COMMUN. MATH. SCI.
Vol. 17, No. 1, pp. 227–242

OPTIMAL HUMAN NAVIGATION IN STEEP TERRAIN: A HAMILTON-JACOBI-BELLMAN APPROACH*

CHRISTIAN PARKINSON†, DAVID ARNOLD‡, ANDREA L. BERTOZZI§, YAT TIN CHOW¶, AND STANLEY OSHER‖

Abstract. We present a method for determining optimal walking paths in steep terrain using the level set method and an optimal control formulation. By viewing the walking direction as a control variable, we can determine the optimal control by solving a Hamilton-Jacobi-Bellman equation. We then calculate the optimal walking path by solving an ordinary differential equation. We demonstrate the effectiveness of our method by computing optimal paths which travel throughout mountainous regions of Yosemite National Park. We include details regarding the numerical implementation of our model and address a specific application of a law enforcement agency patrolling a nationally protected area.

Keywords. Optimal path planning; Hamilton-Jacobi-Bellman equation; optimal control; level set method; anisotropic control.

AMS subject classifications. 00A69; 34H05; 35F21; 49L20.

1. Introduction

We consider the problem of determining the optimal walking path between two points given elevation data in a region. If the terrain is fairly flat, this may be very easy as conventional wisdom ("the shortest path between two points is a straight line") will provide a good approximation to the optimal path. However, in mountainous regions, straight line travel is often inefficient or impossible and the optimal path between two points is no longer clear.

The problem of optimal path planning goes back at least as far as Dijkstra [6] who designed an algorithm for optimally traversing weighted graphs. In the years since, significant effort has been devoted to developing and improving algorithms which find optimal or near-optimal paths in a discrete setting [10,22,26]. Others have used modified versions of Dijkstra's algorithm for path planning in a semi-continuous setting [2, 41].

Recently, path planning problems have been largely reframed using control theory and partial differential equations. An early approach was to compute geodesics on triangulated manifolds using an Eikonal equation and gradient descent [14]. One interesting application of path planning problems is modeling simple, autonomous robots. These so-called *Dubins' cars* were constrained by a maximum turning radius so Dubins considered paths with bounded local curvature [7]. Recently, this problem was reformulated using a Hamilton-Jacobi-Bellman equation and adapted to include impassable obstacles [1, 40]. Indeed, Hamilton-Jacobi-Bellman (HJB) equations are now used extensively in optimal path problems. Sethian and Vladmirsky compute paths by realizing a HJB equation as a continuous version of Dijkstra's algorithm [33, 34]. Recent research employs HJB equations in determining reachable and avoidable sets when traveling from a given ground state [5, 19]. Tomlin et al. also use HJB equations in

*Received: May 9, 2018; Accepted (in revised form): November 5, 2018. Communicated by Jianfeng Lu.

†Department of Mathematics, UCLA, Los Angeles, CA 90095, USA (chparkin@math.ucla.edu).
‡Department of Mathematics, UCLA, Los Angeles, CA 90095, USA (darnold@math.ucla.edu).
§Department of Mathematics, UCLA, Los Angeles, CA 90095, USA (bertozzi@math.ucla.edu).
¶Department of Mathematics, UC Riverside, Riverside, CA 92521, USA (ytchow@math.ucla.edu).
‖Department of Mathematics, UCLA, Los Angeles, CA 90095, USA (sjo@math.ucla.edu).

adversarial reach-avoid games wherein a group of attackers attempt to reach a target set while also avoiding defenders [42]. Others have considered optimal travel in regions which randomly switch between different states; for example, this randomness could account for the effect or weather patterns on a sailboat [35].

There has been some research into path planning in a geographical or terrain-based setting though most previous work is focused on discrete, graph-based methods employing Dijkstra's algorithm and its many variants: so-called A^* and D^* algorithms [15,30]. Such methods have long been used for vehicular navigation and can be adapted to include real-time obstacle recognition [20]. This problem is also of particular interest to those working on unmanned aerial vehicles (UAVs) and other autonomous robots [4,9, 12,18]. In a continuous approach, Popović et al. [28] propose a path planning algorithm for UAVs by maximizing an information functional which measures the amount of data a UAV can collect. However, the methods of control theory and HJB equations have yet to be applied to terrain-based path planning which means, for example, that no previous approach has been able to dynamically account for the optimal direction of travel along a path.

In Section 2, we present a model which uses the level set method and a HJB formulation to compute optimal walking paths in a continuous setting where travel direction can be considered dynamically and walking speed is dependent on slope of the local terrain. This is as opposed to other terrain-based path planning methods which are fully or partially discrete and do not account for directional movement. In Section 3, we discuss the numerical simulation of the model. We begin by testing the model against toy problems using synthetic elevation data specifically designed so that the "correct answer" is somewhat clear *a priori* and move on to use real elevation data of Yosemite National Park. Results of numerical simulations are presented in Section 4.2. The motivation for this work was to aid law enforcement agencies in efficiently patroling protected areas such as parks or forests, but with small adjustments, our method could be applied to optimal path planning in any number of scenarios.

2. Mathematical model

Our primary mathematical tool is the level set method of Osher and Sethian [24]. The level set method models propagation of fronts by treating them as the zero level set of auxiliary function ϕ, known as the *level set function*. We will discuss the method in two spatial dimensions since this is relevant for terrain-based path planning, but this can be easily generalized to higher dimensions.

2.1. The level set method. Suppose that $\Omega \subset \mathbb{R}^2$ is open and bounded with Lipschitz continuous boundary $\Gamma(0) = \partial\Omega$ which is the curve that will evolve via some level set motion. To begin, we find a Lipschitz continuous function $\phi_0 : \mathbb{R}^2 \to \mathbb{R}$ such that $\phi_0 < 0$ in Ω and $\phi_0 > 0$ in $\mathbb{R}^2 \setminus \Omega$. Continuity of ϕ_0 implies that $\Gamma(0) = \{x \in \mathbb{R}^2 : \phi_0(x) = 0\}$; that is, Γ is the zero level contour of the initial function. Next, we evolve the function $\phi : \mathbb{R}^2 \times \{t > 0\} \to \mathbb{R}$ using the Hamilton-Jacobi equation

$$\begin{aligned} \phi_t + H(x, \nabla\phi) &= 0, \\ \phi(x, 0) &= \phi_0(x), \end{aligned} \tag{2.1}$$

where the *Hamiltonian* $H(x,p)$ is homogeneous of degree one in the variable p; here p is a proxy for $\nabla\phi$. As ϕ evolves according to the PDE, we define $\Gamma(t) = \{x \in \mathbb{R}^2 : \phi(x,t) = 0\}$ (so that, in particular, $\Gamma(0) = \Gamma$) and the curve $\Gamma(t)$ evolves with level set motion which is prescribed by the Hamiltonian H [23].

