

**FAST MARCHING FOR HYBRID CONTROL\***

Michael S. Branicky and Ravi Hebbar

Electrical Engineering and Computer Science Dept.

Case Western Reserve University

Cleveland, OH 44106-7221 USA

{msb11,rxh20}@po.cwru.edu

**Abstract**

This paper describes an approach to solving optimal hybrid control problems using level set methods. Level set methods are a powerful set of techniques for generating equipotential contours with applications in the realm of fluid mechanics, computer vision, material science, robotics and geometry. This paper specifically deals with the problem of determining an optimal control path in a hybrid system by extending a particular level set algorithm, known as the "fast marching" method, to a hybrid setting. Several representative examples are solved.

**1. Introduction to Level Sets**

Level set methods are numerical techniques designed to track the evolution of interfaces between two regions [1]. Consider a patch of ice in a square tank of water as shown in Figure 1. As time progresses the boundary of the ice patch keeps morphing (increases/decreases) with time. The evolution of the boundary between ice and water can be solved by using level set methods. Thus, in essence, level set methods provide a solution to the movement of fronts.

A special case of the general level set method is the fast marching level set method [2], in which the front always moves in one direction. The concept of the fast marching method can be visualized as the motion of a wave front emanating from a source. A simple example is the evolution of the wave fronts when a pebble is thrown in a pond of still water.

Related work in robotic obstacle avoidance includes the algorithms of Trovato [4]. There has also been some work in extending level set methods to deal with the evolution of Hamilton-Jacobi equations [5].

\* This work supported by the National Science Foundation, under grant DMI97-20309.

**2. Fast Marching Method**

Let us now focus our attention on the fast marching method, as described in [2]. It forms the basis of the algorithm presented in this paper to solve for the optimal control path in a hybrid setup. Figure 2 shows a wave front in a 2D plane.

Assume  $\phi(x, t)$  to be a distance function such that  $\phi(x, t = t_1) = d$  if the distance from the point  $x$  to the wave front at time  $t = t_1$  is  $d$ . The points on the wave front at time  $t = t_1$  are given by the set

$$\Gamma(t = t_1) = \{x \mid \phi(x, t = t_1) = 0\}.$$

The equation of the evolving contour is obtained by solving the differential equation given as

$$\text{Given: } \phi(x(t), t) = 0, \quad \phi_0 = \phi(x(t), t = 0) \quad (1)$$

$$\Rightarrow \phi_t + \nabla(\phi(x(t), t)) \cdot x'(t) = 0 \quad (2)$$

$$\Rightarrow \phi_t + F |\nabla(\phi)| \quad \text{since } n = \nabla(\phi) / |\nabla(\phi)|. \quad (3)$$

In the above equation,  $F$  is the speed with which the wave can propagate at any point; it can be a function of  $x(t)$ . If  $F > 0$ , it is shown in [1] that the equation reduces to that of a form of the Eikonal equation, given as

$$|\nabla(T)| F = 1, \quad (4)$$

where  $T(x)$  is the time function such that  $T(x_1)$  is the time at which the front passes the point  $x_1$ . A numerical approximation scheme to solve Equation (4) is given in [1,2] and described briefly in the next section.

**3. Fast Marching Method – Numerical Approximation Scheme**

In this section we briefly describe the fast marching level set method explained in [2]. The problem is simplified by considering the propagation of the wave front on a 2D grid. The approximate solution to Equation (4) on a 2D grid is given as

$$[\max(D_{ij}^{-x} T, 0)^2 + \min(D_{ij}^{+x} T, 0)^2 + \max(D_{ij}^{-y} T, 0)^2 + \min(D_{ij}^{+y} T, 0)^2] = 1/F_{ij}^2. \quad (5)$$

However, as mentioned in [2], a different approximation to the gradient, introduced by [3], which is less diffusive is given as

$$[\max(\max(D_{ij}^{-x} T, 0), -\min(D_{ij}^{+x} T, 0))^2 + \max(\max(D_{ij}^{-y} T, 0), -\min(D_{ij}^{+y} T, 0))^2] = 1/F_{ij}^2. \quad (6)$$

In Equations (5) and (6),  $D_{ij}^{-x} T$  and  $D_{ij}^{+x} T$  are defined as

$$D_{ij}^{-x} T = (T_{ij} - T_{i-1,j})/\Delta x \quad \text{and} \quad D_{ij}^{+x} T = (T_{ij} - T_{i+1,j})/\Delta x,$$

where  $\Delta x$  is the grid spacing.

