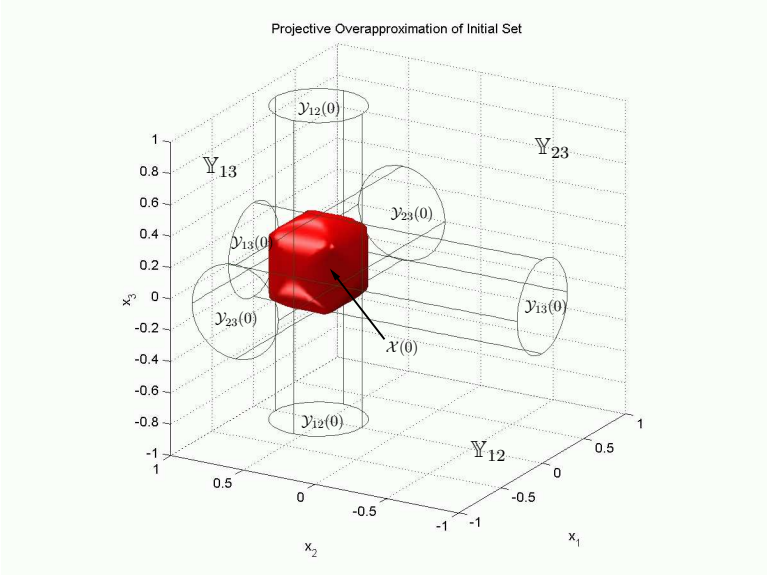

Figure 9: Initial projection sets.

we will instead seek a method of overapproximating $\mathcal{G}_0$ and $\mathcal{G}(t)$ that requires solving PDEs in only two spatial dimensions.

We work on three separate two dimensional projections into the subspaces $\mathbb{Y}_{12}$, $\mathbb{Y}_{13}$, and $\mathbb{Y}_{23}$. The corresponding reachable sets are $\mathcal{Y}_{12}(t)$, $\mathcal{Y}_{13}(t)$, and $\mathcal{Y}_{23}(t)$. The initial sets $\mathcal{Y}_{ij}(0)$ for each of these subspace reachability problems are constructed by projecting the full dimensional initial sphere $\mathcal{G}_0$ down into the subspace as $\mathcal{Y}_{ij}(0) = \mathfrak{p}_{ij}[\mathcal{G}_0]$. These $\mathcal{Y}_{ij}(0)$ and their intersection $\mathcal{X}(0)$ are shown in figure 9. Since $\mathcal{X}(0)$ is restricted by our projective geometry to be the intersection of three axis aligned prisms, it is unavoidably an overapproximation of the initial sphere $\mathcal{G}_0$.

3.3 Evolving a Projection

Our goal in this section is to develop an HJI PDE which can be applied in a lower dimensional subspace to evolve an overapproximative projection of the true reachable set, thus avoiding the need to solve an expensive full dimensional PDE. First, however, we look at how to evolve an overapproximative projection using a PDE defined over the full dimensional space.

Focus on a single projection whose index is ij, and denote the index of the unmodeled dimension as k. If $\mathcal{Y}_{ij}(t)$ is an overapproximative projection of $\mathcal{G}(t)$, then $\mathcal{G}(t) \subseteq \mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(t)]$. Conceptually, $\mathcal{Y}_{ij}(t)$ could be evolved by an inverse projection into $\mathbb{R}^3$, evolution by δt and projection back down into $\mathbb{Y}_{ij}$, written as

$$\mathcal{Y}_{ij}(t+\delta t) = \mathfrak{p}_{ij}\left[S_{\delta t}\left(\mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(t)]\right)\right] \tag{14}$$

Then

$$\begin{aligned}\mathcal{G}(t) \subseteq \mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(t)] &\implies S_{\delta t}(\mathcal{G}(t)) \subseteq S_{\delta t}\left(\mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(t)]\right), \\ &\implies \mathfrak{p}_{ij}[\mathcal{G}(t+\delta t)] \subseteq \mathcal{Y}_{ij}(t+\delta t).\end{aligned}$$

Consequently, we can ensure that $\mathcal{Y}_{ij}(t)$ remains an overapproximative projection of $\mathcal{G}(t)$ provided that we can perform the three steps of (14) on our implicit surface function representation $\phi_{ij}(x,t)$ of $\mathcal{Y}_{ij}(t)$. Projection is accomplished by (9) and inverse projection by (10). For this example $S_{\delta t}(\cdot)$ is accomplished in $\mathbb{R}^3$ by solving (12). Let

$$p(x) = \nabla\left(\mathfrak{p}_{ij}^{-1}[\phi_{ij}](x,t)\right),$$

and $p_i(x)$, $p_j(x)$ and $p_k(x)$ be its components. Since $\mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(t)]$ is a prism in $\mathbb{R}^3$, $p_k(x) = 0$ for all $x \in \mathbb{R}^3$; furthermore, $p_i(x)$ and $p_j(x)$ are independent of x_k. Examining the Hamiltonian of (12) more closely

$$\begin{aligned}H(x,p(x)) =& p(x)\cdot f(x), \\ =& p_i(x_i,x_j,x_k)f_i(x_i,x_j,x_k) + p_j(x_i,x_j,x_k)f_j(x_i,x_j,x_k) \\ &+ p_k(x_i,x_j,x_k)f_k(x_i,x_j,x_k), \\ =& p_i(x_i,x_j)f_i(x_i,x_j,x_k) + p_j(x_i,x_j)f_j(x_i,x_j,x_k).\end{aligned}$$

Thus, the only dependence of the Hamiltonian (and thus the time evolution in general) on dimension k is through the x_k dependence in f_i and f_j. Geometrically, this dependence will manifest itself as a rotation of the prism $\mathfrak{p}_{ij}^{-1}[\mathcal{Y}_{ij}(t)]$ so that it is no longer parallel to e_k. When this rotated prism is projected back down into $\mathbb{Y}_{ij}$, the projection's boundary will be determined by those parts of the prism that rotated the most. Maximum rotation occurs where the flow field is most closely aligned with the outward normal of the initial prism—precisely those states x where $p(x)\cdot f(x)$ is minimized (the gradient $p(x)$ points in the direction of the inward normal).