In the simplest case $H(x,p)=|p|$ and (2.1) is the Eikonal equation $\phi_t+|\nabla\phi|=0$. Rewriting the equation as $\phi_t+\hat{n}\cdot\nabla\phi=0$ where $\hat{n}=\nabla\phi/|\nabla\phi|$, it is clear that locally this equation gives advection in the outward normal direction with velocity 1. This causes $\Gamma(t)$ to deform outward with normal velocity 1. In this case, for $t>0$, $\Gamma(t)$ represents the set of all points which are at distance t from the original curve $\Gamma(0)$. Equivalently, since we are considering people traveling throughout regions, $\Gamma(t)$ is the set of points which can be reached if one travels from $\Gamma(0)$ with normal velocity 1 for time t. To prescribe a different normal velocity $v(x)$ rather than allowing individuals to travel with normal velocity 1, one can simply modify the Hamiltonian by setting $H(x,p)=v(x)|p|$. Now $\Gamma(t)$ represents the set of points which can be reached if one travels from $\Gamma(0)$ with normal velocity $v(x)$ for time t.

Using the level set equation, one can compute the (approximate) time that it takes to travel from one point to another in our domain. Let $a\in\mathbb{R}^2$ represent a starting point and $b\in\mathbb{R}^2$ represent an ending point. For some small $\delta>0$, let $\phi_0(x)=|x-a|-\delta$ so that $\Gamma(0)=\{|x-a|=\delta\}$ is a small circle around the point a. When $\Gamma(t)$ evolves outward with prescribed normal velocity $v(x)$, there will be some time $t^*>0$ such that $b\in\Gamma(t^*)$; that is, at some positive time t^*, the level set will hit our ending point. This time t^* is the time required to travel from point a to point b when traveling in the normal direction with velocity $v(x)$ (neglecting the small parameter δ). This approach can be seen as a continuous analog of Dijkstra's algorithm, assigning optimal travel times to points as the level sets sweep through the region. This gives a method for calculating travel times, but this model is too simple for our purposes, only allowing for travel in the normal direction which is potentially far from optimal. For example, if in a physical setting there is a large mountain between the points a and b, one may wish to walk around the mountain rather than over the mountain, as normal direction travel may suggest. Thus at each point, one must not only consider the speed of travel, but also the direction of travel. Considering direction, it no longer makes sense to simply specify a velocity $v(x)$ at each point. Instead, we assume that walking velocity depends on both the gradient of the terrain at the current point and the direction of travel as we search for the optimal travel direction.

2.2. Our model. For our model, assume that in addition to the starting and ending points $a,b\in\mathbb{R}^2$, there is an elevation profile $E(x)$ and a velocity function $V(S)$ which gives human walking velocity as a function of terrain slope S. Let $\theta\in[0,2\pi]$ be a control variable which represents walking direction and let $s(\theta)=(\cos(\theta),\sin(\theta))$ be the corresponding direction vector. Now if one is standing at a point x and desires to walk in the direction θ, they can walk with velocity $V(s(\theta)\cdot\nabla E(x))$ since $s(\theta)\cdot\nabla E(x)$ represents the slope at x in the direction of θ. For each $\theta\in[0,2\pi]$, define the directional Hamiltonian $H_\theta(x,p)=V(s(\theta)\cdot\nabla E(x))[s(\theta)\cdot p]$. Note that using this Hamiltonian, the corresponding Hamilton-Jacobi equation models advection in the direction of θ. To consider optimal travel, we take the supremum over all θ. Define the *optimal path Hamiltonian*

$$H(x,p):=\sup_{\theta\in[0,2\pi]}H_\theta(x,p)=\sup_{\theta\in[0,2\pi]}\{V(s(\theta)\cdot\nabla E(x))[s(\theta)\cdot p]\}. \tag{2.2}$$

This results in a Hamilton-Jacobi-Bellman equation, which, after resolving the supremum in θ takes the form (2.1); it is indeed a level set equation, since this optimal path Hamiltonian is still homogeneous of degree one in the variable p. Now to find the optimal travel time between points a and b, one can use the same method described above: letting $\Gamma(0)=\{x\in\mathbb{R}^2:|x-a|=\delta\}$ for small δ, evolve $\Gamma(t)$ using the level set

equation with the optimal path Hamiltonian until the time $t^* > 0$ such that $b \in \Gamma(t^*)$. This t^* is the minimal time required to travel from a to b. This procedure is displayed in Figure 2.1.

What remains is to compute the optimal path from a to b: the path which requires time t^* to traverse. In order to do this, one simply needs to follow the characteristics of the Hamilton-Jacobi-Bellman equation. We would like to travel along characteristics originating from a toward b. However, with this small parameter δ, we have removed a small neighborhood of a and instead initiate the motion from the circle $\Gamma(0)$. Note for example, that ϕ_0 is non-differentiable at a. Accordingly, one should follow the characteristics backwards from b to $\Gamma(0)$. The characteristic equations are

$$\begin{aligned} \dot{x} &= -\nabla_p H(x,p),\ x(0) = b, \\ \dot{p} &= \ \ \nabla_x H(x,p),\ p(0) = \nabla\phi(b,t^*). \end{aligned} \tag{2.3}$$

Physically, one can imagine starting at the point b, considering what was the direction of the optimal step which led to the current point, stepping backwards in that direction and updating the direction in real time as one is walking backwards. Running this system of ODEs to time t^*, one will have backtracked optimally from b to $\Gamma(0)$.

To summarize, once we have defined the optimal path Hamiltonian (2.2), the algorithm for finding the optimal path consists of two steps:

(1) Find the optimal travel time by advancing the PDE

$$\begin{aligned} \phi_t + H(x,\nabla\phi) &= 0, \\ \phi(x,0) &= |x-a| - \delta, \end{aligned}$$

until the time $t^* > 0$ such that $b \in \Gamma(t^*)$.

(2) Find the optimal travel path by advancing the ODE system

$$\dot{x} = -\nabla_p H(x,p),\ x(0) = b,$$

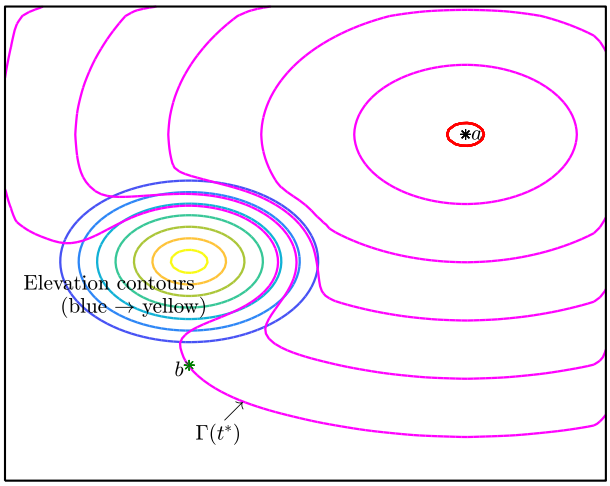


Fig. 2.1: *To find optimal travel time, begin with a small circle around a (red) and evolve level sets* $\Gamma(t)$ *(magenta) outward until the time* $t^* > 0$ *at which* $b \in \Gamma(t^*)$.

$$\dot{p} = \quad \nabla_x H(x,p),\ p(0) = \nabla\phi(b,t^*)$$

until time t^*.

