Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Optimal Path Planning under Different Norms in Continuous State Spaces

Ken Alton and Ian M. Mitchell
Department of Computer Science
University of British Columbia
2366 Main Mall, Vancouver, BC, V6T 1Z4, Canada
{kalton,mitchell}@cs.ubc.ca

***Abstract*— Optimal path planning under full state and map knowledge is often accomplished using some variant of Dijkstra's algorithm, despite the fact that it represents the path domain as a discrete graph rather than as a continuous space. In this paper we compare Dijkstra's discrete algorithm with a variant (often called the fast marching method) which more accurately treats the underlying continuous space. Analytically, both generate a value function free of local minima, so that optimal path generation merely requires gradient descent. We also investigate the use of optimality metrics other than Euclidean distance for both algorithms. These different norms better represent optimal paths for some types of problems, as demonstrated by planning optimal collision-free paths for a multiple robot scenario. When considering approximations consistent with the underlying state space, our conclusion is that fast marching places fewer constraints upon grid connectivity, and that it achieves better accuracy than Dijkstra's discrete algorithm in many but not all cases.**

I. Introduction

Path planning is one of the most studied areas of robotics research and nobody aspires to follow poor paths, but the concept of optimality is in many cases qualitative or is relegated to an afterthought by competing concerns of mapping, localization and/or safety. While these are important issues, in this paper we focus on quantitatively optimal path planning on continuous domains. In order to do so in this constrained conference venue, we make the simplifying assumptions of full state and map knowledge. The recommendations resulting from our analysis should be simple enough to incorporate into more general robotic systems that take account of these other concerns.

The grandfather of techniques for this full knowledge setting is Dijkstra's algorithm [1], which is an efficient implementation of dynamic programming to find the shortest path through a discrete graph. In this paper we explore the use of this algorithm and an adaptation [2]—commonly called the fast marching method [3]—which more accurately approximates the underlying continuous space.

Both algorithms are typically used to create paths which are optimal in the Euclidean or 2-norm, a job for which the discrete nature of connectivity in Dijkstra's algorithm is particularly poorly suited. However, there are tasks for which this norm is inappropriate; for example, planning a minimum time path through configuration space for a robot arm whose joints can be actuated independently. Therefore, we permit a slightly broader definition of optimality than simply minimum Euclidean distance between states. In fact, there are models in which different norms should be applied to different subsets of the coordinates.

The two main contributions of this paper are an exploration of the commonalities and differences between Dijkstra's discrete algorithm and the fast marching method in the context of optimal path planning on continuous state (configuration) spaces, and derivation of update equations for both algorithms for robots whose actions are constrained by a variety of different norms. In addition, we demonstrate how combinations of these norms can be used to generate collision-free optimal paths for multiple robots.

II. Optimal Path Planning in a Continuous Domain

We formulate our problem as an optimal cost to reach some closed *target set* $\mathcal{T}$ over *paths* or *trajectories* $y(\cdot)$ which travel through some *domain* $\mathcal{D}$. We present results for Euclidean space $\mathcal{D} \subset \mathbb{R}^d$, although modifications to handle periodic dimensions—such as rotational coordinates $[0, 2\pi)^d$—are straightforward, and manifolds can be accomodated [4]. The paths are judged to be optimal according to an additive cost metric (total cost is the sum of costs encountered along the path):

$$V(x_0) = \inf_{y(\cdot)} \int_{t_0}^{t_f} c(y(s))ds, \tag{1}$$

where $y(t_0) = x_0$, $y(t_f) \in \mathcal{T}$, and $y(\cdot)$ are drawn from feasible paths such that $y(t) \in \mathcal{D} \backslash \mathcal{T}$ for $t_0 \leq t < t_f$. The *cost function* $c(x)$ is assumed to be Lipschitz continuous in $\mathcal{D}$, and generates a *value function* $V(x)$ which measures the minimum cost to go from a point x to anywhere on the boundary of the target set. It can be shown that this value function satisfies a *dynamic programming principle* (DPP)

$$V(x_0) = \inf_{y(\cdot)} \left[V(y(t + \Delta t)) + \int_t^{t+\Delta t} c(y(s))ds \right], \tag{2}$$

which says that the optimal cost to go from the current point x_0 is given by the cost of the trajectory which minimizes the cost to go from a future point on that trajectory plus the cost over the next Δt time units.

Rearrange this equation and let $\Delta t \to 0$ to get an infinitesimal version of the DPP in the form of a *partial differential*

equation (PDE)

$$\min_{\dot{y}} \nabla V(x) \cdot \dot{y} = c(x), \tag{3}$$

where the choice of $\dot{y} = dy(t)/dt$ is the choice of action for the robot. In this paper we restrict ourselves to the isotropic problem, where the constraint on robot action is given by

$$||\dot{y}||_p \leq 1 \tag{4}$$

for some p-norm defined for $z \in \mathbb{R}^d$ by

$$||z||_p = \left(\sum_{i=1}^{d} |z_i|^p \right)^{\frac{1}{p}}. \tag{5}$$

Typical problems use $p = 1$ (Manhattan norm), $p = 2$ (Euclidean norm) or $p = \infty$ (max norm). After inserting the constraint (4) into (3), we arrive at the PDE

$$\begin{aligned} ||\nabla V(x)||_{p*} &= c(x), \quad \text{for } x \in \mathcal{D} \setminus \mathcal{T}; \\ V(x) &= 0, \qquad \text{for } x \in \partial\mathcal{T}; \end{aligned} \tag{6}$$

where $|| \cdot ||_{p*}$ is the *dual norm* [5, p. 637] of the action constraint norm $|| \cdot ||_p$ from (4). For the cases of interest, the 1-norm and ∞-norm are duals, and the 2-norm is its own dual. Once $V(x)$ is computed, the optimal path is found by gradient descent $\dot{y} = -\nabla V(y(t))/||\nabla V(y(t))||_{p*}$.

