Robotics and Autonomous Systems 70 (2015) 202–214

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Direction-dependent optimal path planning for autonomous vehicles

Alex Shum [a,*], Kirsten Morris [a], Amir Khajepour [b]

[a] Department of Applied Mathematics, University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1, Canada

[b] Department of Mechanical and Mechatronics Engineering, University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1, Canada

HIGHLIGHTS

- Optimal rover path planning is extended to consider direction in tip-over risk.
- Solar energy in net consumed energy is also considered in the optimal problem.
- The Ordered Upwind Method (OUM) is used to solve the rover path planning problem.
- A novel bi-directional (OUM) is introduced and is faster than the original algorithm.
- The bi-directional OUM is shown to outperform genetic algorithm and Bi-RRT*

ARTICLE INFO

Article history:
Received 5 February 2014
Received in revised form
3 December 2014
Accepted 4 February 2015
Available online 12 February 2015

Keywords:
Rover path planning
Ordered Upwind Method
Tip-over stability axes
Optimal path planning

ABSTRACT

The optimal path planning problem is considered for rovers. Tip-over risk is accurately modelled using direction dependence. In the previous direction-independent model, the value function was approximated using the Fast Marching Method (FMM). The risk was not accurately modelled. Solar energy is considered here for the first time. Minimizing path length, obstacle avoidance and soil risk are also considered. For a direction-dependent model, the value function in the optimal path planning problem can be approximated accurately using the Ordered Upwind Method (OUM) but not FMM. The value function is used to synthesize the optimal control, which is shown to have no local minima. A novel algorithmic improvement, OUM-BD over the OUM to include a bi-directional search is presented. The OUM-BD is slightly slower than the FMM, but can accurately solve a larger class of problems. The OUM-BD is faster than the existing OUM, an optimal bi-directional RRT path planner (Bi-RRT*), and a genetic algorithm (GA) path planner in terms of time, and outperforms both the GA and Bi-RRT* planner in cost in tested examples.

1. Introduction

Efficient and safe path planning is an important area of research for autonomous vehicles. The motivating application for the research described in this paper is path planning for rovers. However, the problem formulation and its solution can also be applied to path planning of other autonomous vehicles.

A path planning problem where the continuous environment is approximated by an undirected graph can be solved optimally using Dijkstra's [1] or A* [2] algorithms. The optimal path is found on the edges of the graph. Since the directions of travel are restricted to the edges, the optimal solution(s) of the discrete problem do not in general converge to the optimal solutions of the continuous problem even as the graph is refined [3].

Mixed-Integer Linear Programming (MILP) is often used to solve a discrete-time path planning problem for unmanned aerial vehicles (UAVs) [4–7]. In each time step, the time required to reach a goal position from its current position using a fixed number of waypoints is minimized. The dynamics of the vehicle are described using linear constraints. A set of linear and integer constraints are used to model both reaching the goal and polygonal obstacles. A cluttered environment for the MILP problem results in a significant increase in complexity. MILP is often used with receding horizon control to reduce computational effort [6,7]. The goal constraints are removed and replaced by a weight in the objective function. The resulting path is no longer globally optimal [4]. The path planning problem for rovers here places a strong emphasis on the properties of the terrain and is not easily described by the problem formulation of the MILP problem.

In rapidly-exploring random trees (RRTs) [8], collision-free connections are added to a growing tree data structure that stems from the start configuration. The goal configuration is added to the tree

* Corresponding author. Tel.: +1 519 888 4567x39244.
E-mail addresses: a5shum@uwaterloo.ca (A. Shum), kmorris@uwaterloo.ca (K. Morris), akhajepour@uwaterloo.ca (A. Khajepour).

http://dx.doi.org/10.1016/j.robot.2015.02.003
0921-8890/

when the growing tree reaches close enough to the goal configuration. A collision-free path is found using backtrack along the tree. A more efficient bi-directional algorithm called RRT-connect [9] grows a tree from each of the start and goal configuration, connecting the points of the two trees when they become close to one another. More recently, optimality for the RRT algorithm in RRT* [10] and its bi-directional variant Bi-RRT* [11] have been considered by rewiring existing connections on the tree, favouring connections with lower cost. Both RRT* and Bi-RRT* are asymptotically optimal, that is, the best path will converge to an optimal solution given an infinite number of iterations.

The continuous path planning problem can also be formulated as an optimal control problem [12], where aspects of the environment are modelled by a weight function. The value function according to the continuous dynamic programming principle (DPP) is first found. The optimal path is found by using the value function as a navigation function and solving an ODE related to the corresponding static Hamilton–Jacobi–Bellman (HJB) equation. Navigation functions may have local minima leaving the robot stuck without finding a feasible path [13]. Under a weak assumption on the weight function, the value function has no local minima [14]. Thus, the optimal path can always be recovered from the value function.

The static HJB equation reduces to the Eikonal equation for a direction-independent weight. In this case, the solution can be approximated using the Fast Marching Method (FMM) [15]. Such weights include obstacle avoidance and shortest path [16,17]. Weights for soil type and solar energy absorption as presented in this paper are also direction-independent. The solution is finalized one grid/mesh point at a time. The FMM applied to problems with weights that depend on direction will lead to residual error [12].

The Ordered Upwind Method (OUM) [12] was introduced to solve both a time-invariant continuous dynamic programming principle (DPP) problem, as well as a static Hamilton–Jacobi equation. In both these instances, the weight could have varied with not only position, but also direction. In the work here, the formulation regarding the DPP is considered. The OUM produces an approximation to the solution of the static Hamilton–Jacobi–Bellman equation [12] on an unstructured mesh. It has been shown [12] that the approximation produced by the OUM converge to the true solution of the static HJB equation as the mesh is refined. The characteristics to the solution of the Hamilton–Jacobi–Bellman equation are optimal paths given specific boundary conditions, and hence characteristics of the approximated solution from OUM are approximations of the optimal paths.

In this paper, the OUM is used to solve path planning problems for rovers. Contributions include a more accurate model of tip-over risk and the introduction of solar energy used in optimal path planning for rovers. Minimizing path length, obstacle avoidance and soil risk are also considered. A novel algorithmic improvement, OUM-BD, combining a bi-directional search using OUM is introduced. The OUM-BD is compared against OUM, FMM (for applicable problems), a genetic algorithm (GA) path planner for rovers [18], and Bi-RRT* [11] in both performance and timing. The OUM-BD is observed to be slightly slower than the FMM, but much faster than the original OUM. All of FMM, OUM-BD and OUM outperform GA and Bi-RRT* in timing and performance in all examples.

The path planning problem for rovers will be presented in Section 2. The optimal control problem, continuous DPP and static HJB are described in Section 3. In Section 4, a brief review of the Ordered Upwind Method will be presented. The OUM-BD algorithm to find the optimal path will be proposed in Section 5. Numerical examples will be presented in Section 6. Conclusions and extensions for future work are discussed in Section 7. Finally, the solution of the static HJB equation will be shown to have no local minima in the Appendix.

2. Problem statement

The environment, also known as the workspace, in which the path is planned is assumed to be fully known. For the rover path planning problem, it is assumed that the elevation of the terrain, position of the sun and locations of potential obstacles are all known. Let $|\cdot|$ denote the Euclidean norm.

Definition 2.1. The **workspace** $\Omega \subset \mathbb{R}^2$ is a closed, bounded and convex set of possible locations of the vehicle.

Let $\mathbf{x} : \mathbb{R}_+ \to \mathbb{R}^2$ describe the trajectory of the vehicle. For notational purposes, $\mathbf{x}(t)$ will denote the trajectory at time t and $\mathbf{x}$ will denote a point in $\mathbb{R}^2$. The direction in which the vehicle is facing is defined $\mathbf{u} \in \mathcal{U}$, where $\mathcal{U} = \{\mathbf{u}(\cdot) : \mathbb{R}_+ \to \mathbb{S}^1 | \mathbf{u}(\cdot) \text{ is measurable}\}$ and $\mathbb{S}^1 = \{\mathbf{u} \in \mathbb{R}^2 \,\big|\, |\mathbf{u}| = 1\}$. The dynamics of the rover for $t \geq 0$ are

$$\dot{\mathbf{x}}(t) = \mathbf{u}(t) \tag{1}$$
$$\mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}_0 \in \Omega.$$

Rovers have slow maximum speed (5 cm/s) and assume this speed quickly [19].

The control problem is to find a control $\mathbf{u}(\cdot) \in \mathcal{U}$ such that the vehicle will travel from a start position, $\mathbf{x}_0$ to a final position, $\mathbf{x}_f$ under the dynamics of the system (1).

Definition 2.2. The **exit-time** $T : \Omega \times \mathcal{U} \to \mathbb{R}_+$ is

$$T(\mathbf{x}_0, \mathbf{u}(\cdot)) = \inf\{t \in \mathbb{R}_+ | \, \mathbf{x}(t) = \mathbf{x}_f\}.$$

The exit-time is the first time the vehicle reaches $\mathbf{x}_f$ (from $\mathbf{x}_0$) under the influence of $\mathbf{u}(\cdot)$.

Definition 2.3. The **cost function**

$$\text{Cost}(\mathbf{x}_0, \mathbf{u}(\cdot)) = \int_0^{T(\mathbf{x}_0, \mathbf{u}(\cdot))} g(\mathbf{x}(s), \mathbf{u}(s)) ds, \tag{2}$$

where $\mathbf{x}(T(\mathbf{x}_0, \mathbf{u}(\cdot))) = \mathbf{x}_f$ and the **weight function** $g : \Omega \times \mathbb{S}^1 \to \mathbb{R}_+$ is a real-valued continuous function that satisfies

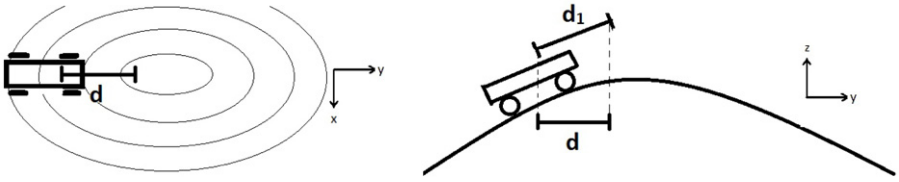
$$0 < G_{min} < g_{min}(\mathbf{x}) \leq g(\mathbf{x}, \mathbf{u}) \leq g_{max}(\mathbf{x}) < G_{max} < \infty,$$
$$\text{for every } (\mathbf{x}, \mathbf{u}) \in \Omega \times \mathbb{S}^1, \tag{3}$$

where $g_{min}(\mathbf{x}) = \min_{\mathbf{u} \in \mathbb{S}^1} g(\mathbf{x}, \mathbf{u})$ and $g_{max}(\mathbf{x}) = \max_{\mathbf{u} \in \mathbb{S}^1} g(\mathbf{x}, \mathbf{u})$ are continuous, and G_{min} and G_{max} are positive constants.

The assumption that there is a maximum bound on the weight G_{max} over Ω to traverse a point is equivalent to small-time local controllability (STLC) over all of Ω. States with infinite weight cannot be traversed and in general cannot be part of the solution to any path planning problem. This assumption is not reasonable in general for wheeled robots with a minimum turning radius. The STLC property is however justified for rovers which can turn in place. Since the problem formulation has no fixed exit-time, a negative weight may imply that the cost is not bounded below. Regions of weight zero can imply arbitrarily large exit-times (the rover remains in such regions without accruing additional cost). The assumption of a positive weight ensures that there are no positions that can be traversed for zero cost.