From this argument, we deduce that using the modified Hamiltonian

$$H'(x,p(x)) = \min_{x_k} p_i(x_i,x_j)f_i(x_i,x_j,x_k) + p_j(x_i,x_j)f_j(x_i,x_j,x_k) \tag{15}$$

in (12) for all $x \in \mathbb{R}^3$ will not modify the projection into $\mathbb{Y}_{ij}$ of the time evolved prism. Although the time evolved prism itself would not be the same, in the end we are only concerned with its projection.

The only reason that one would have to work with the projective overapproximation in $\mathbb{R}^3$ would be the dependence of the time evolution operation $S_{\delta t}(\cdot)$ on the missing dimension x_k. After substituting the Hamiltonian (15) into the evolution PDE (12), $S_{\delta t}(\cdot)$ no longer has any dependence on x_k, and we can therefore work entirely in the lower dimensional $\mathbb{Y}_{ij}$.

The final concern is how to bound the range of x_k when minimizing in (15). We know that $x_k \in \mathbb{Y}_k$, but minimizing over such an unbounded range could lead to a negative value of arbitrarily large magnitude for (15). Fortunately, we have access to some sets within which all feasible reachable states should lie. If it were available, $\mathcal{G}(t)$ would provide a tight bound on possible values of x_k. In practice, we will have to settle for the overapproximation $\mathcal{X}(t)$; however, expanding the interval of feasible x_k by using $\mathcal{X}(t)$ instead of $\mathcal{G}(t)$ can only cause the Hamiltonian (15) to be more negative and hence $\mathcal{Y}_{ij}(t)$ to grow more than necessary during the time evolution step. Since $Y_{ij}(t)$ was an overapproximative projection of $\mathcal{G}(t)$ to begin with, further growth cannot cause the overapproximation to fail.

To formalize the bounds on x_k, define the set valued *slice function* for some $\mathcal{M} \subset \mathbb{R}^3$ and $y_{ij} \in \mathbb{Y}_{ij}$ as

$$\begin{aligned} \mathcal{F}_k(\mathcal{M}, y_{ij}) &= \{y_k \in \mathbb{Y}_k | \exists x \in \mathcal{M} \text{ with } \mathfrak{p}_{ij}[x] = y_{ij} \text{ and } \mathfrak{p}_k[x] = y_k\}, \\ &= \{\mathfrak{p}_k[x] \in \mathbb{Y}_k | x \in \mathfrak{p}_{ij}^{-1}[y_{ij}] \cap \mathcal{M}\}. \end{aligned} \tag{16}$$

In words, $\mathcal{F}_k(\mathcal{M}, y_{ij})$ is a slice through $\mathcal{M}$ along the subspace $\mathbb{Y}_k$ at the point y_{ij}; its value will therefore be an interval in $\mathbb{Y}_k$. If $\mathcal{M}$ is described by the zero sublevel set of function $\phi_{\mathcal{M}} : \mathbb{R}^3 \to \mathbb{R}$, then we can write a mathematical description of $\mathcal{F}_k$

$$\mathcal{F}_k(\mathcal{M}, y_{ij}) = \{y_k \in \mathbb{Y}_k | \phi_{\mathcal{M}}(y_i, y_j, y_k) \leq 0\}. \tag{17}$$

Given this definition, we can formulate a time evolution HJI PDE operating entirely in $\mathbb{Y}_{ij}$ for the implicit surface function $\phi_{ij}(y_{ij}, t)$ of the overapproximative projection $\mathcal{Y}_{ij}(t)$. Instead of (12), use

$$\begin{aligned} \frac{\partial \phi_{ij}(y_{ij}, t)}{\partial t} + H(y_{ij}, \nabla \phi_{ij}(y_{ij}, t)) = 0, \\ \phi_{ij}(y_{ij}, 0) = \mathfrak{p}_{ij}[\phi_0](y_i j), \end{aligned} \tag{18}$$

for those $y_{ij} \in \mathfrak{p}_{ij}[\mathcal{X}(t)]$, with Hamiltonian

$$H(y_{ij}, p) = \min_{y_k \in \mathcal{F}_k(\mathcal{M}, y_{ij})} p_i f_i(y_i, y_j, y_k) + p_j f_j(y_i, y_j, y_k), \tag{19}$$

where $\mathcal{M}$ is either $\mathcal{G}(t)$ or $\mathcal{X}(t)$.

The derivation above is informal, but its conclusion has a fascinating implication. Comparing (19) with (5), we see that the unmodeled dimension is in effect a disturbance input to the dynamics in the lower dimensional subspace.