In another approach, Sethian and Vladimirsky [33, 34] devise a static Hamilton-Jacobi-Bellman formulation for path planning. In doing so, they are able to solve similar problems without the time-dependence that is present in our model. While removing the time-dependence eliminates a dimension, since we are only solving problems in two spatial dimensions, our algorithm is sufficiently efficient for our purposes, and has the advantage that it is simple to implement at any order of accuracy one desires. If efficiency is a concern, local level set methods [17, 21, 27] could be used to speed up the level set computation.

2.3. The associated control problem. Since we are determining optimal travel, we know that underlying the formalism of Section 2.2, there is a control problem that is being solved and a payoff function which is being maximized. As above, let $a \in \mathbb{R}^2$ be the initial point. If one is standing at the point $x \in \mathbb{R}^2$, then traveling optimally away from the point a for a time t is the same as maximizing the distance $|\mathbf{x}(t) - a|$, where $\mathbf{x}(\tau)$, $0 \le \tau \le t$ is a path with $\mathbf{x}(0) = x$. At each time along the path, denote the direction of travel by $\theta(\tau)$. As discussed above, the travel velocity at the point $\mathbf{x}(\tau)$ and in the direction $\theta(\tau)$ is given by $V(s(\theta(\tau)) \cdot \nabla E(\mathbf{x}(\tau)))$. Thus, the problem can be phrased as such: maximize the payoff function

$$P_{x,t}(\theta(\cdot)) = |\mathbf{x}(t) - a| \tag{2.4}$$

among measurable functions $\theta : [0,t] \to [0, 2\pi]$ and subject to the constraint

$$\begin{aligned} \dot{\mathbf{x}}(\tau) &= V\big(s(\theta(\tau)) \cdot \nabla E(\mathbf{x}(\tau))\big) s(\theta(\tau)),\ 0 \le \tau \le t \\ \mathbf{x}(0) &= x. \end{aligned} \tag{2.5}$$

Computing formally, one sees that the HJB equation associated with the value function $u(x,t) = \sup_\theta P_{x,t}(\theta(\cdot))$ for this control problem is (2.1) with the optimal path Hamiltonian (2.2) and initial condition $\phi_0(x) = |x - a|$. We then make the slight modification $\phi_0(x) = |x - a| - \delta$ for small positive δ so that we may utilize the level set method to track optimal travel away from a for every point on $\Gamma(0)$ simultaneously. To make this rigorous, one could require that the map $x \mapsto V(s(\theta) \cdot \nabla E(x))$ is Lipschitz with a Lipschitz constant which is uniform in θ. When this holds, the value function will be the viscosity solution of the HJB Equation (2.1). However, this requirement will, in turn, depend upon the smoothness of the elevation data E, which is something we cannot guarantee, so we emphasize that, in our case, the connection between the value function for the optimal control problem and our HJB equation is merely formal.

2.4. Accounting for uncertainty in the starting point. The above algorithm will compute a path for one who wishes to travel optimally throughout a region. We would like to incorporate some uncertainty into the model to account for a real world situation which law enforcement agents may encounter. Consider a scenario wherein a law enforcement agency has knowledge that environmental criminals (for example, poachers or illegal loggers) are operating within a protected region but can only identify the criminals' location with some uncertainty. Supposing that the criminals perpetrate a crime within the region and then travel to a known final destination, the law enforcement agency may want to predict which paths the criminals will take.

In this situation, assume that rather than a starting point a, we have a compact set A of possible starting points along with a probability distribution from which one

can sample elements of A. The algorithm described above requires a starting point which could be drawn from A upon which one could calculate an optimal path to the end point b. However, we wish to calculate the optimal path to b from each point in A and according to our procedure, this will require solving (2.1) for each point in A. Computationally, this would be very inefficient, so instead one can invert the problem: rather than starting from a point $a \in A$ and evolving level sets outward toward b, one should evolve level sets outward from b. As the level sets $\Gamma(t)$ evolve outward from b, they sweep through the region A so that for each $a \in A$, we find a time $t^*(a)$ such that $a \in \Gamma(t^*(a))$. The computation can be stopped when A is inside $\Gamma(t)$ and for each $a \in A$, we will have found the time $t^*(a)$ required to travel from a to b. This is pictured in Figure 2.2. Having done this, we can draw points N points $\{a_i\}_{i=1}^N$ from A and calculate the optimal paths using (2.3) with starting values $x(0) = a_i$, $p(0) = \nabla\phi(a_i, t^*(a_i))$. In this way, we can calculate optimal paths to N points in A with only one level set computation.

As a minor note, walking velocity is maximal when one is walking on a slight decline. Thus reversing the direction of the level set evolution means we must also reverse our sense of slope since the walking direction is now opposite the direction of the level set evolution. Hence, we replace E with $-E$. In doing so, when we evolve level sets outward from b, we are actually calculating optimal travel as if one was traveling inward toward b. Thus we can still compute the optimal path from a to b even though we use b as the "starting point" for the level sets.

3. Numerical framework

We discuss in detail the numerical methods that we use to simulate our model. The first obstacle is deciding how to calculate our Hamiltonian since this requires a maximization over $\theta \in [0, 2\pi]$. If the velocity function V is sufficiently simple, it may be possible to resolve this maximization explicitly using calculus. When this is not possible (as with our simulations), one can maximize H discretely. That is, rather than maximize over $\theta \in [0, 2\pi]$, we maximize $H_\theta(x, p)$ over the finite set $\theta \in \{\frac{2\pi m}{M} : m = 1, \ldots, M\}$. This

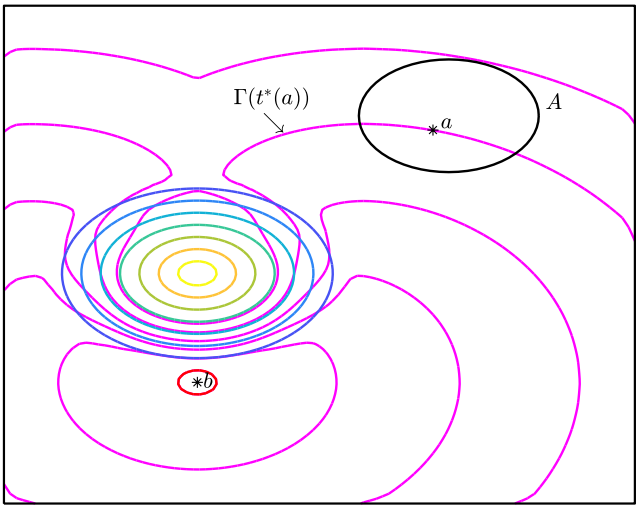


Fig. 2.2: *If there is uncertainty in the location of the starting point, we evolve level sets outward from b until they cover A, recording optimal times for each $a \in A$ as we go.*

causes some approximation error, but as long as $V(s(\theta)\cdot\nabla E(x))$ is continuous in θ for fixed x, this discrete maximization will tend to the exact supremum as $M\to\infty$.

