Multi–Valued Distance Maps for Motion Planning on Surfaces with Moving Obstacles

Ron Kimmel*   Nahum Kiryati†   Alfred M. Bruckstein‡

Abstract

This paper presents a new algorithm for planning the time–optimal motion of a robot traveling with limited velocity from a given location to a given destination on a surface in the presence of moving obstacles. Additional constraints such as space variant terrain traversability and fuel economy can be accommodated. A *multi-valued* distance map is defined and applied in computing optimal trajectories. The multi-valued distance map incorporates constraints imposed by the moving obstacles, surface topography and terrain traversability. It is generated by an efficient numerical curve propagation technique.

1 Introduction

Searching for shortest paths on surfaces with stationary obstacles is a classical problem in Robotic Navigation. Solutions to the problems are based on computational geometry methods [23, 26, 29, 18], differential geometry and hybrid techniques [14, 2], as well as graph search based algorithms, see e.g. Latombe book [17] as a good pointer for many classical techniques. The problem of finding the time–optimal path of a robot in the presence of moving obstacles can be studied using the *configuration space* representation. Consider the simple case of a $2D$ obstacle moving on a plane. Given the obstacle's position in time, it is possible to consider the time axis as the $\hat{z}$ axis of a three dimensional configuration space also known as *Cspace*. The moving planar obstacle can then be represented as a three dimensional structure in the Cspace as shown in Figure 1.

Several attempts were made to reduce the complexity involved in the construction of, and the search in, the full Cspace for path planning among moving obstacles. These approaches are usually based on heuristic assumptions and provide 'a' path rather than an optimal one. One approach is to decompose the problem into *find path* and *move along path* problems as suggested in [8, 9]. The *move along path* involves a search for a parameterization along a given trajectory. This idea was further explored in [15, 16] and used to design the velocity for a robot that navigates on a curved surface, avoids moving obstacles, and is subject to

*Mail-stop 50A-2152, Lawrence Berkeley National Laboratory, and Dept. of Mathematics, University of California, Berkeley, CA 94720 Email: ron@math.lbl.gov FAX: (510)486-5401 Tel: (510)486-5453

†Electrical Engineering Department, Technion, Haifa 32000, Israel. Email: nk@tx.technion.ac.il

‡Computer Science Department, Technion, Haifa 32000, Israel. Email: freddy@cs.technion.ac.il

dynamic constraints. A different approach iteratively extrapolates the behavior of the moving obstacles that are assumed to be circles that travel along straight lines. This assumption simplifies the navigation task as demonstrated in [28].

This paper considers the problem of finding the time-optimal path between two points on a surface (not necessarily planar) in the presence of moving obstacles. We show that without making any heuristic assumption, the complexity of the problem can be reduced from a search through the $3D$ Cspace to searching for the path in a finite set of bounded $2D$ regions forming a multi–valued distance map. This map assigns each point a vector of numbers representing "first" times that the point may be reached by a robot moving with a limited velocity of V_{max}, though it may stop or decrease its velocity in order to avoid moving obstacles. Additional constraints such as terrain traversability and surface topography can be easily incorporated. The multi–valued distance map provides sufficient information for finding a time optimal path.

In the suggested method distance maps are obtained by curve evolution techniques that are based on the Huygens principle. A "wave–front" propagates in time and describes the farthest points the robot could reach by moving in all possible ways away from the source. The minimal path to the destination will be determined when the wave–front first meets the destination. Although it is obvious that this process will discover the best navigation course, it is far from trivial to realize how one could actually carry out this program in a computationally efficient and accurate way. A possible conceptual analogy is the way particles use their dual wave/particle nature in quantum mechanics to determine their path when moving from one point to another minimizing so-called action integrals. In the words of the great physicist R. Feynman [7] *"How does the particle find the right path?... Does it 'smell' the neighboring paths to find out whether or not they have the same action?"... "The miracle of it all is, of course, that it does just that... It isn't that a particle takes the path of least action but that it smells all the paths in the neighborhood and chooses the one that has the least action by a method analogous to the one by which light chose the shortest time."* In the suggested method the calculation of distance maps follows the same basic principle as a propagating light wave, efficiently 'smelling' the shortest among all possible paths.

2 The Multi–valued Distance Map

A distance map from a point $\mathbf{p}$ is a function that assigns to each point $\mathbf{q}$ in a given domain, a number equal to the length of the path of minimal length (minimal geodesic) between $\mathbf{p}$ and $\mathbf{q}$. We assume that the distance map is obtained by propagating an equal distance contour that assigns a distance to each point in the domain as it passes through it. The contour evolves in time at a constant velocity V_{max} from a small circle around the source point $\mathbf{p}$, and the distance corresponds to the time of propagation.

Due to the movement of the obstacles, the various ways to reach a point will be time dependent. In order to reach a destination point, the robot may be compelled to pass more than once through the same point. The optimal path will coincide with minimal geodesics on the surface only in regions not affected by the moving obstacles. Thus, a simple distance map is insufficient. The multi–valued distance map is a function that may assign more than a single value to each point in the domain, and can serve as a 'distance' function for time

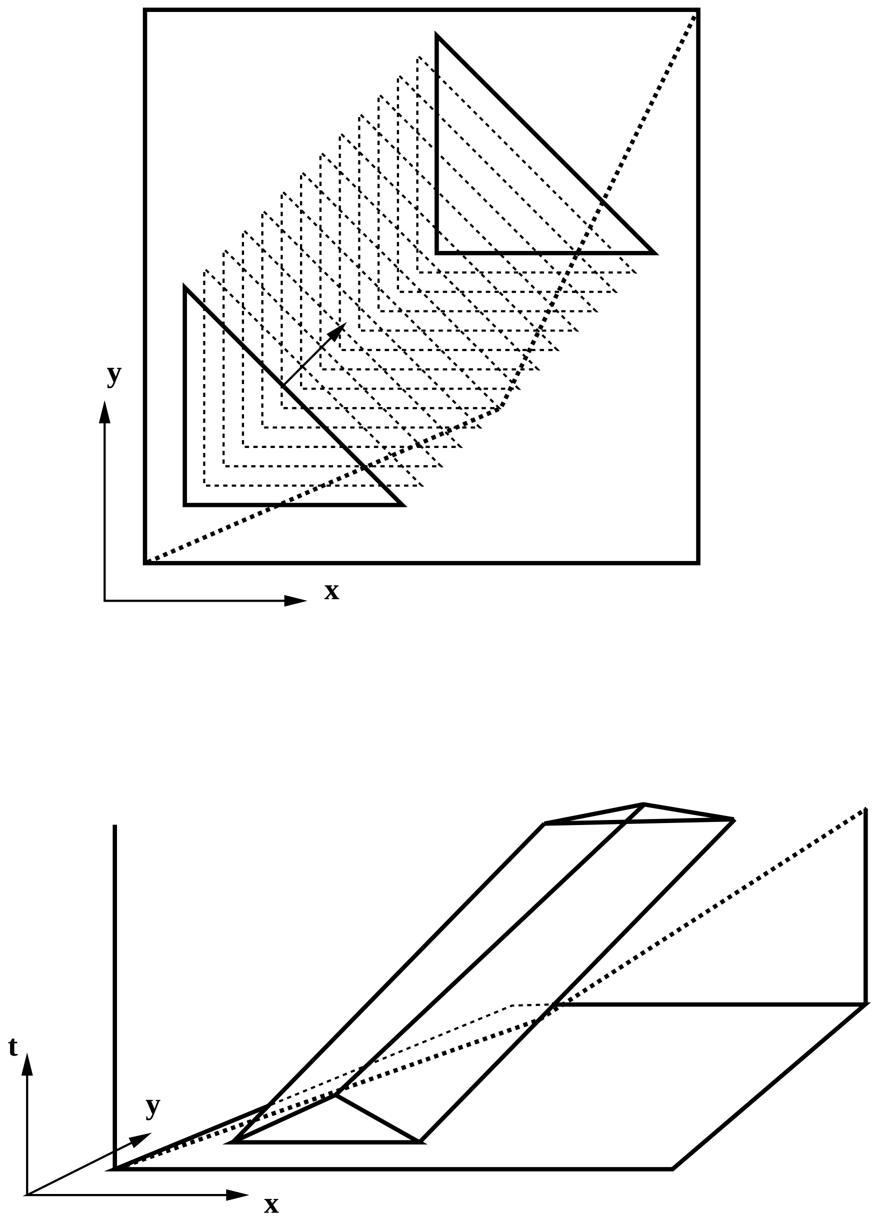

Figure 1: A triangular obstacle moving on the plane is represented by a prism in the Cspace. Searching for a path between two given points on the plane amounts to searching for a $3D$ path in the Cspace. This path should connect the two vertical lines in the Cspace that represent the starting and ending points, and should reach the end line with minimal t. The angle between the path and the t-axis is upper bounded by the maximal velocity of the robot V_{max}, see [2].

varying domains. The domain boundary is determined at any given time by the union of the boundaries of all the obstacles at that time and the boundary of the area of interest. The multi–valued distance map is thus formally defined as $\mathcal{M}_{\mathbf{p}}((x,y),m) : \Omega \times \{1,2,...,\mathcal{E}_{max}\} \rightarrow \mathbb{R}^+$, where $\Omega \subset \mathbb{R}^2$ is a finite domain over which the surface is defined as a function, and $\mathcal{E}_{max}$ is the maximal number of "layers" in the distance map. A point in the first layer, $\mathcal{M}_{\mathbf{p}}(\mathbf{q},1)$, represents the first time that the point $\mathbf{q}$ in the domain may be reached, $\mathcal{M}_{\mathbf{p}}(\mathbf{q},2)$ is a point in the second layer and represents the first time that point $\mathbf{q}$ may be reached after being covered by an obstacle after $\mathcal{M}_{\mathbf{p}}(\mathbf{q},1)$, and so on. See Figures 2 and 3.

