Simplicial Dijkstra and A* Algorithms: From Graphs to Continuous Spaces

Dmitry S. Yershov Steven M. LaValle

Department of Computer Science at University of Illinois at Urbana-Champaign,
201 North Goodwin Avenue, Urbana, IL 61801,
yershov2@illinois.edu *lavalle@uiuc.edu*

Abstract

This paper considers the optimal feedback planning problem of a point robot among polygonal obstacles in $\mathbb{R}^n$. In this problem the Euclidean distance traveled by the robot is minimized. The approximate optimal feedback plan is computed using a piecewise linear approximation of the cost-to-go function. The approximate cost-to-go function, in turn, satisfies the discrete version of dynamic programming principle defined using a simplicial decomposition of the environment. Adaptations of Dijkstra's and A* algorithms are introduced that solve the nonlinear system of discrete dynamic programming equations. Interpolation methods are carefully designed and analyzed so that they are proven to converge numerically. As the result, the computed feedback plan produces approximately optimal trajectories. The methods are implemented and demonstrated on 2D and 3D examples. As expected, the simplicial A* algorithm significantly improves performance over the simplicial Dijkstra's algorithm.

keywords: shortest paths, feedback motion planning, Bellman's principle, Dijkstra's algorithm, A* algorithm

1 INTRODUCTION

Computing the Euclidean shortest path to a given goal is a recurring problem in robotics. In addition to optimal robot navigation and manipulation, it is also useful in image processing, financial modeling, physics, etc. We focus on finding the shortest path between a given point and a polygonal goal set in an n-dimensional environment with polygonal obstacles. For $n = 3$ this problem is already PSPACE-hard [3]; thus, approximation methods have been developed [4, 17, 21]. We, therefore, consider only approximate shortest paths.

In robotics, algorithms that compute approximately optimal paths are based on a common approach to discretizing the problem: construct a reachability graph for a robot in a given environment using, for example, regular grids [14, 18]; and apply a graph search algorithm to find the shortest path. Although

this approach is appealingly intuitive, the computed control signals do not necessarily converge to the optimal control signal as the graph resolution increases.

An improved approach is the continuous optimal control formulation of the shortest path problem [5, 7]. The optimal feedback plan (solution to the optimal control problem) gives the shortest path when integrated. In this case, the space of possible feedback plans is discretized instead of the path space. This provides flexibility in choosing the optimal path from a continuum of all possible paths. Moreover, this approach does not require an implementation of path following controllers; instead, the motion strategy is naturally given by the feedback plan. In this sense, the control-theoretic approach can be considered as an optimal version of a motion planning using navigation functions [19].

Algorithms proposed in this paper expand upon the control-theoretic approach outlined above. The main advantages of the new method are:

- The feedback plan is defined through interpolation, thereby providing control values at every point in space, not just at a discrete set of points. Moreover, the interpolation technique is introduced for spaces of any dimension and a general simplicial decomposition of the environment, extending existing interpolation techniques beyond regular grids in 2D or 3D [21] and 2D triangulations of manifolds [13].

- The interpolation scheme inherits the causality property of the original problem. We exploit this property to build a Dijkstra-like [6] algorithm to solve the resulting system of nonlinear equations in one sweep through the domain. This provides an extremely efficient algorithm for computing a feedback plan with an asymptotic running time $O(N \log N)$ (here N is the number of vertices in the simplicial decomposition).

- In case the initial point is known, we further reduce the computational cost by proposing a continuous version of the A* algorithm [10]. Introducing the heuristic into the interpolation-based algorithm is nontrivial, and care must be taken to ensure that the system of discrete dynamic programming equations is solved correctly.

- The theoretical framework sketched in this paper provides error bounds for the proposed algorithms. This analysis paves the way for showing convergence of the approximate path to the optimal path as the resolution of the simplicial discretization is refined.

The proposed approach concerns computations only, and it closely resembles the numerical analysis framework for the Eikonal equation in [20]. In this respect, our method is different from [7, 14, 18], which address the problem of simultaneous plan computation, path execution and dynamic replanning. However, the approach can be thought of as a planner that, if embedded into simultaneous execution and replanning framework, possibly leads to an even more general interpolation-based methodology.

2 PROBLEM FORMULATION

Consider the problem of optimal feedback planning in a n-dimensional Euclidean space. High dimensional environments may arise, for example, from considering the configuration space of a robot. Assume that the robot's trajectory, $x : [0, T_f] \to \mathbb{R}^n$ (here T_f is a final time), are restricted by global constraints only. Hence, the motion model is

$$\dot{x}(t) = u(t)\,, \quad x(0) = x_{\text{init}}\,, \tag{1}$$

in which $u : [0, T_f] \to U$ is a control signal, and $U \subset \mathbb{R}^n$ is a compact input set. The global constraints, however, require $x(t) \in X_{\text{free}} = \mathbb{R}^n \setminus X_{\text{obs}}$ for all $t \in [0, T_f]$, in which X_{obs} is an open set with $(n-1)$-dimensional polygonal boundary that represents obstacles. Finally, assume the goal set, $X_{\text{goal}} \subset X_{\text{free}}$, is closed and has a nonempty interior and polygonal boundary. The problem is to find u that navigates the robot along the shortest path from x_{init} to X_{goal}, while avoiding obstacles. In this paper, we focus on computing the optimal feedback plan to find such u.

Formally a feedback plan can be described as a vector valued function $F : X_{\text{free}} \to U$, which returns the control signal at given position, that is, $u = F(x)$. Thus, the motion model (1) becomes

$$\dot{x}(t) = F(x(t))\,, \quad x(0) = x_{\text{init}}\,. \tag{2}$$

The problem of optimal feedback planning is to find F, such that x is the shortest collision-free path between x_{init} and X_{goal}. Generally, we may consider discontinuous feedback plans, and hence the Filippov solution in (2) is assumed [8].

It is well known that the optimal feedback plan can be derived from the *cost-to-go* function, $V(x)$, which satisfies Bellman's equation [2]:

$$V(x) = \liminf_{\delta \to 0} \inf_{h \in B(\delta)} \left\{ V(x+h) + \|h\| \right\}, \tag{3}$$

in which $B(\delta)$ is a ball of radius δ centered at the origin. Once $V(x)$ is known, $F(x) = -\nabla V(x)$ is the optimal feedback plan.

To summarize, the main goal of this paper is to compute the cost-to-go function, and use the result to derive the optimal feedback plan.

3 NUMERICAL APPROACH

Equation (3) admits an analytical solution in special cases. For example, if there are no obstacles in the environment, $V(x)$ is simply given by the distance function to X_{goal}. For a two-dimensional environment with polygonal obstacles, this problem can be solved exactly by visibility graph methods [12] or continuous Dijkstra method [11, 15, 16]. However, the exact solution is unknown under general conditions, and thus we must rely on a numerical approximation. In this section we construct a piecewise linear approximate solution, and prove that it converges to the exact solution.