At this point the notation may have obscured the simplicity of the formulation. Consider planning a two dimensional path through the known corridors of a building to a point $\mathcal{T}$. The robot may travel at some maximum speed $s_{\max}$ in any direction; in other words $||\dot{y}||_2 \leq s_{\max}$. To put this problem in the format described above, the map is represented by the cost function defined in $\mathcal{D} \subset \mathbb{R}^2$. To use (4) as the action constraint, set $c(x) = 1/s_{\max}$ in the middle of the corridor and to some very large value inside the walls. To keep the robot away from the walls, a smooth transition of appropriate width is chosen between these two values. Then we can solve (6) with $p* = 2$ to find $V(x)$. An example of this type is described in section VI.

Now consider planning a kinematic path through configuration space for a robot arm whose state is given by d joint angles. Then $\mathcal{D} = [0, 2\pi)^d$ and $c(x)$ is defined as above to encode obstacles in $\mathcal{D}$. However, a Euclidean constraint on action is inappropriate in this case, since each joint can be actuated independently at maximum speed. Therefore, we choose to constrain the maximum speed over all joint angles using the $p = \infty$ (max) norm: $||\dot{y}||_\infty \leq s_{\max}$. Solving (6) with $p* = 1$ generates an appropriate value function, and gradient descent on this value function creates an optimal path. Apart from the simple adaptation of the algorithms to the toroidal domain $\mathcal{D}$ and the different choice of norm, this problem is the same as the one in the previous paragraph, so we do not further investigate it here due to limited space. However, in section VII we examine how mixtures of norms can be used to plan optimal paths for multiple robots.

The derivation of (6) given above was extremely informal and was intended primarily to provide intuition for the use of this equation in path planning problems. Readers should not infer from this informality a lack of rigourous theoretical support for the formulation. The general PDE (3) is known as a *Hamilton-Jacobi-Bellman* equation, and it can be modified to describe much more general types of optimal trajectory control problems [6]. The specific PDE (6) is known as an *Eikonal* equation, and it can be shown that even under the assumptions described above it may not have a classical solution that is differentiable everywhere; however, for shortest path problems there exists a unique weak solution for $V(x)$ guaranteed to be Lipschitz continuous, bounded, and differentiable almost everywhere, called the *viscosity solution* [6]. Of more relevance to path generation by gradient descent, this solution has a global minimum at the target and no other local minima, although it may contain saddle points and ridges (in both cases there are multiple paths to the target with the same optimal cost). The algorithms discussed below are designed to approximate the viscosity solution.

The constraint (4) is not quite as limiting as it first appears, since varying robot speed can be incorporated by scaling the cost function as described above. However, a significant limitation of this formulation is that the cost function cannot depend on the choice of action; this limitation rules out, for example, the planning of optimal paths for nonholonomic vehicles. We have chosen this restricted formulation in order to focus on the challenges and benefits of using different norms. With appropriate edge weighting Dijkstra's algorithm can handle anisotropic cost functions $c(x, \dot{y})$ and hence nonholonomy. Modification of fast marching to treat this situation is possible at the cost of added complexity [7] or multiple iterations [8].

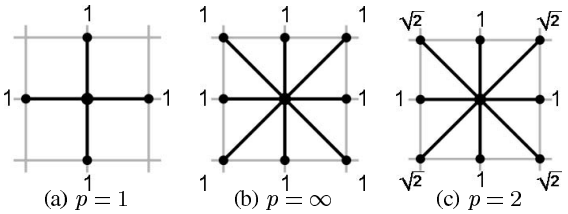
III. Related Work

Path planning is a central endeavor in robotics research [9], so we mention only the most closely related work. The value function solution of (6) is an example of a navigation function [10], and gradient descent is simply an optimal descent path through this potential field which contains no spurious local minima. There are other ways of approximating the value function. An alternative discretization of (2) is used in [11] to provide the update equation for Dijkstra's algorithm. This discretization permits more general types of motion constraints but requires calculation of the preimage set of the current reachable nodes under all available motions and the updates may involve optimizations, so the algorithm may be much more expensive than the simple, explicit algebraic equations used here. An alternative to Dijkstra's algorithm, wavefront propagation [12], keeps an active list of nodes, but does not require that the node removed from this list is always the one with minimum value. It therefore avoids the cost of sorting the list, but may need to revisit nodes multiple times.