The weight function $g(\mathbf{x}, \mathbf{u}) \equiv 1$, for all $(\mathbf{x}, \mathbf{u}) \in \Omega \times \mathbb{S}^1$ corresponds to the optimal control problem for shortest time (and hence shortest path, since the vehicle travels at constant speed). The shortest path is not necessarily the optimal path.

The above formulation applies to path planning for any autonomous vehicle. Weightings specific to rovers are considered in this paper, including shortest path on terrain, obstacle avoidance, soil risk, solar energy input and tip-over stability risk. Some specific types of weights are as follows.



(a) Level sets of terrain $z(\mathbf{x})$, two-dimensional view (xy-plane).

(b) Terrain $z(\mathbf{x})$, three-dimensional view (yz-plane).

Fig. 1. Path Length on Terrain—The rover is traversing a hill. The true distance traversed d_1 in the direction of greatest increase is larger than the distance projected onto the xy-plane, d.

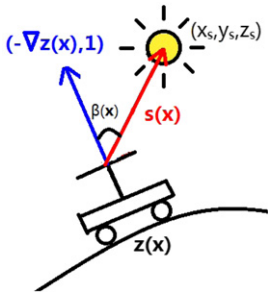


Fig. 2. Solar energy definitions—The sun is located at (x_s, y_s, z_s). The function $\beta(\mathbf{x})$ is the angle in radians between the normal to the tangent plane $(-\nabla z(\mathbf{x}), 1)$ and the vector between the rover's panel and the sun $\mathbf{s}(\mathbf{x})$.

Definition 2.4. The **terrain** $z : \Omega \to \mathbb{R}$ is a continuously differentiable function representing the elevation experienced by the rover on the set Ω.

Path Length on terrain. The weight $g(\mathbf{x}, \mathbf{u}) \equiv 1$ on Ω corresponds to the shortest path on Ω, however the rover must often traverse uneven terrain. The position of the rover on terrain is constrained to $(\mathbf{x}, z(\mathbf{x}))$ for $\mathbf{x} \in \Omega$. On a hill, a small distance d in the direction of greatest increase of z on the xy-plane is longer than its projection d_1 onto the terrain (see Fig. 1). As in [17], the problem here is solved directly on a discretization that approximates the surface.

Obstacles. Obstacles are regions in Ω where travel is forbidden. These include impassable regions or high-risk zones such as quicksand. Let $O \subset \Omega$ be the region(s) of obstacles. An obvious weight function is to use a large value M for obstacles, and 0 outside of obstacles. To maintain continuity of g, a buffer zone where the danger decreases gradually away from the obstacle can be included. Obstacle weights are independent of direction. Another implementation of obstacles is to consider the path planning problem on $\Omega \setminus O$.

Soil Risk. The soil risk is a measure of the difficulty to traverse a type of soil. For example, rocky soils are easier to manoeuvre than soft sand. The weight can be independent of direction where a value is provided for each type of soil. A weight that considers the risk associated with the slope experienced on a particular soil is dependent of direction.

Solar Energy. Energy absorbed by a solar panel is a source of power for rovers. For a panel parallel to the frame of the rover, the weight is independent of direction. The normal vector of the panel is $(-\nabla z(\mathbf{x}), 1)$. The sun, located at (x_s, y_s, z_s), will be at the same angle to the panel for a position $\mathbf{x}$ regardless of direction. The vector from the rover to the sun is

$$\mathbf{s}(\mathbf{x}) = (x_s, y_s, z_s) - (\mathbf{x}, z(\mathbf{x})),$$

while the angle between the vector normal to the tangent plane at $z(\mathbf{x})$ and $\mathbf{s}(\mathbf{x})$ is

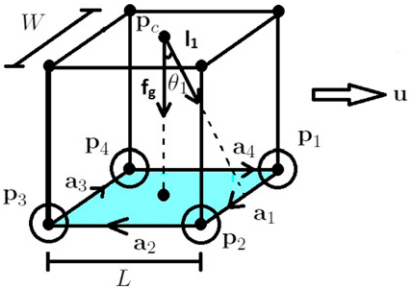
$$\beta(\mathbf{x}) = \arccos\left(\frac{\mathbf{s}(\mathbf{x}) \cdot (-\nabla z(\mathbf{x}), 1)}{|\mathbf{s}(\mathbf{x})||(-\nabla z(\mathbf{x}), 1)|}\right).$$

A smaller angle $\beta(\mathbf{x})$ will result in more direct sunlight reaching the panel (see Fig. 2). If $\beta(\mathbf{x})$ is obtuse, then no direct sunlight will reach the panel.

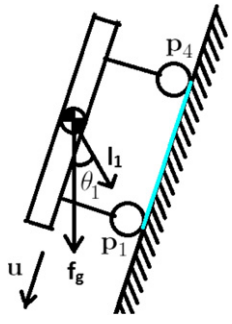
If the panel is not parallel to the rover frame, the weight is dependent on the direction in which the rover is facing.

Stability tip-over risk. The risk due to tip-over is dependent on direction. The rover is in a stable position if the downward projection of its centre of gravity onto the terrain lies within the convex hull defined by the rover's contact points (see Fig. 3). Assume that the convex hull of the contact points made by the rover on the terrain is rectangular. A common model for rovers known as the rocker-bogie model [18] satisfies this assumption. Assume also that the centre of mass is at its centroid.

A suitable measure of tip-over risk is the force–angle stability margin described in [20]. Suppose the rover is at a position $\mathbf{x}$



(a) Convex hull (shaded) of rover contact points with terrain. The rover has length L, and width W. The tip-over stability measure θ_1 is shown for the side $\mathbf{p}_1\mathbf{p}_2$. The vector $\mathbf{f}_g$ is pointing into the convex hull made by the contact points. Hence the stability measure is positive.



(b) A rover is on tilted ground. The projection of the centre of mass falls outside of the convex hull made of the contact points with the terrain, resulting in tip-over. The stability margin is the negative angle θ_1. A negative angle indicates that the orientation of the rover is unstable.

Fig. 3. Tip-over stability risk.

facing a direction $\mathbf{u}$. Let the centre of gravity of the rover be $\mathbf{p}_c = (x_c, y_c, z_c)$ and the four corners of the convex hull be labelled clockwise (viewed from above): $\mathbf{p}_i$, $i \in \{1, 2, 3, 4\}$ (see Fig. 3a). Let the tip-over axes be

$$\mathbf{a}_i = \mathbf{p}_{i+1} - \mathbf{p}_i, \quad \text{for } i = \{1, 2, 3\}, \qquad \mathbf{a}_4 = \mathbf{p}_1 - \mathbf{p}_4$$

and $\hat{\mathbf{a}}_i = \frac{\mathbf{a}_i}{|\mathbf{a}_i|}$. The normal vector to tip-over axis $\mathbf{a}_i$ that intersects the centre of gravity is

$$\mathbf{l}_i = (\mathbf{I}_3 - \hat{\mathbf{a}}_i \hat{\mathbf{a}}_i^T)(\mathbf{p}_{i+1} - \mathbf{p}_c) \quad \text{for } i = \{1, 2, 3\},$$
$$\mathbf{l}_4 = (\mathbf{I}_3 - \hat{\mathbf{a}}_4 \hat{\mathbf{a}}_4^T)(\mathbf{p}_1 - \mathbf{p}_c).$$

For $i \in \{1, 2, 3, 4\}$, let $\hat{\mathbf{l}}_i = \frac{\mathbf{l}_i}{|\mathbf{l}_i|}$ and let $\hat{\mathbf{f}}_g = [0, 0, -1]$. The force–angle stability measure corresponding to each tip-over axis is

$$\theta_i = \sigma_i \cos^{-1}(\hat{\mathbf{f}}_g \cdot \hat{\mathbf{l}}_i)$$

where

$$\sigma_i = \begin{cases} +1 & \text{if } \left(\hat{\mathbf{l}}_i \times \hat{\mathbf{f}}_g\right) \cdot \hat{\mathbf{a}}_i < 0 \\ -1 & \text{otherwise.} \end{cases}$$

Instability occurs when θ_i is negative, indicating that the vectors $\hat{\mathbf{f}}_g$ and $\hat{\mathbf{l}}_i$ have switched sides (compared to when θ_i was positive), projecting the centre of gravity outside of the convex hull. The overall stability margin α at the position $\mathbf{x}$ with direction $\mathbf{u}$ is given by

$$\alpha(\mathbf{x}, \mathbf{u}) = \min_i \theta_i, \quad i \in \{1, 2, 3, 4\}. \tag{4}$$

The weights described above will be used for path planning in the examples in Section 6.

3. Optimal path planning

Definition 3.1. The control $\mathbf{u}^*(\cdot) \in \mathcal{U}$ is **optimal** if it minimizes the cost function (2) subject to (1).

Definition 3.2. The **value function** $V : \Omega \to \mathbb{R}$ at $\mathbf{x} \in \Omega$ is the cost associated with the optimal control $\mathbf{u}^*(\cdot)$ for reaching the final position $\mathbf{x}_f$ from $\mathbf{x}$,

$$V(\mathbf{x}) = \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \text{Cost}(\mathbf{x}, \mathbf{u}(\cdot)). \tag{5}$$

The value function satisfies the continuous *Dynamic Programming Principle* (DPP).

Theorem 3.3 (*Dynamic Programming Principle [21, Theorem 10.3.1]*)**.** *For $h > 0$, $t \geq 0$, such that $0 \leq t + h \leq T(\mathbf{x}, \mathbf{u}^*(\cdot))$,*

$$V(\mathbf{x}(t)) = \inf_{\mathbf{u}(\cdot) \in \mathcal{U}} \left\{ \int_t^{t+h} g(\mathbf{x}(s), \mathbf{u}(s)) ds + V(\mathbf{x}(t+h)) \right\}. \tag{6}$$

To obtain the optimal path, the following static Hamilton–Jacobi–Bellman equation is required. For points $\mathbf{x} \in \Omega$ where $\nabla V(\mathbf{x})$ is defined,

$$\min_{\mathbf{u} \in \mathbb{S}^1} \{(\nabla V(\mathbf{x}) \cdot \mathbf{u}) + g(\mathbf{x}, \mathbf{u})\} = 0, \quad \mathbf{x} \in \Omega \setminus \{\mathbf{x}_f\}, \tag{7}$$
$$V(\mathbf{x}_f) = 0.$$

The solution to the static HJB (7) is the value function described in the DPP (6) [22]. If $\nabla V(\mathbf{x})$ is undefined, V will satisfy (7) in the weaker sense of viscosity solutions [21, Chapter 10]. Combining (1) and (7),

$$\dot{\mathbf{x}}(t) = \mathbf{u}^*(t) = \arg\min_{\mathbf{u} \in \mathbb{S}^1} \{\nabla V(\mathbf{x}(t)) \cdot \mathbf{u} + g(\mathbf{x}(t), \mathbf{u})\}, \tag{8}$$
$$\mathbf{x}(0) = \mathbf{x}_0.$$

If the weight function is of the form $g(\mathbf{x}, \mathbf{u}) = g(\mathbf{x})$, then the minimizing control of (8) is $\mathbf{u}^*(t) = -\frac{\nabla V(\mathbf{x}(t))}{|\nabla V(\mathbf{x}(t))|}$. The optimal path can be found by following the negative gradient of ∇V. The HJB equation (7) trivially reduces to the simpler Eikonal equation

$$|\nabla V(\mathbf{x})| = g(\mathbf{x}), \quad \mathbf{x} \in \Omega \setminus \{\mathbf{x}_f\} \tag{9}$$
$$V(\mathbf{x}_f) = 0.$$

If the weight function does depend on direction, then a minimization problem must be solved to find $\mathbf{u}^*$ in (8). An optimal direction $\mathbf{u}^*$ (8) can be defined for each position $\mathbf{x}$ to navigate the rover if ∇V is defined. In this process, V is known as a navigation function.