3.1 Piecewise linear approximation

First, we construct a simplicial discretization of X_{free} by choosing a set of vertices $X_{\text{d}} = \{x_i \in X_{\text{free}} \mid 1 \leq i \leq N\}$ (a subset of X_{free}). Define an abstract simplicial complex, $\mathcal{T} = \{T \subseteq \{1, \ldots, N\}\}$, such that if $T' \subseteq T \in \mathcal{T}$, then $T' \in \mathcal{T}$. In this case, T' is called a *face* of T; if additionally $T' \neq T$, then it is called a *proper face* of T. In this regard, the notion of (proper) faces is parallel to the notion of (proper) subsets. Next, denote a geometric representation of simplex $T \in \mathcal{T}$ as X_T such that X_T is the convex hull of the set $\{x_i\}_{i \in T}$. The tuple $(X_{\text{d}}, \mathcal{T})$ is called a simplicial complex if any two simplices intersect over the common proper face only, that is, for any T and T' in $\mathcal{T}$, $X_T \cap X_{T'} = X_{T \cap T'}$. Finally, a simplicial complex discretizes X_{free}, if $\bigcup_{T \in \mathcal{T}} X_T = X_{\text{free}}$.[1] For simplicity, we refer to a simplicial discretization of X_{free} as a mesh on X_{free}.

Second, build a piecewise linear approximation $\hat{V}$ of the cost-to-go function, using a mesh on X_{free}. Let the approximation take value $\hat{V}_i$ at vertex x_i. Define $\hat{V}(x)$ by linear interpolation within any simplex T, the geometric description of which contains x:

$$\hat{V}(x) = \hat{V}\Big(\sum_{i \in T} \alpha_i x_i\Big) \triangleq \sum_{i \in T} \alpha_i \hat{V}_i \,, \tag{4}$$

in which $\alpha_i \geq 0$ for all i, and $\sum_{i \in T} \alpha_i = 1$. The values α_i are called *barycentric coordinates* of x within X_T. The approximation is completely determined by its values at the vertices of a mesh through (4).

Since a piecewise linear function cannot satisfy (3) under general conditions, we introduce a discrete version of Bellman's principle by considering (3) at points of X_{d} only:

$$\hat{V}(x_i) = \min_{T \in \mathcal{N}(i)} \inf_{x \in X_{T_i}} \Big\{ \hat{V}(x) + \big\| x_i - x \big\| \Big\} \,, \tag{5}$$

in which $\mathcal{N}(i) = \{T \in \mathcal{T} \mid i \in T\} \subset \mathcal{T}$ is a set of simplices incident to vertex x_i and $T_i = T \setminus \{i\}$ is a proper face of simplex X_T opposite vertex x_i.

To construct a fully discrete numerical method we closely follow [20] by using linear interpolation (4) to solve minimization problem (5) at each vertex. A similar discretization for the Hamilton-Jacobi-Bellman equation is introduced in [1] based on upwind differencing, whereas our approach is based directly on discretization of the dynamic programming principle (firstly introduced in [21]), and it generalizes to high-dimensional meshes.

3.2 Numerical convergence

As with most mesh-based numerical methods, *computational error* depend crucially on the mesh quality. The computational error can be defined using a function space norm of the difference between the cost-to-go function and its approximation. Unlike the standard norms defined on vector spaces, norms on function spaces are not all equivalent. Thus, it is possible that an algorithm converges in one norm but

[1] If the boundary of X_{free} is not a polygonal set, then a simplicial discretization may not exist. Although, it is still possible to find a simplicial complex such that $\bigcup_{T \in \mathcal{T}} X_T \subset X_{\text{free}}$, and the difference $X_{\text{free}} \setminus \bigcup_{T \in \mathcal{T}} X_T$ is "small". In this case the shortest path in X_{free} can be approximated by the shortest path in the simplicial complex.

not in another. In this paper, we define the computational error using sup-norm

$$\text{Error} = \sup_{x \in X_{\text{free}}} |V(x) - \hat{V}(x)| \,. \tag{6}$$

Moreover, the convergence in L_p-norm for all $p \geq 1$ follows from the convergence in sup-norm. Thus, proving the convergence in sup-norm is a stronger result.

Further, we introduce a mesh quality parameter

$$h = \max_{T \in \mathcal{T}} \max_{i,j \in T} \|x_i - x_j\| \,. \tag{7}$$

This parameter is defined by the diameter of the largest simplex in a given mesh. Additionally, h can be reduced if mesh refinement techniques are applied, for example, in 2D each triangle can be split by its midsegments into four triangles effectively reducing h by a factor of two.

Before we formulate our main result it is useful to introduce some common notation. Let, V be a solution to (3) such that the norm of the Hessian matrix of V is uniformly bounded by M in X_{free}[2]. Further, let $\hat{V}$ be a solution to (5) and $\hat{V}'$ be a piecewise linear function such that $\hat{V}'$ is linear on each X_T and $\hat{V}'(x_i) = V(x_i)$ for all $i \in \{1, \ldots, N\}$. Finally, we define $[x, x'] = \{tx + (1-t)x' | 0 \leq t \leq 1\}$ for any two points x and x'.

The next three lemmas provide the basis, essential to prove the main global error result.

Lemma 1 (Local Interpolation Error)**.** *In the notation above,*

$$|V(x) - \hat{V}'(x)| \leq Mh^2 \,. \tag{8}$$

Proof. Assume $x \in X_T$ for some simplex $T \in \mathcal{T}$, and the barycentric coordinates of x in X_T are $\{\alpha_i\}_{i \in T}$. By definition

$$x = \sum_{i \in T} \alpha_i x_i \quad \text{and} \quad \hat{V}'(x) = \sum_{i \in T} \alpha_i V(x_i) \,. \tag{9}$$

On the other hand, using Taylor series expansion, we establish that

$$V(x) = \sum_{i \in T} \alpha_i V(x) = \sum_{i \in T} \alpha_i \big(V(x_i) + \nabla V(x) \cdot (x - x_i) + \frac{1}{2} (x - x_i)^T \mathrm{H}\, V(x)(x - x_i) + \mathcal{O}(h^3) \big) \,. \tag{10}$$

Consider the second term in the expansion:

$$\sum_{i \in T} \alpha_i \nabla V(x) \cdot (x - x_i) = \nabla V(x) \cdot \Big(\sum_{i \in T} \alpha_i (x - x_i) \Big) = \nabla V(x) \cdot \Big(x - \sum_{i \in T} \alpha_i x_i \Big) = 0 \,. \tag{11}$$

Thus, it follows, from (9)–(11) and the triangle inequality, that

$$|V(x) - \hat{V}'(x)| \leq \sum_{i \in T} \alpha_i \frac{1}{2} |(x - x_i)^T \mathrm{H}\, V(x)(x - x_i)| + |\mathcal{O}(h^3)| \,. \tag{12}$$

[2]The Hessian matrix of V may not be uniformly bounded in a small neighborhood of a critical point, in which the optimal trajectory changes its direction, for example, a corner of an obstacle. Nevertheless, the set of all critical points has measure zero. Moreover, it is possible to approximate V with a smoother function V', which has the uniformly bounded Hessian matrix. Thus, we may assume without loss of generality, that V is sufficiently smooth.