This observation leads to an alternative interpretation of (18) and (19). For the linear rotation example, $\mathcal{G}(t)$ is the set of trajectory points $x(t)$ for those trajectories with initial points $x(0) \in \mathcal{G}_0$. If $\mathcal{Y}_{ij}(t)$ is to be a projective overapproximation of $\mathcal{G}(t)$, then $\mathcal{Y}_{ij}(t)$ must contain $\mathfrak{p}_{ij}[x(t)]$ for all these trajectories. Consider any time $s \in [0, t]$ and the point $x(s)$ along the full dimensional trajectory. By choosing the unmodeled dimension y_k from the set $\mathcal{F}_k(\mathcal{G}(s), y_{ij})$, we allow $y_k = \mathfrak{p}_k[x(s)]$. Therefore $\mathfrak{p}_{ij}[\dot{x}(s)] = \mathfrak{p}_{ij}[f(\mathfrak{p}_i[x(s)], \mathfrak{p}_j[x(s)], \mathfrak{p}_k[x(s)])]$ will be among the possible flow fields for the subspace's dynamics. Since s was arbitrary, $\mathfrak{p}_{ij}[x(\cdot)]$ is a feasible trajectory of the subspace's dynamic system, and so $\mathfrak{p}_{ij}[x(t)] \in \mathcal{Y}_{ij}(t)$.

Conjecture. *Let $\mathcal{G}(t)$ be time evolved by some HJI PDE in $\mathbb{R}^3$ and $\mathcal{Y}_{ij}(t)$ by some HJI PDE in $\mathbb{Y}_{ij}$. If the unmodeled dimension $x_k \in \mathbb{Y}_k$ of the full dimensional system dynamics $\dot{x} = f(x)$ is treated as a disturbance input to the subspace's dynamics, then*

$$\mathfrak{p}_{ij}[\mathcal{G}(t)] \subseteq \mathcal{Y}_{ij}(t),$$

where that input x_k is drawn from a slice $\mathcal{F}_k(\mathcal{M}, y_{ij})$ of an appropriate $\mathcal{M}$ for points $y_{ij} \in \mathbb{Y}_{ij}$.

We initially formulated this conjecture based on our numerical success in computing overapproximating projections. Sections 3.4 and 3.5 showcase some of those results. In the remainder of this section we outline what might be required to prove the conjecture, and then discuss some implementation details.

If $\mathcal{M} = \mathcal{G}(t)$, proving the conjecture requires showing that $\mathcal{F}_k(\mathcal{G}(t), y_{ij})$ is a valid set from which to draw disturbance inputs such that the viscosity solution of the appropriate HJI PDE (either (18)–(19) or (4)–(6)) will still result in the reachable set in which we are interested. The problem is that

the input constraint set $\mathcal{F}_k(\mathcal{G}(t), y_{ij})$ depends on both time t and state y_{ij}. In [3] we turned the computation of backwards reachable sets into a terminal payoff differential game, and used results in [10] to show that the differential game could be solved with an HJI PDE; however, those results assumed that the control and disturbance input constraint sets were constant. State dependent input constraints were investigated in [43], but only for the optimal control case (no disturbance inputs). It is not clear whether a differential game with time and state dependent input constraints would satisfy a dynamic programming principle. Without satisfying such a principle, it is unlikely that the viscosity solution of an HJI PDE would solve the differential game.

However, in practical terms, we do not have access to $\mathcal{G}(t)$ and must use $\mathcal{M} = \mathcal{X}(t)$. To prove the conjecture in this case would require the additional step of showing that

$$\mathcal{G}(t) \subseteq \mathcal{X}(t) \implies S_{\delta t}\left(\mathcal{G}(t)\right) \subseteq S_{\delta t}\left(\mathcal{X}(t)\right).$$

We are currently investigating methods of proving or disproving the conjecture. In addition, we are concentrating our efforts on implementation of the projection technique, in order to determine whether it can be applied to practical problems. A number of implementation details arise when solving (18) and (19), of which we briefly describe the three most important.

- In practice, the unmodeled dimension should be chosen from a slightly bloated version of $\mathcal{X}(t)$ to avoid the chance that $\mathcal{F}_k(\mathcal{X}(t), y_{ij}) = \emptyset$ for some y_{ij} on the boundary of $\mathcal{Y}_{ij}(t)$. Choosing d as a small multiple of the grid spacing, we use $\mathcal{F}_k(\mathcal{X}^d(t), y_{ij})$ instead.

- The computational domain in $\mathbb{Y}_{ij}$ is always larger than $\mathcal{Y}_{ij}(t)$. Assuming that d is chosen to be relatively small (to avoid excessive overapproximation), for those $y_{ij} \notin \mathfrak{p}_{ij}\left[\mathcal{X}^d(t)\right]$, we will still get $\mathcal{F}_k(\mathcal{X}^d(t), y_{ij}) = \emptyset$. One way of solving (18) and (19) in those cases is to use velocity extension [44] to extend the vector field artificially into $\mathcal{Y}_{ij}(t)^{\complement}$.

- Some projections approximate the reachable set better than others; however, each projection is individually an overapproximation of the reachable set, so if $\mathfrak{p}_i\left[\mathcal{Y}_{ij}(t)\right] \subset \mathfrak{p}_i\left[\mathcal{Y}_{ik}(t)\right]$, then we know that the extra range in $\mathfrak{p}_i\left[\mathcal{Y}_{ik}(t)\right]$ is not actually feasible. Thus, we can clip $\mathcal{Y}_{ik}(t)$ along dimension x_i until $\mathfrak{p}_i\left[\mathcal{Y}_{ij}(t)\right] = \mathfrak{p}_i\left[\mathcal{Y}_{ik}(t)\right]$. More generally, we can safely clip any portions of $\mathcal{Y}_{ij}(t)$ which lie outside of $\mathfrak{p}_{ij}\left[\mathcal{X}(t)\right]$.

Without this clipping process, poorly behaved projections can quickly grow larger than practical computational domains.