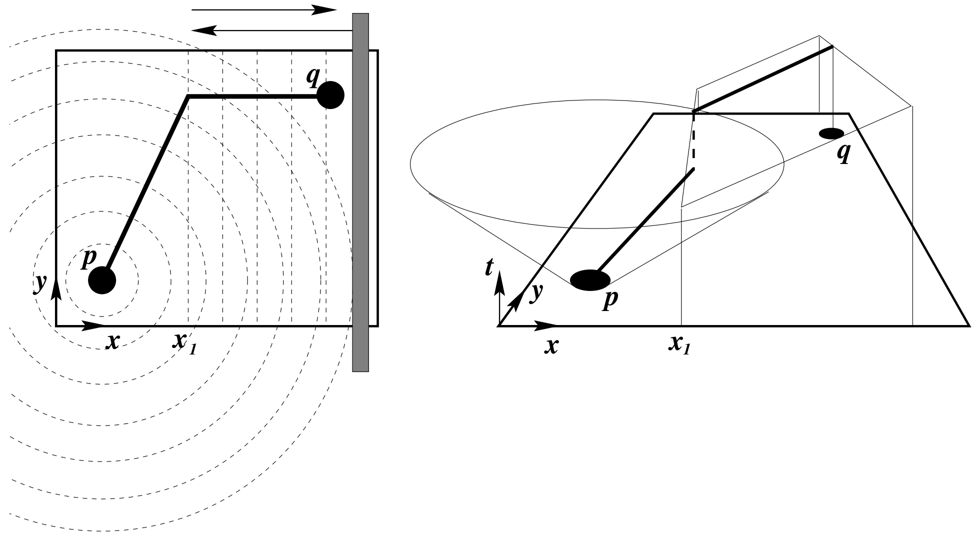

Figure 2: A robot is located at a point $\mathbf{p}$ on a plane at $t = 0$. It should travel from $\mathbf{p}$ to $\mathbf{q}$. The gray rectangle represents an obstacle that waits for a while, then translates to the left along the x direction from its position to the point $x = x_1$ and back to its original place. The right frame presents the resulting double valued map (the cone and the tilted plain) and the optimal path connecting $\mathbf{q}$ to $\mathbf{p}$. The left frame shows the projection of the optimal path and the equal distance contours that correspond to the two layers.

In terms of the prairie fire model [3], the equal distance contour is a propagating fire front in dry grassland. Each point in the grassland gets a time stamp at the time it is burnt. Now, consider a moving obstacle in burnt grassland. The obstacle causes each point it covers to instantaneously grow new un-burnt grass. The boundary of this new grass patch will be ignited as it is uncovered by the obstacle. Each point in the domain may thus get time stamped more than once, forming the multi–valued distance (time) map.

Consider a simple smooth surface $\mathcal{S} \subset \mathbb{R}^3$, where $\mathcal{S}$ is given by the function $z : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$, and Ω is a given finite domain. Each point $\mathbf{p}$ in the domain Ω gets a time stamp t as the projection of the wavefront passes through it. Denote such an event as $\mathcal{E}(\mathbf{p},t)$. Without moving obstacles, the event $\mathcal{E}(\mathbf{p},t)$ occurs only once for each point in the domain during the first (and only) time the wavefront's projection passes through it. With moving obstacles, consider a point $\mathbf{p} \in \Omega$ through which the wavefront had already passed at t_1, and therefore the event $\mathcal{E}(\mathbf{p},t_1)$ occurred. Assume that an obstacle steps over $\mathbf{p}$ and then moves away and reveals the point. The moving obstacle caused the fire to be re-ignited near $\mathbf{p}$ at

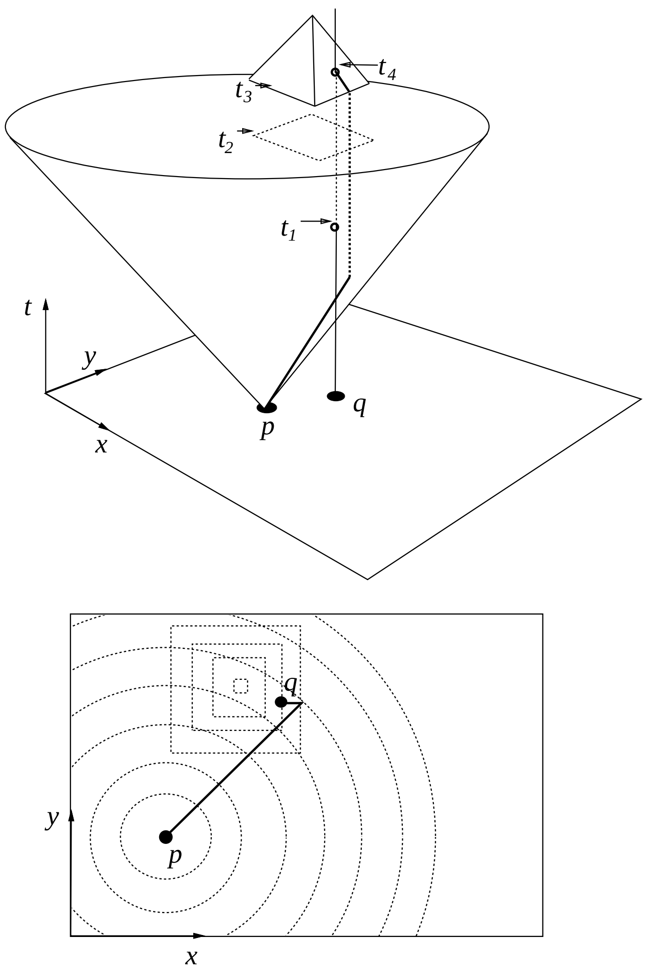

Figure 3: A robot is located at a point $\mathbf{p}$ on a plane at $t = 0$. It should travel from $\mathbf{p}$ to another point $\mathbf{q}$ and stay there. At t_2 an obstacle lands on the plane in a square that includes point $\mathbf{q}$ and disappears at t_3. From t_2 to t_3 the robot cannot stay inside the occupied square. The upper frame shows the resulting double valued distance map (the cone and the pyramid) and the optimal path connecting $\mathbf{q}$ (reached at t_4) to $\mathbf{p}$. The distance values at $\mathbf{q}$, are $\mathcal{M}_{\mathbf{p}}(\mathbf{q}, 1)$ and $\mathcal{M}_{\mathbf{p}}(\mathbf{q}, 2)$ are t_1 and t_4 respectively. The lower frame shows the projection of the optimal path and the equal distance contours that correspond to the two layers.

the boundary of the obstacle. The new fire front will reach $\mathbf{p}$ at t_2, causing a new event $\mathcal{E}(\mathbf{p}, t_2)$, etc. An example is shown in Figure 4.

Define $\mathcal{N}(\mathbf{p})$ as the number of events $\mathcal{E}(\mathbf{p}, t)$. Denote by $\mathcal{E}_{max}$ the maximal number of events that occur at a point in the domain (we assume that each point my be uncovered by obstacles only a finite number of times), that is

$$\mathcal{E}_{max} = \max_{\mathbf{p}\in\Omega}\{\mathcal{N}(\mathbf{p})\}.$$

The new "configuration space" over which it is sufficient to perform the search is $\Omega \times \{1, 2, ..., \mathcal{E}_{max}\}$. The complexity of the problem can thus be reduced from a search over $\Omega \times [0, T)$, which is a volume quantity, to a search over a finite number $\mathcal{E}_{max}$ of bounded two dimensional domains.