Study has also been devoted to modifications of Dijkstra's algorithm to focus computational effort when both source and destination state are known, in which case a feedback map for the entire state space is superfluous. For example, the focused dynamic A* (D*) algorithm [13] not only avoids calculating the value function for states far from the optimal path, but also allows for updates to the value function when new map

foreach $x_i \in \mathcal{G}_n \setminus \mathcal{T}$ **do** $V(x_i) = +\infty$
foreach $x_i \in \mathcal{T}$ **do** $V(x_i) = 0$
$\mathcal{Q} \leftarrow \mathcal{G}_n$
while $\mathcal{Q} \neq \emptyset$ **do**
  $x_i \leftarrow$ `ExtractMin(`$\mathcal{Q}$`)`
  foreach $x_j \in \mathcal{N}_n(x_i)$ **do**
    $V(x_j) \leftarrow$ `Update(`x_j, $\mathcal{N}(x_j)$, V, c`)`

Algorithm 1: Generic shortest path dynamic programming algorithm. Depending on the choice of update function, this algorithm will produce either Dijkstra's algorithm or the fast marching method.



(a) $p = 1$ (b) $p = \infty$ (c) $p = 2$

Fig. 1. Local connectivity and weighting for the node at the center of each subfigure. If these patterns are extended to all nodes in the orthogonal grid $\mathcal{G}$ on $\mathcal{D} = \mathbb{R}^2$, then running the discrete Dijkstra's algorithm on $\mathcal{G}$ approximates value functions on $\mathcal{D}$ corresponding to shortest path problems under the labelled continuous p-norm bounds on the action, as shown in figure 2.

information becomes available. It should be possible to combine these methods with those outlined here to create solutions of (6) which are both focused and accurately approximate the underlying continous domain.

As an alternative to value functions, [14] uses harmonic functions to find navigation functions without local minima. While we prefer value functions because the paths generated by these harmonic function methods are not optimal in any obvious metric, it is possible that the sophisticated sampling strategy used there to decide where to refine their adaptive 2^d-tree discretization of the configuration space could improve on the simplistic grid refinement strategy used here.

The techniques outlined in this paper apply in theory to arbitrary dimension, but their grid size scales exponentially with dimension and so they are practical only in dimensions less than four or five. Probabilistic roadmap methods (PRM) [15] have demonstrated some success at path planning in high dimensional spaces. The update formulae outlined below for different norms for Dijkstra's algorithm can be used in PRM, but those for fast marching require simplicial grids and would therefore be difficult and inaccurate on probabilistic or sparse samplings. However, naive probabilistic sampling of the configuration space does not provide as good dispersion and discrepancy as some deterministic sampling strategies [16]. Consequently, PRM cannot escape the necessity of having a dense sampling of the configuration space in order to find near optimal paths and narrow channels, and so will not be as accurate and efficient for lower dimensional problems as the grid based techniques mentioned here.

IV. Dijkstra's Algorithm

Dijkstra's Algorithm (DA) [1] is an easy to program, efficient and accurate method for solving shortest path problems on a discrete graph using *dynamic programming* (DP). In this section we briefly review the algorithm and then examine its effectiveness for approximating the solution of (6) on a continuous domain for various norms.

A. Discrete Domain

Let $\mathcal{G}$ be a discrete graph with nodes $\mathcal{G}_n = \{x_i\}$ and edges $\mathcal{G}_e$. The target is a subset of nodes $\mathcal{T} \subset \mathcal{G}_n$. Each node has an associated cost $c(x_i)$ and set of neighbor nodes $\mathcal{N}_n(x_i)$. A path consists of a sequence of neighboring nodes $y(t_j) = x_{k_j}$ such that $y(t_J) \in \mathcal{T}$. Dijkstra's algorithm constructs by DP a value function $V(x_i)$ which is the discrete analog of (1): $V(x_i) = \min_{y(\cdot)} \sum_{j=0}^{J} y(t_j)$, where $y(t_0) = x_i$. Algorithm 1 gives a generic version of DP. For a set of nodes $\mathcal{Q}$, the `ExtractMin(`$\mathcal{Q}$`)` function removes from $\mathcal{Q}$ a node $x_i \in \mathcal{Q}$ with minimum $V(x_i)$. The algorithm is specialized to Dijkstra's discrete setting by using the update function

$$\texttt{Update}(x_j, \mathcal{N}(x_j), V, c) = c(x_j) + \min_{x_k \in \mathcal{N}_n(x_j)} V(x_k), \quad (7)$$

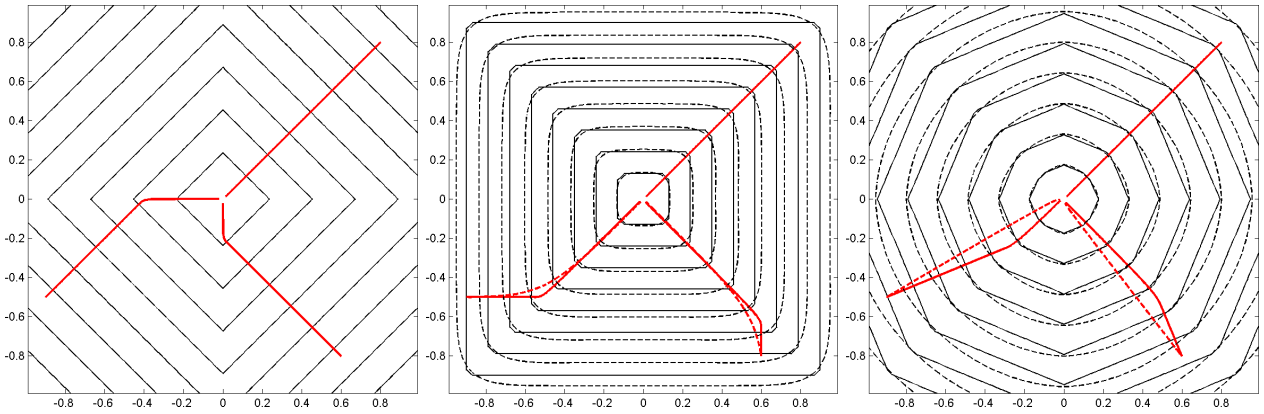
which is the discrete analog of (2). Once $V(x_i)$ is calculated for all $x_i \in \mathcal{G}_n$, the optimal path from a node can be found by the discrete analog of gradient descent: the next node in the path is always the neighbor with smallest value.