3.4 Evolving the Linear Rotation Example's Projections

The presentation in the previous section was somewhat abstract, so in this section we will apply the algorithm to the example from section 3.2. Consider how to evolve the initial projective overapproximation $\mathcal{Y}_{13}(0)$. From (9) and (13)

$$\phi_{13}(x_1, x_3, 0) = \sqrt{(x_1 - c_1)^2 + (x_3 - c_3)^2} - r,$$

which is a circle in $\mathbb{Y}_{13}$. We can evolve $\mathcal{Y}_{13}(t)$ by solving the HJI PDE

$$\frac{\partial \phi_{13}(x_1, x_3, t)}{\partial t} + H\left(x_1, x_3, \frac{\partial \phi_{13}(x_1, x_3, t)}{\partial x_1}, \frac{\partial \phi_{13}(x_1, x_3, t)}{\partial x_3}\right) = 0, \quad (20)$$

with Hamiltonian (using the dynamics (11))

$$H(x_1, x_3, p_1, p_3) = \min_{x_2 \in \mathcal{F}_2(\mathcal{X}(t), x_1, x_3)} \pi(-p_1 x_2 + p_3 0). \quad (21)$$

While $\mathcal{F}_2(\mathcal{X}(t), x_1, x_3)$ is a set valued function of x_1 and x_3, for illustration we can describe its value (an interval of $\mathbb{Y}_2$) at a few points for $t = 0$ based upon (9) and (13)

$$\mathcal{F}_2(\mathcal{X}(0), 0, 0) = [c_2 - r, c_2 + r],$$
$$\mathcal{F}_2(\mathcal{X}(0), r, 0) = [c_2, c_2].$$

Similar PDEs are used for $\mathcal{Y}_{12}(t)$ and $\mathcal{Y}_{23}(t)$.

Figure 10 shows the results of applying the projective evolution algorithm to the linear rotation example. The upper left figure shows the initial conditions and is the same as figure 9. The remaining subplots show the progress of the overapproximation through a half rotation of the dynamics. By $t = 1$, the projection $\mathcal{Y}_{13}(t)$ has grown from its initial circle into a rectangle. This growth occurs because of the freedom in choosing x_2 in (21). Similar growth occurs in $\mathcal{Y}_{23}(t)$ because there is freedom in choosing x_1 for the dynamics in $\mathbb{Y}_{23}$. In contrast, $\mathcal{Y}_{12}(t)$ remains a circle, because the free dimension x_3 in $\mathbb{Y}_{12}$ has no effect on the dynamics (11). In fact, $\mathcal{Y}_{13}(t)$ and $\mathcal{Y}_{23}(t)$ would grow larger than the squares shown were it not for the clipping procedure mentioned in the previous section. Figure 11 compares $\mathcal{X}(t)$ with the true reachable set $\mathcal{G}(t)$ at a variety of times in a closer view. As advertised, $\mathcal{G}(t) \subseteq \mathcal{X}(t)$.

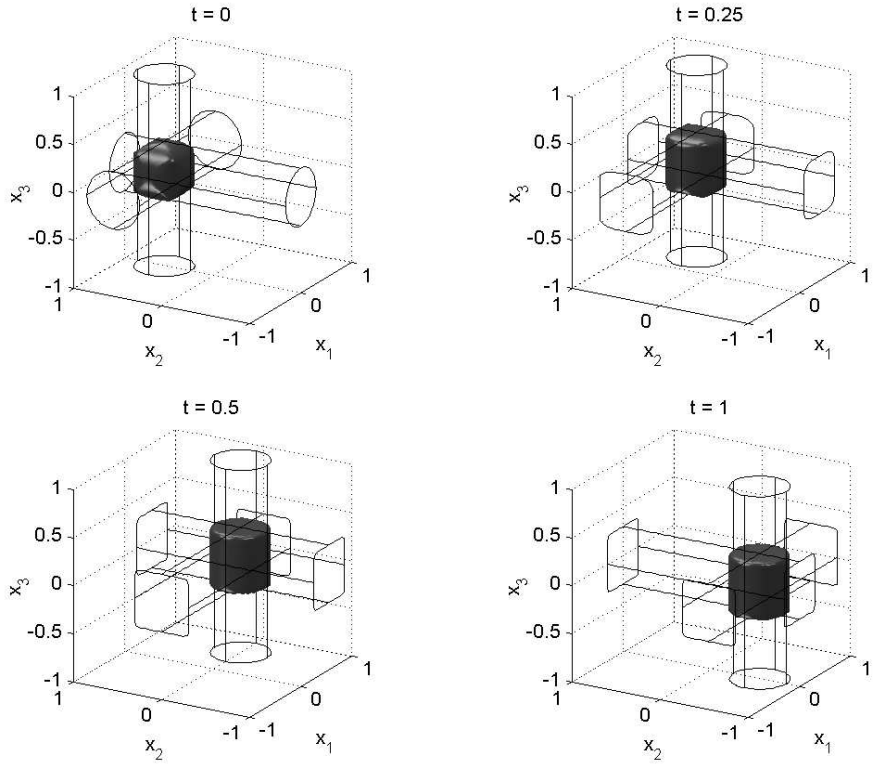

Figure 10: Evolution of the linear rotation example's projective overapproximations $\mathcal{Y}_{ij}(t)$ (contours on the walls) and $\mathcal{X}(t)$ (solid object).