In the expression above, the first term on the right-hand-side is bounded by $\frac{1}{2}Mh^2$. Moreover, we can bound the second term on the right-hand-side by $\frac{1}{2}Mh^2$, for sufficiently small h. Substituting these bounds in (12), we finalize the proof. ■

Lemma 2 (Local Minimization Problem Error). *In the notation above, for all $i \in \{1, \dots, N\}$*

$$\left| \min_{T \in \mathcal{N}(i)} \min_{x \in X_{T_i}} \left\{ \hat{V}'(x) - \hat{V}'(x_i) - \|x - x_i\| \right\} \right| \leq Mh^2 \,. \tag{13}$$

Proof. The proof follows from Lemma 1 and (3). ■

Lemma 3 (Local Dependency). *Assume $\hat{V}$ is a linear function defined on X_T for some simplex $T \in \mathcal{T}$. For all $k \in T$, let*

$$x_k^* = \arg\min_{x \in X_{T_k}} \{ \hat{V}(x) - \hat{V}(x_k) + \|x - x_k\| \} \,. \tag{14}$$

If $\nabla \hat{V}$ is a nonzero covector field, then for any $i, j \in T$ at most one of the following holds:

1. *The intersection of $[x_i, x_i^*]$ with the interior of X_T is nonempty;*
2. *The intersection of $[x_j, x_j^*]$ with the interior of X_T is nonempty.*

Proof. Since $\hat{V}$ is linear in X_T, $\nabla \hat{V}$ is a constant covector field in X_T, which we denote as $\nabla_T \hat{V}$. Thus, (14) is equivalent to

$$x_k^* = \arg\min_{x \in X_{T_k}} \left\{ \nabla_T \hat{V} \cdot \frac{x - x_k}{\|x - x_k\|} \right\} \,. \tag{15}$$

Assume to the contrary that conditions 1 and 2 hold. Thus, both $[x_i, x_i^*]$ and $[x_j, x_j^*]$ pass through the interior of X_T, and the constraints $x \in X_{T_i}$ and $x \in X_{T_j}$ are inactive[3]. Therefore, the result of (15) can be achieved considering the corresponding unconstrained minimization problems. Thus, both $x_i - x_i^*$ and $x_j - x_j^*$ must be collinear with $\nabla_T \hat{V}$, which is impossible for nondegenerate X_T and $\nabla_T \hat{V} \neq 0$. ■

Theorem 4 (Global Error). *In the notation above, there exists C such that for sufficiently small $h > 0$,*

$$\text{Error} \leq Ch \,. \tag{16}$$

Proof. From the definition of $\hat{V}$ and compactness of $\bigcup_{T \in \mathcal{N}(i)} X_{T_i}$ it follows that for all $i \in \{1, \dots, N\}$ there exist $x_i^* \in \bigcup_{T \in \mathcal{N}(i)} X_{T_i}$ such that

$$\hat{V}(x_i) = \hat{V}(x_i^*) + \|x_i^* - x_i\| \,. \tag{17}$$

Similarly, using Lemma 2, we find $x_i^{**} \in \bigcup_{T \in \mathcal{N}(i)} X_{T_i}$ such that

$$\left| \hat{V}'(x_i^{**}) - \hat{V}'(x_i) + \|x_i^{**} - x_i\| \right| \leq Mh^2 \,. \tag{18}$$

[3]See complementary slackness in Karush-Kuhn-Tucker conditions.

We define T_i^* and T_i^{**} to be such that $x_i^* \in X_{T_i^*}$ and $x_i^{**} \in X_{T_i^{**}}$.

For given x_i, consider two piecewise discontinuous paths: Let the first path be defined as $\tilde{x} : [0, K] \to X_{\text{free}}$ such that $\tilde{x}(k-0) = x_{j(k)}$ and $\tilde{x}(k+0) = x^*_{j(k)}$[4], in which $\{j(k)\}_{k=0}^K$ is such that $j(k+1) \in T^*_{j(k)}$, $j(0) = i$, and $x_{j(K)} \in X_{\text{goal}}$. An example of $\tilde{x}$ is sketched in Fig. 1. Similarly, the second path is defined as $\tilde{x}' : [0, K'] \to X_{\text{free}}$, with the difference that $\tilde{x}'(k-0) = x_{j'(k)}$ and $\tilde{x}'(k+0) = x^{**}_{j'(k)}$, in which $\{j'(k)\}_{k=0}^{K'}$ is such that $j'(k+1) \in T^{**}_{j'(k)}$, $j'(0) = i$, and $x_{j'(K')} \in X_{\text{goal}}$.

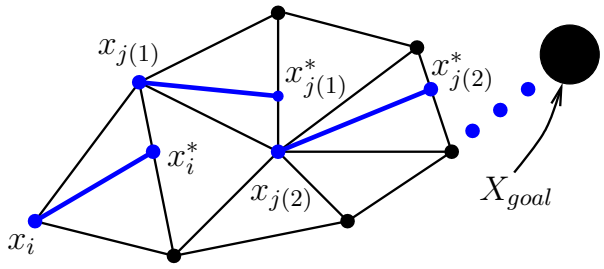

Figure 1: An example of a piecewise discontinuous path $\tilde{x}$ such that $\tilde{x}(0) = x_i$ and $\tilde{x}(K) \in X_{\text{goal}}$.