3 Algorithm Development

In this section the differential equation that describes the propagation of equal distance contours on a given surface is determined. Tracking the evolution of a $3D$ curve is a complicated task. However, it is possible to consider the evolution of the projection of this $3D$ curve onto the (x, y)-plane. This planar evolution must incorporate information about the surface within the propagation rule. Actually, the only information about the surface that is required for constructing the planar evolution rule is the gradient of the surface. The planar evolution rule is obtained by first projecting the $3D$ propagating curve onto the plane, and then, considering only the velocity component normal to the planar curve (the trace of the evolving curve depends only on the normal component of the velocity). The justification for eliminating the tangential component from the evolution rule is that it affects only the internal representation of the curve, *i.e.* the parameterization, while the trace (the geometric shape) of the evolving curve depends on the normal component of the velocity alone, [6, 22].

The procedure for finding the optimal path involves two steps: 1. The construction of the multi valued distance map. 2. Given the multi valued distance map, the optimal path is extracted by a back tracking procedure. Here, we first present the continuous partial differential equation (PDE) that describes the relation between level sets of the distance map, and the continuous back tracking ordinary differential equation. Then, Subsection 3.1 introduces a simplified discretization for the back tracking procedure. Next, Subsection 3.2 reviews the level set formulation for the distance map reconstruction: The level sets of the distance map are reconstructed via the Osher-Sethian level set formulation. It is simple to discretize the resulting PDE in a consistent way.

Let us first give the formulation of the problem of finding the evolution rule of the equal distance contours on a given surface. We begin with a definition of the equal distance contour. Given a source point $\mathbf{p}$ on a surface $\mathcal{S} \subset \mathbb{R}^3$ (that can be described by a function $\mathcal{S} = \{(x, y, z(x, y))|(x, y) \in \Omega \subset \mathbb{R}^2\}$), let the $3D$ equal distance contour at distance t from $\mathbf{p}$ be defined as

$$\alpha(t) = \{\mathbf{q} \in \mathcal{S} | d_{\mathcal{S}}(\mathbf{q}, \mathbf{p}) = t\},$$

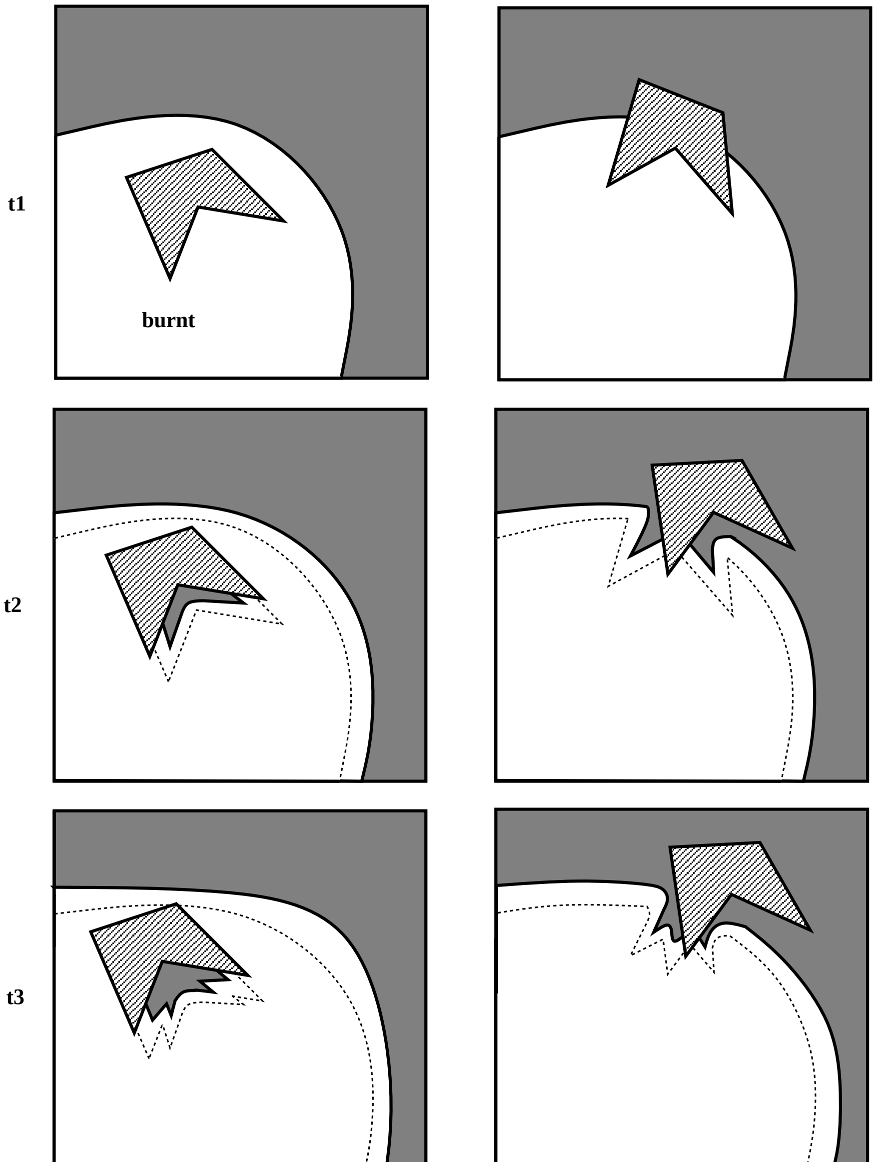

Figure 4: In column **a** an obstacle (indicated in gray) is moving in a “burnt” (white) area. The fire front and the obstacle's position are displayed at three successive times. Observe that the fire front at t_2 is the obstacle's boundary and an offset of the fire front and the obstacle's boundary at t_1. Column **b** displays an obstacle moving along the propagating fire front. The obstacle's position in time deforms the wavefront shape as it stops the front from propagating at t_1. At t_2 the wavefront is further deformed by the current position of the obstacle. The outline of the obstacle is impressed into the wavefront.

where $d_{\mathcal{S}}(\mathbf{q},\mathbf{p})$ is the length of the minimal geodesic between the point $\mathbf{q}$ and the point $\mathbf{p}$ on the surface $\mathcal{S}$. It can be proved [10, 11] that given a $3D$ parametric curve $\alpha(u)$ on $\mathcal{S}$, the equal distance contour propagation rule is

$$\frac{\partial \alpha(u,t)}{\partial t} = \vec{\mathbf{N}} \times \vec{\mathcal{T}} \quad \text{given} \quad \alpha(u,0) = \alpha(0),$$

where $\vec{\mathcal{T}}$ is the unit vector tangent to α, and $\vec{\mathbf{N}}$ is the surface unit normal.

Starting from a small circle, $\alpha(u,0)$, on the surface around the source point $\mathbf{p}$, it is possible to find the equal distance contour for any desired distance d, by using the evolution equation to calculate $\alpha(u,t)|_{t=d}$. We are interested in formulating the evolution rule of the projection of the $3D$ curve on the (x,y)–plane,

$$C(t) = \{(x,y)|(x,y,z(x,y)) \in \alpha(t)\} \equiv \pi \circ \alpha(t),$$

where π is the projection operation. Let us consider the projection of the $3D$ evolution on the (x,y)–plane. The knowledge of how this projected contour behaves allows us to construct a simple, accurate and stable numerical algorithm that can be used to produce the equal distance contours on the surface.

The planar normal component of the projected velocity of the evolving equal distance contour is

$$V_N = \langle \pi \circ (\vec{\mathbf{N}} \times \vec{\mathcal{T}}), \vec{\mathbf{n}} \rangle.$$

Writing the unit normal to the planar curve in terms of its components $\vec{\mathbf{n}} = (n_1, n_2)$, and the surface normal as $\vec{\mathbf{N}} = (-p,-q,1)/\sqrt{1+p^2+q^2}$, where $p \equiv \partial z(x,y)/\partial x$ and $q \equiv \partial z(x,y)/\partial y$, it is straightforward [10, 11] to show that

$$V_N = \sqrt{\frac{(1+q^2)n_1^2 + (1+p^2)n_2^2 - 2pqn_1n_2}{1+p^2+q^2}}.$$

Using this expression we construct a differential equation (in *Lagrangian* formulation) describing the projected equal distance contour evolution

$$\frac{\partial}{\partial t} C(t) = V_N \vec{\mathbf{n}},$$

given

$$C(0) = \{(x,y)|(x,y,z(x,y)) \in \alpha(u,0)\} \equiv \pi \circ \alpha(u,0).$$

The procedure that calculates the equal distance contours allows us to build a distance map (multi–valued in the presence of moving obstacles) from a given point. In our search for the time optimal path we make use of the fact that the minimal geodesics are perpendicular to the equal distance contours on the surface. This result is obtained by an expansion of Gauss Lemma [10, 11]. It is used to track the optimal trajectory by starting at the destination point that is reached by an equal distance contour, and proceeding backwards using the orthogonal property of the optimal path and the equal distance contours.