3.5 Solving the Collision Avoidance Example Projectively

In this section we examine the projective overapproximation algorithm's application to the reachable set problem from section 2.1. We will use the single projection into the relative location plane $\mathbb{Y}_{12}$. Because the unmodeled dimension—the relative heading x_3—is already restricted to $\mathcal{Y}_3 = [0, 2\pi]$, there is no need to keep track of any other projections. We simply solve

$$\frac{\partial \phi_{12}(y_{12}, t)}{\partial t} + \min\left[0, H(y_{12}, \nabla \phi_{12}(y_{12}, t))\right] = 0,$$

with Hamiltonian

$$H(y_{12}, p) = \max_{a \in \mathcal{A}} \min_{b \in \mathcal{B}} \min_{y_3 \in \mathcal{Y}_3} p_1 f_1(y_1, y_2, y_3, a, b) + p_2 f_2(y_1, y_2, y_3, a, b)$$

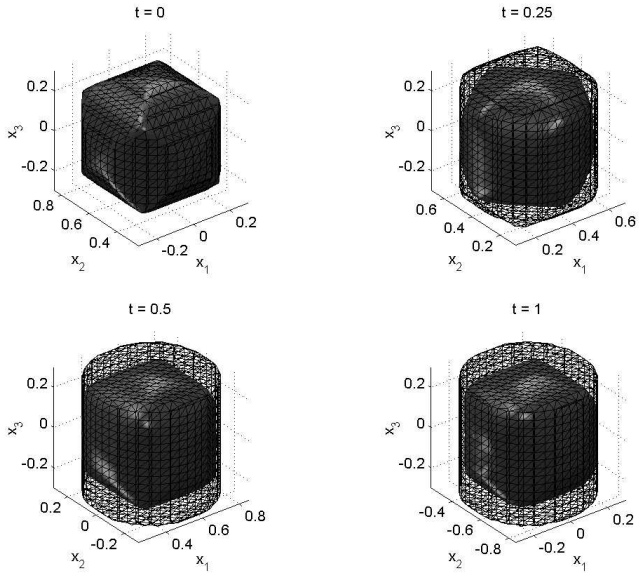

Figure 11: Comparing the projection based approximation $\mathcal{X}(t)$ (mesh) to the true reachable set $\mathcal{G}(t)$ (solid) at several times.

(where $f(x, a, b)$ is given by (2)) and terminal conditions

$$\phi(y_{12}, 0) = \sqrt{y_1^2 + y_2^2} - d_c.$$

The leftmost subplot of figure 12 shows the initial capture circle $\mathcal{Y}_{12}(0)$, while the remaining subplots show the growth of $\mathcal{Y}_{12}(t)$ until it converges to a fixed point $\mathcal{Y}_{12}$ in the rightmost for $t \gtrsim 2.6$. Figure 13 compares the overapproximation of the reachable set $\mathfrak{p}_{12}^{-1}[\mathcal{Y}_{12}]$ to the true reachable set $\mathcal{G}$ from two angles. Although $\mathfrak{p}_{12}^{-1}[\mathcal{Y}_{12}]$ is significantly larger than $\mathcal{G}$, in the

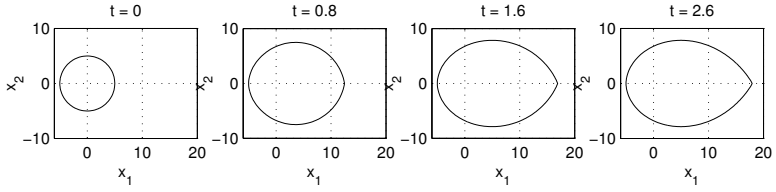

Figure 12: Growth of $\mathcal{Y}_{12}(t)$ for the collision avoidance example.

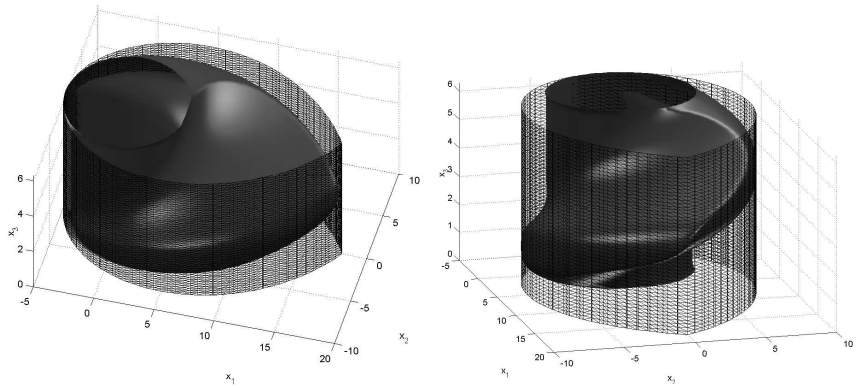

Figure 13: Two views comparing the true reachable set $\mathcal{G}$ (solid) with the back projection $\mathfrak{p}_{12}^{-1}[\mathcal{Y}_{12}]$ of the approximation (mesh).

Figure 14: Evader keeps pursuer from entering the projective overapproximation $\mathcal{Y}_{12}$ of the reachable set, and hence conservatively avoids collision.

left hand view it can be seen that to within grid resolution, $\mathcal{Y}_{12} = \mathfrak{p}_{12}[\mathcal{G}]$, which is the best that any projective representation could hope to achieve. The real payoff is computational time. While the full dimensional reachable set $\mathcal{G}$ takes about 20 minutes to compute on a three dimensional grid, the projective overapproximation $\mathcal{Y}_{12}$ takes less than one minute on a higher resolution two dimensional grid.