The algorithm to compute the arrival times of all the grid points of an N by N grid, using (6), is reproduced from [2]:

1. Initialize
  - a. Alive points: All start points are tagged as alive points and the value of  $T$  for those points is assigned to be 0.
  - b. Narrow band points: All points that are members of the four-connectivity neighborhood of the alive points, that are not alive, are narrow band points. The time of arrival,  $T$ , for those points are evaluated as  $dx/F_{ij}$  or  $dy/F_{ij}$ .
  - c. Faraway points: All other points on the grid are tagged as faraway points and their time of arrival,  $T$ , is set equal to  $\infty$ .
2. Marching forward
  - a. Begin loop: Find  $i$  and  $j$  of the narrow band point with the smallest value of  $T$ .
  - b. Tag the point as alive point and remove it from the narrow band list.
  - c. Tag its four-connectivity neighbors as narrow band points if they are either faraway or narrow band points. Remove the neighbors in the faraway list and put them in the narrow band list.
  - d. Compute the values of  $T$  for the neighbors by solving for the largest possible solution to Equation (6).
  - e. Return to step a.

We will now look into the problem of solving Equation (6) to compute the arrival times of the neighbors of an alive point. Figure 3 shows five points on the grid, where the arrival time of the point E needs to be evaluated. Let  $T_A$ ,  $T_B$ ,  $T_C$  and  $T_D$  be the arrival times of the four points. It is required to compute the arrival time of the point E, i.e.  $T_E$ .

We define  $T_x$  and  $T_y$  as  $T_x = \min(T_A, T_C)$  and  $T_y = \min(T_B, T_D)$ . Further, if  $T_x < T_y - 1/F_E$  then  $T_E = T_x + 1/F_E$  and if  $T_y < T_x - 1/F_E$  then  $T_E = T_y + 1/F_E$ . Otherwise,  $T_E$  is computed by solving the following quadratic equation and taking the largest possible root.

$$(T_E - T_x)^2 + (T_E - T_y)^2 = 1/F_E^2$$

It can be easily verified that  $T_E$  computed according to the above-mentioned rules is the exact solution of Equation (6).

#### 4. Robotic Navigation Examples

The fast marching level set method provides an attractive solution to time optimal path generation for robotic navigation. Obstacles in the path of the robot can be represented as regions of very low speed and the rest of the configuration space (C-space) can be modeled as regions with very high speed. Knowing the start point of the robot, level sets can be generated based on the fast marching method. The trajectory from the goal point to the start point which is orthogonal to the level set contours at every point gives the time optimal path from the start point to the goal point. Figure 4 illustrates a few examples in which the start point is the grid point represented by a triangle and the goal point is the grid point represented by a circle.

#### 5. Extension to Hybrid System Problems

We are now in a position to extend the concept to solve for time optimal paths for hybrid systems. A *hybrid system* is one that has both continuous dynamics and discrete dynamics associated with it [6]. Hybrid systems can be modeled as *hybrid automata* which have finite set of control locations with edges between them [6,7]. The finite set of *control locations* is the finite set of discrete states and the *edges* are the discrete transitions between the states. Each state has continuous dynamics encoded in it and the discrete transitions are characterized by a guard and a reset or jump relation. Figure 5 illustrates a hybrid automaton representing a hybrid system.

The hybrid automaton of Figure 5 has two states with the continuous dynamics encoded in the states. The *guards* are the condition checks associated with the edges. A *reset relation* is one in which the value of a variable is set to some constant value and a *jump relation* is one in which the value of a variable is set based on the values of other variables or itself. The *jump predecessors* of a point, say  $p$ , are all those points from which there is a discrete transition (reset or jump) that leads to  $p$ . A *switch* is a special kind of jump in which the discrete state changes but all continuous states maintain their values.