Consider the function $\hat{V}(\tilde{x}) : [0, K] \to \mathbb{R}$. On one hand, using $\hat{V}(\tilde{x}(K)) = 0$ ($\tilde{x}(K) = x_{j(K)} \in X_{\text{goal}}$), we derive

$$\int_0^K -\frac{\mathrm{d}}{\mathrm{d}t}\hat{V}(\tilde{x}(t))\,\mathrm{d}t = \sum_{k=0}^{K-1}\left(\hat{V}(x_{j(k)}) - \hat{V}(x^*_{j(k)})\right) = \hat{V}(x_i) + \sum_{k=1}^{K-1}\left(\hat{V}(x_{j(k)}) - \hat{V}(x^*_{j(k-1)})\right). \quad (19)$$

On the other hand, using the linearity of $\hat{V}$ on $[x_{j(k)}, x^*_{j(k)}]$ for all $0 \le k < K$, we derive from (17)

$$\int_0^K -\frac{\mathrm{d}}{\mathrm{d}t}\hat{V}(\tilde{x}(t))\,\mathrm{d}t = \int_0^K -\nabla\hat{V}(\tilde{x}(t)) \cdot \dot{\tilde{x}}(t)\,\mathrm{d}t =$$
$$= \sum_{k=0}^{K-1}\int_k^{k+1} -\nabla\hat{V}(\tilde{x}(t)) \cdot (x^*_{j(k)} - x_{j(k)})\,\mathrm{d}t = \sum_{k=0}^{K-1}\|x^*_{j(k)} - x_{j(k)}\| = L\,, \quad (20)$$

in which L is a total length of the line segments of $\tilde{x}$. It follows from the above that

$$\hat{V}(x_i) = \sum_{k=1}^{K-1}\left(\hat{V}(x^*_{j(k-1)}) - \hat{V}(x_{j(k)})\right) + L\,. \quad (21)$$

Similarly, considering $\hat{V}(\tilde{x}')$ and repeating the above steps, we show that

$$\hat{V}(x_i) \le \sum_{k=1}^{K'-1}\left(\hat{V}(x^{**}_{j'(k-1)}) - \hat{V}(x_{j'(k)})\right) + L'\,, \quad (22)$$

in which L' is a total length of the line segments of $\tilde{x}'$. The inequality arises due to the nonoptimality of x_j^{**} when substituted into (5).

Relations for $\hat{V}'(x_i)$ similar to (21) and (22) are established by considering functions $\hat{V}'(\tilde{x})$ and $\hat{V}'(\tilde{x}')$ and using Lemma 2 and (18):

$$\hat{V}'(x_i) \le \sum_{k=1}^{K-1}\left(\hat{V}'(x^*_{j(k-1)}) - \hat{V}'(x_{j(k)})\right) + L(1 + Mh^2) \quad (23)$$

[4]Here the notation $\tilde{x}(k-0)$ and $\tilde{x}(k+0)$ refers to the left and right limits of function $\tilde{x}$ at point k respectively.

and

$$\hat{V}'(x_i) \geq \sum_{k=1}^{K'-1} \left(\hat{V}'(x^{**}_{j'(k-1)}) - \hat{V}'(x_{j'(k)}) \right) + L'(1 - Mh^2)\,. \tag{24}$$

From relations (21) – (24), the upper and lower bounds on the difference $\hat{V}' - \hat{V}$ at point x_i are provided as follows:

$$\hat{V}'(x_i) - \hat{V}(x_i) \leq \sum_{k=1}^{K-1} \left(\hat{V}'(x^{*}_{j(k-1)}) - \hat{V}(x^{*}_{j(k-1)}) + \hat{V}(x_{j(k)}) - \hat{V}'(x_{j(k)}) \right) + LMh^2 \tag{25}$$

and

$$\hat{V}(x_i) - \hat{V}'(x_i) \leq \sum_{k=1}^{K'-1} \left(\hat{V}(x^{**}_{j'(k-1)}) - \hat{V}'(x^{**}_{j'(k-1)}) + \hat{V}'(x_{j'(k)}) - \hat{V}(x_{j'(k)}) \right) + L'Mh^2\,. \tag{26}$$

It follows from the definition of $\{j(k)\}_{k=0}^{K}$ and $\{j'(k)\}_{k=0}^{K'}$, the linearity of $\hat{V}$ and $\hat{V}'$, and Lemma 3 that $\{j(k)\}_{k=0}^{K}$ and $\{j'(k)\}_{k=0}^{K}$ have no cycles. Moreover,

$$|\hat{V}(x^{*}_{j(k-1)}) - \hat{V}'(x^{*}_{j(k-1)})| \leq |\hat{V}(x_{j(k)}) - \hat{V}'(x_{j(k)})| \tag{27}$$

and

$$|\hat{V}(x^{**}_{j'(k-1)}) - \hat{V}'(x^{**}_{j'(k-1)})| \leq |\hat{V}(x_{j'(k)}) - \hat{V}'(x_{j'(k)})|\,. \tag{28}$$

Therefore, inequalities (25) and (26) can be solved recursively in K and K'.

Note that both L and L' are bounded by the maximum geodesic distance in X_{free}, denoted as $\text{diam}(X_{\text{free}})$, and independent of h. Let C_1 be the ratio between the largest and the smallest simplex diameters, which can be fixed for various h. It follows that K and K' are bounded by $C_1\,\text{diam}(X_{\text{free}})/h$. Thus, we establish that

$$|\hat{V}'(x_i) - \hat{V}(x_i)| \leq 2C_1(\text{diam}(X_{\text{free}}))^2 Mh\,. \tag{29}$$

Finally, using Lemma 1 and the triangle inequality, the upper bound on the computational error is derived from (29):

$$\text{Error} \leq \sup_{x \in X_{\text{free}}} |\hat{V}(x) - \hat{V}'(x)| + \sup_{x \in X_{\text{free}}} |\hat{V}'(x) - V(x)| \leq 2C_1(\text{diam}(X_{\text{free}}))^2 Mh + Mh^2\,. \tag{30}$$

This result proves the theorem if we let $C = (2C_1(\text{diam}(X_{\text{free}}))^2 + 1)M$ and $h \leq 1$. ■

4 ALGORITHMIC APPROACH

The discrete dynamic programming principle (5), considered at all vertices of the discretization, describes a system of nonlinear equations. An application of standard iterative nonlinear solvers suffers from several shortcomings: it requires a sufficiently accurate initial guess, running time is high, and the result is only an approximate solution. By contrast, discrete graph search methods, such as Dijkstra's algorithm [6] or A* algorithm [10], solve a similar system of dynamic programming equations in optimal time without requiring an initial guess. In this paper we implement a modification of these algorithms to solve the given system.

4.1 Simplicial Dijkstra algorithm

We propose a Simplicial Dijkstra algorithm (SDA) that evaluates the function $\hat{V}$ in increasing order of its values using a priority queue, similarly to Dijkstra's graph search algorithm. Under minimal conditions, which we establish later in this paper, our implementation guarantees that equation (5) is solved only once for each vertex. Thus, the entire computation is done in one "sweep" through the mesh; see Algorithm 1 for details.