Next, one must decide how to solve (2.1) numerically. There has been much research into efficient and accurate numerical methods for Hamilton-Jacobi equations [3, 13, 27, 39]. Since these equations are (in general) nonlinear, naive differencing methods will not always work. Instead, we trade the Hamiltonian $H(x,\phi_x,\phi_y)$ for a numerical Hamiltonian $\hat{H}(x,\phi_x^+,\phi_x^-,\phi_y^+,\phi_y^-)$ which somehow averages the forward difference and backward difference approximations to ϕ_x and ϕ_y, represented here by ϕ_x^+,ϕ_x^- and ϕ_y^+,ϕ_y^- respectively. We then advance the PDE via explicit time-stepping. Osher and Shu [25] give several suggestions for different types of numerical Hamiltonians and describe methods for attaining higher-order accuracy. For our purposes, we use the Godunov Hamiltonian given by

$$\hat{H}(x,\phi_+^x,\phi_-^x,\phi_+^y,\phi_-^y)=\operatorname*{ext}_{u\in I(\phi_-^x,\phi_+^x)}\ \operatorname*{ext}_{v\in I(\phi_-^y,\phi_+^y)} H(x,u,v) \tag{3.1}$$

where

$$I(a,b)=[\min(a,b),\max(a,b)] \tag{3.2}$$

and

$$\operatorname*{ext}_{x\in I(a,b)}=\begin{cases}\min_{a\le x\le b} & \text{if } a\le b,\\ \max_{b\le x\le a} & \text{if } a>b.\end{cases} \tag{3.3}$$

These extrema are designed to take into account the direction in which information is flowing and, as a result, the Godunov Hamiltonian gives a fully upwind scheme. Again, we need to perform minimization or maximization computationally and again, in certain cases, these can be resolved explicitly (for example, if $H(x,u,v)$ is monotone in the arguments (u,v)), but this is not possible in our case, so we do this discretely. The Godunov Hamiltonian $\hat{H}$ gives a first-order approximation to the Hamiltonian H. Following Osher and Shu [25], we use second-order, essentially non-oscillatory approximations for the derivatives ϕ_x,ϕ_y and second-order total variation diminishing Runge-Kutta time stepping to evolve the solution. In doing so, we have constructed a second-order accurate scheme for (2.1). Finally, one can solve the optimal path ODE system (2.3) using any method one wishes. For relatively jagged elevation data E, the equation can become stiff, so it is recommended that one uses a stiff solver with accuracy which matches that of the numerical solution to (2.1).

While this describes the basics of the numerical implementation, there are some minor adjustments required to obtain our results which we discuss in Section 4.3. Some of these issues are caused by roughness in the terrain data. Indeed, rough terrain data may violate the smoothness conditions required to ensure second-order accuracy, and for this reason, first-order schemes may be sufficient to solve this problem. However, there is no significant difficulty in implementing the above scheme at second-order. This will provide second-order accuracy in regions where terrain is relatively smooth, and in some cases we observed slightly improved results (in terms of optimal travel time) using the second-order scheme.

4. Implementation & results

The model was implemented in MATLAB and in the succeeding section we discuss the results of the simulations and some issues which one may encounter. Before this, it remains to decide what elevation data to use and what form the velocity function takes.

4.1. Elevation & velocity. For the velocity function, we use a slight modification of the function Irmischer and Clarke [11]. Irmischer and Clarke analyze human walking speed data and suggest the function

$$V_{IC}(S) = 0.11 + \exp\left(-\frac{(100S+2)^2}{1800}\right) \tag{4.1}$$

where $S = \frac{\text{rise}}{\text{run}}$. However, in their paper, they only considered slopes up to 45° (grades up to 100%), and their function is bounded below by 0.11. This is a good starting point, but we would like to consider slopes much higher than 45° where walking speed may become very small. Accordingly, using a slightly different ansatz and fitting the denominator in the exponential, we have arrived at our own velocity function which approximates the Irmischer and Clarke function for small slopes, but which decays to zero for more extreme slopes:

$$V(S) = 1.11\exp\left(-\frac{(100S+2)^2}{2345}\right). \tag{4.2}$$

While this function is never exactly zero, it is no longer bounded from below by any positive number. It bears mentioning that the exact form of the velocity function is not terribly important for the model so long as the function $V(S)$ that we choose has $\max V(S) \approx 1$, is fairly near the maximum for all $S \approx 0$ and is nearly zero for $|S|$ large. Our velocity function is plotted against the Irmischer-Clarke velocity function in Figure 4.1.

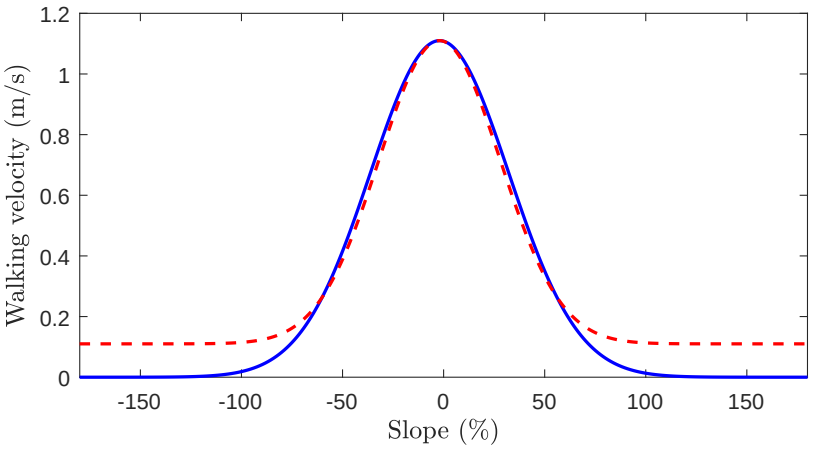


Fig. 4.1: *Comparison of our velocity function (blue, solid) with Irmischer & Clarke's velocity function (red, dotted).*

To test our code, we first ran simulations with synthetic (and very simple) elevation data. This allows us to gauge whether our model aligns with our intuition. When we were confident that our model and numerical methods were correct, we were able to download real elevation data from the United States Geological Survey and run simulations in a real national park. For our simulations, we chose Yosemite National Park and we ran optimal path simulations in the direct vicinity of the mountain El Capitan. Specifically, we use data spanning longitude 119°W - 120°W and lattitude

38°N - 39°N with 1/3 arcsecond resolution which we obtained from the USGS National Map Viewer. The data was processed and re-formatted using QGIS [29] and imported to MATLAB using TopoToolbox [31, 32].