Let us recall the definition of the multi–valued distance map from $\mathbf{p}$,

$$\mathcal{M}_{\mathbf{p}}(x, y, m) = d_{\mathcal{S}}((x, y, z(x, y)), \mathbf{p}),$$

where the index m refers to the layer in the multi-valued distance map. Every time a point (x, y) is reached by the propagating contour, the corresponding m index is incremented and a time stamp is assigned to that point. Given the multi–valued distance map, the remaining task is to determine the optimal path.

Assume that there are no moving obstacles, i.e., that the distance map is single–valued. Let $\beta(t) : I \to \mathbb{R}^3$ denote the continuous $3D$ curve representing the optimal path. Starting at the destination point, the direction of backtracking on the surface is given by

$$\vec{\eta} = -\vec{\mathbf{N}} \times \vec{\mathcal{T}},$$

where $\vec{\mathbf{N}}$ is the surface normal and $\vec{\mathcal{T}}$ is the tangent unit vector to the equal distance contour. The following result can be computed by using arclength parameterization of curve α. Denote the first derivatives of the distance map as $r \equiv \partial\mathcal{M}_{\mathbf{p}}/\partial x$ and $l \equiv \partial\mathcal{M}_{\mathbf{p}}/\partial y$, and a normalization function

$$\psi(p, q, r, l) = \frac{1}{\sqrt{(p^2 + q^2 + 1)\,(l^2(1 + p^2) + r^2(1 + q^2) - 2pqrl)}}.$$

Using these terms

$$\vec{\eta} = \psi(p, q, r, l)\,[r(1 + q^2) - pql,\ l(1 + p^2) - pqr,\ pr + ql]\,,$$

and the backtracking rule, the optimal path is given by

$$\frac{\partial\beta}{\partial t} = -\vec{\eta},$$

where $\beta(0) = \mathbf{q}$ is the destination point from which backtracking towards the source point begins.

In the presence of moving obstacles the distance map is multi–valued. The value index m may change along the optimal path. Therefore, one generally has to consider all distance values at a candidate point.

3.1 Example for a simple backtracking procedure

Let us present a simple discrete approximation for the back propagation procedure. Suppose that the surface and the distance map are given as samples on a rectangular grid. Denote $z_{ij} \equiv z(i\Delta x, j\Delta y)$, and similarly for the distance map $\mathcal{M}_{ij}^m \equiv \mathcal{M}_{\mathbf{p}}(i\Delta x, j\Delta y, m)$. A possible definition of the neighborhood of a point (i, j) is $\mathcal{N}_{ij} = \{(i, j+1), (i-1, j+1), (i-1, j), (i-1, j-1), (i, j-1), (i+1, j-1), (i+1, j), (i+1, j+1)\}$. Also assume without loss of generality that $V_{max} = 1$.

We shall not make an explicit usage of the analytic results describing the back tracking as presented in the previous section. Alternatively, we open an imaginary sphere around the last detected point in the back tracked path, and locate the neighbor from which the current point was reached. The optimal path is thus created by a repeated neighbor selection process, starting at the destination point, according to the following rule: Given the point (i,j) with a distance value $\mathcal{M}_{ij}^m$, its neighbor (k,l) (with distance value $\mathcal{M}_{kl}^n$), is selected by

$$\min_{n,(k,l)\in\mathcal{N}_{ij}} \left|\mathcal{V}_{ij,kl}^{m,n} - 1\right|, \tag{1}$$

subject to the two following constraints

$$\mathcal{M}_{ij}^m > \mathcal{M}_{kl}^n \quad \text{and} \quad \mathcal{V}_{ij,kl}^{m,n} < 1+\delta,$$

where

$$\mathcal{V}_{ij,kl}^{m,n} \equiv \frac{\sqrt{(z_{ij}-z_{kl})^2 + (i-k)^2\Delta x^2 + (j-l)^2\Delta y^2}}{\mathcal{M}_{ij}^m - \mathcal{M}_{kl}^n}.$$

The selection is carried out between all neighbors, and for each neighbor through all distance values (all different layers n). δ is a small number less than 1 that compensates for the metrication error due to the grid. It describes the wavefront error that might occur between neighboring grid points. The first constraint ensures that the next candidate point, generated by the back tracking procedure, is in a decreasing order in time, while the second constraint guarantees that the next candidate will not be along the same wavefront (traveling along the same wavefront is possible only with an infinite velocity $\mathcal{V}$).$\mathcal{V}_{ij,kl}^{m,n}$ represents the velocity speed the robot would have had to travel, in order to get from the position in the C-space defined by $t = \mathcal{M}_{kl}^n$, $(x,y) = (k\Delta x, l\Delta y)$ to the current position $t = \mathcal{M}_{ij}^m$, $(x,y) = (i\Delta x, j\Delta y)$. The fact that we limit the velocity to 1 enables us to exclude traveling along the wavefront. While the minimization (1) is used to select the steepest decent, and thereby the selection of the optimal path.

The robot travels with a limited velocity, and by our construction it practically moves with its maximal velocity while it can. The wavefront describes the front of possibilities moving at the maximal possible velocity. The wavefront itself is the set of all possible locations for the robot at a given time. Moving along a given wavefront means that the robot can move in zero time from one point to another, which is an obvious violation of the limited velocity assumption. In principle, we could check for the computed velocity to be less than 1, which was our general assumption. However, since we restrict the reconstructed path to pass through grid points on a rectangular grid, we should allow some flexibility in the back tracking procedure. The parameter δ may be considered as a measure for this flexibility, see Figure 5. The discrete back tracking procedure presented here is simple but not consistent. Consistent sub-grid methods for back tracking that are based on second order ODE integrators are possible (see e.g. [13]).

An interesting observation is the following: Consider an obstacle moving at a velocity $\tilde{v} < 1$ and forcing the robot to follow it. Using the above approximation, *i.e.* preferring a unit velocity, may cause the robot to travel back and forth (juggle) along the boundary of the moving obstacle while accompanying it as it moves. Although this is not the natural way to follow an obstacle, the result is still a time optimal path.

An important advantage of the suggested approach is that, unlike potential based methods, it is inherently immune against being trapped in local minima. This follows from the

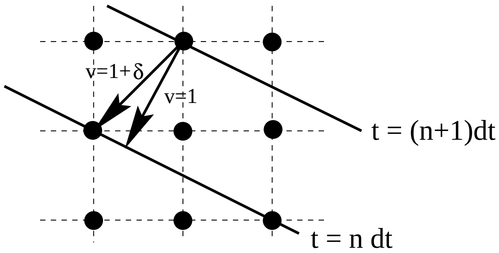

Figure 5: δ corresponds to the error caused by forcing the path to pass through the grid points. The bold lines represent two level sets of the distance map projected to the coordinate plane. In our example of a flat domain $\delta = 0.08\Delta x$.

fundamental property of distance maps, namely that the distance between two points corresponds to the globally shortest paths between them. This is implied by the definition of distance, see e.g. [30]. Similarly, singular points that in our case can only be maximum or saddle points, i.e. points at which there is more than one possible way to continue, do not cause any difficulty since they correspond to multiple paths of equal lengths that are all globally shortest.

Angular errors that are accumulated by the back propagation procedure can be corrected using geodesic curvature flow [12, 4] to shorten the generated curve into a geodesic (between the moving obstacles). We remark that the above backtracking approximation is only one simple example, and the one used in our examples. For constraints imposed by different considerations like terrain traversability see Appendix.

3.2 The Eulerian Formulation

The construction of the multi–valued distance map is a curve evolution process. When implementing curve evolution equations of Lagrangian formulation on a digital computer, one should take care of some numerical problems and topological changes in the evolving contour. Moreover, the moving obstacles generate new boundary conditions as they move and the question is how to efficiently incorporate these in the evolution procedure. Sethian and Osher [21, 24] proposed an algorithm for curve and surface evolution that elegantly solves these problems. As a first step in constructing the algorithm, the curve is embedded in a higher dimensional function. Then, evolution equations for the implicit representation of the curve are solved using numerical techniques derived from hyperbolic conservation laws.