Algorithm 1 Simplicial Dijkstra Algorithm

Input: Simplicial complex $(X_{\mathrm{d}}, \mathcal{T})$, goal set X_{goal}

Output: Approximation of cost-to-go function, $\hat{V}_i$, at all vertices x_i of simplicial complex

```
Initialize priority queue Q of all vertex indices. Set priority key K̂_i ← 0, for all x_i ∈ X_goal, and K̂_i ← ∞, otherwise
while Q is not empty do
    Pop j with least key K̂_j from Q
    Set V̂_j ← K̂_j
    for all T ∈ 𝒩(j) do
        for all i ∈ T \ {j} do
            V̂* ← minloc(i, T, X_d)
            if V̂* < K̂_i then
                Update key of i to V̂* in Q
```

Algorithm 1 is identical to Dijkstra's graph search algorithm if the complex is a graph (i.e., a 1-complex), and **minloc** computes the distance to the neighboring vertex. In our case, however, **minloc** is defined to satisfy (5) for general meshes.

4.2 Local minimization problem

To satisfy (5), **minloc** must return a solution to the local minimization problem

$$\hat{V}^* = \inf_{\alpha_j} \left\{ \sum_{j \in T_i} \alpha_j \hat{V}_j + \Big\| x_i - \sum_{j \in T_i} \alpha_j x_j \Big\| \right\} \tag{31}$$

for any given i and T, subject to linear constraints $\alpha_j \geq 0$ for all $j \in T_i$ and $\sum_{j \in T_i} \alpha_j = 1$. Note that the local minimization problem is equivalent to the shortest path problem between vertex x_i and the proper face of simplex T opposite x_i. The terminal cost on the face is given by linear interpolation of values $\hat{V}_j$ at vertices previously computed. We propose a geometric algorithm to solve (31) exactly for simplices of any dimension; see Algorithm 2 for details.

To interpret Algorithm 2, consider a two-dimensional simplex (triangle). Assume x_3 is a vertex with unknown $\hat{V}_3$, and without loss of generality, consider $\hat{V}_1 \leq \hat{V}_2$ to be known at vertices x_1 and x_2, respectively. In this setting, the problem is to find the shortest path from x_3 to the line segment between points x_1 and x_2, given a linear terminal cost $\hat{V}$ such that $\hat{V}(x_1) = \hat{V}_1$ and $\hat{V}(x_2) = \hat{V}_2$.

Algorithm 2 Function **minloc**

Input: Vertex i, simplex T, vertex coordinates X_d

Output: Solution to minimization problem (31)

1: Restrict T to face $J \subset T$ such that all $\hat{V}_j$ for $j \in J$ are known

2: Let $\hat{V}_{\max} = \max_{j\in J} \hat{V}_j$ and $j_{\max} = \arg\max_{j\in J} \hat{V}_j$

3: Calculate normal vector $\vec{n}$ to planar section of $\hat{V}_{\max}$ level set of cost-to-go function (Fig. 2).

4: Calculate distance vector from x_i to plane orthogonal to $\vec{n}$ and passing through $x_{j_{\max}}$ (Fig. 3).

5: **if** no barycentric coordinate of distance vector within simplex is negative **then**

6: **return** $\hat{V}_{\max} + \langle x_i - x_{j_{\max}}, \vec{n}\rangle$

7: **else**

8: Restrict J to subset of non-negative barycentric coordinates and repeat from step 2

For the considered shortest path problem, the level sets of the cost-to-go function $\hat{V}$ are illustrated in Fig. 2. Each level set consists of two line segments bitangent to two circular arcs, one of which may be of zero radius. Two cases are considered: x_3 belongs to a line segment or x_3 belongs to a circular arc. In the first case, the shortest path is orthogonal to the line segment of the level set $\{x \mid \hat{V}(x) = \hat{V}_2\}$; see Fig. 3. Hence, the solution is given by the signed distance to the line segment, $\langle x_3 - x_2, \vec{n}\rangle$, plus the cost-to-go function value on the segment, $\hat{V}_2$. In the second case, the shortest path terminates either at x_1 or at x_2; see Fig. 4. Thus, the solution to the local minimization problem is the lower of $\hat{V}_1 + \|x_3 - x_1\|$ and $\hat{V}_2 + \|x_3 - x_2\|$. Finally, consider the distance vector from vertex x_3 to the line embedding the linear segment of $\{x \mid \hat{V}(x) = \hat{V}_2\}$. This vector is within the triangle in the first case, and outside otherwise. Using barycentric coordinates of the distance vector, we have thus found a criterion to distinguish between the two cases considered.

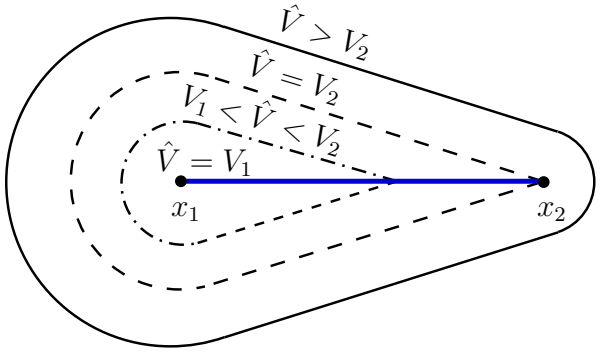

Figure 2: Level sets of $\hat{V}(x)$ consist of two line segments bitangent to two circular arcs. One of the circular arcs is of zero radius if $\hat{V}(x) \leq \hat{V}_2$.

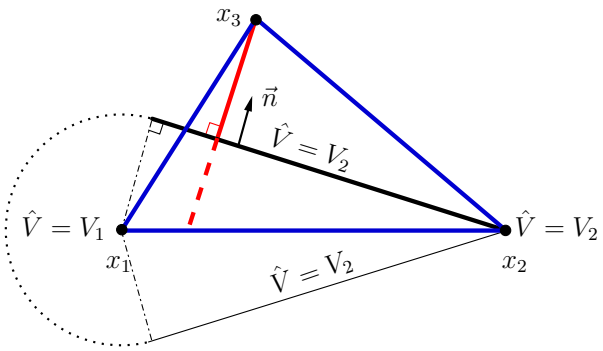

Figure 3: The shortest path intersects the linear segment of the level set. In this case $\hat{V}_3 = \hat{V}_2 + \langle x_3 - x_2, \vec{n} \rangle$, in which $\langle \cdot, \cdot \rangle$ is a scalar product of two vectors.