Since path recovery uses gradient descent in (9) for the direction-independent case and a similar process for the direction-dependent case in (8), positions $\mathbf{x} \in \Omega \setminus \{\mathbf{x}_f\}$ that are local minima of V are problematic. The path recovery process would become trapped at these local minima, not yielding a feasible path. Navigation functions in general may have positions that are local minima, local maxima and saddle points such as the Artificial Potential Field formulation in [13]. Fortunately, in this formulation, V cannot have any local minima on $\Omega \setminus \{\mathbf{x}_f\}$. This was shown in [14, Theorem 3.3.5] and is given in the Appendix.

Local maxima and saddle points of V in this formulation are non-differentiable points where multiple optimal paths meet [23]. A very small perturbation from such critical points will allow the path search to continue, approximating one of the optimal paths.

An analytic solution V to (7) or (9), may be difficult to find. A method to approximate V will be described in the next section.

4. Review of Ordered Upwind Method

The Ordered Upwind Method (OUM) was first introduced in [12] to obtain an approximation $\widetilde{V}$ of V (6) on the vertices of a mesh X that discretizes Ω. The advantage of OUM over the Fast Marching Method (FMM) [17] (which approximates the solution to the Eikonal equation (9)), is that weight functions g that are dependent on direction $\mathbf{u}$ can be considered.

The domain of the solution $\widetilde{V}$ (provided by the OUM) on the vertices of X is extended to all of Ω by linear interpolation within each triangular element of the mesh X. The final point of the path $\mathbf{x}_f$ is a vertex of the mesh X, while the initial point $\mathbf{x}_0$ is any point in the domain Ω. The vertices of the mesh X are assigned and updated with labels during the execution of the OUM.

Far. The set of vertices in X where computation of $\widetilde{V}$ has not yet begun. These vertices have tentative values $\widetilde{V}(\mathbf{x}_i) = K$, where K is a large number.

Considered. The set of vertices where $\widetilde{V}$ has yet to be finalized and $\widetilde{V} < K$.

Accepted. The set of vertices where $\widetilde{V}(\mathbf{x}_i)$ has been finalized.

Though vertices of X may be relabelled, they must each have exactly one of *Accepted*, *Considered* or *Far* label at any instance of the algorithm.

The following definitions are further classifications of vertices with the *Accepted* label.

Accepted Front. The vertices labelled *Accepted* that have a neighbour labelled *Considered*.

AF. The set of edges $\mathbf{x}_j\mathbf{x}_k$ made of adjacent vertices $\mathbf{x}_j$ and $\mathbf{x}_k$ on X in the *Accepted Front*.

Near Front *of* $\mathbf{x}_i$ ($\mathbf{NF}(\mathbf{x}_i)$). Let $\mathbf{x}_i$ be labelled *Considered* and the maximum edge length between any two connected vertices of X be h_{max}. Then,

$$\mathbf{NF}(\mathbf{x}_i) = \left\{ \mathbf{x}_j\mathbf{x}_k \in \mathbf{AF} \,\middle|\, \exists\, \widetilde{\mathbf{x}} \text{ on } \mathbf{x}_j\mathbf{x}_k \,\middle|\, |\widetilde{\mathbf{x}} - \mathbf{x}_i| \leq \Gamma(\mathbf{x}_i) h_{max} \right\} \tag{10}$$

where $\Gamma(\mathbf{x}_i) = \frac{g_{max}(\mathbf{x}_i)}{g_{min}(\mathbf{x}_i)}$ and $g_{max}(\cdot), g_{min}(\cdot)$ are described in (3).

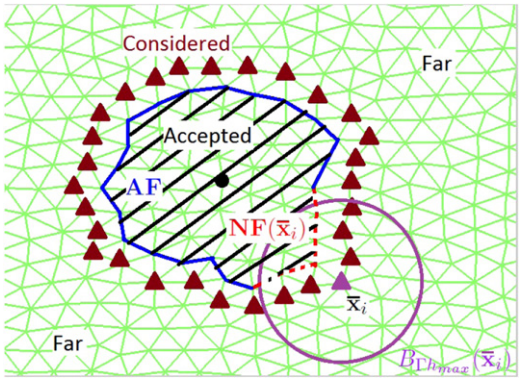


Fig. 4. Vertex labels in Ordered Upwind Method ($X \subset \mathbb{R}^2$)—Unmarked vertices have *Far* label. Vertices marked with a triangle have *Considered* label. Shaded vertices (including those on the border) have *Accepted* label. Vertex $\bar{\mathbf{x}}_i$ with *Considered* label is being updated. The ball $B_{\Gamma h_{max}}(\bar{\mathbf{x}}_i)$ has radius $\Gamma(\bar{\mathbf{x}}_i)h_{max}$ and centre $\bar{\mathbf{x}}_i$. The *Near Front* $\mathbf{NF}(\bar{\mathbf{x}}_i)$, shown dotted is the subset of edges in $\mathbf{AF}$ that contain a point within Γh_{max} of $\bar{\mathbf{x}}_i$ used in the computation (13).

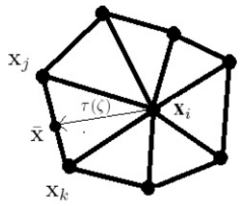


Fig. 5. OUM Update—Vertex $\mathbf{x}_i$ in the *Considered* list is updated from edge $\mathbf{x}_j\mathbf{x}_k$ on the *Near Front*, where $\mathbf{x}_j$ and $\mathbf{x}_k$ are on the *Near Front* $\mathbf{NF}(\mathbf{x}_i)$. The point $\bar{\mathbf{x}}$ is the point on the edge $\mathbf{x}_j\mathbf{x}_k$ that minimizes the update (12) with $\tau(\mathbf{x}_i, \zeta) = |\bar{\mathbf{x}} - \mathbf{x}_i|$.

See Fig. 4. The set of edges $\mathbf{AF}$ and $\mathbf{NF}(\mathbf{x}_i)$ change at each step of the algorithm as a result of vertices being relabelled between *Far*, *Considered*, *Accepted*.

The process to calculate $\mathbf{u}^*$ at the vertices of X will now be described. Suppose $\mathbf{x}_i \in X$ is the vertex in the *Considered* list to be updated (see Fig. 5). Let $\bar{\mathbf{x}} = \zeta\mathbf{x}_j + (1-\zeta)\mathbf{x}_k$ for some $\zeta \in [0, 1]$ be the point at which the optimal trajectory that crosses the vertex $\mathbf{x}_i$ exits triangle $\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k$. The distance between $\mathbf{x}_i$ and $\bar{\mathbf{x}}$, and optimal constant control $\mathbf{u}^*(\zeta)$ that bring the trajectory $\mathbf{x}(\cdot)$ from $\mathbf{x}_i$ to the edge $\mathbf{x}_j\mathbf{x}_k$ optimally, are

$$\tau(\mathbf{x}_i, \zeta) = |\bar{\mathbf{x}} - \mathbf{x}_i| \quad \text{and} \quad \mathbf{u}^*(\zeta) = \frac{\bar{\mathbf{x}} - \mathbf{x}}{\tau(\mathbf{x}_i, \zeta)}$$

respectively. The approximated value is

$$\widetilde{V}(\mathbf{x}_i) \approx \tau(\mathbf{x}_i, \zeta)g(\mathbf{x}_i, \mathbf{u}^*(\zeta)) + \zeta\widetilde{V}(\mathbf{x}_j) + (1-\zeta)\widetilde{V}(\mathbf{x}_k). \tag{11}$$

A search for the optimal direction $\mathbf{u}^*(\zeta)$ is performed. The update for $\mathbf{x}_i$ provided by an edge $\mathbf{x}_j\mathbf{x}_k$ is defined by the function $\widetilde{C} : \{\mathbf{x}_i\} \subset X \to \mathbb{R}$,

$$\widetilde{C}_{\mathbf{x}_j\mathbf{x}_k}(\mathbf{x}_i) = \min_{\zeta\in[0,1]} \left\{\tau(\mathbf{x}_i, \zeta)g(\mathbf{x}_i, \mathbf{u}(\zeta)) + \zeta\widetilde{V}(\mathbf{x}_j) + (1-\zeta)\widetilde{V}(\mathbf{x}_k)\right\}. \tag{12}$$

It was proven in [12] that the minimizing direction for $\mathbf{x}_i$ must come from its *Near Front*. The update formula over all of the *Near Front of* $\mathbf{x}_i$, $\mathbf{NF}(\mathbf{x}_i)$ is

$$\widetilde{C}(\mathbf{x}_i) = \min_{\mathbf{x}_j\mathbf{x}_k\in\mathbf{NF}(\mathbf{x}_i)} \widetilde{C}_{\mathbf{x}_j\mathbf{x}_k}(\mathbf{x}_i). \tag{13}$$

The OUM algorithm is as follows.

Initialization

1. Label all vertices $\{\mathbf{x}_i\}$ of X *Far*, with $\widetilde{V}(\mathbf{x}_i) = K$ (where K is a large number).
2. Relabel $\mathbf{x}_f$ from *Far* to *Accepted*, and let $\widetilde{V}(\mathbf{x}_f) = 0$.
3. Relabel all neighbours of *Accepted* vertices $\mathbf{x}_i$ that have *Far* label, to *Considered*. Compute the values $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$ according to (13).
4. Find the vertex $\bar{\mathbf{x}}_i$ with *Considered* label that has the lowest value $\widetilde{V}(\bar{\mathbf{x}}_i)$.
5. Relabel $\bar{\mathbf{x}}_i$ to the *Accepted* label. If all vertices in X are labelled *Accepted*, then terminate.
6. Relabel all vertices $\mathbf{x}_i$ adjacent to $\bar{\mathbf{x}}_i$ that are in the *Far* label to the *Considered* label.
7. Compute $\widetilde{C}(\mathbf{x}_i)$ using (13) for all vertices $\mathbf{x}_i$ that have *Considered* label such that $\bar{\mathbf{x}}_i \in \mathbf{NF}(\mathbf{x}_i)$. If $\widetilde{V}(\mathbf{x}_i) > \widetilde{C}(\mathbf{x}_i)$, then update $\widetilde{V}(\mathbf{x}_i) = \widetilde{C}(\mathbf{x}_i)$.
8. Go to Step 4.

The different labels of Ordered Upwind Method are shown in Fig. 4. The algorithm is terminated once all the vertices of X have been labelled *Accepted*. The solution produced by the OUM, $\widetilde{V}$ converges to the viscosity solution V of (7) as the mesh is refined ($h_{max} \to 0$) provided the elements do not become arbitrarily degenerate [12, Theorem 7.7].

To find the optimal path on $\widetilde{V}$, Eq. (8) is solved replacing V with $\widetilde{V}$, where points in each triangle have been linearly interpolated between the values at its vertices found using OUM. A minimization problem (8) is solved for $\mathbf{u}^*$ each time the path exits its current triangle and enters a new adjacent triangle. The path is not restricted to the edges of the mesh and is found from $\mathbf{x}(0) = \mathbf{x}_0$ and complete when $\mathbf{x}(T) = \mathbf{x}_f$ is reached. Recall that the solution V does not contain any local minima (see the Appendix). If local minima are encountered in $\widetilde{V}$, then OUM can be executed again using a finer mesh. As the solutions produced by OUM converge to V using progressively refined meshes, all local minima will disappear.