To keep things simplified we still consider the problem in 2D. We now have a 2D-grid space for each of the discrete states in the hybrid system, where the start point is in one state and the goal point in the same or some other state. Further, the fast marching method described in Section 3 needs to be modified to incorporate the discrete transitions between the states. The discrete transitions between the states are explicitly specified by the jump relation. The arrival time computation of a grid point (say  $p$ ) is done in two steps. First, compute the arrival time (say  $T_1$ ) of the point by considering the four neighbors in the same state as explained in Section 3. In the next step, evaluate the sums of the arrival time of its jump predecessors in other states and their corresponding jump costs and determine the minimum among them (say  $T_2$ ). Now assign the arrival time of  $p$  as the minimum of  $T_1$  and  $T_2$ .

The optimal path is determined in the same manner as in the case of a single state but with a small modification. In a particular state, a path that is orthogonal to the level sets is traversed and as soon as the path hits a grid point whose jump predecessor has smaller arrival time, a discrete transition is made to the state corresponding to the jump predecessor.

The proof of the modified fast marching algorithm is given in [8]. In this paper we look at two different simplified hybrid system models.

### 5.1. Restricted Jumps Between States

In this model, we consider a start point in one discrete state and a goal point in some other discrete state, with transitions from one discrete state to another being explicitly defined at a specific subset of locations. To explain the concept, let us consider four floors of a building as four discrete states. Let us assume that the start point is the building entrance. Each floor is connected to the other floor by stairs. It is now

required to find the time optimal path from the entrance to anywhere in the building.

Each floor has obstacles in it which define the continuous dynamics for the state corresponding to the floor. The stairs are assumed to be free of obstacles and hence have constant dynamics associated with them. The stairs represent the discrete transitions between the states where the time of arrival is defined by a jump relation. The grid point corresponding to the building entrance is the one from which the wave front propagation starts and continues on that level the same way as is done in the case of the single state case explained at the beginning of the section. As soon as the wave front hits a stair, however, it enters the next level and the propagation continues on both the levels. As far as a grid point does not have a stair to any of the other levels, it is considered to have only four neighbors. If a grid point, say  $p$ , has a stair associated with it to the next level, then the point in the other level, say  $q$ , associated with the stair is also a neighbor of  $p$ . The arrival time of the point  $q$  is then the arrival time of the point  $p$  plus the time required to climb the stair which is referred as stair cost. If the point  $q$  is a faraway point, then the computed arrival time is assigned to it and the point  $q$  is removed from the faraway list and added to the narrow band list. If the point  $q$  is a narrow band point, then the computed time is assigned to it if its arrival time is greater than the computed arrival time; else its arrival time is retained as it is.

After completion of propagation through all the four levels, given the goal point, a backwards trajectory from the goal point to the start point such that the trajectory is orthogonal to each of the level set contours gives the time optimal path from the start point to the goal point. The discrete transitions between the levels are made if the arrival time of a stair point in one level is greater than its corresponding stair point in another level. Figure 6 shows the four floors with obstacles (outlined regions), the stairs connecting the floors (dotted lines), and a time optimal path (bold lines) from the entrance to a point on the fourth floor.

In the above example we considered only one stair connecting each floor to the next floor. If more than one stair connects a floor to the next floor, the algorithm still remains the same.

## 5.2. Arbitrary Switches Between States

In this model, we consider the case where the start and goal points are replicated in all the discrete states and *switching* between discrete states can occur everywhere on the grid (to another discrete state while maintaining the same continuous state). We consider an example of maneuvering a vehicle with four gears from a start point to a goal point. The four distinct continuous dynamics for each gear are encoded in each of the four discrete states in terms of a speed profile, specified at each grid point. Note that the transition between discrete states is arbitrary, e.g., one may switch directly from first to fourth gear.