B. Continuous Domain

The physical position, configuration or state of most real robot systems lies in a continuous domain, and not at the nodes of a discrete graph. Despite this crude approximation, DA is a popular and often successful approach to optimal path planning for robots. We will examine the case where $\mathcal{G}$ is a orthogonal grid discretization of $\mathcal{D}$, and for concreteness we will work in $\mathbb{R}^2$ with node spacing Δx (so there are $\mathcal{O}(\Delta x^{-2})$ nodes in $\mathcal{G}_n$). Graph connectivity is examined below. All of the conclusions apply to higher dimensional domains, and most apply to alternative discretizations as well. The algorithm is identical to that described above, except that $c(x_i)$ should be scaled by Δx. However, two concerns arise when we consider using the resulting value function for path planning in $\mathcal{D}$.

The first concern, which is more of a mathematical detail, is that the value function is not formally defined for states in $\mathcal{D}$ which are not nodes in $\mathcal{G}$. In practice, some kind of linear interpolation among neighboring grid nodes is used to assign a value to these states. We note in passing that the `Update` function for DA was not designed with this interpolation in mind; however, we do not further analyze the resulting approximation error here.

The second concern is related to the first, but more practical. Not only do the states of typical physical robots lie in a continuous domain, but so do their actions. The set of actions considered by discrete DP for a particular node $x_i \in \mathcal{G}_n$ includes only those corresponding to edges leading to $\mathcal{N}_n(x_i)$. Interpolation of these actions to points $x \in \mathcal{D} \setminus \mathcal{G}_n$ (by



(a) $p = 1$ (b) $p = \infty$ (c) $p = 2$

Fig. 2. Contour plots of approximations to the value function $V(x)$ for $x \in \mathcal{D} = \mathbb{R}^2$ with the origin as $\mathcal{T}$ and $c(x) = 1$ for various continuous p-norm bounds on action. The solid lines are the contours of the approximation generated by Dijkstra's algorithm with the connectivities shown in figure 1, and the dashed lines are the contours of the approximation generated by the fast marching method. The curves that lead toward the center are sample paths $y(t)$ generated by gradient descent on the two value function approximations. The results are identical for $p = 1$, Dijkstra's algorithm is more accurate for $p = \infty$, and fast marching is more accurate for $p = 2$.

interpolating $\nabla V(x)$) may not lead to actions optimal under continuous constraint (4).

Fortunately, the value function generated by DA can be interpreted as the solution of (6) with continuous p-norm constraint for certain choices of p, provided that certain connectivities are chosen for $\mathcal{G}$. Figure 1 shows local views of the appropriate connectivity patterns. Connecting a node to its four nearest neighbors in the orthogonal mesh produces a value function equivalent to the $p = 1$ case, while connection to the eight nearest neighbors produces the $p = \infty$ case. Figure 2 provides contour plots of the resulting value function approximations (solid lines) for the time to reach the origin on a 101^2 grid of $\mathcal{D} = [-1, +1]^2 \subset \mathbb{R}^2$. A few sample paths $y(t)$ are also shown.

While the $p = 1$ and $p = \infty$ cases do occur, the most common norm of interest is $p = 2$. If we allow variable weighting of edges, it is possible to construct an approximation of this case using DA. A common connectivity pattern and edge weighting are shown in figure 1(c), and the resulting value function appears in figure 2(c). Note that the contours in the latter figure are octagonal, when they should be circular. This faceting is not an artifact of the coarse grid, but will persist as the grid is refined. The only way to reduce it is to add more neighbors for each node. Similar persistent inaccuracies may arise for the $p = 1$ and $p = \infty$ cases when grid spacing or the cost function is nonuniform; for further discussion see section VI.

V. The Fast Marching Method

Rather than trying to include all possible optimal paths between nodes in the discrete grid $\mathcal{G}$, an alternative is to incorporate the continuity of the domain $\mathcal{D}$ and use interpolation during the construction of the value function. We follow the approach of [2] for the $p = 2$ norm case to develop intuition, and specialize to a orthogonal grid of $\mathbb{R}^2$ for concreteness. Using the connectivity shown in figure 1(a), each node x_i will have four neighbor nodes in $\mathcal{N}_n(x_i)$. In addition, the neighbor data structure $\mathcal{N}(x_i)$ is augmented to contain in $\mathcal{N}_s(x_i)$ (s is for "simplex") the four right triangles created by connecting the nodes in $\mathcal{N}_n(x_i)$ with virtual edges.