The computational complexity of OUM for the rover path planning problem is $\mathcal{O}(\Gamma N \log N)$ [12] for h_{max} small enough where

$$\Gamma = \frac{G_{max}}{G_{min}}$$

and G_{max}, G_{min} are defined in (3) and N is the number of vertices in the mesh X. The vertices labelled *Considered* are maintained in a minimum binary heap array [15]. The first entry of the array always contains the smallest value of the *Considered* vertices. Only $\mathcal{O}(\log N)$ operations are required to maintain the heap ordering when a vertex is added, updated or removed.

5. Bi-directional search

While the idea of a bi-directional search on discrete graphs is not new [24], it has not been used previously with OUM. Instead of finding $\widetilde{V}$ on all of Ω, any region inside Ω that contains the optimal path is sufficient to solve (8) to recover the optimal path. An obvious improvement to the OUM is an uni-directional search which terminates when the initial point $\mathbf{x}_0$ is reached. An even smaller limiting region will be found by also considering the path planning problem where the roles of $\mathbf{x}_0$ and $\mathbf{x}_f$ in (1), (2) in Section 2 are reversed.

Suppose both problems provide the same optimal path. See Fig. 6. By the DPP (Theorem 3.3), the point $\widetilde{\mathbf{x}}_i$ representing the lowest sum between the two problems must be on the optimal path. The points in OUM are finalized in the manner of an advancing front based on cost value. The point $\widetilde{\mathbf{x}}_i$ is the instance for which the two fronts meet for the first time. The optimal path is recovered by solving (8), using their respective value functions and the initial condition $\widetilde{\mathbf{x}}_i$ for both problems.

Suppose the optimal path from an initial position $\mathbf{x}_0$ to a final position $\mathbf{x}_f$ is required. Let $\mathbf{x}_1 = \mathbf{x}_0$, and $\mathbf{x}_2 = \mathbf{x}_f$. For $k \in \{1, 2\}$,

$$\dot{\mathbf{x}}_k(t) = \mathbf{u}_k(t) \tag{14}$$
$$\mathbf{x}_k(0) = \mathbf{x}_k, \quad \mathbf{x}_k \in \Omega$$

Table 1
Performance—Costs for GA and Bi-RRT* in all examples were averaged over 5 trials, with the bracketed value being the lowest cost of the 5 trials. Example 1: Path length with obstacle avoidance, NPF indicates that no (feasible) path was found, Example 2: Tip-over stability risk, Example 3: All components.

Cost	FMM	FMM-BD	OUM	OUM-BD	GA	Bi-RRT*
Example 1	1404.1	1405.2	1421.5	1417.8	NPF	1448.1 (1434.7)
Example 2	–	–	61.38	61.43	61.69 (61.56)	63.52 (62.83)
Example 3	–	–	796370	794120	822810 (820000)	814790 (800970)

Table 2
Time required in seconds—All timings were averaged over 5 trials. Example 1: Path length with obstacle avoidance, Example 2: Tip-over stability risk, Example 3: All components. For Bi-RRT*, all simulations were terminated after 30 min (1800 s).

Timings	FMM	FMM-BD	OUM	OUM-BD	GA
Example 1	21.22	7.15	45.82	15.97	1257
Example 2	–	–	106.9	68.53	3615
Example 3	–	–	57.11	49.61	5299

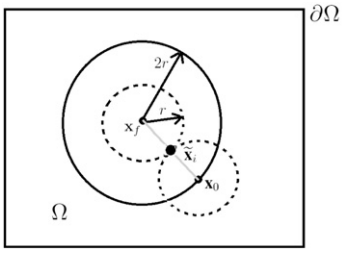


Fig. 6. Bi-directional Search—Let $g(\mathbf{x}, \mathbf{u}) \equiv 1$. The solid circle is the level set of the uni-directional problem when $\mathbf{x}_0$ has been reached. The dotted circles are the respective level sets in each of the problems in the bi-directional problem. The value of the two level sets are the same. If the initial point $\mathbf{x}_0$ and final point $\mathbf{x}_f$ are far enough from $\partial\Omega$, the areas between the bi-directional search and uni-directional search for this problem are reduced by a factor of 2. The optimal path (straight line) is shown in grey.

where $\mathbf{u}_k \in \mathcal{U}$. The objective is to reach $\mathbf{x}_k(T_k) = \mathbf{x}_{\{1,2\}\setminus k}$, while minimizing their respective cost functions,

$$Cost_k(\mathbf{x}_k, \mathbf{u}_k(\cdot)) = \int_0^{T_k(\mathbf{x}_k, \mathbf{u}_k(\cdot))} g_k(\mathbf{x}_k(s), \mathbf{u}_k(s))ds \tag{15}$$

where $g_k : \Omega \times \mathbb{S}^1$ are the weights of (14). The value function $V_k : \Omega \to \mathbb{R}_+$ for each problem is

$$V_k(\mathbf{x}) = \inf_{\mathbf{u}_k(\cdot)\in\mathcal{U}} Cost_k(\mathbf{x}, \mathbf{u}_k(\cdot)).$$

The following theorem is required to guarantee that both problems (14) produce the same optimal path.

Theorem 5.1. *Assume $g_1(\mathbf{x}, \mathbf{u}) = g_2(\mathbf{x}, -\mathbf{u})$. Let $\mathbf{u}_1^*(\cdot), \mathbf{u}_2^*(\cdot) \in \mathcal{U}$ be optimal controls for the problems (14) for $k = 1$ and $k = 2$ respectively. Then $-\mathbf{u}_2^*(T_2 - \cdot)$ and $-\mathbf{u}_1^*(T_1 - \cdot)$ are also optimal controls for (14) for $k = 1$ and $k = 2$ respectively.*

Proof.

$$\begin{aligned} V_2(\mathbf{x}_2) &= Cost_2(\mathbf{x}_2, \mathbf{u}_2^*(\cdot)) \le Cost_2(\mathbf{x}_2, -\mathbf{u}_1^*(T_1 - \cdot)), \\ &= \int_0^{T_1} g_2(\mathbf{x}_1(T_1 - s), -\mathbf{u}_1^*(T_1 - s))ds \\ &= \int_{T_1}^{0} -g_2(\mathbf{x}_1(v), -\mathbf{u}_1^*(v))dv, \\ &= \int_0^{T_1} g_1(\mathbf{x}_1(s), \mathbf{u}_1^*(s))ds = Cost_1(\mathbf{x}_1, \mathbf{u}_1^*(\cdot)) = V_1(\mathbf{x}_1). \end{aligned}$$

Similarly, $V_1(\mathbf{x}_1) \le V_2(\mathbf{x}_2)$. Therefore $V_1(\mathbf{x}_1) = V_2(\mathbf{x}_2)$. Hence $-\mathbf{u}_2^*(T_2 - \cdot)$ and $-\mathbf{u}_1^*(T_1 - \cdot)$ are optimal controls for (14) with $k = 1$ and $k = 2$ respectively. □

Furthermore, if $\mathbf{u}_1^*(\cdot) = -\mathbf{u}_2^*(T_2 - \cdot)$, then $T = T_1 = T_2$ and the optimal path for both problems are the same. This may not always be the case. If $\nabla V_1(\mathbf{x}_1)$ or $\nabla V_2(\mathbf{x}_2)$ are not defined, then multiple optimal paths may exist.

The assumption in Theorem 5.1, $g_1(\mathbf{x}, \mathbf{u}) = g_2(\mathbf{x}, -\mathbf{u})$ is automatically satisfied for direction-independent weights and weights that are symmetric in opposite directions. The weights used in Section 6 satisfy this assumption. Otherwise, set $g_2(\mathbf{x}, \mathbf{u}) = g_1(\mathbf{x}, -\mathbf{u})$. The two problems (14) are solved concurrently in the form of a bi-directional search. Let $\widetilde{T}$ be the time at which the fronts in the problems (14) meet. Then the control

$$\mathbf{u}^*(t) = \begin{cases} \mathbf{u}_1^*(t), & t \in [0, \widetilde{T}] \\ -\mathbf{u}_2^*(T - t), & t \in (\widetilde{T}, T] \end{cases}$$

is the solution to the optimal path planning problem for both problems in (14).

The OUM-BD labels and algorithm are as follows. For $k = \{1, 2\}$, the labels Far_k, $Considered_k$ and $Accepted_k$ are the same labels as described in Section 4 for their respective problems.

1. Label all vertices in $\mathbf{x}_i \in X$ in both Far_1 with $\widetilde{V}_1(\mathbf{x}_i) = K$ and Far_2 with $\widetilde{V}_2(\mathbf{x}_i) = K$, where K is a large value.
2. Label $\mathbf{x}_2$ as $Accepted_1$, setting $\widetilde{V}_1(\mathbf{x}_2) = 0$ and label $\mathbf{x}_1$ as $Accepted_2$, setting $\widetilde{V}_2(\mathbf{x}_1) = 0$.
3. Place the neighbours of $\mathbf{x}_2$ in $Considered_1$ and the neighbours of $\mathbf{x}_1$ in $Considered_2$ and update each from their respective $Accepted_i$ vertex using (13).
4. Let $\bar{\mathbf{x}}_i$ be the vertex in $\{Considered_1 \cup Considered_2\}$ with minimum tentative value ($\widetilde{V}_1$ or $\widetilde{V}_2$). Let $k \in \{1, 2\}$ be the $Considered$ label with the minimizing $\bar{\mathbf{x}}_i$.
5. Relabel $\bar{\mathbf{x}}_i$ from its respective $Considered_k$ label to its respective $Accepted_k$ label. Relabel the neighbours of $\bar{\mathbf{x}}_i$ with Far_k label to $Considered_k$ label.
6. Update the vertices in $Considered_k$ affected by adding $\bar{\mathbf{x}}_i$ to $Accepted_i$.
7. If no vertices have both $Accepted_1$ and $Accepted_2$ labels, then go to Step 4. Otherwise, let $\widetilde{\mathbf{x}}_i$ be the vertex labelled both $Accepted_1$ and $Accepted_2$. Repeat Steps 4–6 until all vertices labelled $Considered_k$ within $\frac{g_{max}(\widetilde{\mathbf{x}}_i)}{g_{min}(\widetilde{\mathbf{x}}_i)} h_{max}$ of $\widetilde{\mathbf{x}}_i$ (at the time $\widetilde{\mathbf{x}}_i$ first belonged to both $Accepted_k$ labels) have been relabelled $Accepted_k$. Then terminate.

To find the optimal path, the triangles of X in which the optimal path traverses must be made entirely of vertices labelled $Accepted$. This is guaranteed in the last step of OUM-BD. The optimal path is found by solving (8) using $\widetilde{V}_1$ and $\widetilde{V}_2$ with their respective problems and connecting the two paths. The vertex $\widetilde{\mathbf{x}}_i$ may not lie on the actual optimal path of either problem in (14). However as $h_{max} \to 0$, $\widetilde{V}_1$ and $\widetilde{V}_2$ will converge to V_1 and V_2 [12, Theorem 7.7]. The two value functions can be solved independently using parallel processors with the termination condition in the last step of OUM-BD checked periodically.

For the Fast Marching Method, a brief description of a bi-directional search is provided in [25]. The OUM-BD here can easily be modified to a bi-directional FMM algorithm, FMM-BD. The termination condition in Step 4 is replaced with immediate termination, and the update formula in Step 3 with the FMM update in [17].

Aside from fewer total vertices labelled $Accepted_k$, the set of vertices labelled $Considered_k$ have fewer elements in each problem for OUM-BD. Fewer operations would be required to both maintain the minimum heap array as well as to solve the update formula (12).

The amount of computation saved is dependent on the problem, and location of $\mathbf{x}_0$ and $\mathbf{x}_f$ relative to the boundary of the workspace $\partial\Omega$. Additional computational savings would be gained by implementing a heuristic for OUM, in a manner similar to A* [2]. Finding a heuristic for OUM that will yield the same approximated value function is not straightforward.