4.2. Results. As in Figure 2.1 above, in the following images, the starting point a is represented by the black asterisk surrounded with a red circle which denotes the starting contour $\Gamma(0)$. Next, the magenta contours represent several steps in the evolution of the contours $\Gamma(t)$. The green asterisk represents the point b and the thick black line represents the optimal path from a to b. The elevation contours are plotted in colors ranging from blue representing low elevation to yellow representing high elevation. In our first simulations, we place mountains in certain areas and our intuition tells us that the optimal path should likely bend around the mountains since it would require too much effort to climb up the mountain. Our simulations do indeed reflect this; see Figure 4.2.

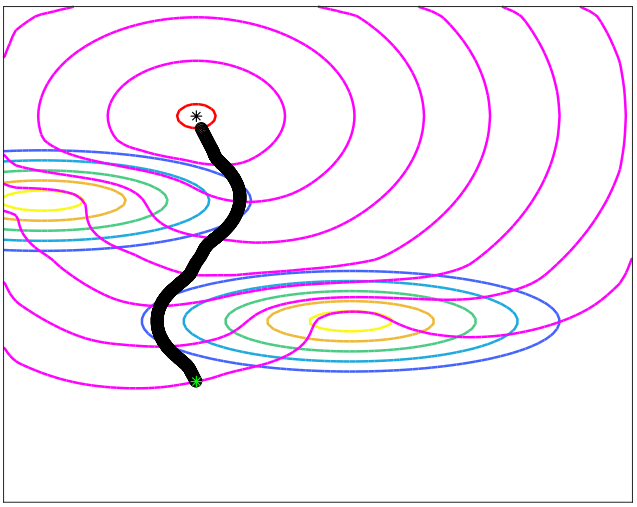


Fig. 4.2: *Optimal path winding around two mountains (toy problem).*

Next, we use actual elevation data from the area surrounding El Capitan. Before showing the result of the simulation, we show the elevation profile and the starting and ending points in Figure 4.3. Note that directly above the endpoint, there is a very steep cliff face which should be nearly impossible to traverse. Thus we would expect the optimal path to travel to the east or west, descending down a gully rather than a cliff. Indeed, this is shown to happen in Figures 4.4a, 4.4b, wherein the path travels down the eastern or western slope depending on the location of the initial point.

Finally, we ran our algorithm which accounts for uncertainty in the location of the starting point. Before we display our results, we remind the reader of the distinction here. In all of these above results, we are calculating one optimal path from the point a to the point b. Now we wish to calculate several optimal paths from the region A to the point b. Whereas previously we evolved level sets outward from point a until they reach the point b, now we evolve level sets outward from b until they subsume the region A and record the optimal travel time for each $a \in A$ as the level sets sweep through the region. This is shown in Figure 2.2. We ran our algorithm in two different areas within Yosemite. We let A be a circle near the peak of El Capitan and calculated the optimal

path down the mountain from 100 random points drawn uniformly from A. We then did the same thing but using the elevation profile of Half Dome, another peak in Yosemite National Park. The results are pictured in Figures 4.5a, 4.5b.

Note that in both cases, while there are 100 randomly chosen starting positions, all of the paths eventually conform to one of very few routes. We seek to quantify this similarity between some paths. Suppose we have calculated several paths $\{P_i\}_{i=1}^N$ starting from different locations. For each path, we know the time $T_i > 0$ required to traverse the path, so that $P_i : [0,T_i] \to \mathbb{R}^2$ and the paths are oriented so that $P(0) = b$ and $P(T_i) = a_i$, the i^{th} starting location. However, we are not concerned with where a path began, we would like to classify the route that the path eventually takes. Thus, we only consider points along the path which are outside of the set A; that is, if we define $T_i^* = \sup\{t : P_i(t) \notin A\}$, we would only like to compare the paths on the intervals $[0,T_i^*]$. After making this restriction, one can then re-parametrize using $\tau = t/T_i^*$ so that for each i, $P_i : [0,1] \to \mathbb{R}^2$ denotes a path from the ending point b to the set A. Then it is easy to define a metric to judge whether two such paths lie nearby each other: for two paths P,Q, define $d(P,Q) = \int_0^1 ||P(\tau) - Q(\tau)|| \mathrm{d}\tau$. With this metric, one can evaluate the pairwise distances between our paths, $\{d(P_i,P_j)\}_{i,j=1}^N$. Now, using basic clustering algorithms, one can categorize the paths into collections which are morally the same, in the sense that they eventually collapse onto the same route. We performed this clustering for the above two examples. Specifically, we used k-means clustering with $k = 2$ clusters in each case, though other clustering methods could be used. The results are included in Figures 4.6a, 4.6b, where the first cluster of paths is depicted in red and the second in blue. Here, we have plotted each of the 100 paths as well as the mean path for each cluster. Thus any initial point which is marked with a blue circle has a corresponding optimal path which eventually closely resembles the bright blue path and any initial point which is marked with a red asterisk has a corresponding optimal path which eventually closely resembles the bright red path. Returning to our original motivation, these graphics could be of great interest to law enforcement agencies who are tracking criminal movement. For example, in the case of El Capitan (Figure 4.6a),