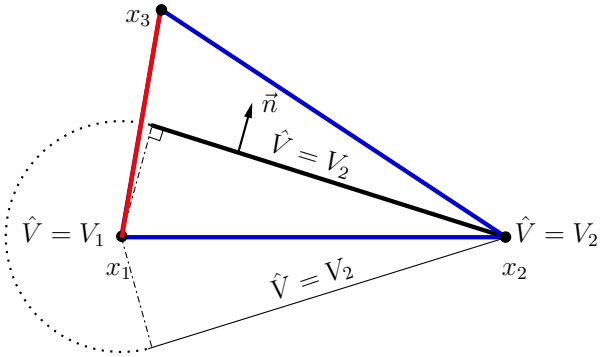

Figure 4: The shortest path intersects the circular arc of the level set. In this case $\hat{V}_3 = \hat{V}_1 + \|x_1 - x_3\|$.

4.3 Simplicial A* algorithm

The SDA outlined in Section 4.1 computes the approximate cost-to-go function in the entire environment regardless of the robot's initial configuration. If x_{init} is known, however, then it is desirable to perform costly computations only in the vicinity of the optimal path. In the discrete case, the A* graph search algorithm accomplishes this by employing a heuristic at each iteration of Dijkstra's algorithm [10]. We propose a Simplicial A* algorithm (SAA) by invoking a similar heuristic at each iteration of the SDA that narrows the focus of computations to vertices along the shortest path; see Algorithm 3 for details.

Algorithm 3 Simplicial A* Algorithm

Input: Simplicial complex $(X_{\text{d}}, \mathcal{T})$, goal set X_{goal}, initial position of robot x_{init}.

Output: Approximation of cost-to-go function, $\hat{V}_i$, in all vertices x_i in neighborhood of optimal path.

$\hat{H}_i \leftarrow$ **heuristic**(x_i, x_{init}) for all $x_i \in X_{\text{goal}}$
Initialize priority queue Q of all vertices. Set priority key $\hat{K}_i \leftarrow \hat{H}_i$, for all $x_i \in X_{\text{goal}}$, and $\hat{K}_i \leftarrow \infty$, otherwise
while Q is not empty **do**
  Pop i with least key $\hat{K}_i$ from Q
  Set $\hat{V}_i \leftarrow \hat{K}_i - \hat{H}_i$
  for all $T \in \mathcal{N}(i)$ **do**
    for all $j \in T \setminus \{i\}$ **do**
      $\hat{V}^* \leftarrow$ **minloc**(j, T, X_{d})
      $\hat{H}_j \leftarrow$ **heuristic**(x_j, x_{init})
      if $\hat{V}^* + \hat{H}_j < \hat{K}_j$ **then**
        Update key of j to $\hat{V}^* + \hat{H}_j$ in Q

Algorithm 3 is identical to Algorithm 1 in case of the trivial heuristic corresponding to no prior knowledge of the initial configuration. Although, if the heuristic approximates the *cost-to-come* function (i.e., the optimal cost of reaching the point from the initial state), then the SAA advances towards the initial configuration since $\hat{H}(x) + \hat{V}(x)$ is generally lower in this direction than in any other directions. Moreover, if the heuristic is admissible ($\hat{H}$ is an underestimate of the true cost-to-come function [10]) and consistent (covered in the next section), then the solution given by the SAA is identical to the solution given by the SDA at the evaluated vertices.

4.4 Requirements on meshes

If there is no nonnegative cycle in the graph, Dijkstra's graph search algorithm computes the cost-to-go function correctly in finite time. We refer to this property as *correctness* of Dijkstra's algorithm. Similarly to Dijkstra's algorithm, the correctness of the SDA and the SAA requires extra conditions on a simplicial mesh.

The correctness of the SDA is guaranteed, provided the discrete Bellman's principle satisfies the

causality property: for any i and j sharing a simplex, value $\hat{V}_i$ depends on value $\hat{V}_j$ if $\hat{V}_j \leq \hat{V}_i$. This property parallels to the edge weight positivity condition for Dijkstra's algorithm, which implies the nonnegative cycle condition. For the SDA, the causality property is satisfied if the following holds for (31):

$$\hat{V}^* > V_j \text{ ,for all } j \text{ such that } \alpha_j > 0\,. \tag{32}$$

The acute simplicial mesh guaranties (32). We demonstrate this for the two-dimensional case in the setting of the geometric construction from Section 4.2. First, notice that $\hat{V}^* \geq \hat{V}_1$. Second, $\hat{V}^*$ depends on $\hat{V}_2$ only if vertex x_3 belongs to a linear segment of the corresponding level set. Hence, it follows from Fig. 3 that $\hat{V}^* \geq \hat{V}_2$ if the projection of $x_3 - x_2$ on $\vec{n}$ is positive, which holds if the angle between edges incident at vertex x_3 is acute. Thus, as in [1], in 2D the correctness of the SDA is guaranteed if the triangulation is acute. Moreover, this geometric argument extends to higher dimensions, in which case we say a discretization is acute if the angles between all pairs of incident edges are acute.

The correctness of the SAA is implied by the modified causality property: for any i and j sharing a simplex, the value $\hat{V}_i$ depends on the value $\hat{V}_j$ if

$$\hat{H}_i + \hat{V}_i \geq \hat{H}_j + \hat{V}_j\,, \text{ or } \; \hat{H}_j - \hat{H}_i \leq \hat{V}_i - \hat{V}_j\,. \tag{33}$$

In (33) we replace $\hat{V}_i - \hat{V}_j$ with its minimum provided $\hat{V}_i$ depends on $\hat{V}_j$. In the two-dimensional case, the minimum is achieved if $\vec{n}$ is parallel to the side opposite x_j; see Fig. 3. Hence,

$$\hat{V}_i - \hat{V}_j \leq \|x_i - x_j\| \cos(\beta)\,, \tag{34}$$

in which β is the angle between edges incident at vertex x_i. It follows from (33) and (34) that

$$\hat{H}_j - \hat{H}_i \leq \|x_i - x_j\| \cos(\beta) \tag{35}$$

must be satisfied for correctness. If H satisfies (35), then it is called *consistent*. Higher-dimensional cases are analogous, except that β must be replaced with the maximum angle (minimum cosine) between any two edges incident at vertex x_i. In addition to the mesh requirements for the SDA, the heuristic must be consistent for the SAA correctness.

All of the requirements on meshes can be summarized in the following theorem.

Theorem 5 (Correctness)**.** *If the mesh is acute, then the SDA is correct. If additionally $\hat{H}$ is consistent, then the SAA is correct.*

Proof. The derivations above prove the theorem. ■

The *airline distance* is a commonly used heuristic for the A* algorithm [10], but it fails to satisfy (35). Nevertheless, a rescaled airline distance provides a consistent heuristic. We introduce a mesh quality parameter

$$\gamma = \min_{T \in \mathcal{T}} \min_{i,j,k \in T} \cos(\angle(x_i, x_j, x_k))\,, \tag{36}$$

in which $\angle(x_i, x_j, x_k)$ is the radian measure of the angle between vectors $x_i - x_j$ and $x_k - x_j$. The parameter γ measures the regularity of a mesh, with $\gamma < 0$ indicating there is at least one non-acute simplex and $\gamma = 1/2$ for the "perfect" 2D equilateral triangulation. It follows, from the triangle inequality and (36), that the airline distance multiplied by γ satisfies (35). Furthermore, as the mesh regularity improves, the parameter γ increases, and the rescaled heuristic becomes increasingly usable.