The wave front propagation starts from the start point in one state. Every grid point now has four neighbors in the same discrete state and three other neighbors in each of the other discrete states, corresponding to the ability to switch gears. The arrival time of the neighbors in the same level is updated according to the level set formulation. The arrival time of a neighbor, say  $q$ , in another state is evaluated as the sum of the arrival time of the point, say  $p$ , in the state under consideration and a jump cost to the other state. If the point  $q$  is a faraway point, then the computed arrival time is assigned to it and the point  $q$  is removed from the faraway list and put in the narrow band list. If the point  $q$  is already in the narrow band list, then its time is updated with the computed time if the computed time is lesser than the already assigned time of  $q$ .

Once the propagation is complete, the time optimal path is determined in a similar fashion as that of the single state case. A backward trajectory from the goal point to the start point gives the time optimal path from the start point to the goal point. At every grid point (say  $p$ ), the three neighbors in the other discrete states are considered and a discrete transition is made if the arrival time of any of those three points is less than that of  $p$ . If more than one of the three points have arrival times less than the arrival time of  $p$ , then the discrete transition is made to the state with the point that has the least arrival time.

The process is done for four different cases. In each case the start point is assumed to be in one of the four discrete states. However, in each case the goal point is considered in all the states. The time optimal path is then the optimal path among the four solutions thus obtained. Figure 7 illustrates the speed profile for the four states. Figures 8 through 11 illustrate the time

optimal path in the four different cases. The four sub-figures for each figure represent the four states. Based on the arrival time of the goal point, the path of Figure 8 was found out to be the optimal one.

## 6. Conclusions

This paper was aimed at studying level set methods and investigating the possibility of employing level sets in solving time optimal path problems in hybrid systems. The examples presented show that the level set formulation with the modification of jump between states can be employed to solve those problems. Further, by combining the procedure of propagation in the two models the solution to the time optimal path can be determined for any complex hybrid system. A constructive proof of the algorithm in the general case is given in [8].

## 7. References

1. J.A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Material Sciences*. Cambridge University Press, 1996.
2. J.A. Sethian, A fast marching level set method for monotonically advancing fronts, *Proc. Nat. Acad. Sci.*, 93(4):1591-1595, 1996.
3. E. Rouy and A. Tourin, *SIAM J. Num. Anal.*, 29:867-884, 1992.
4. K.I. Trovato, *A' Planning in Discrete Configuration Spaces of Autonomous Systems*, Ph.D. Thesis, Univ. of Amsterdam, Sept. 1996.
5. C. Tomlin, J. Lygeros, and S. Sastry. Computing controllers for nonlinear systems. *Proc. Hybrid Systems: Computation and Control*, Nijmegen, The Netherlands, March 1999.
6. M.S. Branicky, *Studies in Hybrid Systems: Modeling, Analysis, and Control*, Sc.D. Thesis, Electrical Engineering and Computer Science Dept., M.I.T., June 1995.
7. R. Alur *et. al.*, Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: R. L. Grossman *et. al.*, eds., *Hybrid Systems*, LNCS 736, pp. 209-229, Springer, 1993.
8. M.S. Branicky, R. Hebbbar and G. Zhang, A fast marching algorithm for hybrid systems, *Proc. 1999 IEEE Conference on Decision and Control*, Phoenix, AZ, Dec. 1999. Submitted.