Figure 14 shows a series of frames from an animation of the collision avoidance scenario when the evader uses the projective overapproximation $\mathcal{Y}_{12}$ of the backwards reachable set. When comparing figure 14 to figure 7, notice that the slice of reachable set in the frames of figure 14 does not depend on relative heading, since that is the unmodeled dimension in the projection. By construction, the evader can keep the pursuer from entering $\mathcal{Y}_{12}$, and as long as the pursuer does not enter a collision is impossible. Using $\mathcal{Y}_{12}$ is a conservative strategy—it is an overapproximation of the true reachable set—but it is guaranteed to be safe, and in the event that model parame-

ters change, it can be recomputed much more rapidly than the true three dimensional reachable set.

4 Discussion

While the outline of the projective overapproximation algorithm above was specific to projecting a three dimensional space into coordinate aligned two dimensional subspaces, the power of this HJI based approach is that it can be generalized so easily. Both the full dimensional space and the projection subspaces can be higher dimensional. The projection subspaces need not be aligned with the coordinate axes, nor need all subspaces be of the same dimension; in fact, there are systems in which it might be useful to allow the projection subspaces to change smoothly with time. In a projection with multiple unmodeled dimensions, all the unmodeled dimensions would be treated as a disturbance input vector constrained by the appropriate projection of $\mathcal{X}(t)$ into the subspace spanned by the unmodeled dimensions. There is no theoretical reason to constrain the number of projections—for example, we could add to the linear rotation problem a projection into the subspace whose coordinate axes are $e_1 + e_3$ and $e_1 - e_3$, if we thought that such a projection would help restrict excessive overapproximation in $\mathcal{X}(t)$. The only constraints are implementation complexity and computational resources.

All this flexibility in the choice of projections leads to the question of how to choose appropriate projections for a particular system. For the linear rotation example, the natural coordinate axis projections turned out to be very effective (see section 3.4). In particular, the $\mathbb{Y}_{12}$ projection captured the relevant system dynamics and thus constrained the other two less effective projections through the clipping procedure. We can simulate the effect of poorly chosen projections by using the same three coordinate axis aligned projections, but rotating the system dynamics counterclockwise by 45° around the e_1 axis. To do that, replace the matrix A in (11) by

$$A' = GAG^T,$$

where

$$G = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix},$$

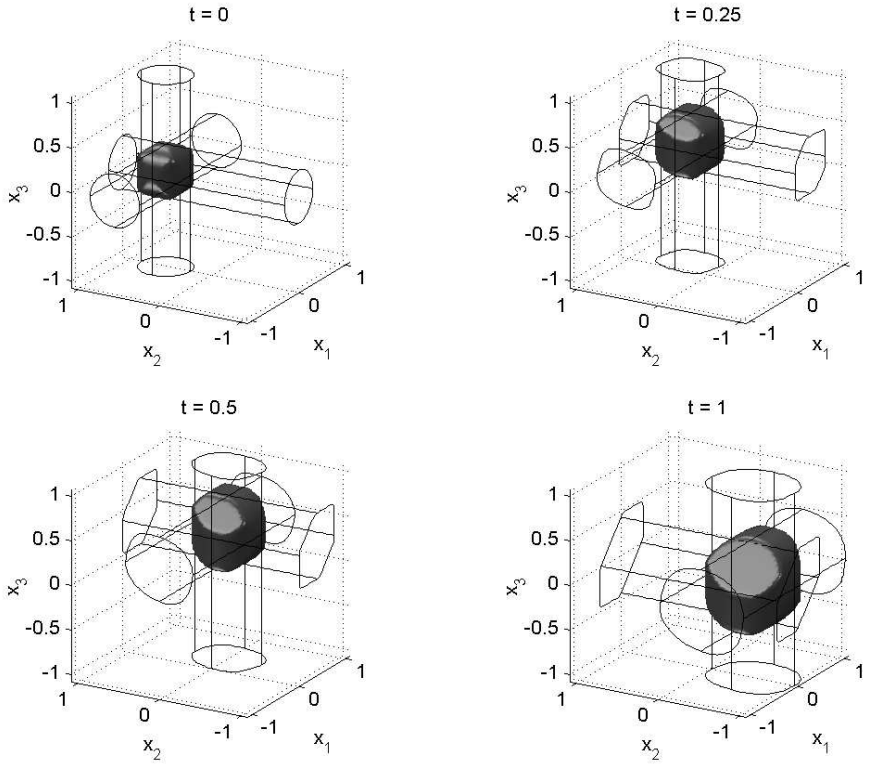

Figure 15: Evolution of the linear rotation example with poorly chosen projections.

and $\theta = \pi/4$. Figure 15 shows the growth of the projective reachable set $\mathcal{X}(t)$ for this version of the linear rotation example. Comparing it with figure 10 we can see how much greater the overapproximation becomes when none of the projections capture the system's dynamics.

There is also some concern, based on results from topology, that the projections' evolution may be pathological even if the true reachable set is well behaved under the system's dynamics. While we believe that this problem is unlikely to occur in practice when we are working with $\mathcal{X}(t)$—which is an intersection of prisms derived from the projections themselves—we are still investigating techniques for identifying appropriate projections for general systems.