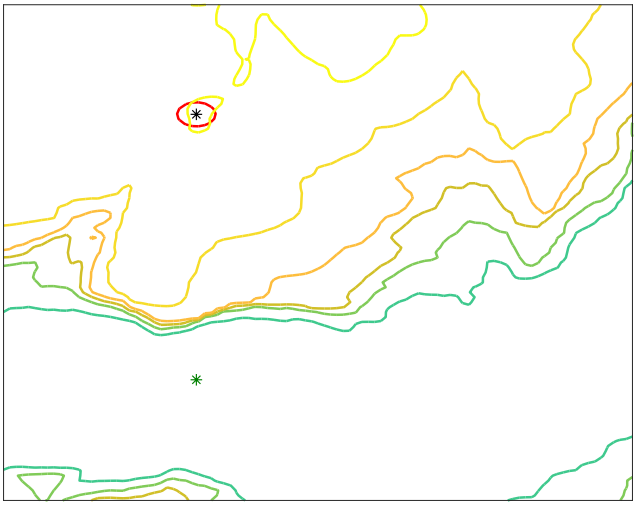
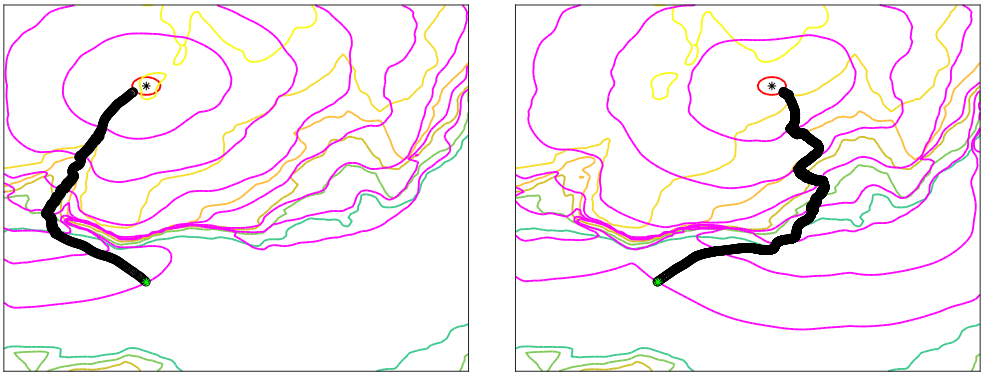
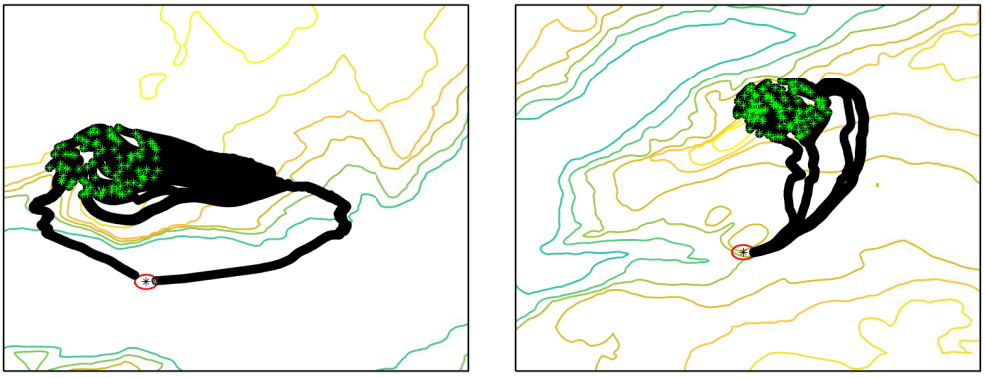


Fig. 4.3: *Elevation profile of El Capitan.*



(a) *Optimal path down the western slope.* (b) *Optimal path down the eastern slope.*

Fig. 4.4: *Optimal paths down from El Captian avoid the steep cliff.*



(a) *100 optimal paths traveling down from the summit of El Capitan.* (b) *100 optimal paths traveling down from the summit of Half Dome.*

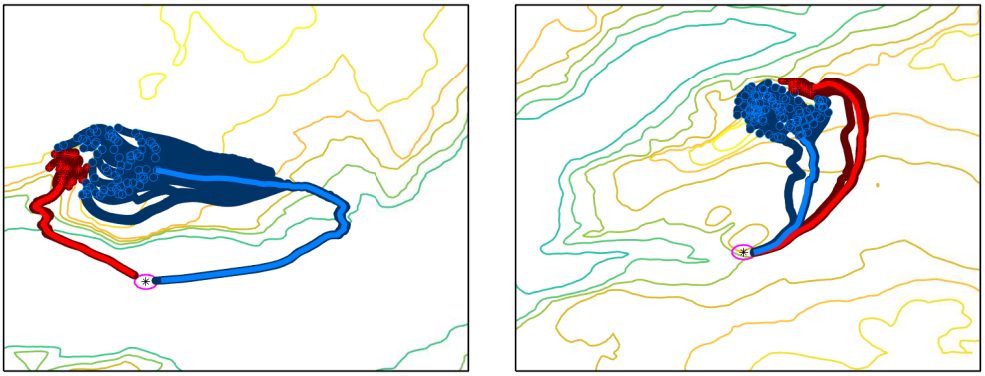
Fig. 4.5: *Calculation optimal paths accounting for uncertainty in the initial location.*

we observe that 20% of the paths travel down the western slope while 80% travel down the eastern slope. This may suggest to law enforcement that they should patrol the eastern slope with roughly four times the resources which they devote to the western slope.

4.3. Implementation notes. There are a few specific issues that arise when implementing the model numerically. We discuss three such issues (and their resolutions) and demonstrate their effects in Figures 4.7a-4.9b. First, note that the initial function $\phi(x,0)$ gives precisely the signed distance from x to $\Gamma(0)$; that is

$$\phi(x,0) = \text{dist}(x,\Gamma(0)) := \begin{cases} \inf_{y\in\Gamma(0)} |x-y|, & x \text{ inside } \Gamma(0), \\ -\inf_{y\in\Gamma(0)} |x-y|, & x \text{ outside } \Gamma(0). \end{cases}$$

As the level sets evolve, there is some distortion so that for $t>0$ we no longer have $\phi(x,t) = \text{dist}(x,\Gamma(t))$. This distortion happens when $|\nabla\phi|$ becomes too large or too



(a) *Clustering the paths down from El Capitan into two collections.*

(b) *Clustering the paths down from Half Dome into two collections.*

Fig. 4.6: *Clustering can help us identify which paths are morally the same. The bright blue path is the representative path for the blue points and the bright red path is the representative path for the red points.*

small near the zero level contour $\Gamma(t)$ and can cause the level set results to become unreliable. We can fix this by occasionally replacing ϕ with the signed distance function to $\Gamma(t)$. That is, we occasionally halt the time integration, reset $\phi(x,t) = \text{dist}(x,\Gamma(t))$ and continue. This process is known as *re-distancing*. The typical strategy for computing the distance function to the current contour $\Gamma(t)$ is to set $\sigma(x) = \text{sign}(\phi(x,t))$ and, initializing $d(x,0) = \phi(x,t)$, solve the equation

$$d_\tau = \sigma(x)(1 - |\nabla d|), \ \tau > 0, \tag{4.3}$$

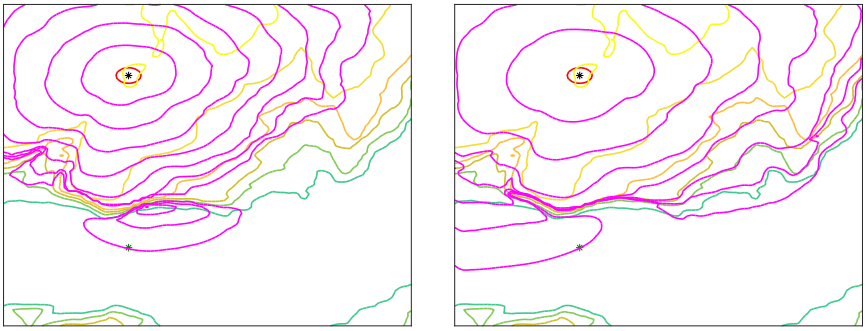
until steady state. The steady state solution will then be the signed distance function to the current contour. One advantage of this method is that we do not need to explicitly compute the contour $\Gamma(t)$; the contour is resolved implicitly by observing the sign of $\phi(x,t)$. If there is no difficulty computing the contour $\Gamma(t)$, and $\phi(\cdot,t)$ is being resolved on a grid (x_i,y_j), one can explicitly calculate the distance from each grid point to the contour, eliminating the need to solve (4.3). However, this will only be computationally feasible in very low dimension. The re-distancing problem is well established in the literature and several schemes have been developed to solve the problem [8, 16, 36–38]. Figure 4.7 shows the effect of re-distancing.