Let the planar curve $C(t)$ be represented by the zero level set of a smooth Lipschitz continuous function $\phi : \mathbb{R}^2 \times [0, T) \to \mathbb{R}$, so that ϕ is negative in the interior and positive in the exterior of the zero level set $\phi = 0$. Consider the zero level set defined by

$$\{C(t) \in \mathbb{R}^2 : \phi(C, t) = 0\}.$$

We have to find the evolution rule of ϕ, so that the evolving curve $C(t)$ can be represented by the evolving zero level set $\phi(C, t) = 0$. Using the chain rule on $\phi(C(t), t) = 0$ we get

$$\nabla\phi(C, t) \cdot C_t + \phi_t(C, t) = 0.$$

Note that for any level set the planar normal can be written as $\vec{\mathbf{n}} = \nabla\phi/\|\nabla\phi\|$. Using this relation in conjunction with the Lagrangian evolution equation we obtain

$$\phi_t = -V_N\|\nabla\phi\|,$$

where the curve $C(t)$ is obtained as the zero level set of ϕ. This procedure is known as the Eulerian formulation [24]. This formulation of planar curve evolution processes frees us from the need to take care of the possible topological changes in the propagating curve, see Figure 6. It also frees us from the need to construct special procedures that handle the boundaries conditions imposed by the moving obstacles. In fact, the implicit representation of the propagating wavefront allows us to include the effect of the moving obstacles in a natural way. It is straightforward to construct the Eulerian formulation and the appropriate numerical implementation at any dimension. This is an important property for path planning problems that usually involve several degrees of freedom.

The numerical implementation [10, 11] of the Eulerian formulation is based on *monotone* and *conservative* numerical algorithms, derived from hyperbolic conservation laws and from the Hamilton–Jacobi type of equations [21, 20]. Using the normal component of the velocity in the Eulerian formulation equation yields the following evolution rule:

$$\phi_t = \sqrt{\frac{(1+q^2)\phi_x^2 + (1+p^2)\phi_y^2 - 2pq\phi_x\phi_y}{1+p^2+q^2}}. \tag{2}$$

This equation describes the propagation rule for the surface ϕ. It was implemented on a rectangular grid $(i\Delta x, j\Delta y, n\Delta t)$ over the domain, using forward finite derivative approximation in time, and slope limiters in approximating the spatial derivatives [10, 11]. Every time step the moving obstacles force the $\phi_{ij}^n \equiv \phi(i\Delta x, j\Delta y; n\Delta t)$ function to become negative in areas covered by them. The whole procedure of finding an optimal path may be summarized as follows:

- Evolve the discretization of Eq. 2 in time $(n\Delta t)$, until the destination point is reached. In each iteration check:
  If $\phi_{ij}^{n+1}\phi_{ij}^n < 0$ then update the multi valued distance map $\mathcal{M}_{ij}^m$ at the point $(i\Delta x, j\Delta y)$, and increment by 1 the number of values counter m at that point (denoted by m_{ij}),

$$\begin{aligned} \mathcal{M}_{ij}^m &= \left(n + \frac{|\phi_{ij}^n|}{|\phi_{ij}^n| + |\phi_{ij}^{n+1}|}\right)\Delta t; \\ m_{ij} &= m_{ij} + 1. \end{aligned} \tag{3}$$

- Given the multi valued distance map $\mathcal{M}_{ij}^m$, back track the optimal path.

It is important to note that while constructing the numerical scheme for the above evolution equation, one must take care of satisfying the consistency condition. It guarantees that while the time and distance steps in the numerical scheme approach zero, the scheme converges to the continuous case. As discussed in [14, 19], this may be difficult in alternative approaches.

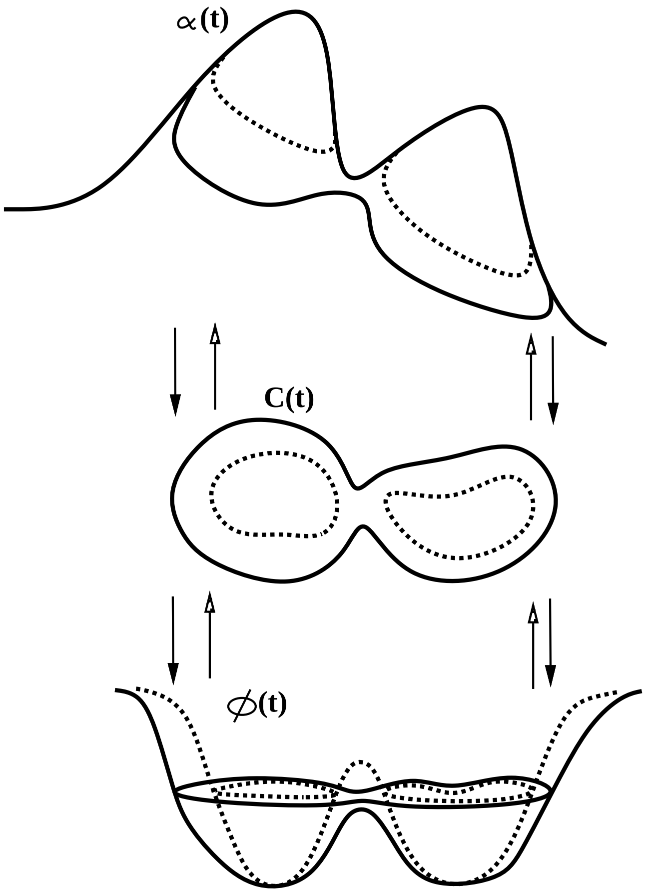

Figure 6: Propagating the $3D$ contour $\alpha(t)$ on the surface is done by first projecting the contour onto the plane to obtain $C(t)$, and then, as suggested by the Eulerian formulation, to propagate its implicit representation $\phi(t)$. This formulation avoids difficulties due to topological changes: observe how a single connected contour propagates and splits gracefully into two separate contours.

3.3 Computational Considerations

The order of the computational complexity of a serial generation of the multi–valued distance map is $O(\frac{T}{\Delta t}n)$, where T is the time duration for the optimal path, Δt is the time step in the numerical scheme and n is the number of grid points (the size of the planar domain Ω). Observe that T corresponds to the first time the propagating contour reaches the destination point. The upper bound of the memory complexity is $O(\mathcal{E}_{max} \cdot n)$, which is the finite, usually small, number of layers. The accuracy in the generation of the multi–valued distance map depends on the numerical scheme. For first order schemes the local truncation error at each iteration is of order $O(\Delta t)$. It can be reduced by using higher order schemes [20, 21].

Under the assumption that the back propagation complexity is smaller than the map generation (which is a reasonable assumption for non cluttered environments) the complexity is as indicated above $O(\frac{T}{\Delta t}n)$. It can be reduced by using the narrow band technique developed by Adalsteinsson and Sethian [1]. For flat weighted domains it can be reduced to $O(\mathcal{E}_{max} n \log n)$ with Sethian's Fast Marching Method, see concluding remarks. The back propagation, for which the complexity is $O(\frac{L}{\Delta x} \log \mathcal{E}_{max})$, that involves a search over the different layers, can become dominant for cluttered environments. Here L is the length of the optimal path, and Δx is the spatial discretization interval.

4 Examples

The following examples demonstrate the operation of the suggested algorithm. A 64×64 grid was used in the numerical approximation of the multi–valued distance map. In all the examples, a robot should travel from a starting point to a destination in the presence of time varying obstacles. The projection of the starting point is $(16, 16)$ and that of the destination point is $(48, 48)$. For simplicity of presentation, the obstacles are constructed on the xy coordinate plane and back projected onto the surface: Imagine an obstacle moving on the surface while deforming its shape so that its 'shadow' on the coordinate plane is always a rectangle.

In the first example, Figure 7, an obstacle lands on a plane at $t = 44$ and disappears at $t = 73$. The optimal motion plan includes traveling at an area just before it is covered by the obstacle, then waiting at the boundary of the covered area at the point closest to the destination. When the obstacle moves away the shortest path to the destination point is taken. This example is similar to the one sketched in Figure 3.

The upper left frame shows the projection of the equal distance contours onto the plane. A 3-D visualization of these contours appears in the lower left frame. The multi-valued distance map $\mathcal{M}_{ij}^m$ is shown in the (x, y, t)-space at the lower right frame. The optimal path appears as a black curve in all the frames, and is highlighted in the upper right frame. The starting point is black and the destination point is white.

In this example we have used an obstacle that appears and vanishes in order to demonstrate the ideas we put forward in the introduction of this paper. It is by no means limits the cases the proposed method can handle. As we will see in the following examples, moving obstacles are considered, and an optimal path is extracted. We also note that the paths being extracted are indeed time optimal paths, however, they are not unique (one may walk along the sidewalk waiting for the stop light to switch to green, and still get to the other side of the road in a time optimal path).

In the examples shown in Figures 8 to 10, an 'egg-box' like surface, shown in the upper left frames, is used. The organization of the other three frames is similar to that of Figure 7. In the upper right frames the projection of the movement of an obstacle's boundary is shown in white. The gray levels represent the elevation of the surface.

In Figure 8, motion is constrained by a moving rectangular obstacle. The robot follows a local geodesic, then gets away from the approaching obstacle and finally reaches the destination.

In Figure 9, three moving rectangular obstacles cause the path to avoid the left obstacle, track the middle obstacle for a while, then follow the right obstacle towards the destination point, using geodesics when possible.