The idea of subspace projections works well when we are trying to overap-

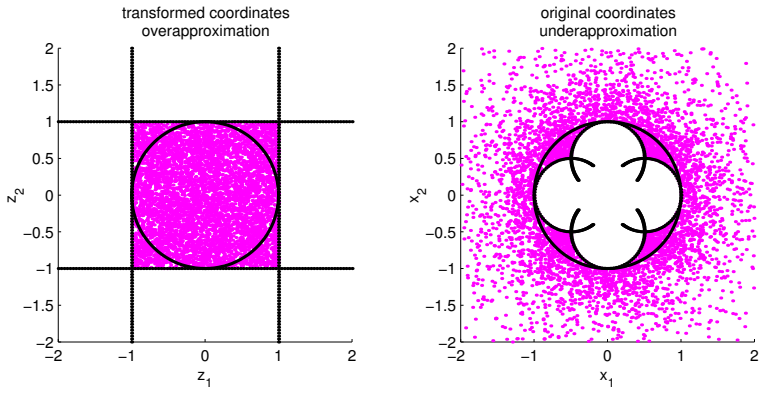

Figure 16: A square overapproximating the unit circle in z space becomes a cloverleaf underapproximating the same circle in x space.

proximate reachable sets, because inverting these projections back up into the full dimensional space generates a prism overapproximating the true reachable set. There are problems in which we wish to underapproximate the reachable set; for example, in aircraft envelope protection [5], safety requires that we stay within the flight envelope. If we are going to approximate that envelope we need an underapproximation, since an overapproximation would incorrectly mark as safe some states outside the true envelope. Safe flight envelopes are just one example of *controlled invariant sets*, and to compute these sets we need underapproximations of the true reachable set.

The projection scheme outlined above cannot directly compute underapproximations, since the back projected prisms are unbounded in the projection's unmodeled dimensions; thus, those prisms could never represent underapproximations of the true reachable set. We are instead investigating a coordinate inversion that could turn overapproximations into underapproximations. Consider underapproximating a circle centered at the origin in $\mathbb{R}^2$ by a pair of one dimensional projections (intervals of $\mathbb{R}$). Map $x \in \mathbb{R}^2$ to $z \in \mathbb{R}^2$ through the transformation

$$z_i = \frac{x_i}{\|x\|_2^2}. \tag{22}$$

While the circle stays a circle, this transformation could be applied to more general shapes by transformation of their implicit surface function representation, provided that the coordinate origin did not lie on the boundary of

the shape (we could shift the origin if it did). System dynamics can likewise be mapped through this nonlinear transformation, so that reachability could be calculated in the transformed coordinates. Now build a projective overapproximation of the circle in z space, using projections onto the coordinate axes. The left side of figure 16 shows the slabs that are the inverse projections of the two overapproximating intervals. The intersection of these two slabs is a square overapproximating the circle. The key observation is that we can invert (22) back into the original coordinate frame, and in the process the overapproximation in z space becomes an underapproximation in x space—the square that was an intersection of slabs becomes a cloverleaf made from a union of circles. The right side of figure 16 shows this underapproximation of the circle. The gray points on the left side map to the gray points on the right, and lie in the region of each state space that would be considered unsafe in an envelope protection problem. If the circle represents the true safe flight envelope, notice that the projective safe region (no gray points) on the left is an underapproximation of the true safe region.

Projective schemes based on Hamilton-Jacobi-Isaacs equations are a powerful way to tackle Bellman's "curse of dimensionality" and calculate approximations to reachable sets for systems larger than dimension two or three. The goal of this paper was to present motivation for and a gentle introduction to the computation of reachable sets, and to outline the basic ideas behind projective approximation algorithms. We continue to work on the many remaining theoretical and implementation details, and hope that this paper will stimulate further innovation in accurate, scalable schemes for calculating approximations of reachable sets.

Acknowledgments: We would like to thank Professors Ronald Fedkiw and Stanley Osher for extensive discussions about the details of numerical schemes for solving the Hamilton-Jacobi PDE, and we thank Professors John Lygeros, L. C. Evans, Shankar Sastry and Alexander Kurzhanski for discussions about the previous and current time dependent Hamilton-Jacobi formulations. The original idea for tackling continuous reachability in higher dimensional systems using projections came out of work with Professor Mark Greenstreet. The frames in figures 4 and 5 and their associated animations were generated using a renderer written by Professor Ronald Fedkiw.

References

[1] C. J. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, pp. 110–120, 2001. June.

[2] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, pp. 677–691, August 1986.

[3] I. Mitchell, A. Bayen, and C. J. Tomlin, "Computing reachable sets for continuous dynamic games using level set methods." Submitted January 2002 to *IEEE Transactions on Automatic Control*.

[4] I. Mitchell and C. Tomlin, "Level set methods for computation in hybrid systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 310–323, Springer Verlag, 2000.

[5] I. Mitchell, A. Bayen, and C. J. Tomlin, "Validating a Hamilton-Jacobi approximation to hybrid system reachable sets," in *Hybrid Systems: Computation and Control* (M. D. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), no. 2034 in Lecture Notes in Computer Science, pp. 418–432, Springer Verlag, 2001.

[6] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[7] S. Osher and C.-W. Shu, "High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations," *SIAM Journal on Numerical Analysis*, vol. 28, no. 4, pp. 907–922, 1991.

[8] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[9] M. G. Crandall, L. C. Evans, and P.-L. Lions, "Some properties of viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 282, no. 2, pp. 487–502, 1984.

[10] L. C. Evans and P. E. Souganidis, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations," *Indiana University Mathematics Journal*, vol. 33, no. 5, pp. 773–797, 1984.

[11] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[12] C. Tomlin, J. Lygeros, and S. Sastry, "A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, vol. 88, pp. 949–970, July 2000.

[13] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations*. Boston: Birkhäuser, 1997.

[14] M. Broucke, M. D. D. Benedetto, S. D. Gennaro, and A. Sangiovanni-Vincentelli, "Optimal control using bisimulations: Implementation," in *Hybrid Systems: Computation and Control* (M. D. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), no. 2034 in Lecture Notes in Computer Science, pp. 175–188, Springer Verlag, 2001.