6. Examples

The methods described in this section were programmed in MATLAB®. All timed computations were performed on an ASUS X550L Laptop with Intel® Core™ i5 -4210U CPU Processor (1.7 GHz/2.4 GHz) with 4 GB RAM. All path planning problems were executed on the square workspace $\Omega = [-500, 500] \times [-500, 500] \subset \mathbb{R}^2$ discretized by a mesh X made of 16,765 vertices and 32,888 triangles generated using Mesh2D [26]. The FMM [17] and FMM-BD (when possible), OUM [12], OUM-BD (introduced in this work), Bi-RRT* [11], a genetic algorithm (GA) path planner [18] were used to solve the problem.

The OUM algorithm is programmed as discussed in this paper, except that the algorithm is terminated once the *Accepted Front* reaches $\mathbf{x}_0$, so unnecessary computation that is unrelated to the optimal path is avoided. The same is true for the FMM algorithm.

In the GA path planner [18], candidate paths were defined using cubic splines between a set of control points in Ω. All candidate paths had their ends fixed at $\mathbf{x}_0$ and $\mathbf{x}_f$. A minimization was performed over these control points according to the cost (2). The best results were obtained using 6–8 control points [18]. In the following examples, 8 control points were used with a population of 50 candidate paths, evolved over 100 generations. The crossover factor and mutation probabilities were chosen to be 0.9 and 0.1 respectively. The best candidate path was returned. Since a different path was produced in each instance and resulting paths were not necessarily minimums, both the average cost and the cost of the best path over 5 trials are presented.

In the Bi-RRT* algorithm [11], two trees (one stemming from the start configuration and the other from the goal configuration) are built by sampling random points in the workspace, adding feasible connections to the corresponding trees. The connections between the points are reconfigured to ensure the paths along the tree are optimal within the tree itself. When the sampled points of the trees become close enough to one another, they are connected using local optimality within a predefined radius. The algorithm is guaranteed to produce the optimal path given infinite time. The reader is referred to [11] for additional detail. Following the algorithm [11], Bi-RRT* was programmed without the use of the described heuristics. Better performance was found without the use of the local biasing heuristic, and the more complicated weights described used in this work do not allow for the node rejection heuristic. Since the optimal cost is not known a priori, the termination criteria is usually set by number of iterations or run time. The best path with the first initial path is presented after 30 minutes along with a graph to show the best cost progression throughout that time. All cost values shown were averaged over 5 trials.

For all path planning examples, the start and final positions were $\mathbf{x}_0 = (450, -450)$ and $\mathbf{x}_f = (-450, 450)$ respectively. The minimizations in (8) and (12) were performed using the golden section search [27, Chapter 10.2]. Timings for each example were measured using `tic` and `toc`. The values presented in Table 2 were each averaged over 5 trials. Using code from [18], the paths in all examples were smoothed using cubic splines and sampled roughly every one unit. The costs of the paths were approximated on the continuous cost function (2) using trapezoidal rule over the sampled points.

Example 1 (*Cluttered Environment*)**.** A hundred circular obstacles with radius 30 were generated with random locations in Ω. Let the set of obstacle regions be denoted O. The weight used to model obstacles was $g_o(\mathbf{x}) = 10^6$ for $\mathbf{x} \in O$, with linear decay to $g_o(\mathbf{x}) = 0$ at $\epsilon = 1$ from O to maintain continuity. The weight function used was

$$g(\mathbf{x}) = 1 + g_o(\mathbf{x}),$$

where obstacle locations are shown in Fig. 7.

Since the weight associated with this example did not depend on direction, all of FMM, FMM-BD, OUM, OUM-BD, GA and Bi-RRT* were compared. The cost (path length) between the paths found were approximately the same for all of FMM, FMM-BD, OUM and OUM-BD while the cost for the path found by Bi-RRT* was higher, despite having the algorithm run for 30 min. See Fig. 7 and Table 1. The GA planner was executed using 8 control points (as described above) and using 25 control points. All other parameters were the same. A feasible path was not found by GA in either case (over 5 trials each). In the case of 8 control points, there was likely not enough degrees of freedom to form the required path. For 25 control points, the problem was likely too complex to solve within 100 generations.

Timings for Example 1 are shown in Table 2. The FMM and FMM-BD are faster than OUM, due to the large *Accepted Front* in OUM. The OUM-BD was only slightly slower than the FMM algorithm. The time required for the GA planner (shown for 8 control points) is much higher, despite not producing a feasible path. The 25 control point example took an even longer time. The Bi-RRT* algorithm was timed for the preset 30 min. The progression of best cost is shown every minute in Fig. 7d. Despite a longer run time, a better cost is achieved using OUM-BD in a significantly shorter amount of time.

Example 2 (*Tip-over Stability Risk*)**.** Using the force–angle stability margin α as defined in (4), let $m(\mathbf{x}) = \alpha\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})_\perp}{|\nabla z(\mathbf{x})_\perp|}\right)$ and $n(\mathbf{x}) = \alpha\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})}{|\nabla z(\mathbf{x})|}\right)$ where the vector $\nabla z(\mathbf{x})_\perp$ is in a level set direction of z at $\mathbf{x}$. Though there are two level set directions, due to rectangular symmetry of the rectangular rover, both level set directions yield the same stability margin $m(\mathbf{x})$.

The tip-over risk weight $g_{risk} : \Omega \times \mathbb{S}^1 \to \mathbb{R}_+$ is

$$g_{risk}(\mathbf{x}, \mathbf{u}) = \frac{1}{n(\mathbf{x})}\sqrt{1 + \left(\frac{n(\mathbf{x})^2}{m(\mathbf{x})^2} - 1\right)\left(\frac{\nabla z(\mathbf{x})}{|\nabla z(\mathbf{x})|} \cdot \mathbf{u}\right)^2}. \tag{16}$$

The stability margins in directions $\nabla z(\mathbf{x})_\perp$ and $\nabla z(\mathbf{x})$ are

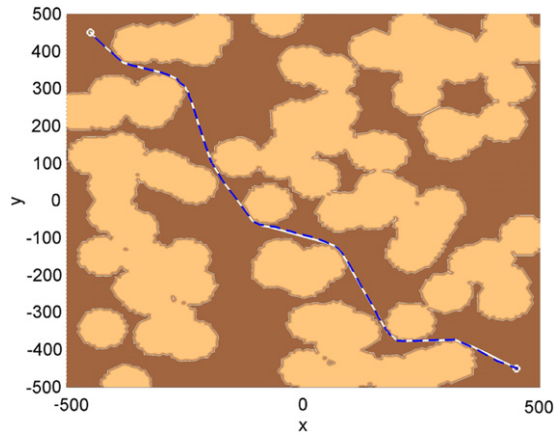
$$g_{risk}\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})_\perp}{|\nabla z(\mathbf{x})_\perp|}\right) = \frac{1}{n(\mathbf{x})} \quad \text{and} \quad g_{risk}\left(\mathbf{x}, \frac{\nabla z(\mathbf{x})}{|\nabla z(\mathbf{x})|}\right) = \frac{1}{m(\mathbf{x})}$$

respectively. The maximum and minimum of $\alpha(\mathbf{x}, \cdot)$ over $\mathbb{S}^1$ correspond to directions $\nabla z(\mathbf{x})_\perp$ and $\nabla z(\mathbf{x})$ depending on the length and the width of the rover. As the rover approaches tip-over, g_{risk} becomes large. Finally, if either $m(\mathbf{x})$ or $n(\mathbf{x})$ is negative, then the corresponding weight in (16) is given the value 10^6.

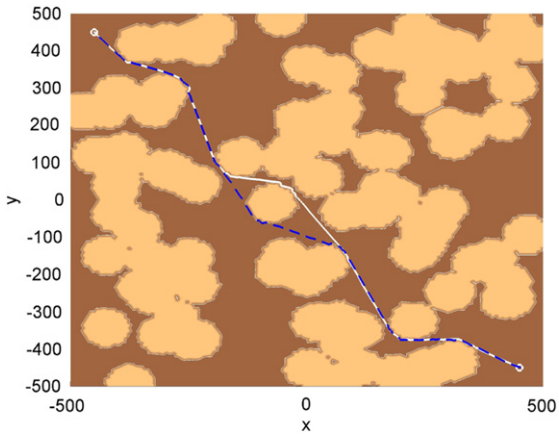
The weight used in Example 2 was

$$g(\mathbf{x}, \mathbf{u}) = g_{risk}(\mathbf{x}, \mathbf{u})$$

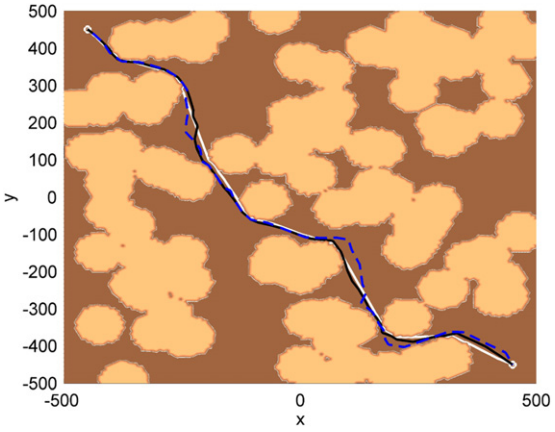
for a rectangular rover with width 6 units and length 3 units, travelling in its length-wise direction. The terrain is shown in



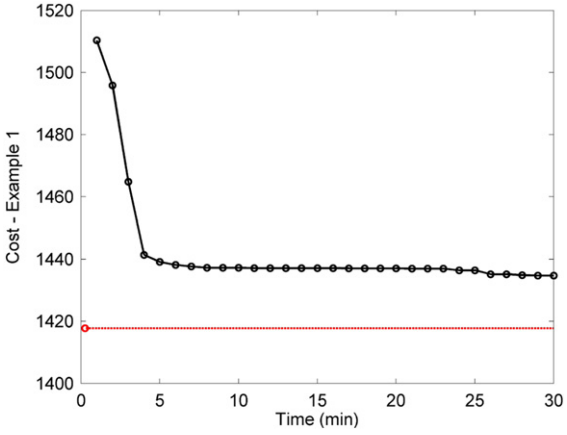
(a) FMM in solid line, FMM-BD in dotted line.



(b) OUM in solid, OUM-BD in dotted.

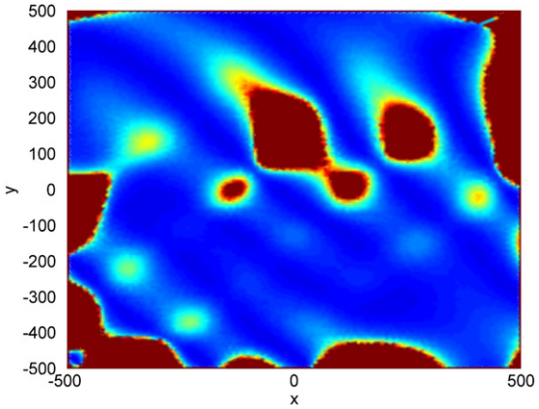


(c) OUM-BD in white, Best Bi-RRT* in black, First feasible Bi-RRT* in dotted.