The basic procedure is still given by algorithm 1, but with a different `Update` function. Consider the update for a node $x_0 \in \mathcal{D} \setminus \mathcal{T}$. Choose a neighbor triangle $S \in \mathcal{N}_s(x_0)$ and label its other two nodes x_1 and x_2. Using the values $V(x_1)$ and $V(x_2)$, construct a linear interpolant $V_{12}(x)$ for the value function along the (virtual) edge $[x_1, x_2]$. Then use a locally linearized version of (2) to approximate $V(x_0)$

$$V(x_0) = \min_{\tilde{x} \in [x_1, x_2]} \left(V_{12}(\tilde{x}) + c(x_0) \|\tilde{x} - x_0\|_2 \right). \qquad (8)$$

The corresponding optimal path enters S at $\tilde{x}$ and travels in a straight line to x_0.

While (8) is based on (2) and is hence quite general, we find it algebraically easier to adopt the treatment of [3] to derive our update equation for (6), as well as the name used there for this adaptation of DA: the *Fast Marching Method* (FMM). The basic idea is to plug a finite difference approximation of $\nabla V(x)$ for $x \in S \in \mathcal{N}_s(x_0)$ into (6), and then solve for $V(x_0)$ in terms of $c(x_0)$ and $V(x_i)$ for $x_i \in S \setminus \{x_0\}$. For the orthogonal mesh of $\mathbb{R}^2$ described above, a first order accurate finite difference approximation of (6) is

$$\|\nabla V(x_0)\|_2 = \sqrt{\left(\tfrac{V_0 - V_1}{\Delta x}\right)^2 + \left(\tfrac{V_0 - V_2}{\Delta x}\right)^2} = c(x_0),$$

where $V_i = V(x_i)$. This equation is quadratic in $V(x_0)$, and so can be solved easily

$$V_0|_{p*=2} = \tfrac{1}{2}\left(V_1 + V_2 + \sqrt{2\Delta x^2 c(x_0)^2 - (V_1 - V_2)^2}\right) \qquad (9)$$

It can be shown that (8) generates the same equation when applied to (6) for $p* = 2$. The full `Update` function for FMM in this setting is given in algorithm 2. Adaptation to more dimensions or variable grid spacing merely requires slight modification of (9) [3]; the triangles of $\mathcal{N}_s(x_0)$ become simplices in higher dimension.

Input: x_0, $\mathcal{N}(x_0)$, V, c
Output: $V(x_0)$
foreach $S \in \mathcal{N}_s(x_0)$ **do**
    Compute $V^{(S)}(x_0)$ from (9)
return $\min_S V^{(S)}(x_0)$

Algorithm 2: The `Update` function for the fast marching method. The use of (9) specializes this version to a orthogonal grid of $\mathbb{R}^2$ with action bounded in the $p = 2$ norm, but with appropriately modified equation it can be used for other grids, dimensions and/or norms.

Method	Grid	Figure	Path Number 1	2	3	4
DA	O	3(c)	–	2274	4989	3347
DA	S	3(d)	3145	2272	5131	3427
FMM	O	3(e)	–	2258	4906	3281
FMM	S	3(f)	3053	2252	4873	3307

TABLE I

METHOD VS TRAJECTORY LENGTH FOR THE TRAJECTORIES IN FIGURE 3. "O" IS THE ORTHOGONAL GRID AND "S" IS THE SIMPLICIAL GRID.

To treat the alternative norms, we write finite difference approximations of $||\nabla V(x)||_{p*}$

$$||\nabla V_0||_1 = \tfrac{1}{\Delta x}(|V_0 - V_1| + |V_0 - V_2|),$$
$$||\nabla V_0||_\infty = \tfrac{1}{\Delta x}\max(|V_0 - V_1|, |V_0 - V_2|),$$

plug them into (6), and rearrange to solve for $V(x_0)$

$$V_0|_{p*=1} = \tfrac{1}{2}(\Delta x c(x_0) + V_1 + V_2), \quad (10)$$
$$V_0|_{p*=\infty} = \Delta x c(x_0) + \min(V_1, V_2). \quad (11)$$

Either of these equations can be substituted for (9) in algorithm 2. Figure 2 shows contours and sample paths for value function approximations generated by FMM (dashed lines). For action constraint (4) with $p = 1$, use (11), since (6) requires the dual norm $p* = \infty$. In the resulting figure 2(a), the approximation and paths are invisible because they are identical to those generated by DA with connectivity from figure 1(a), which is not surprising since (11) is exactly the same as (7) for this choice of grid. The result in figure 2(b) for $p = \infty$, using (10) since $p* = 1$, is less accurate than the discrete approximation, and there is noticable rounding of what should be sharp corners. However, the result in figure 2(c) for $p = 2 = p*$ using (9) is much better than the corresponding discrete approximation. The contours are essentially circular and the paths are straight.

VI. ADAPTIVE GRID REFINEMENT

The discussion thus far has focused on uniform orthogonal discretizations of $\mathcal{D}$; however, the distribution of obstacles in the world is rarely uniform, and an adaptive grid which has more nodes in cluttered regions than in open regions can achieve better paths than can a uniform grid.