In Figure 10, several "layers" are formed in the multi-valued distance map due to a vertical line obstacle that moves from right to left and then left to right, twice. The optimal path (from the destination to the source) starts at the upper "layer" and then jumps to a lower layer. The robot runs away from the obstacle and waits at a point that connects to the destination with the shortest possible geodesic on the surface (the projection of the line obstacle is moving at a speed of $v = 1$).

5 Concluding Remarks

We dealt with the problem of finding optimal paths on surfaces while avoiding obstacles that are moving on them. To the best of our knowledge, there is no numerically consistent solution (that converges, or in other words 'resolution complete' in a systematic way) for this case, with a lower complexity. We use the property that the only needed knowledge for the construction of an optimal path in a temporal-spatial configuration space is the first time a point can be visited (or revisited after being covered by an obstacle). This way we compressed the space-time Cspace domain into what we named multi-valued distance map. Indeed we lose some important properties of the Cspace and limit our class of optimal solutions. Note that in many cases there is more than one optimal solution, and actually the class of these solutions may even be continuous, i.e. a net of connected manifolds rather than curves.

Since this manuscript was first submitted there were some new developments in the field of numerical approximations to wave propagation, especially for the Eikonal case that is relevant to our discussion. Coupled with the proposed method it offers a complexity equivalent to that of Dijkstra's algorithm [5] plus the important convergence property. In other words a systematically 'resolution complete' algorithm for the stationary search problem with a complexity of $O(n \log n)$. It is based on recent works by Sethian [25], Tsitisiklis [27], and was recently used for efficiently solving navigation problems in [13]. These algorithms apply to flat domains, as well as flat weighted domains; domains with a different cost assigned to each point, referred to as 'traversability' in the appendix. Still, for non flat domains, e.g. surfaces, the level set implicit propagation, with Adalsteinsson-Sethian [1] 'narrow band' implementation for tracking the wavefront, is the most efficient numerically consistent implementation on a rectangular grid known to the authors.

Acknowledgments

We thank referees for all their valuable comments. RK's work is supported in part by the Applied Mathematics Subprogram of the Office of Energy Research under DE-AC03-76SFOOO98, and ONR grant under NOOO14-96-1-0381

Appendix: Terrain Traversability

We have shown an analytic formulation of an equal distance contour evolution on a surface. The contour describes possible positions of a robot's traveling at a constant velocity. Now consider the constraints imposed by terrain traversability. A simple model may be given according to which at each point on the surface the robot's velocity is limited by a space variant traversability value. Let the traversability map be given by $F : \Omega \subset \mathbb{R}^2 \to [0,1]$, where each point on the surface $(x, y, z(x, y)) \in \mathcal{S}$ is characterized by a traversability factor of $F(x, y)$. At each point the robot velocity is limited by $V_{max} \cdot F(x, y)$ (it is assumed without loss of generality that $V_{max} = 1$). This constraint can be incorporated in the equal distance contour evolution equation by simply multiplying the propagation velocity. Using these arguments the planar evolution rule can now be written as

$$\frac{\partial C}{\partial t} = F(x, y) V_N(p, q, \vec{\mathbf{n}})\vec{\mathbf{n}}.$$

The robot's loss of weight and increase of maximal speed as it travels and uses its fuel can also be considered. The planar projection becomes

$$\frac{\partial C}{\partial t} = F(x, y) V_N(t, p, q, \vec{\mathbf{n}})\vec{\mathbf{n}}.$$

Time dependent terrain traversability $F(x, y, t)$ can be handled by generalizing the planar evolution rule to

$$\frac{\partial C}{\partial t} = F(x, y, t) V_N(t, p, q, \vec{\mathbf{n}})\vec{\mathbf{n}}.$$

Observe that the moving obstacles may not be considered as part of the changing traversability function. This is due to the fact that the robot can wait at a point of zero traversability for the traversability to increase at that area, but it can not be present at a point that is covered by an obstacle.

References

[1] D Adalsteinsson and J A Sethian. A fast level set method for propagating interfaces. *J. of Comp. Phys.*, 118:269–277, 1995.

[2] E Adin and A M Bruckstein. Navigation in a dynamic environment. Center for Intelligent Systems Report CIS #9101, Technion–Israel Institute of Technology, Israel, January 1991.

[3] H Blum. Biological shape and visual science (part I). *J. theor. Biol.*, 38:205–287, 1973.

[4] D L Chopp. Computing minimal surfaces via level set curvature flow. *J. of Computational Physics*, 106(1):77–91, May 1993.

[5] E W Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematic*, 1:269–271, 1959.

[6] C L Epstein and M Gage. The curve shortening flow. In A Chorin and A Majda, editors, *Wave Motion: Theory, Modeling, and Computation*. Springer-Verlag, New York, 1987.

[7] R P Feynman, R B Leighton, and M Sands. *The Feynman Lectures on Physics*. Addison Westly, Massachusetts, 1964.

[8] K Kant and S Zucker. Towards efficient trajectory planning: The path–velocity decomposition. *The International Journal of Robotics Research*, 5(3):72–89, 1986.

[9] K Kant and S Zucker. Planning collision–free trajectories in time–varying environments: A two–level hierarchy. In *Proceedings International Conference on Robotics and Automation*, pages 1644–1649, 1988.

[10] R Kimmel, A Amir, and A M Bruckstein. Finding shortest paths on surfaces. In P. Laurent, A. Le Méhauté, and L.L. Schumaker, editors, *Curves and Surfaces in Geometric Design*, pages 259–268. A K Peters, Ltd., Wellesley, Massachusetts, 1994.

[11] R Kimmel, A Amir, and A M Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Trans. on PAMI*, 17(6):635–640, June 1995.

[12] R Kimmel and G Sapiro. Shortening three dimensional curves via two dimensional flows. *International Journal: Computers & Mathematics with Applications*, 29(3):49–62, 1995.

[13] R Kimmel and J A Sethian. Applications of fast marching methods to robotic navigation and construction of optimal paths. *submitted*, 1996.

[14] N Kiryati and G Székely. Estimating shortest paths and minimal distances on digitized three dimensional surfaces. *Pattern Recognition*, 26(11):1623–1637, 1993.

[15] K J Kyriakopoulos and G N Saridis. An integrated collision prediction and avoidance scheme for mobile robots in non-stationary environments. *Automatica*, 29(2):309–322, 1993.

[16] K J Kyriakopoulos and G N Saridis. Optimal and suboptimal motion planning for collision avoidance of mobile robots in non-stationary environments. *J. of Intelligent and Robotic Systems: Theory and Applications*, 11(3):223–267, 1995.

[17] J C Latombe. *Robot motion planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[18] J S B Mitchell, D M Mount, and C H Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, 1987.

[19] J S B Mitchell, D Payton, and D Keirsey. Planning and reasoning for autonomous vehicle control. *International Journal of Intelligent Systems*, 2:129–198, 1987.

[20] S Osher and C W Shu. High–order essentially nonoscillatory schemes for Hamilton–Jacobi equations. *SIAM J. Numer. Analy.*, 28(4):907–922, August 1991.

[21] S J Osher and J A Sethian. Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. of Comp. Phys.*, 79:12–49, 1988.

[22] G Sapiro and A Tannenbaum. Affine invariant scale–space. *International Journal of Computer Vision*, 11(1):25–44, 1993.

[23] J T Schwartz, M Sharir, and J Hopcroft, eds. *Planning, Geometry, and Complexity of Robot Motion.* Ablex Series in Artificial Intelligence. Ablex, Norwood, NJ, 1987.

[24] J A Sethian. A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed. *J. of Diff. Geom.*, 33:131–161, 1990.

[25] J A Sethian. A marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93(4), 1996.

[26] M Sharir and A Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 1(15):193–215, 1986.

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

[28] T Tsubouchi and S Arimoto. Behavior of a mobile robot navigated by an "iterated forecast and planning" scheme in the presence of multiple moving obstacles. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 2470–2475, 1994.

[29] E Wolfson and E L Schwartz. Computing minimal distances on polyhedral surfaces. *IEEE Trans. on PAMI*, 11(9):1001–1005, 1989.

[30] A Zelinsky. Using path transforms to guide the search for findpath in 2D. *The Int. J. of Robotic Research*, 13(4):315–315, 1994.