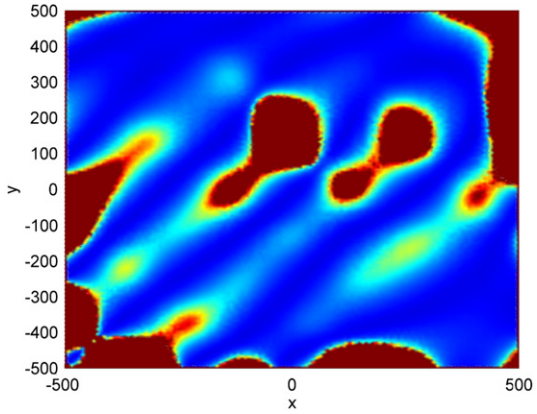


(d) Progression of best cost using Bi-RRT* over 30 min sampled every minute. The OUM-BD cost and time (the circle on the bottom line) are shown for comparison.

Fig. 7. Example 1: Cluttered Environment—The GA planner was unable to find a feasible path. The costs of the paths are approximately the same between FMM, FMM-BD, OUM, OUM-BD. The best paths produced by Bi-RRT* after 30 min have higher cost than the other found paths. See Tables 1 and 2.



(a) The weight $g(\mathbf{x}, \mathbf{u}_1)$ for $\mathbf{u}_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$.

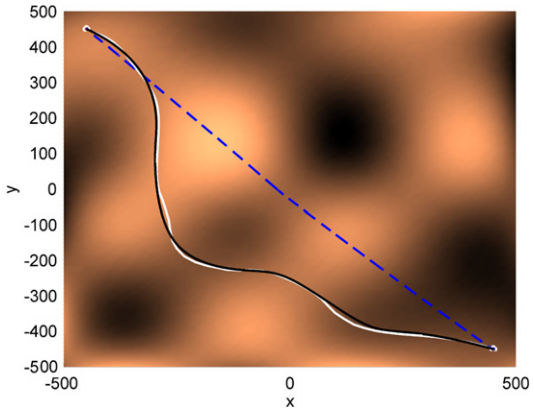


(b) The weight $g(\mathbf{x}, \mathbf{u}_2)$ for $\mathbf{u}_2 = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$.

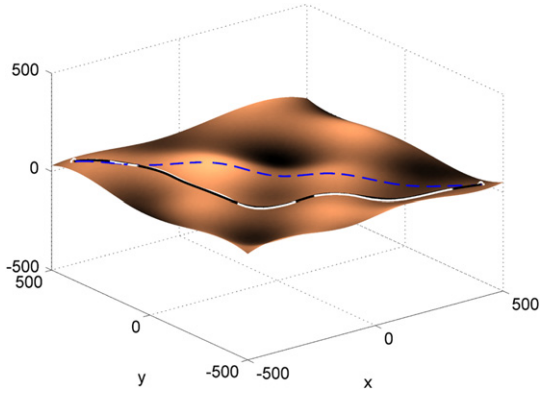
Fig. 8. *Weight for Example 2*: The weight $g(\mathbf{x}, \mathbf{u})$ for Example 2 is shown for two sample directions.

Fig. 9. The weight for two sample directions, $\mathbf{u}_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ and $\mathbf{u}_2 = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ are shown in Fig. 8. Since the weight is dependent on direction, FMM cannot be used. The safest path calculated using the OUM and OUM-BD approach is compared to the shortest path in Fig. 9c and 9d. A comparison is made with both GA and Bi-RRT* using the same weight. Though the costs of the paths are similar (Table 1) between OUM, OUM-BD, GA, and Bi-RRT*, the time required to find these paths using Bi-RRT* and GA is much higher. See Table 2. Note the difference in paths in Fig. 9c and 9d.

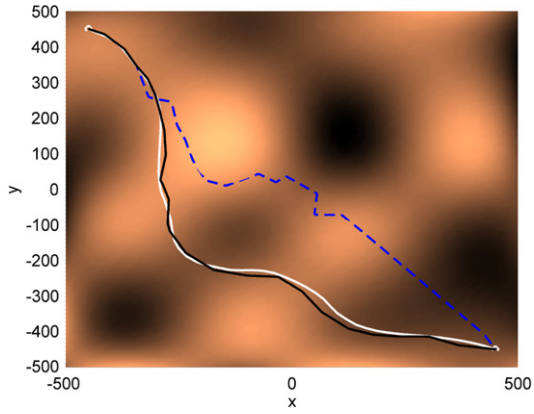
Example 3. Soil risk, solar energy and path length on terrain were considered for a rectangular rover with width 6 and length 3 in



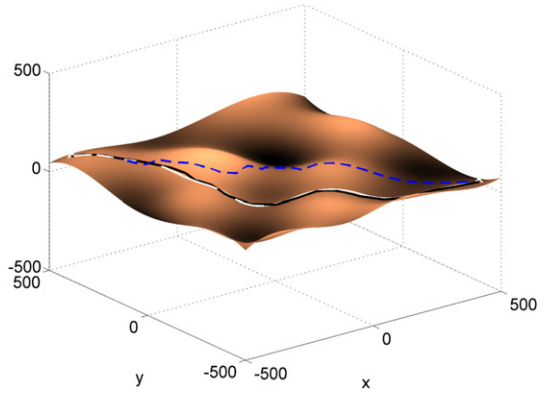
(a) OUM-BD—White, GA—Black, Shortest Path—Dotted.



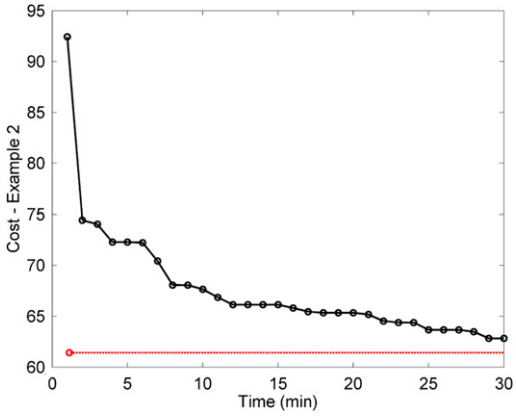
(b) OUM-BD—White, GA—Black, Shortest Path—Dotted, 3D view.



(c) OUM-BD—White, Final Bi-RRT*—Black, Initial Bi-RRT*—Dotted.



(d) OUM-BD—White, Final Bi-RRT*—Black, Initial Bi-RRT*—Dotted in 3D.



(e) Progression of best cost using Bi-RRT* over 30 min sampled every minute. The OUM-BD cost and time (the circle on the bottom line) are shown for comparison.

Fig. 9. *Example 2*: Tip-over stability risk. The paths for OUM and OUM-BD are virtually identical, so only OUM-BD has been shown. Both the best GA planner path and Bi-RRT* path are shown with the OUM-BD path.

addition to obstacle avoidance, and tip-over stability risk as described in Examples 1 and 2. The weight $g_{sr} : \Omega \to \mathbb{R}_+$ used to model soil risk was independent of direction where a value is assigned for each type of soil. The solar panel was chosen to be parallel to the frame of the rover. The solar energy weight $g_{so} : \Omega \to \mathbb{R}_+$ was

$$g_{so}(\mathbf{x}) = \begin{cases} (\beta(\mathbf{x}))^4, & \text{if } \beta(\mathbf{x}) \leq \dfrac{\pi}{2} \\ \left(\dfrac{\pi}{2}\right)^4, & \text{if } \beta(\mathbf{x}) > \dfrac{\pi}{2}. \end{cases} \tag{17}$$

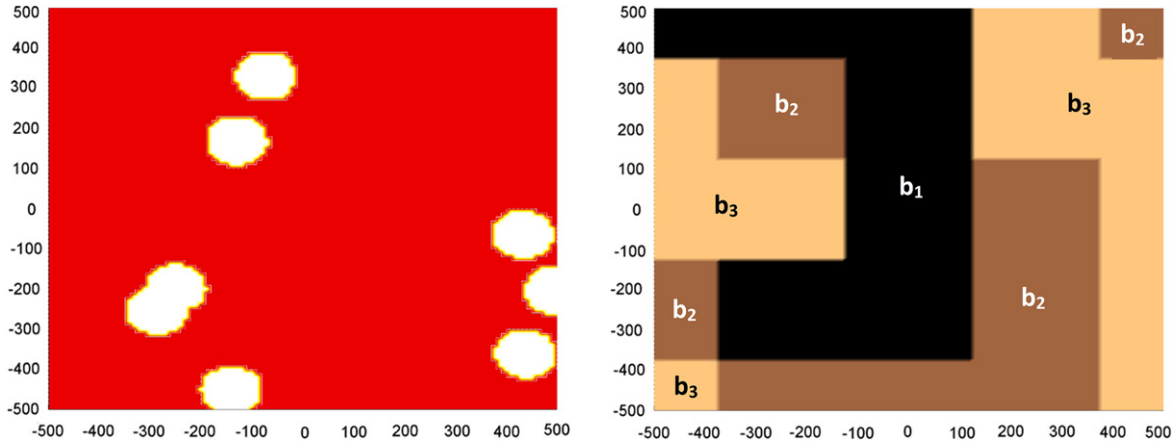
A quartic was chosen to mimic a drastic drop in energy absorbed for an angle $\beta(\mathbf{x}) > \pi/4$.

The weight function used in Example 3 was

$$g(\mathbf{x}, \mathbf{u}) = 1 + a_o g_o(\mathbf{x}) + a_{sr} g_{sr}(\mathbf{x}) + a_{so} g_{so}(\mathbf{x}) + a_{risk} g_{risk}(\mathbf{x}, \mathbf{u}) \tag{18}$$

where $a_o, a_{sr}, a_{so}, a_{risk} \geq 0$ were chosen to reflect the relative importance of the various weights.

The parameters $a_{sr} = 100$, $a_{risk} = 1000$, $a_{so} = 10$ with soil and obstacle weighting maps [18] in Fig. 10 were used. The paths are



(a) Obstacles $g_o(\mathbf{x})$.

(b) Soil Weightings $g_{sr}(\mathbf{x})$, $b_1 = 0.5$, $b_2 = 1.5$ and $b_3 = 2.5$.

Fig. 10. *Example* 3—Weighting maps, [18].

shown in Fig. 11. The cost and timings are found in Tables 1 and 2 respectively. Again, OUM and OUM-BD outperform GA and Bi-RRT* in both performance and time. The paths are quite different, and neither GA nor Bi-RRT* were able to find a path with lower cost on the other side of the obstacles in the allotted time. In some trials lasting over an hour, Bi-RRT* was able to find paths on the other side, but the cost was still higher.

Both the OUM and OUM-BD outperform GA and Bi-RRT* in cost within a reasonable time frame. Given more time, GA and Bi-RRT* will produce paths with lower cost, while the cost performance of OUM and OUM-BD can be improved by executing the algorithm on a finer mesh. Comparing OUM-BD with the original OUM algorithm, the regions Ω for which $\widetilde{V}$ is found using OUM-BD are shown in Fig. 12 for each example. In terms of timing, OUM-BD was faster than OUM for all examples while maintaining very similar cost.

7. Conclusions and future work

The FMM can be used to solve the HJB for positive, continuous, direction-independent weights. The solution found by OUM yields approximately the same path, but is slower than FMM. Thus, for problems where either approach can be used, FMM is preferable. The value function has been shown to not have any local minima (aside from the final point). Local minima in the solution would correspond to positions where the path finding process would become stuck. The OUM can be used to find solutions for a more general class of weights. The OUM is about 60 times (depending on the weight) faster than GA on the problems tested and yielded similar or better results. The OUM also outperformed Bi-RRT* by producing closer-to-optimal paths in significantly less time in all shown examples. The Bi-RRT* and GA planners are capable of handling even more general weights including negative weights however. The faster computational times with FMM and OUM could allow for replanning, allowing the algorithm to be executed in real time in response to moving obstacles or a moving sun.