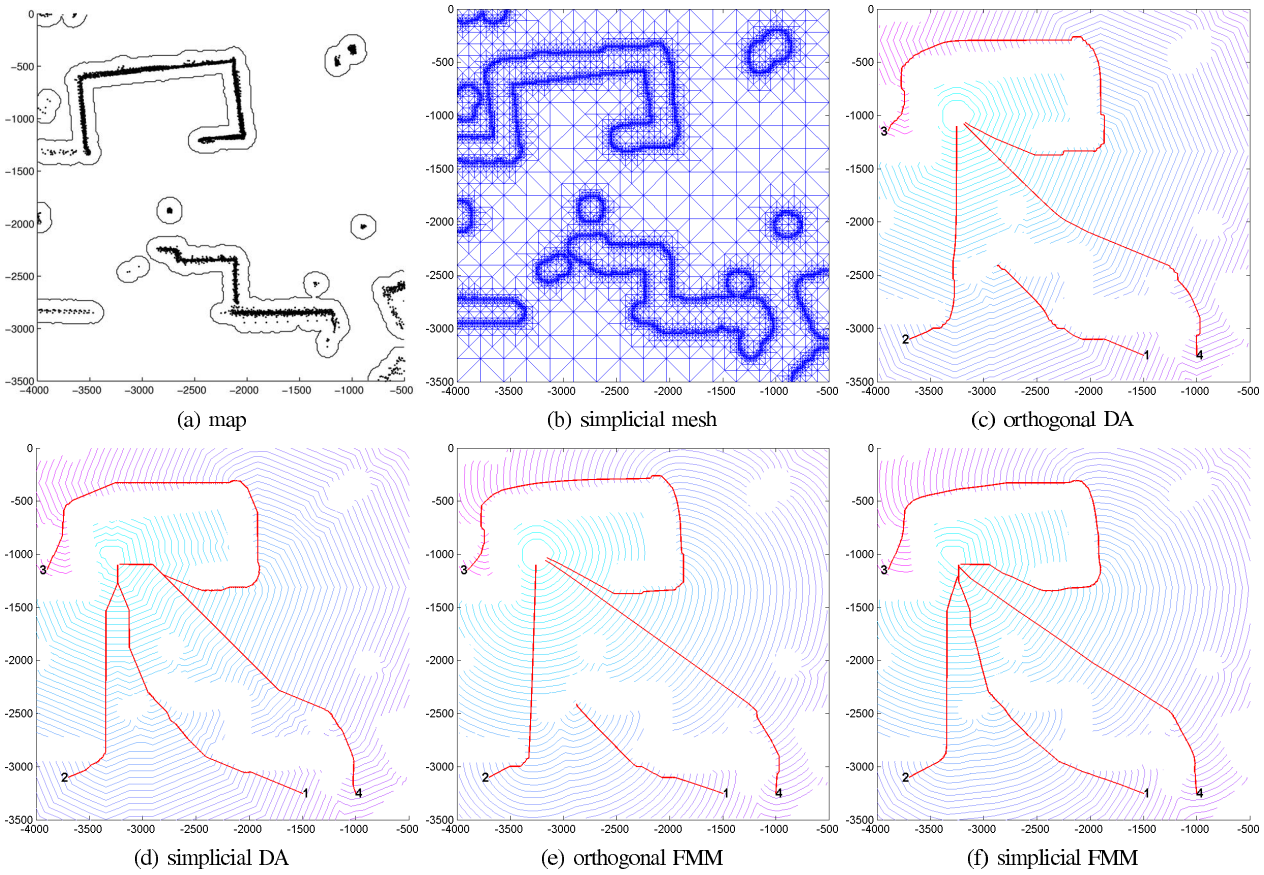
While the choice of where to place additional nodes is important, we do not investigate refinement strategies here and instead concentrate on the challenges of modifying DA and FMM to accomodate adaptive grids. For the example below, we refine the grid where the gradient of the cost function has a large magnitude. This simple strategy is easy to implement for cost functions generated by laser or sonar point data, and results in dense node placement near obstacle boundaries, where accuracy of the value function is more likely to matter when choosing robot actions.

DA easily accomodates adaptive meshes, since it makes no assumptions about the relative placement of nodes or local connectivity. However, its accuracy for the $p = 1$ and $p = \infty$ cases in the orthogonal mesh depended on the fact that the connectivity patterns in figures 1(a) and 1(b) included the optimal paths between nodes in these metrics. The discrete algorithm's failure for the $p = 2$ case occurred because it could not represent all optimal paths between nodes without turning the grid into a clique. On an adaptive grid, it may not be possible to create efficient connectivity patterns for the $p = 1$ and $p = \infty$ cases without accepting paths that are similarly suboptimal at the local level of resolution of the grid.

FMM requires more information about the local neighborhood of each node; specifically, a collection of neighboring acute simplices which cover all possible directions of approach to that node (a *simplex* in $\mathbb{R}^d$ is a convex polytope with $d+1$ vertices). The procedures in section V must be modified in two ways: a formula based on approximating $||\nabla V(x)||_{p*}$ in a general acute simplex to replace (9), (10) or (11) and find $V(x_0)$, and some method of generating simplicial grids in arbitrary dimension. An update formula to replace (9) for the $p* = 2$ case on an acute simplex is given in [17]. We are currently investigating adaptations of this formula to the $p* = 1$ and $p* = \infty$ cases. We use the simplicial mesh refinement algorithm developed in [18] to create an appropriate adaptive grid. This algorithm works in any dimension, and under mild assumptions guarantees a grid free of malformed simplices which might introduce large approximation errors.