If parameter γ is small, then the rescaled airline distance is closer to the trivial heuristic, and the SAA has very little advantage over the SDA. Nevertheless, the rescaling coefficient and hence the quality of the heuristic can be improved significantly by implementing a virtual edge flip [1]. Figure 5 illustrates the idea of the virtual edge flip in 2D: the local minimization problem at vertex x_1 is solved for red simplices instead of the blue simplex. Red simplices are built using vertex x_4, which is opposite face (x_2, x_3) within the green simplex (blue and green simplices are required to share the face (x_2, x_3)). Using a virtual edge flip we can improve γ for a 2D equilateral triangulation up to $\sqrt{3}/2$. In higher dimensions the improvement is less pronounced, but three-dimensional experiments show that a virtual edge flip still provides reasonable $\gamma > 1/2$.

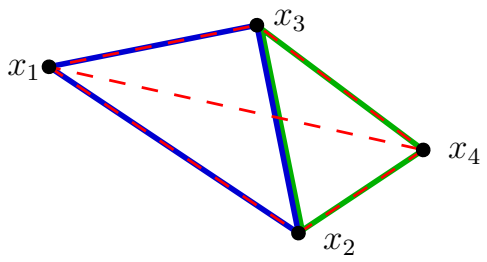

Figure 5: Virtual edge flip trick in 2D.

5 RESULTS and DISCUSSION

In this section, we investigate the convergence of the proposed numerical algorithms and their applicability to real-world problems. The convergence was tested on a simple 2D environment with a circular goal region and no obstacles. In this case, the cost-to-go function is known, and, hence, the error can be computed precisely in any standard function space norm, namely sup-norm, L_1-norm, and L_2-norm. Numerical solutions were computed using the SDA on multiple meshes with various mesh quality parameters h. The convergence results along with the slope 1 lines, as a visual aid, are plotted on Fig. 6. Our experiment confirms the linear convergence rate, proven in Theorem 4.

We further test the proposed algorithms on real-world problems by considering three different scenarios: 1) a two-dimensional environment, 2) a two-dimensional manifold, and 3) a three-dimensional environment. In all test cases, polygonal obstacles were introduced. Meshes were generated using Gmsh software [9]. The same algorithms were applied, regardless of a problem's dimensionality or topology.

Figure 7 shows level sets of the approximate cost-to-go function in the 2D environment with obstacles. The black level sets are computed using the SDA, and the white level sets are computed using the SAA.

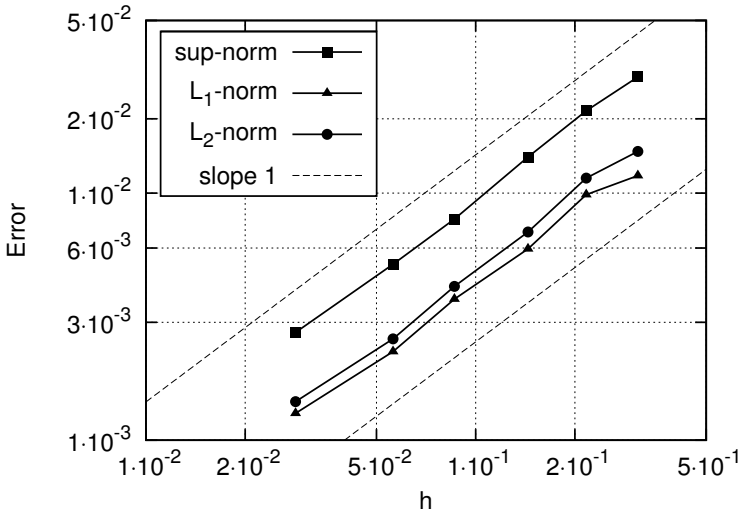

Figure 6: The plot of error in sup-norm, L_1-norm, and L_2-norm. Dashed lines are the slope 1 reference.

The thick white line surrounds vertices computed by the SAA. As we can see, implementing a heuristic focuses the SAA on vertices primarily in the direction of the robot's location. Moreover, values of $\hat{V}$ at vertices computed by the SAA are identical to those computed by the SDA, and level sets coincide. Hence, the resulting optimal paths are identical.

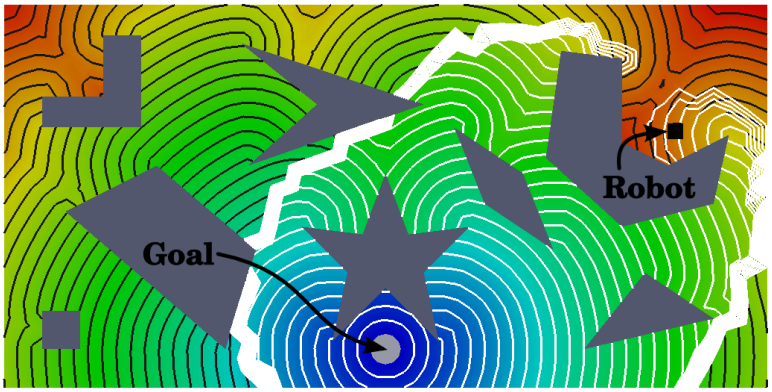

Figure 7: Level sets of $\hat{V}$ in the 2D environment with obstacles (gray) computed using the SDA (black) and the SAA (white).

Level sets of $\hat{V}$ computed on a 2D torus with obstacles are shown on Fig. 8. From this experiment, it is evident that the proposed interpolation-based approach generalizes to finding shortest paths (geodesics) on manifolds. As expected, the SAA outperforms the SDA by exploring fewer vertices.

In Fig. 9 two slices of level sets of $\hat{V}$ for a 3D environment with obstacles are illustrated. The black level sets correspond to SDA computations and white level sets correspond to SAA computations. As we can see, both algorithms extend to 3D cases. Moreover, the SDA explores more vertices than the SAA, and the trend remains in higher dimensions.