A novel bi-directional search for OUM, OUM-BD specific to path planning was presented. Depending on the problem, computational savings of about 25%–70% were observed in terms of finalized vertices on a mesh. The percentage of time saved over regular OUM was even greater than the percentage of finalized vertices. The OUM-BD search algorithm can be implemented using parallel processors which would increase its efficiency. A future activity is to improve speed by rewriting the code into C in parallel for the OUM-BD. An FPGA would allow for real-time path planning.

Two novel improvements to rover optimal path planning were considered: solar energy input and tip-over stability risk. Safe paths are important in rover path planning because of the high costs incurred in unmanned extraterrestrial exploration. The safest path was shown to be different from the shortest path.

An area for future investigation include more accurate modelling of the rover to include energy consumption and a moving sun in the solar weight. For the latter, time-dependence in the weight function as done in [28] would be required. As a rover traverses the environment, it will become more imperative to recharge the solar panels. Including a moving sun $(x_s(t), y_s(t), z_s(t))$ in the solar energy weight is realistic as rover speeds are slow and paths are planned across large environments. Constraints could be considered as in [23,29].

Acknowledgements

The authors thank the Natural Sciences and Engineering Research Council of Canada, Ontario Government, Canadian Space Agency and Maplesoft® for their partnership and funding of this research. The authors would also like to thank Orang Vahid and M. Saber Fallah for the use of their genetic algorithm path planner [18]. Finally, the authors would like to thank the anonymous reviewers for their comments on this work.

Appendix. Local minima free value function

A proof that the value function V described in (7) cannot contain local minima aside from $\mathbf{x}_f$ based on the definition of viscosity solutions was first shown in [14] is presented here. A smooth solution V (where ∇V is defined) may not exist on all of Ω. Let the Hamiltonian be

$$H(\mathbf{x}, \mathbf{p}) = -\min_{\mathbf{u}\in\mathbb{S}^1}\{\mathbf{p}\cdot\mathbf{u} + g(\mathbf{x}, \mathbf{u})\}. \tag{A.1}$$

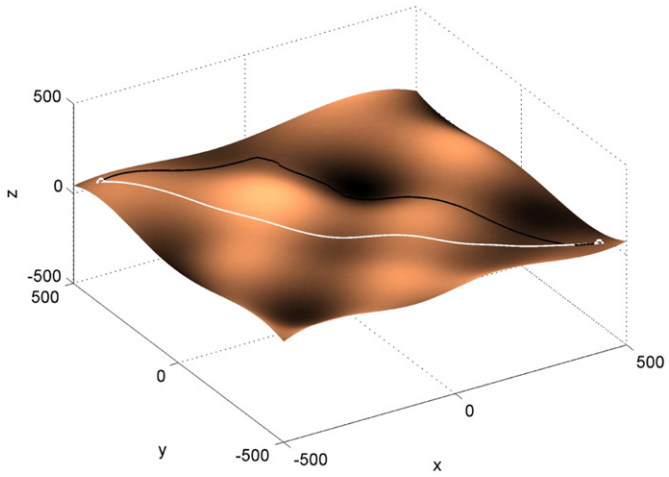
For $\mathbf{x} \in \Omega$ such that $\nabla V(\mathbf{x})$ exists, (7) can be rewritten $H(\mathbf{x}, \nabla V) = 0$. Let $C^1(\Omega)$ denote the space of continuously-differentiable functions defined on Ω.

Definition A.1. A **viscosity subsolution** $\underline{V} : \Omega \to \mathbb{R}$ of (7) satisfies the following.

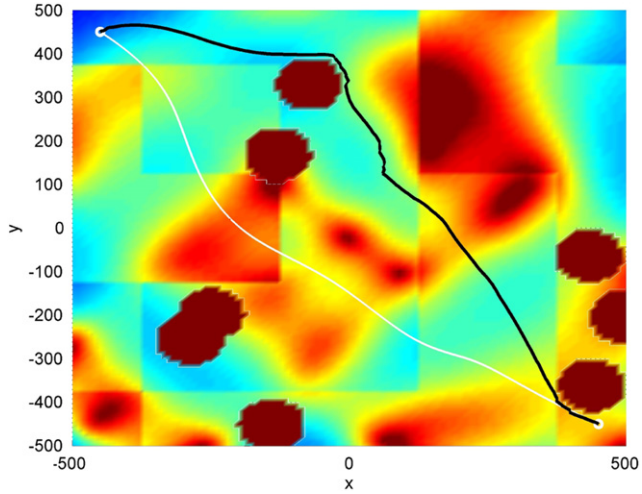
1. $\underline{V}(\mathbf{x}_f) \leq 0$
2. If $\phi \in C^1(\Omega)$ satisfies

$$\begin{cases} \underline{V}(\mathbf{x}) \leq \phi(\mathbf{x}) & \text{for } \mathbf{x} \in \Omega \setminus \{\mathbf{x}_f\}, \\ \underline{V}(\mathbf{x}_0) = \phi(\mathbf{x}_0) & \text{at } \mathbf{x}_0 \in \Omega \setminus \{\mathbf{x}_f\}, \end{cases}$$

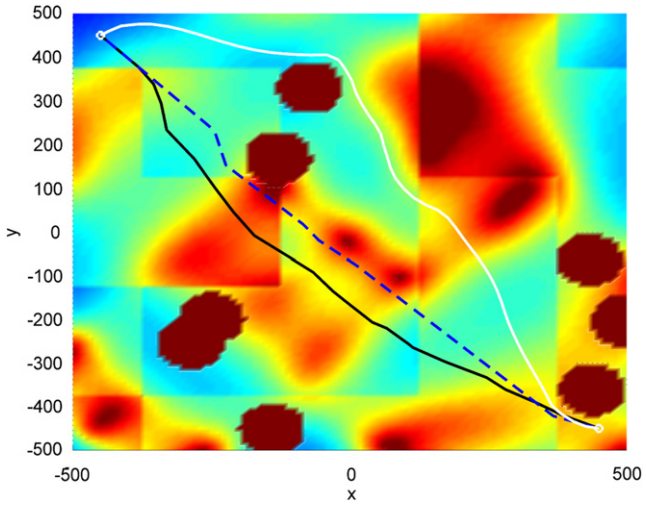
then $H(\mathbf{x}_0, \nabla\phi) \leq 0$.



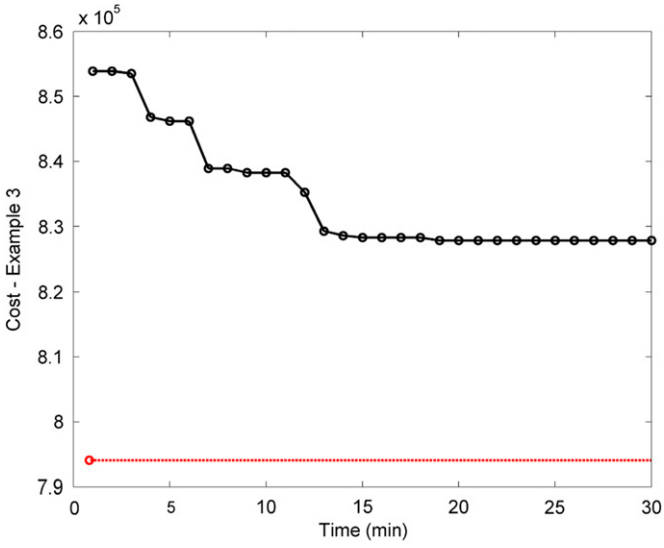
(a) OUMBD/OUM (visually identical)—Black, GA—White superimposed on the terrain.



(b) OUMBD/OUM (visually identical)—Black, GA—White superimposed on the sum of the direction-independent weights.



(c) OUM-BD—White, Final Bi-RRT*—Black, Initial Bi-RRT*- Dotted superimposed on the sum of direction-independent weights.



(d) Progression of best cost using Bi-RRT* over 30 min sampled every minute. The OUM-BD cost and time (the circle on the bottom line) are shown for comparison.

Fig. 11. *Example 3*—Optimal path with all terms included in weight $g(\mathbf{x}, \mathbf{u})$: $a_{sr} = 100$ ($b_1 = 0.5$, $b_2 = 1.5$, $b_3 = 2.5$), $a_{risk} = 1000$, $a_{so} = 10$, with obstacles ($M = 10^6$). The rover has width 6 units and length 3 units.

Definition A.2. A **viscosity supersolution** $\overline{V} : \Omega \to \mathbb{R}$ of (7) satisfies the following.

1. $\overline{V}(\mathbf{x}_f) \geq 0$.
2. If $\phi \in C^1(\Omega)$ satisfies

$$\begin{cases} \overline{V}(\mathbf{x}) \geq \phi(\mathbf{x}) & \text{for } \mathbf{x} \in \Omega \setminus \{\mathbf{x}_f\}, \\ \overline{V}(\mathbf{x}_0) = \phi(\mathbf{x}_0) & \text{at } \mathbf{x}_0 \in \Omega \setminus \{\mathbf{x}_f\}, \end{cases}$$

then $H(\mathbf{x}_0, \nabla\phi) \geq 0$.

Definition A.3. A **viscosity solution** of the HJB (7) is both a viscosity subsolution and a viscosity supersolution of (7).

Definition A.4. The vector $\mathbf{y} \in \mathbb{R}^2$ is a **subgradient** of a function $f : \mathbb{R}^2 \to \mathbb{R}$ at $\mathbf{x}_0 \in \mathbb{R}^2$ if there exists $\delta > 0$ such that for any $\mathbf{x} \in B_\delta(\mathbf{x}_0)$,

$$f(\mathbf{x}) - f(\mathbf{x}_0) \geq \mathbf{y} \cdot (\mathbf{x} - \mathbf{x}_0).$$

Let $D^- f(\mathbf{x}_0)$ denote the set of all subgradients of f at $\mathbf{x}_0$.