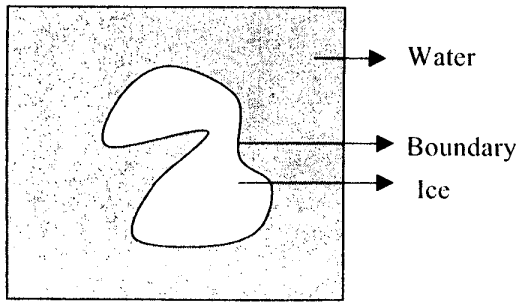


Figure 1

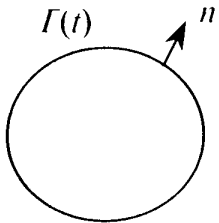


Figure 2

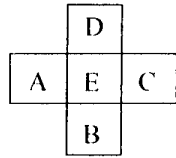


Figure 3

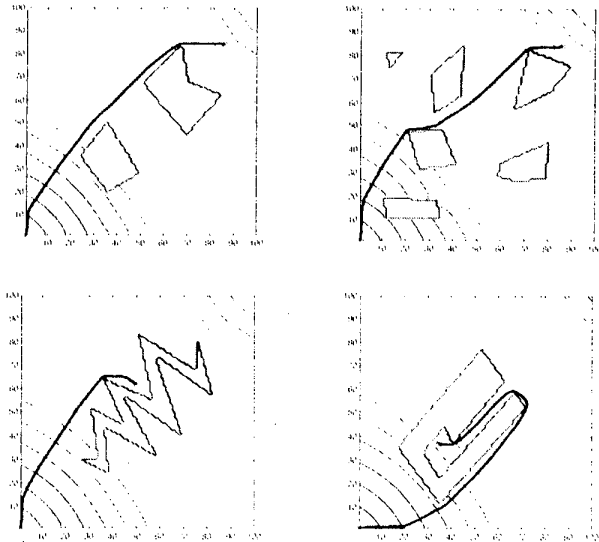


Figure 4

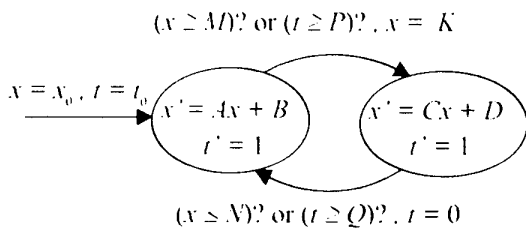


Figure 5

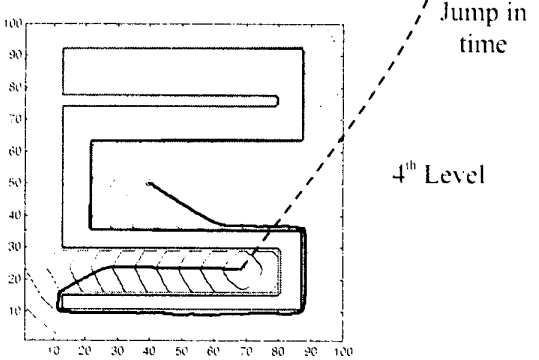
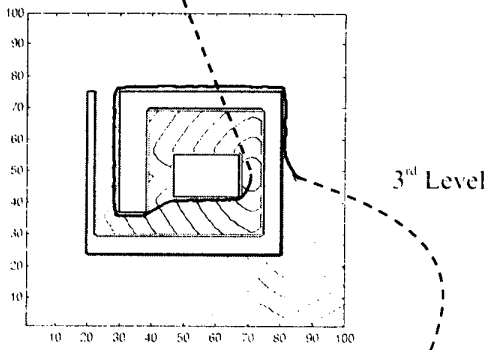
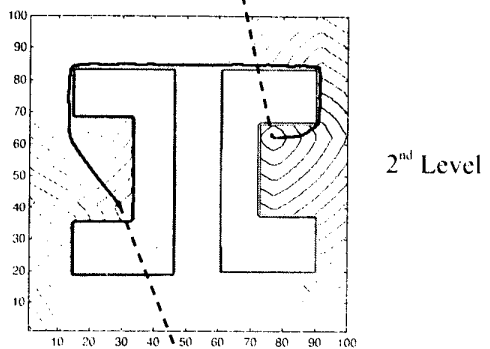
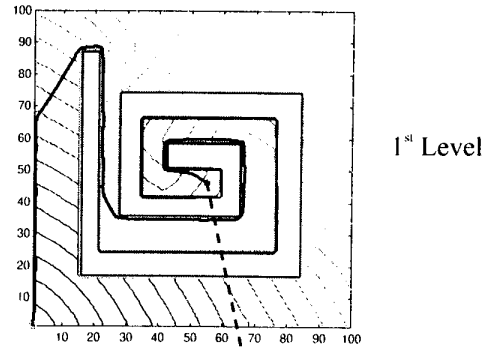