To demonstrate on a realistic example the pros and cons of the various algorithms discussed thus far, we approximate a value function for $p = 2$ on a portion of `newoffice.map`, a collection of point obstacles based on laser rangefinder data which comes with the Saphira software package. Figure 3(a) shows the laser data for the map, as well as the outlines of the resulting obstacles in the configuration space of a circular robot of radius 100. Using a cost function to represent the obstacles, we adaptively refine to the simplicial grid shown in Figure 3(b). The resulting grid has 10326 nodes, and is compared to a uniform orthogonal grid with $101^2 = 10201$ nodes. Value function approximations for a target at $(-3250, -1000)$ are computed with both DA and FMM on both grids, and then paths from four initial locations are computed based on gradient descent. The results are shown in figures 3(c)–3(f).

The most obvious feature of the solutions is that path 1 fails to reach the target for both algorithms on the orthogonal grid, because gradient descent is foiled by inaccurate gradient approximations when traversing a gap narrower than the grid cell size. More generally, it can be seen that the adaptive grids do a better job of optimal path generation near obstacles. The fact that the orthogonal grids appear to do a better job of path generation through open spaces can be somewhat attributed to the fact that they are better refined in these open spaces, but is



(a) map (b) simplicial mesh (c) orthogonal DA

(d) simplicial DA (e) orthogonal FMM (f) simplicial FMM

Fig. 3. Robot navigation using laser point data. Figure 3(a): Laser data and obstacles in configuration space. Figure 3(b): Adaptive simplicial mesh. Figures 3(c)–3(f): Contour plots and example trajectories of value function approximations.

mostly a consequence of our simplistic refinement criteria. At present the paths approach the target from a direction in which the triangles are rather large. Additional grid refinement near the target would create better open space simplicial paths.

The difference in results between DA and FMM is a little more subtle. In practice, the paths generated following DA tend to quickly converge to grid edges, a behavior which can be seen in a few places in figures 3(c) and 3(d). Table I demonstrates that the FMM paths are quantitatively shorter.

VII. An Example of Mixed Norms

To demonstrate the use of a mixed norm, we look at the problem of coordinated control of two robots. Figure 4 shows the scenario of two robots in a ring, where the light shaded robot is constrained to move along a fixed circular path and the dark shaded robot can move freely in $\mathbb{R}^2$ subject to the black obstacles. System state variables x_1 and x_2 are the position of the 2D robot, while x_3 represents the position of the 1D (circular arc) robot. We adapt (6) to this multirobot scenario by using a $p = 2$ norm ($p* = 2$) to constrain the motion of the 2D robot and a $p = \infty$ norm ($p* = 1$) to constrain the combined motion of both robots:

$$\left\| \left(\left\| \left(\frac{\partial V(x)}{\partial x_1}, \frac{\partial V(x)}{\partial x_2} \right) \right\|_2, \frac{\partial V(x)}{\partial x_3} \right) \right\|_1 = c(x).$$

A first order accurate finite difference approximation is

$$\sqrt{\left(\frac{V_0 - V_1}{\Delta x_1}\right)^2 + \left(\frac{V_0 - V_2}{\Delta x_2}\right)^2} + \left|\frac{V_0 - V_3}{\Delta x_3}\right| = c(x_0). \qquad (12)$$

Since $c(x) > 0$ and $V_0 - V_3 > 0$, it is straightforward to solve the quadratic equation for the value V_0 at the current node.

Using this update equation with FMM on a 101^3 orthogonal grid, we compute a value function to navigate the robots from any state to a goal state. In this example, the goal for the 2D robot (dark) is on the left side of the large circular obstacle and the goal for the 1D robot (light) is on the right side. At the top and bottom of the ring are bottlenecks where only one robot may pass. The cost function encodes not only the configuration space shape of the fixed obstacles, but also those states forbidden because they represent a collision between the robots.

In figure 4(a), the robots both start at each other's goal states and avoid one another entirely by going around opposite sides of the central obstacle. In the remaining figures both robots start in the bottom half of the ring. In figure 4(b) the 2D robot is further away from the gap and therefore waits for the 1D robot to pass through first, whereas in figure 4(c) the 2D robot clears the gap and then makes way for the 1D robot.* Note

*Animations of these scenarios are available at `http://www.cs.ubc.ca/~kalton/icra2006.html`