[15] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre, "Set-valued numerical analysis for optimal control and differential games," in *Stochastic and Differential Games: Theory and Numerical Methods* (M. Bardi, T. Parthasarathy, and T. E. S. Raghavan, eds.), vol. 4 of *Annals of International Society of Dynamic Games*, Birkhäuser, 1999.

[16] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," in *Hybrid Systems: Computation and Control* (N. Lynch and B. Krogh, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 21–31, Springer Verlag, 2000.

[17] O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 73–88, Springer Verlag, 2000.

[18] A. Bemporad, F. D. Torrisi, and M. Morari, "Optimization-based verification and stability characterization of piecewise affine and hybrid systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 45–59, Springer Verlag, 2000.

[19] A. Chutinan and B. H. Krogh, "Approximating quotient transition systems for hybrid systems," in *Proceedings of the American Control Conference*, (Chicago, IL), pp. 1689–1693, 2000.

[20] A. Hassibi, S. Boyd, and J. P. How, "A class of Lyapunov functionals for analyzing hybrid dynamical systems," in *Proceedings of the American Control Conference*, (San Diego, CA), pp. 2455–2460, 1999.

[21] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, "Beyond HyTech: Hybrid systems analysis using interval numerical methods," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 130–144, Springer Verlag, 2000.

[22] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 202–214, Springer Verlag, 2000.

[23] O. Shakernia, G. J. Pappas, and S. S. Sastry, "Decidable controller synthesis for classes of linear systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 407–420, Springer Verlag, 2000.

[24] N. Shishido and C. J. Tomlin, "Ellipsoidal approximations of reachable sets for linear games," in *Proceedings of the IEEE Conference on Decision and Control*, (Sydney, Australia), 2000.

[25] A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," in *Hybrid Systems: Computation and Control* (C. J. Tomlin and M. R. Greenstreet, eds.), no. 2289 in Lecture Notes in Computer Science, pp. 465–478, Springer Verlag, 2002.

[26] R. Vidal, S. Schaffert, J. Lygeros, and S. S. Sastry, "Controlled invariance of discrete time systems," in *Hybrid Systems: Computation and Control* (B. Krogh and N. Lynch, eds.), no. 1790 in Lecture Notes in Computer Science, pp. 437–450, Springer Verlag, 2000.

[27] M. Greenstreet and I. Mitchell, "Integrating projections," in *Hybrid Systems: Computation and Control* (S. Sastry and T. Henzinger, eds.), no. 1386 in Lecture Notes in Computer Science, pp. 159–174, Springer Verlag, 1998.

[28] M. Greenstreet and I. Mitchell, "Reachability analysis using polygonal projections," in *Hybrid Systems: Computation and Control* (F. Vaandrager and J. van Schuppen, eds.), no. 1569 in Lecture Notes in Computer Science, pp. 103–116, Springer Verlag, 1999.

[29] S. G. Govindaraju, D. L. Dill, A. J. Hu, and M. A. Horowitz, "Approximate reachability with BDDs using overlapping projections," in *35th Design Automation Conference*, (San Francisco, CA), pp. 451–456, 1998.

[30] S. G. Govindaraju, D. L. Dill, and J. P. Bergmann, "Improved approximate reachability using auxiliary state variables," in *36th Design Automation Conference*, (New Orleans, LA), pp. 312–316, 1999.

[31] S. G. Govindaraju and D. L. Dill, "Counterexample-guided choice of projections in approximate symbolic model checking," in *International Conference on Computer Aided Design*, (San Jose, CA), pp. 115–119, 2000.

[32] P. Burchard, L.-T. Cheng, B. Merriman, and S. Osher, "Motion of curves in three spatial dimensions using a level set approach," *Journal of Computational Physics*, vol. 170, pp. 720–741, 2001.

[33] S. Osher, L.-T. Cheng, M. Kang, H. Shim, and Y.-H. Tsai, "Geometric optics in a phase space based level set and Eulerian framework." preprint, submitted to *J. Computational Physics*, 2001.

[34] B. Engquist, O. Runborg, and A.-K. Tornberg, "High frequency wave propagation by the segment projection method," *Journal of Computational Physics*, vol. 178, pp. 373–390, 2002.

[35] A. W. Merz, "The game of two identical cars," *Journal of Optimization Theory and Applications*, vol. 9, no. 5, pp. 324–343, 1972.

[36] I. Mitchell, "Games of two identical vehicles," Tech. Rep. SUDAAR 740, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, July 2001.

[37] G.-S. Jiang and D. Peng, "Weighted ENO schemes for Hamilton-Jacobi equations," *SIAM J. Sci. Comput.*, vol. 21, pp. 2126–2143, 2000.

[38] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. New York: Cambridge University Press, 1999.

[39] M. G. Crandall and P.-L. Lions, "Two approximations of solutions of Hamilton-Jacobi equations," *Mathematics of Computation*, vol. 43, no. 167, pp. 1–19, 1984.

[40] K. O. Friedrichs and P. D. Lax, "Systems of conservation equations with a convex extension," *Proceedings of the National Academy of Sciences, USA*, vol. 68, pp. 1686–1688, 1971.

[41] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, pp. 439–471, 1988.

[42] `http://cherokee.stanford.edu/~mitchell`.

[43] S. Bortoletto, "The Bellman equation for constrained deterministic optimal control problems," *Differential and Integral Equations*, vol. 6, no. 4, pp. 905–924, 1993.

[44] D. Adalsteinsson and J. A. Sethian, "The fast construction of extension velocities in level set methods," *Journal of Computational Physics*, vol. 148, pp. 2–22, 1999.