Figure 6

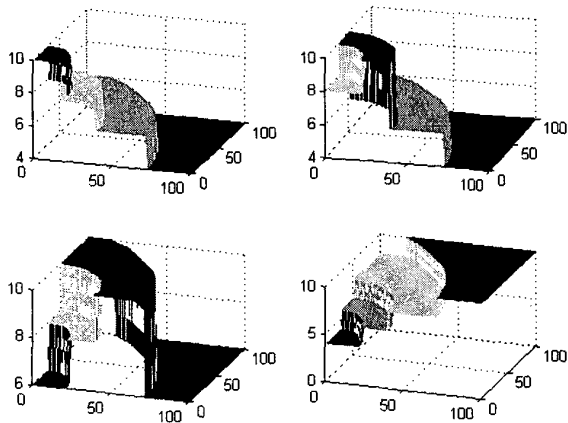


Figure 7

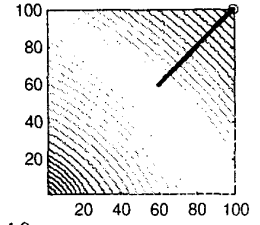
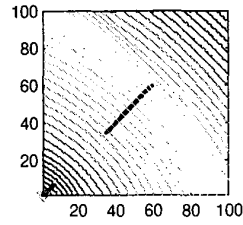
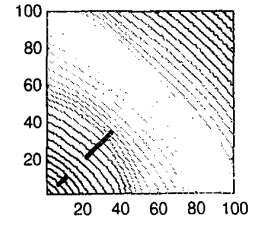
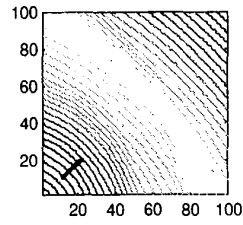


Figure 10

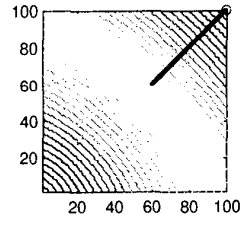
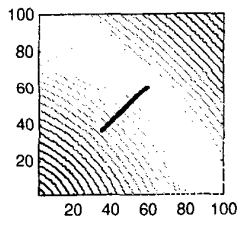
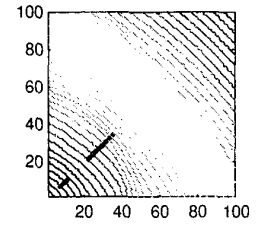
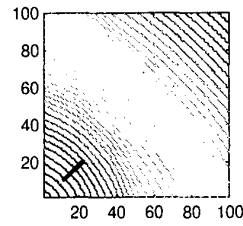
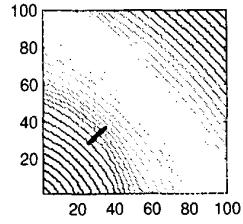
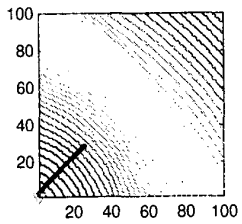


Figure 8

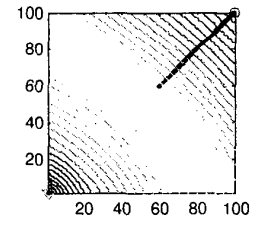
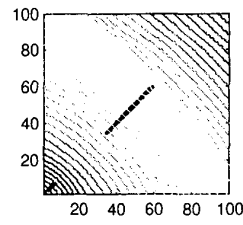


Figure 11

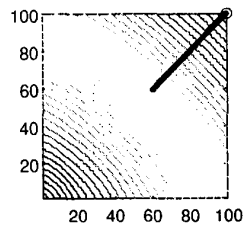
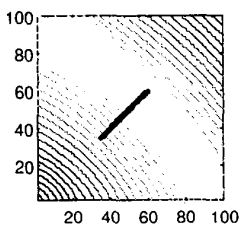
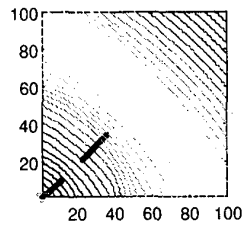
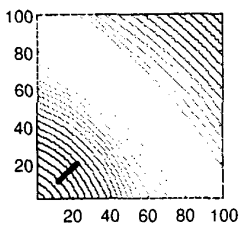


Figure 9