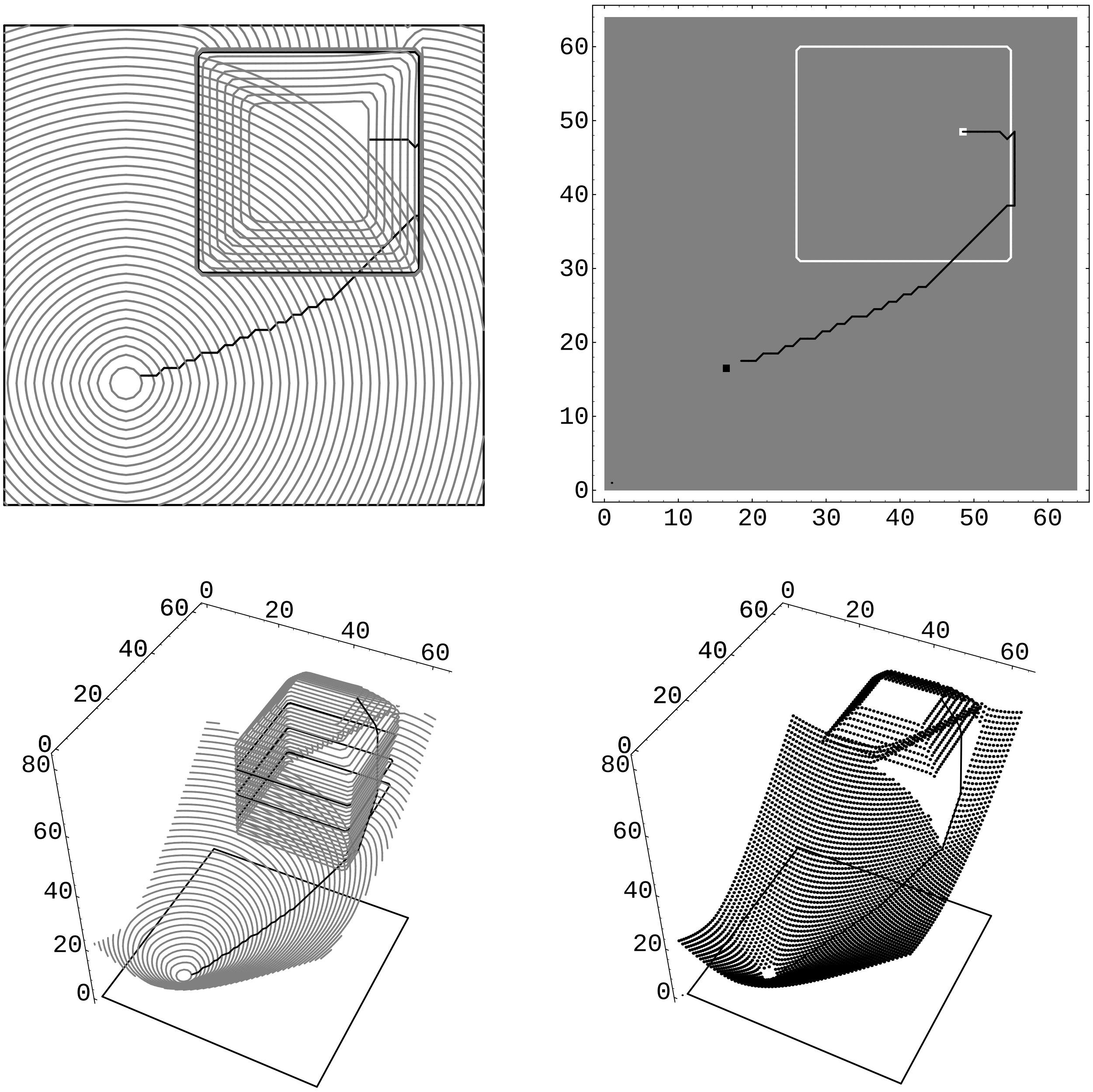

Figure 7: Time optimal path on a plane in the presence of a square obstacle that landed at $t = 44$ and disappeared at $t = 73$. **Upper left:** The gray curves are an overhead view of the equal distance contours. The black curves are the time optimal path and the obstacle boundary. **Upper right:** The time optimal path is the black curve. The white square is the boundary of the obstacle. The black point is the source and the white point is the destination. **Lower left:** The equal distance contours in the (x, y, t) Cspace are shown as gray curves. The optimal path and the boundary of the obstacle are in black. **Lower right:** The $\mathcal{M}_{ij}^m$ multi-valued distance map, in the (x, y, t) Cspace, where a black point corresponds to $(i, j, \mathcal{M}_{ij}^m)$. The black curve is the optimal path.

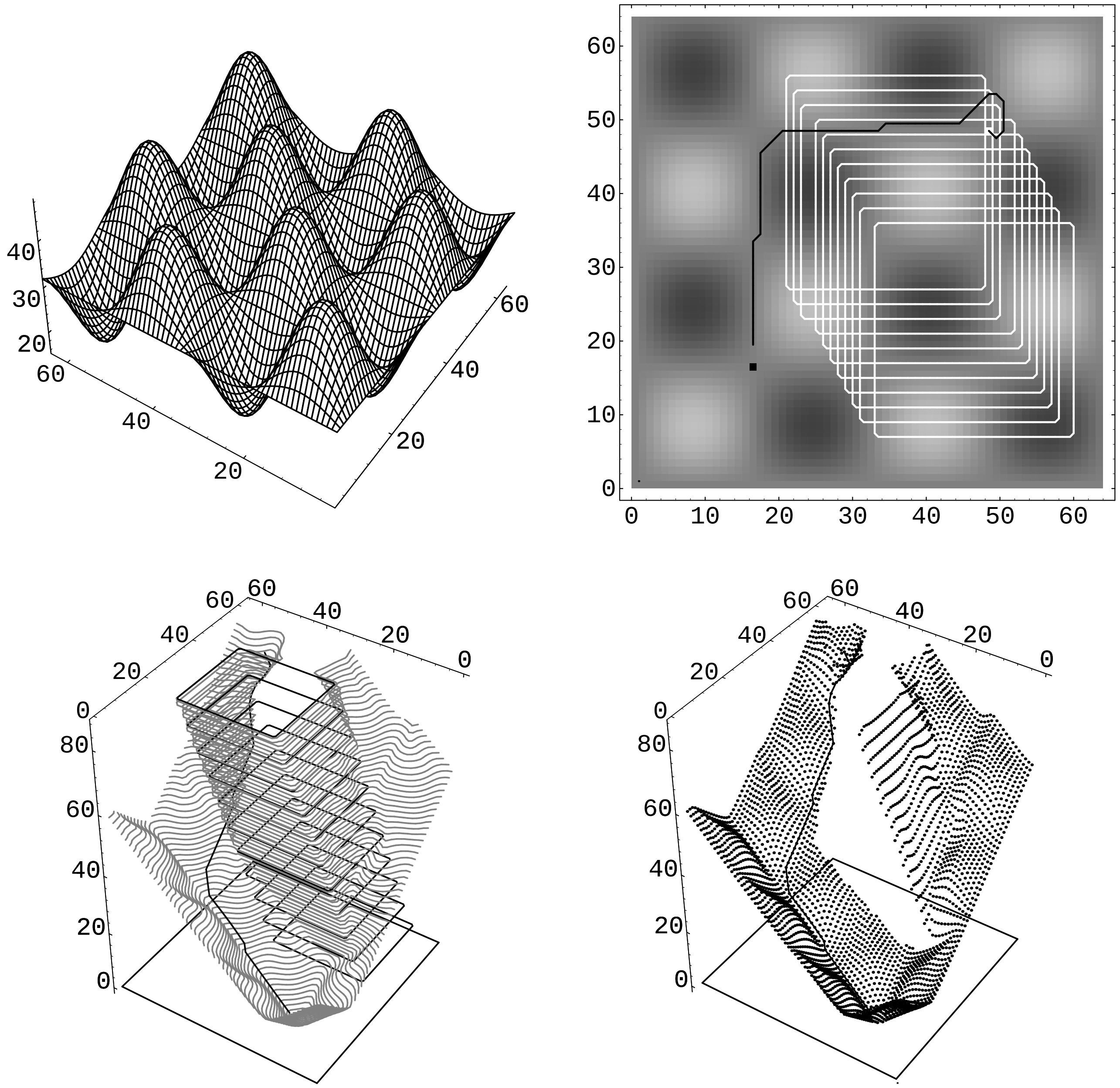

Figure 8: Time optimal path in presence of a moving square obstacle on an 'egg-box' surface. **Upper left:** The surface. **Upper right:** The time optimal path is the black curve. The white square is the boundary of the obstacle. The black point is the source and the white point is the destination. **Lower left:** The equal distance contours in the (x, y, t) Cspace are shown as gray curves. The optimal path and the boundary of the obstacle are in black. **Lower right:** The $\mathcal{M}_{ij}^m$ multi-valued distance map, in the (x, y, t) Cspace, where a black point corresponds to $(i, j, \mathcal{M}_{ij}^m)$. The black curve is the optimal path.