To compare the performance of the SDA vs. the SAA, we introduce two performance measures: the

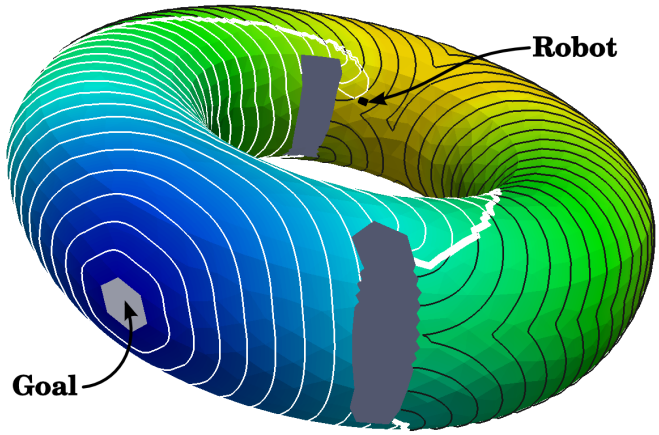

Figure 8: Level sets of $\hat{V}$ on a torus with obstacles computed using the SDA (black) and the SAA (white).

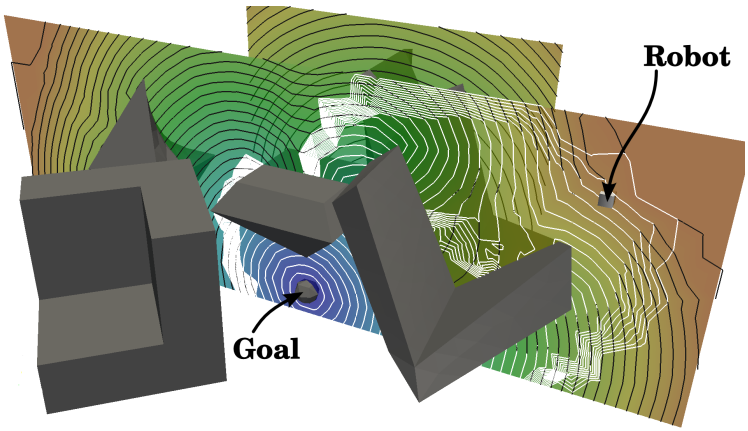

Figure 9: Level sets of $\hat{V}$ in a the 3D environment with obstacles computed using the SDA (black) and the SAA (white).

number of **minloc** function calls and the number of computed vertices. The former is a better metric since the local minimization problem is computationally expensive. The latter, however, is adequate in case memory is limited. As we can see from Table 1, the SAA consistently outperforms the SDA in both measures introduced above for all experiments considered. The running time of a Python implementation on Intel Core i7 3GHz is also illustrated in Table 1. We would like to emphasize that the proposed algorithms are not optimized for online computations, and there is space for improvement.

Table 1: Performance of SDA and SAA

Experiment	**Algorithm**	**Performance Measure**		
		minloc call #	vertex #	running time (sec)
2D obs.	**SAA**	15202	727	2.16
	SDA	31314	1539	4.16
2D torus	**SAA**	274240	11550	72.91
	SDA	557112	23395	138.35
3D obs.	**SAA**	837297	4515	131.82
	SDA	1675890	9693	426.03

6 CONCLUSIONS

In summary, we have developed an interpolation-based method for approximating the cost-to-go function associated with the Euclidean shortest path problem over a simplicial complex. We introduced simplicial versions of Dijkstra's algorithm and the A* algorithm to compute the approximate cost-to-go function from the system of the discrete dynamic programming equations efficiently. We have shown that both algorithms find a first-order accurate solution when provided with an acute mesh, and a consistent and admissible heuristic in case of the SAA. The key features of the proposed framework are:

- The implementation is independent of the dimension or topology of the environment, and it relies solely on a simplicial discretization of the environment.
- For a mesh with N vertices, both algorithms have asymptotic running time $O(N \log N)$.
- The SAA consistently explores fewer vertices and requires fewer **minloc** function calls then the SDA for the same mesh.

Acknowledgments

This work is supported in part by and NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (CNS Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052. The authors also would like to express their gratitude to Prof. Michael T. Heath for valuable input on an earlier draft of this paper.

REFERENCES

[1] T. J. Barth and J. A. Sethian. Numerical Schemes for the HamiltonJacobi and Level Set Equations on Triangulated Domains. *Journal of Computational Physics*, 145(1):1–40, September 1998. 4, 13, 14

[2] Dimitri P. Bertsekas. *Dynamic Programming & Optimal Control, Vol. I.* Athena Scientific, 3rd edition, May 2005. 3

[3] John Canny and John Reif. New lower bound techniques for robot motion planning problems. In *Proceedings of 28th Annual Symposium on Foundations of Computer Science*, pages 49–60, October 1987. 1

[4] Joonsoo Choi, Jürgen Sellen, and Chee K. Yap. Approximate Euclidean shortest path in 3-space. In *Proceedings of the tenth annual symposium on Computational geometry*, SCG '94, pages 41–48, New York, NY, USA, 1994. ACM. 1

[5] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-Angle Path Planning on Grids. *Journal of Artificial Intelligence Research*, 39:533–579, 2010. 2

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959. 2, 8

[7] Dave Ferguson and Anthony Stentz. Field D*: An Interpolation-Based Path Planner and Replanner. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, chapter 22, pages 239–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 2

[8] A. F. Filippov. *Differential Equations with Discontinuous Righthand Sides: Control Systems (Mathematics and its Applications).* Springer, 1 edition, September 1988. 3

[9] Christophe Geuzaine and Jean-Franois Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009. 14

[10] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, February 1968. 2, 8, 12, 13

[11] J. Hershberger and S. Suri. Efficient computation of Euclidean shortest paths in the plane. In *Proceedings of 34th Annual Symposium on Foundations of Computer Science*, pages 508–517, 1993. 3

[12] S. Kapoor, S. N. Maheshwari, J. S. B. Mitchell, S. Kapoor, S. N. Maheshwari, and J. S. B. Mitchell. An Efficient Algorithm for Euclidean Shortest Paths Among Polygonal Obstacles in the Plane. *Discrete & Computational Geometry*, 18(4):377–383, December 1997. 3

[13] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences of the United States of America*, 95(15):8431–8435, July 1998. 2

[14] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 968–975, 2002. 1, 2

[15] Joseph S. B. Mitchell. Shortest paths among obstacles in the plane. In *Proceedings of the ninth annual symposium on Computational geometry*, SCG '93, pages 308–317, New York, NY, USA, 1993. ACM. 3

[16] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The Discrete Geodesic Problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. 3

[17] Christos H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20(5):259–263, June 1985. 1

[18] M. Pivtoraiko and A. Kelly. Fast and Feasible Deliberative Motion Planner for Dynamic Environments. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, May 2009. 1, 2

[19] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992. 2

[20] J. A. Sethian and A. Vladimirsky. Ordered Upwind Methods for Static Hamilton–Jacobi Equations: Theory and Algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003. 2, 4

[21] John N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *Automatic Control, IEEE Transactions on*, 40(9):1528–1538, August 1995. 1, 2, 4