Next, as mentioned before, the system (2.3) used to find the optimal path becomes very stiff when non-smooth elevation profiles are used. Even when using a stiff solver, the results were unreliable in that the value of $p(t)$ corresponding to a location $x(t)$ was straying far from the theoretically correct value $\nabla\phi(x(t),t)$. This was causing the "optimal path" that our code found to be wildly inaccurate, often times not even connecting b to a, opting instead to wander off in some seemingly random direction. To fix this, we do something similar to the above fix: we occasionally stop the time integration, re-initialize $p(t) = \nabla\phi(x(t),t)$ and then restart the time stepping. We refer to this as *re-initialization* and the effect is shown in Figure 4.8. If one can resolve ϕ at all values of (x,t) (as opposed to resolving ϕ only on a discrete grid), one could replace the system (2.3) with the single equation

$$\dot{x} = -\nabla_p H(x, \nabla\phi(x,t)), \ x(0) = b. \tag{4.4}$$

When ϕ is not available at all points, one can solve for p using (2.3) and re-initialize as often as possible which is akin to solving (4.4).

Finally, there is still one shortcoming of our Hamiltonian with respect to directional movement: the slope in the direction of travel and its orthogonal are completely decoupled. For example, consider a situation where there is a steep cliff face in the north-south direction while the slope in the east-west direction is very mild. Our model would allow an individual to walk east-west in this situation even though they may be standing on an prohibitively steep slope. To fix this problem, we add a penalty for walking in locations where the maximum slope in *any* direction is very large. This is as simple as multiplying our Hamiltonian by a pre-factor which is approximately 1 for low slopes and approximately zero for high slopes. We have chosen the penalization function $P(S) = \frac{1}{2} - \frac{1}{2}\tanh(S-1)$ where $S = \frac{\text{rise}}{\text{run}}$ is the slope. Thus we actually solve the Hamilton-Jacobi-Bellman equation with Hamiltonian $P(|\nabla E(x)|)H(x,p)$ where $H(x,p)$ is the optimal path Hamiltonian.

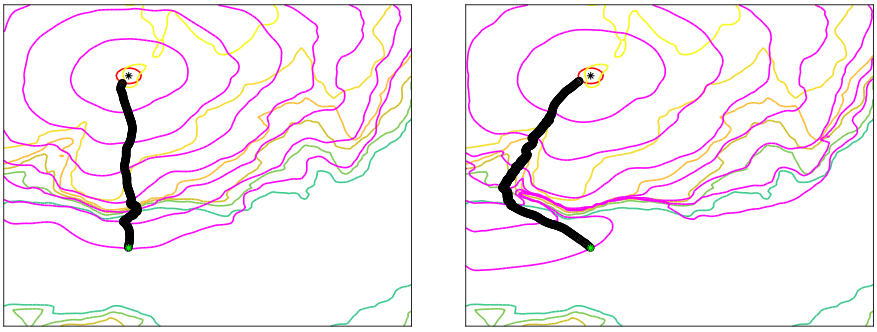


(a) *Level sets without re-distancing "jump over" the cliff.* (b) *Level sets with re-distancing wrap around the cliff.*

Fig. 4.7: *Level sets without, (a), and with, (b), re-distancing.*

(a) *Optimal path without re-initialization veers off the map.* (b) *Optimal path with re-initialization finds its target.*

Fig. 4.8: *Optimal paths without, (a), and with, (b), re-initilialization.*



(a) *Optimal path without high-slope penalization zig-zags up the cliff.*

(b) *Optimal path with high-slope penalization avoids the cliff.*

Fig. 4.9: *Optimal paths without, (a), and with, (b), the high-slope penalization.*

5. Conclusion

We have presented a method for resolving optimal walking paths given terrain data. The key element of the model is a generalization of the level set equation. By representing the direction of travel for the level sets with a control variable, we constructed a Hamiltonian whose corresponding level set equation models optimal travel. Using this, we described a simple algorithm for calculating the optimal walking path between a starting and ending point which consists of numerically solving a Hamilton-Jacobi-Bellman (HJB) equation and then a system of ordinary differential equations. Further, we suggest a method for incorporating uncertainty into the location of the starting point: by modifying the algorithm slightly, we can compute several optimal paths while only solving one HJB equation. We then suggested numerical methods for simulating the HJB equation. We used Godunov's numerical Hamiltonian with second-order, essentially non-oscillatory finite difference approximations for spatial derivatives and second-order, total variation diminishing time integration. We also suggested modifications to the numerical methods which avoid common pitfalls which one may encounter. To test our algorithm, we simulated our model first using artificial elevation data and then using the actual elevation data in certain regions of Yosemite National Park. In both cases, results aligned very well with our physical intuition. Finally, we sampled several different starting locations and calculated optimal paths which travel down from the summits of El Capitan and Half Dome and noticed that in both cases, the paths can be naturally clustered into collections of paths which follow the same basic route. We performed k-means clustering to separate the paths into such collections. Such clustering could suggest simple yet effective patrol strategies for a law enforcement agency tasked with patrolling nationally protected areas.

Acknowledgements. The authors would like to thank the anonymous referees for several helpful comments and suggestions.

Author Yat Tin Chow is supported by ONR grant N000141712162 and NSF grant DMS-1720237. The remaining authors are supported by NSF grant DMS-1737770 and the National Geospatial-Intelligence Agency's Academic Research Program (Award No. HM0210-14-1-0003, Project Title: "Sparsity Models for Spatiotemporal Analysis and Modeling of Human Activity and Social Networks in a Geographic Context." Approved for public release, 18-648).