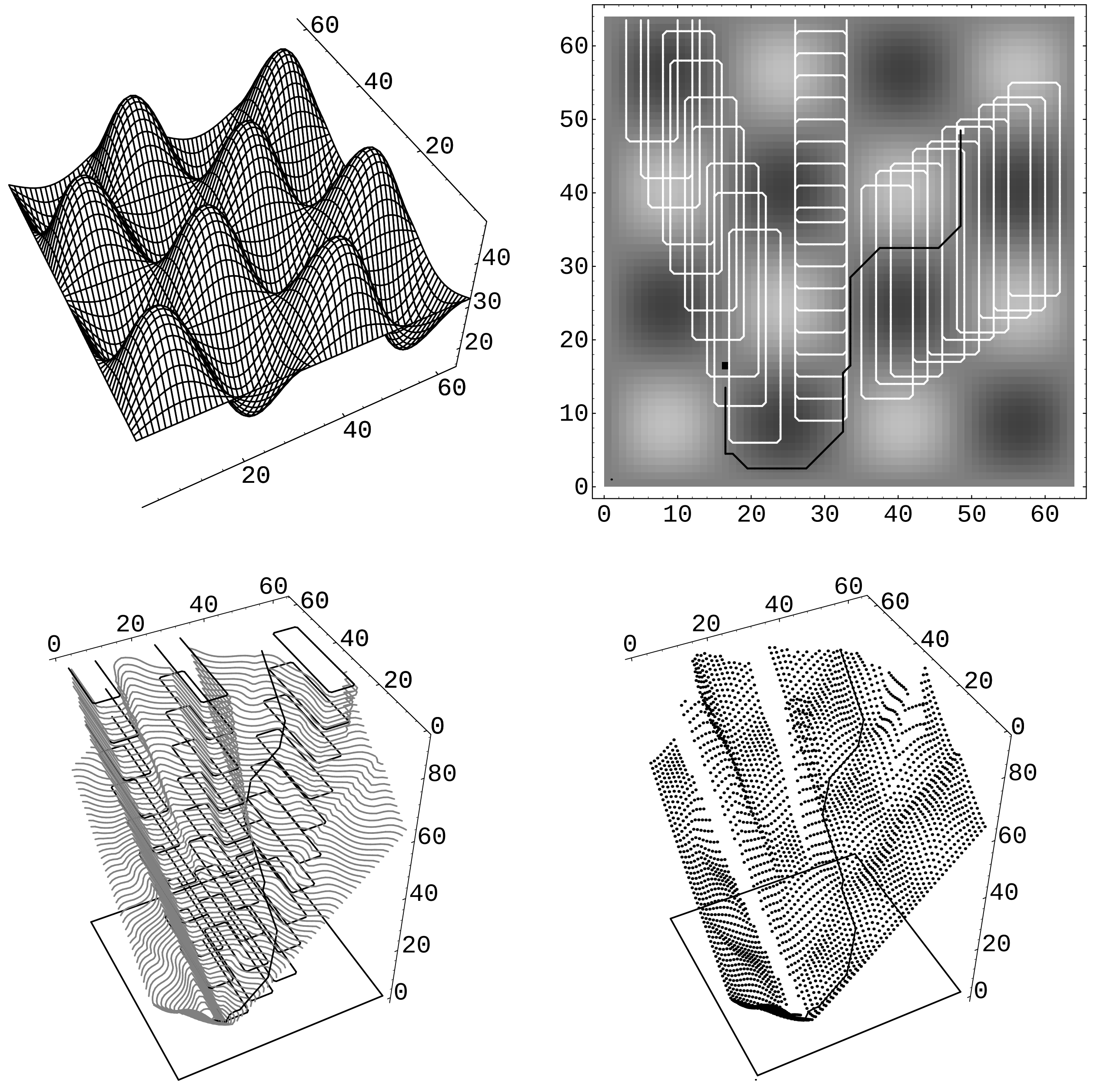

Figure 9: Time optimal path in presence of three moving obstacles on an 'egg-box' surface. **Upper left:** The surface. **Upper right:** The time optimal path is the black curve. The white square is the boundary of the obstacle. The black point is the source and the white point is the destination. **Lower left:** The equal distance contours in the (x, y, t) Cspace are shown as gray curves. The optimal path and the boundary of the obstacle are in black. **Lower right:** The $\mathcal{M}_{ij}^m$ multi-valued distance map, in the (x, y, t) Cspace, where a black point corresponds to $(i, j, \mathcal{M}_{ij}^m)$. The black curve is the optimal path.

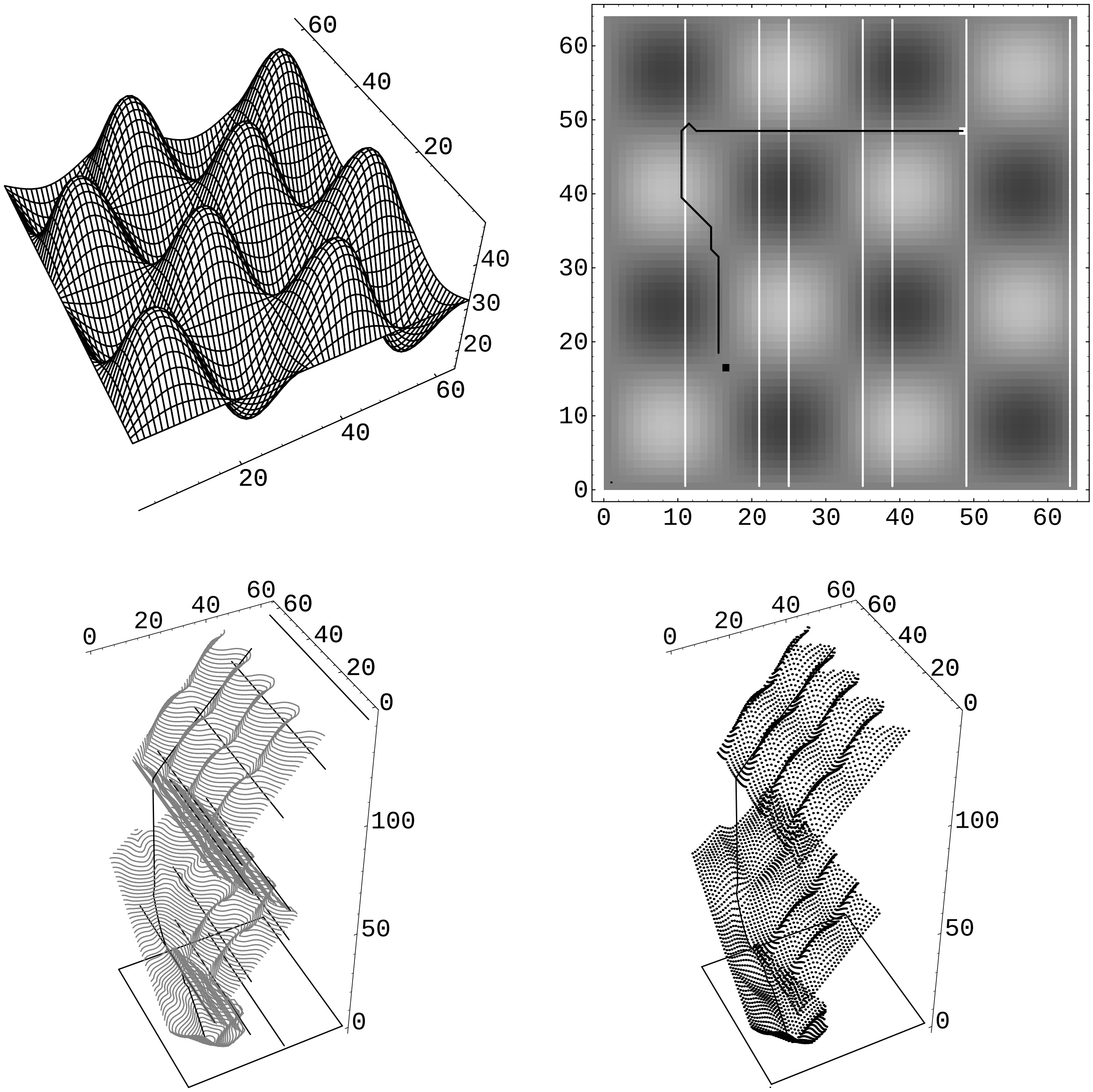

Figure 10: A single line obstacle is moving from $i = 40$ to $i = 10$, then backwards from $i = 10$ to $i = 50$, again to $i = 10$, and at last to the image frame boundary $i = 64$. The motion of the obstacle obviously creates multiple values in the distance map. **Upper left:** The 'egg-box' surface. **Upper right:** The time optimal path is the black curve. The white square is the boundary of the obstacle. The black point is the source and the white point is the destination. **Lower left:** The equal distance contours in the (x, y, t) Cspace are shown as gray curves. The optimal path and the boundary of the obstacle are in black. **Lower right:** The $\mathcal{M}_{ij}^{m}$ multi-valued distance map, in the (x, y, t) Cspace, where a black point corresponds to $(i, j, \mathcal{M}_{ij}^{m})$. The black curve is the optimal path.