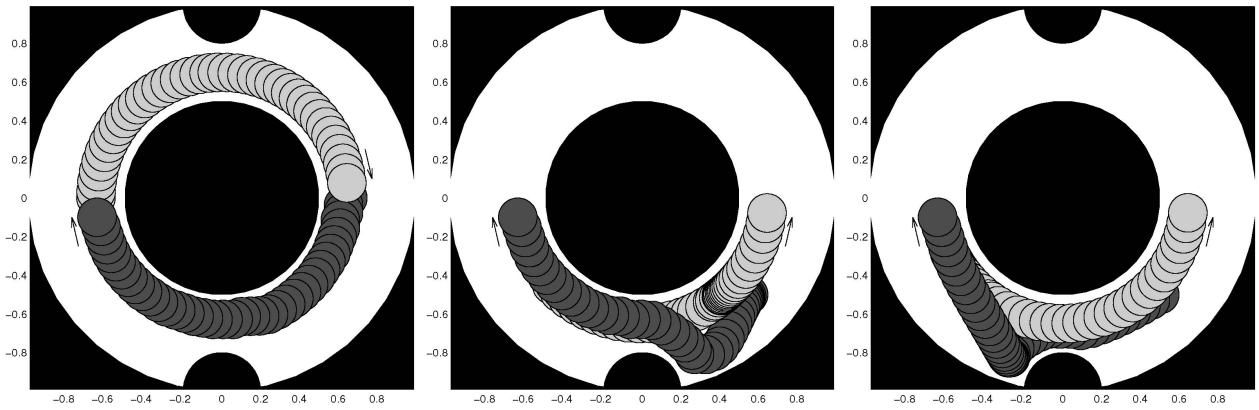


Fig. 4. Motion of two robots in a 2D world. The light robot is constrained to a circular path, while the dark robot may move anywhere within the obstacle free space. The light robot's goal is on the right, and the dark robot's goal is on the left. Arrows show direction of movement, and are placed near goal states.

that the optimal paths in all three cases are generated from the same value function approximation.

We are currently adapting (12) to unstructured grids and higher dimensions. These results could also be approximated with an appropriate grid connectivity and DA, subject to the errors discussed in section IV-B for the $p* = 2$ component of the motion (and $p* = 1$ component on unstructured grids).

VIII. Conclusion

The fast marching method is essentially Dijkstra's algorithm with a different node update formula, a formula which is consistent with the underlying continuous state space and so can represent optimal paths which do not lie on the discretized grid. As a consequence, it tends to produce shorter paths. However, there are some cases where it is less accurate—for example, the max norm constraint ($p = \infty$) on an orthogonal grid—and it does require that the grid data structure keep track of neighboring simplices (although an algorithm for generating such grids in arbitrary dimension is available). Not surprisingly, both Dijkstra's algorithm and fast marching benefited from refinement of the grid near obstacles.

Update equations for paths optimal under actions bounded in various norms were derived for both algorithms on orthogonal grids. The final example showed how mixtures of norms can be used in a multiple robot environment for generating optimal collision-free paths. We hope that for many existing path planning environments, introduction of these alternative update formulae will be straightforward.

In addition to the extensions mentioned previously, we are also investigating methods of improving the performance of the gradient descent phase of path planning.

Acknowledgement: The authors would like to thank Professor Adam Oberman for pointing out the connection between the dual norm and the Eikonal equation with bounds on robot action in alternative norms.

References

[1] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik 1*, pp. 269–271, 1959.

[2] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. AC-40, no. 9, pp. 1528–1538, 1995.

[3] R. Kimmel and J. A. Sethian, "Optimal algorithm for shape from shading and path planning," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 3, pp. 237–244, 2001.

[4] ——, "Computing geodesic paths on manifolds," *Proceedings of the National Academy of Sciences, USA*, vol. 95, no. 15, pp. 8431–8435, 1998.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

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

[7] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations," *Proceedings of the National Academy of Sciences, USA*, vol. 98, no. 20, pp. 11 069–11 074, 2001.

[8] C.-Y. Kao, S. Osher, and Y.-H. Tsai, "Fast sweeping methods for static Hamilton-Jacobi equations," *SIAM Journal on Numerical Analysis*, vol. 42, no. 6, pp. 2612–2632, 2005.

[9] J.-C. Latombe, *Robot Motion Planning*. Boston: Kluwer Academic Publishers, 1991.

[10] J. Barraquand, B. Langlois, and J. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[11] S. M. LaValle, "Numerical computation of optimal navigation functions on a simplicial complex," in *Robotics: The Algorithmic Perspective*, P. Agarwal, L. Kavraki, and M. Mason, Eds. Wellesley, MA: A K Peters, 1998, pp. 339–350.

[12] K. Konolige, "A gradient method for realtime robot control," in *Proceedings of IEEE IROS*, 2000, pp. 639–646.

[13] D. Ferguson and A. Stentz, "The delayed D* algorithm for efficient path replanning," in *Proceedings of IEEE ICRA*, 2005, pp. 2057–2062.

[14] J. Rosell and P. Iniguez, "Path planning using harmonic functions and probabilistic cell decomposition," in *Proceedings of IEEE ICRA*, 2005, pp. 1815–1820.

[15] L. E. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Proceedings of IEEE ICRA*, 1994, pp. 2138–2145.

[16] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7–8, pp. 673–692, 2004.

[17] J. Sethian and A. Vladimirsky, "Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes," *Proceedings of the National Academy of Sciences, USA*, vol. 97, no. 11, pp. 5699–5703, May, 2000.

[18] J. M. Maubach, "Local bisection refinement for n-simplicial grids generated by reflection," *SIAM Journal on Scientific Computation*, vol. 16, no. 1, pp. 210–227, Jan., 1995.