REFERENCES

[1] P. K. Agarwal and H. Wang, *Approximation algorithms for curvature-constrained shortest paths*, SIAM J. Comput., 30(6):1739–1772, 2001. 1

[2] K. Alton and I. M. Mitchell, *Optimal path planning under defferent norms in continuous state spaces*, in Proceedings IEEE Int. Conf. Robot. Autom., ICRA, 866–872, 2006. 1

[3] K. Alton and I. M. Mitchell, *Fast marching methods for stationary Hamilton-Jacobi equations with axis-aligned anisotropy*, SIAM J. Numer. Anal., 47(1):363–385, 2009. 3

[4] V. Balasubramanian, D. V. Kalashnikov, S. Mehrotra, and N. Venkatasubramanian, *Efficient and scalable multi-geography route planning*, in Proceedings of 13th International Conference on Extending Database Technology, 394–405, 2010. 1

[5] A. Bayen, I. M. Mitchell, M. Oishi, and C. J. Tomlin, *Aircraft autolander safety analysis through optimal control-based reach set computation*, J. Guid. Control Dyn., 30:68–77, 2007. 1

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

[7] L. E. Dubins, *On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents*, Amer. J. Math., 79(3):497–516, 1957. 1

[8] V. Estellers, D. Zosso, R. Lai, S. Osher, J. P. Thiran, and X. Bresson, *An efficient algorithm for level set method preserving distance function*, IEEE Trans. Image Process., 21(12):4722–4734, 2012. 4.3

[9] O. Hachour, *A genetic FPGA algorithm path planning of an autonomous mobile robot*, Inter. J. Sys. Appl., Engineering & Development, 2(4):178–190, 2008. 1

[10] C. Hirsch, D. Neuhuser, C. Gloaguen, and V. Schmidt, *Asymptotic properties of Euclidean shortest-path trees in random geometric graphs*, Statist. Probab. Lett., 107:122–130, 2015. 1

[11] I. J. Irmischer and K. C. Clarke, *Measuring and modeling the speed of human navigation*, Cartogr. Geogr. Inf. Sci., 45(2):177–186, 2018. 4.1

[12] G. Ishigami, K. Nagatani, and K. Yoshida, *Path planning and evaluation for planetary rovers based on dynamic mobility index*, in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 601–606, 2011. 1

[13] C. Y. Kao, S. Osher, and J. Qian, *Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations*, J. Comput. Phys., 196(1):367–391, 2004. 3

[14] R. Kimmel and J. A. Sethian, *Computing geodesic paths on manifolds*, Proc. Natl. Acad. Sci. USA, 95(15):8431–8435, 1998. 1

[15] G. Krishnaswamy and A. Stentz, *Resolution independent grid-based path planning*, Technical report, Carnegie-Mellon University Robotics Institute, 1995. 1

[16] B. Lee, J. Darbon, S. Osher, and M. Kang, *Revisiting the redistancing problem using the Hopf–Lax formula*, J. Comput. Phys., 330(1):268–281, 2017. 4.3

[17] C. Lee, J. Dolbow, and P. J. Mucha, *A narrow-band gradient-augmented level set method for multiphase incompressible flow*, J. Comput. Phys., 273:12–37, 2014. 2.2

[18] L. Lin and M. A. Goodrich, *UAV intelligent path planning for wilderness search and rescue*, IEEE/RSJ International Conference on Intelligent robots and systems (IROS), 709–714, 2009. 1

[19] J. Lygeros, *On reachability and minimum cost optimal control*, Automatica, 40(6):917–927, 2004. 1

[20] K. R. Memon, S. Memon, B. Memon, A. R. Memon, and S. M. Z. A. Shah, *Real time implementation of path planning algorithm with obstacle avoidance for autonomous vehicle*, in 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2048–2053, 2016. 1

[21] C. Min, *Local level set method in high dimension and codimension*, J. Comput. Phys., 200(1):368–382, 2004. 2.2

[22] J. S. Mitchell and M. Sharir, *New results on shortest paths in three dimensions*, in Proceedings of the Twentieth Annual Symposium on Computational Geometry, 124–133, ACM, 2004. 1

[23] S. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Appl. Math. Sci., Springer–Verlag, 153, 2003. 2.1

[24] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed: Algorithms based on Hamilton–Jacobi formulations*, J. Comput. Phys., 79:12–49, 1988. 2

[25] S. Osher and C.-W. Shu, *High order essentially non–oscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28(4):907–922, 1991. 3, 3

[26] C. H. Papadimitriou, *An algorithm for shortest-path motion in three dimensions*, Inform. Process. Lett., 20(5):259–263, 1985. 1

[27] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, *A PDE-based fast local level set method*, J. Comput. Phys., 155(2):410–438, 1999. 2.2, 3

[28] M. Popović, T. A. Vidal-Calleja, G. Hitz, I. Sa, R. Siegwart, and J. I. Nieto, *Multiresolution mapping and informative path planning for UAV-based terrain monitoring*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1382–1388, 2017. 1

[29] Quantum GIS Development Team, *Quantum GIS geographic information system*, 2017. 4.1

[30] C. Saranya, M. Unnikrishnan, S. A. Ali, D. Sheela, and D. V. Lalithambika, *Terrain based D* algorithm for path planning*, 4th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2016, IFAC-PapersOnLine, 49(1):178–182, 2016. 1

[31] W. Schwanghart and N. Kuhn, *TopoToolbox: a set of MATLAB functions for topographic analysis*, Environ. Model. Softw., 25(6):770–781, 2010. 4.1

[32] W. Schwanghart and D. Scherler, *TopoToolbox 2: MATLAB-based software for topographic analysis and modeling in Earth surface sciences*, Earth Surf. Dyn., 2(1):1–7, 2014. 4.1

[33] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton–Jacobi equations*, Proc. Natl. Acad. Sci. USA, 98(20):11069–11074, 2001. 1, 2.2

[34] J. A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms*, SIAM J. Numer. Anal., 41(1):325–363, 2003. 1, 2.2

[35] Z. Shen and A. Vladimirsky, *Piecewise-deterministic optimal path planning*, J. Optim. Theory Appl., submitted, arXiv:1512.08734. 1

[36] M. Sussman and E. Fatemi, *An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow*, SIAM J. Sci. Comput., 20(4):1165–1191, 1999. 4.3

[37] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 114(1):146–159, 1994. 4.3

[38] M. Sussman, E. Fatemi, P. Smereka, and S. Osher, *An improved level set method for incompressible two-phase flows*, Comput. & Fluids, 27(5):663–680, 1998. 4.3

[39] A. Szpiro and P. Dupuis, *Second order numerical methods for first order Hamilton-Jacobi equations*, SIAM J. Numer. Anal., 40(3):1136–1183, 2002. 3

[40] R. Takei, R. Tsai, H. Shen, and Y. Landa, *A practical path-planning algorithm for a simple car: a Hamilton-Jacobi approach*, Proc. Am. Control Conf., 6175–6180, 2010. 1

[41] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40(9):1528–1538, 1995. 1

[42] Z. Zhou, J. Ding, H. Huang, R. Takei, and C. Tomlin, *Efficient path planning algorithms in reach-avoid problems*, Automatica, 89:28–36, 2018. 1