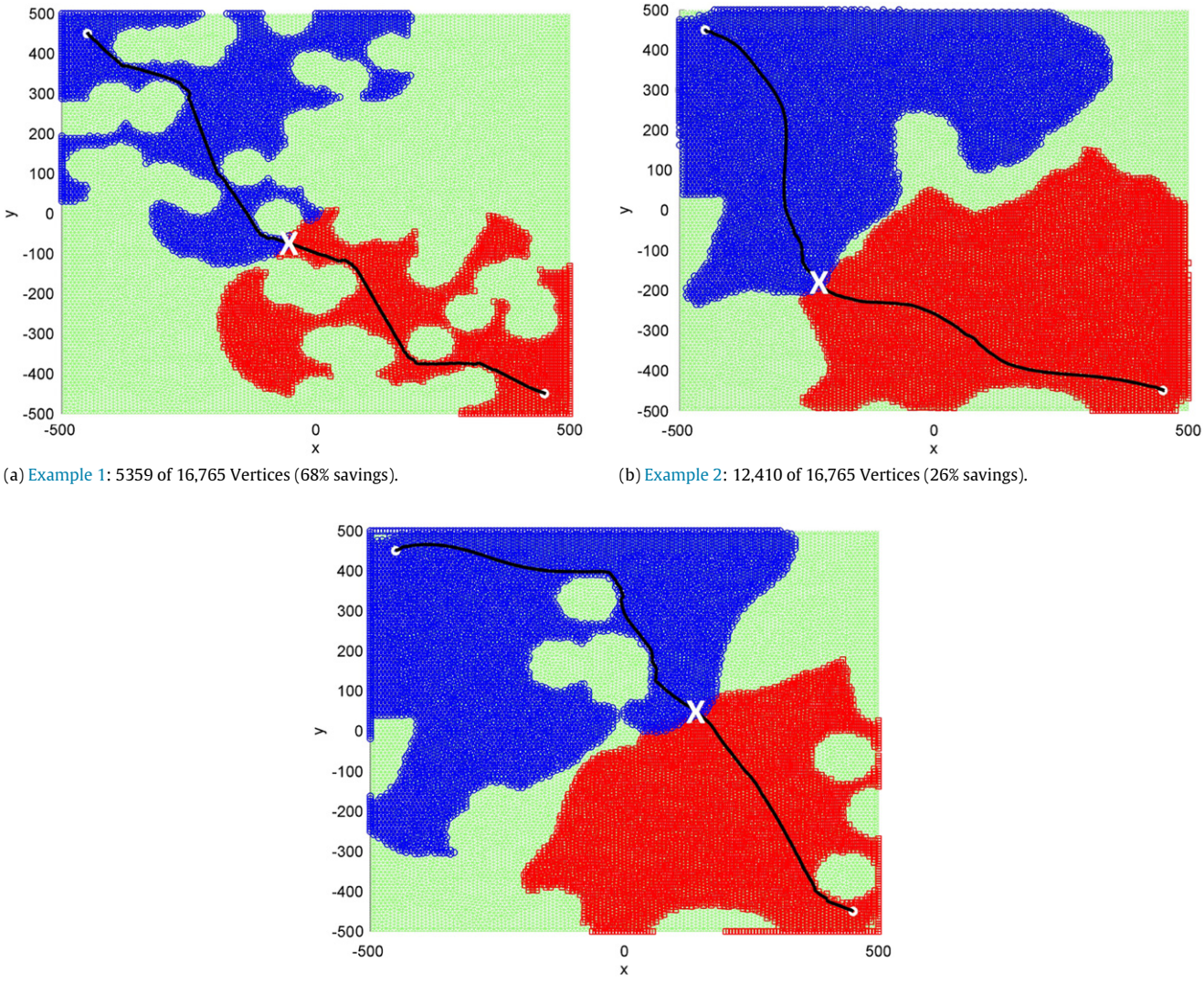
Lemma A.5. *The point $\mathbf{x}_0 \in \mathbb{R}^2$ is a local minima of f if and only if $\mathbf{0}$ is a subgradient of f at $\mathbf{x}_0$.*

The proof is obvious. The following lemma relates the set $D^- f(\mathbf{x}_0)$ and the possible values of $\nabla\phi(\mathbf{x}_0)$ of the test function in the viscosity supersolution given previously.

Lemma A.6 (*[30, Lemma 1.7]*). *The vector $\mathbf{v} \in D^- f(\mathbf{x}_0)$ if and only if there exists $\phi \in C^1(\Omega)$ such that $\nabla\phi(\mathbf{x}_0) = \mathbf{v}$, $f(\mathbf{x}_0) = \phi(\mathbf{x}_0)$ and $f(\mathbf{x}) \geq \phi(\mathbf{x})$.*

Theorem A.7. *If V is a viscosity solution of the HJB equation (7) where the weight function satisfies (3), then V has no local minima on $\Omega \setminus \{\mathbf{x}_f\}$.*

Proof. It will be shown that $\mathbf{x}_0 \in \Omega \setminus \{\mathbf{x}_f\}$ is not a local minimum. The viscosity solution V is a viscosity supersolution. If there is no $\phi \in C^1(\Omega)$ that satisfies the second property of Definition A.2 ($V(\mathbf{x}) \geq \phi(\mathbf{x})$ for $\mathbf{x} \in \Omega$ and $V(\mathbf{x}_0) = \phi(\mathbf{x}_0)$), then by Lemma A.6, $D^- V(\mathbf{x}_0)$ is empty and does not contain $\mathbf{0}$. Hence by Lemma A.5,



(a) Example 1: 5359 of 16,765 Vertices (68% savings).

(b) Example 2: 12,410 of 16,765 Vertices (26% savings).

(c) Example 3: 11,401 of 16,765 Vertices (32% savings).

Fig. 12. Bi-directional Fronts: The vertices in OUM-BD labelled *Accepted* in each example with the optimal path found. The front found from $\mathbf{x}_f = (-450, 450)$ is shown with circles, and the front found from $\mathbf{x}_0 = (450, -450)$ is shown with squares. The point at which the fronts meet is labelled with a white X.

$\mathbf{x}_0$ is not a local minimum. Now suppose such a $\phi \in C^1(\Omega)$ exists ($V(\mathbf{x}) \geq \phi(\mathbf{x})$ for $\mathbf{x} \in \Omega$ and $V(\mathbf{x}_0) = \phi(\mathbf{x}_0)$). Hence,

$$H(\mathbf{x}_0, \nabla\phi) \geq 0,$$

or

$$-\min_{\mathbf{u}\in\mathbb{S}^1}\{\nabla\phi(\mathbf{x}_0)\cdot\mathbf{u} + g(\mathbf{x}_0, \mathbf{u})\} \geq 0,$$

and let $\mathbf{u}^* \in \mathbb{S}^1$

$$-\nabla\phi(\mathbf{x}_0)\cdot\mathbf{u}^* \geq g(\mathbf{x}_0, \mathbf{u}^*) \geq g_1(\mathbf{x}_0) > 0,$$
$$|\nabla\phi(\mathbf{x}_0)| \geq -\nabla\phi(\mathbf{x}_0)\cdot\mathbf{u}^* > 0.$$

Thus $\nabla\phi(\mathbf{x}_0) \neq \mathbf{0}$ which implies by Lemma A.6 that $\mathbf{0} \notin D^- V(\mathbf{x}_0)$. By Lemma A.5, $\mathbf{x}_0$ is not a local minimum of V. Since $\mathbf{x}_0$ was an arbitrary point on $\Omega \setminus \{\mathbf{x}_f\}$, there are no local minima on $\Omega \setminus \{\mathbf{x}_f\}$. □

References

[1] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.

[2] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. SSC44 2 (1968) 100–107.

[3] J.A. Sethian, Level Set Methods and Fast Marching Methods: Computer Vision and Material Sciences, Cambridge University Press, 1999.

[4] J. Bellingham, A. Richards, J. How, Receding horizon control of autonomous aerial vehicles, in: American Control Conference, 2002.

[5] L. Blackmore, H. Li, B. Williams, A probabilistic approach to optimal robust path planning with obstacles. In Proceedings of the American Control Conference, 2006, 2006, pp. 2831–2837.

[6] L. Blackmore, M. Ono, B.C. Williams, Chance-constrained optimal path planning with obstacles, IEEE Trans. Robot. 27 (2011) 1080–1094.

[7] C. Ma, R. Miller, MILP optimal path planning for real-time applications, in: Proceedings of the 2006 American Control Conference, Minneapolis, Minnesota, 2006, pp. 4945–4950.

[8] K. Yang, S. Gan, S. Sukkarieh, An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments, J. Intell. Robot. Syst. 57 (2010) 101–122.

[9] J. Kuffner, RRT-connect: An efficient approach to single-query path planning, in: Proc. IEEE Int'l Conf. on Robotics and Automation, 2000.

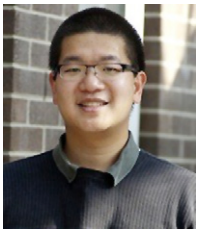
[10] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (2011) 846–894.

[11] B. Akgun, M. Stilman, Sampling heuristics for optimal motion planning in high dimensions, in: IEEE/RSJ Intelligent Conference on Intelligent Robots and Systems, 2011.

[12] J. Sethian, A. Vladimirsky, Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms, SIAM J. Numer. Anal. 41 (2003) 325–363.

[13] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robot. Res. 5 (1) (1986) 90–98.

[14] A. Shum, Optimal direction-dependent path planning for autonomous vehicles (Ph.D. thesis), University of Waterloo, 2014.
[15] J.A. Sethian, A marching level set method for monotonically advancing fronts, Proc. Natl. Acad. Sci. 93 (1996) 1591–1595.
[16] R. Kimmel, J. Sethian, Optimal algorithm for shape from shading and path planning, J. Math. Imaging Vision 14 (2001) 237–244.
[17] R. Kimmel, J.A. Sethian, Computing geodesic paths on manifolds, Proc. Natl. Acad. Sci. USA 95 (1998) 8431–8435.
[18] S. Fallah, B. Yue, O. Vahid-Araghi, A. Khajepour, Energy management of planetary rovers using a fast feature-based path planning and hardware-in-the-loop experiments, IEEE Trans. Veh. Technol. 62 (2013) 2389–2401.
[19] M. Maimone, J. Biesiadecki, E. Tunstel, Y. Cheng, C. Leger, Intelligence for Space Robotics: Surface Navigation and Mobility Intelligence on the Mars Exploration Rovers, TSI Press, 2006.
[20] E. Papadopoulos, D. Rey, A new measure of tipover stability margin for mobile manipulators, Proc. IEEE Int. Conf. Robot. Autom. 4 (1996) 3111–3116.
[21] L.C. Evans, Partial Differential Equations, second ed., in: Graduate Studies in Mathematics, vol. 19, American Mathematical Society, 2010.
[22] A. Vladimirsky, Fast methods for static Hamilton–Jacobi partial differential equations (Ph.D. thesis), University of Califoria, Berkeley, 2001.
[23] A. Kumar, A. Vladimirsky, An efficient method for multiobjective optimal control and optimal control subject to integral constraints, J. Comput. Math. 28 (2010) 517–551.
[24] I. Pohl, Bi-directional search, Mach. Intell. 6 (1971) 127–140.
[25] Z. Clawson, A. Chacon, A. Vladimirsky, Causal domain restriction for Eikonal equations, SIAM J. Sci. Comput. 36 (2014) A2478–A2505.
[26] D. Engwirda, MESH2D—Automatic mesh generation, in: MATLAB Central, 2011.
[27] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes: the Art of Scientific Computing, Cambridge University Press, 2007.
[28] A. Vladimirsky, Static PDEs for time-dependent control problems, Interfaces Free Bound. 8/3 (2006) 281–300.
[29] I. Mitchell, S. Sastry, Continuous path planning with multiple constraints, IEEE Conf. Decis. Control 42 (2003) 5502–5507.
[30] M. Bardi, I. Capuzzo-Dolcetta, Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations, Birkhäuser, 1997.



Alex Shum earned his B.Sc. Eng. at Queen's University in applied mathematics and mechanical engineering in 2007, and his M. Math in applied mathematics at the University of Waterloo in 2009. He completed his Ph.D. in 2014 at the University of Waterloo in the department of applied mathematics. His research interests include control theory, optimal control, numerical approximations of Hamilton–Jacobi–Bellman equations and math education.

Prof. Kirsten Morris' main research interests are systems modelled by partial differential equations and also systems, such as smart materials, involving hysteresis and has written numerous papers on these subjects as well as a textbook "Introduction to Feedback Control", and was editor of the book "Control of Flexible Structures". Professor Morris is a member of the Applied Mathematics Department at the University of Waterloo and is cross-appointed to the Department of Mechanical & Mechatronics Engineering. She was an associate editor with the IEEE Transactions on Automatic Control during 2007–2012, SIAM Journal on Control & Optimization 2010–2012 and has been a member of the editorial board of the SIAM book series Advances in Design & Control since 2005. Prof. Morris has served as a member of the IEEE Control System Society Board of Governors since 2010 and is currently vice-president, membership activities.

Amir Khajepour's area of research is in systems modelling and control of dynamic systems. He has developed an extensive research program that applies his expertise in several key multidisciplinary areas. His research has resulted in several patents, technology transfers, and more than 300 journal and conference publications, as well as several books. He is a recipient of three best paper awards, a fellow of the American Society of Mechanical Engineers (ASME) and the Canadian Society of Mechanical Engineering (CSME).