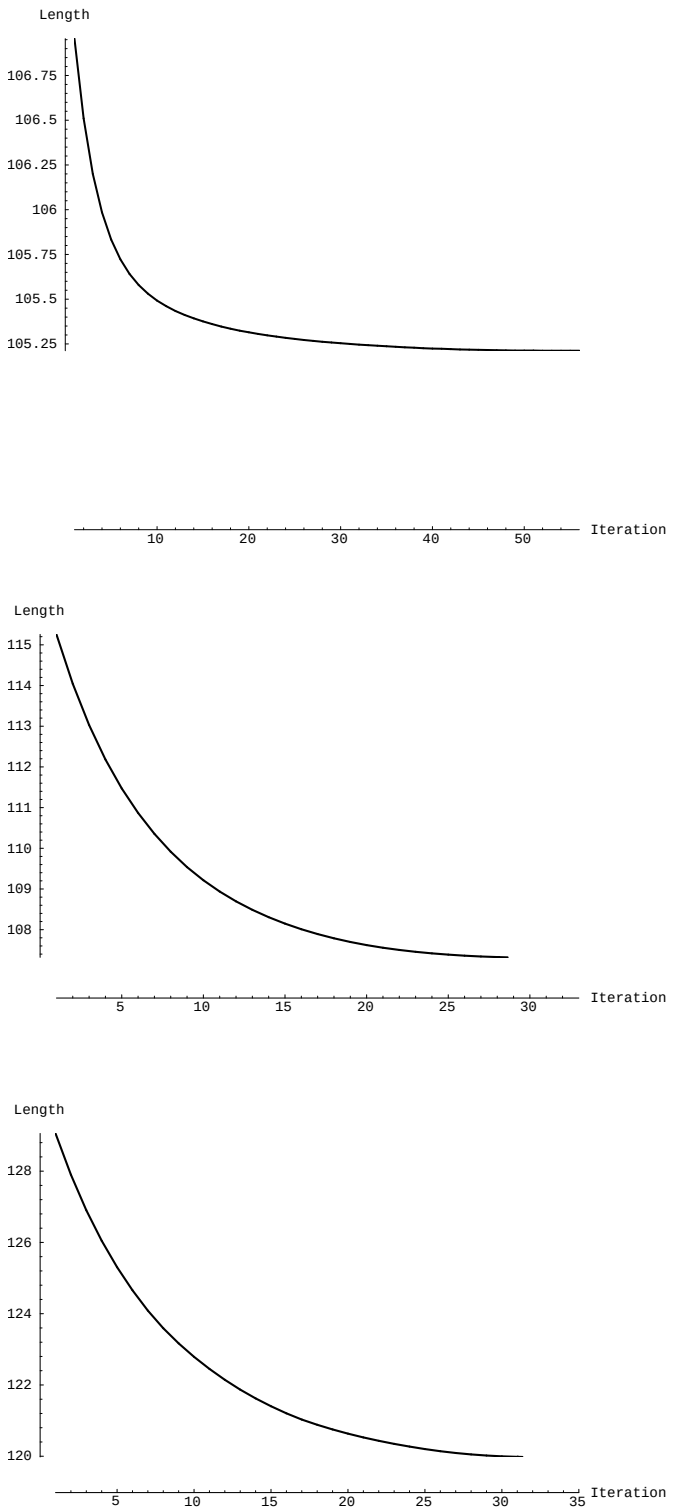

Figure 7.6: The rate of convergence of the second stage of the algorithm for the small (upper left frame), medium (middle frame) and large (lower frame) amplitude egg box surfaces. The length of the path in each iteration is linearly interpolated.

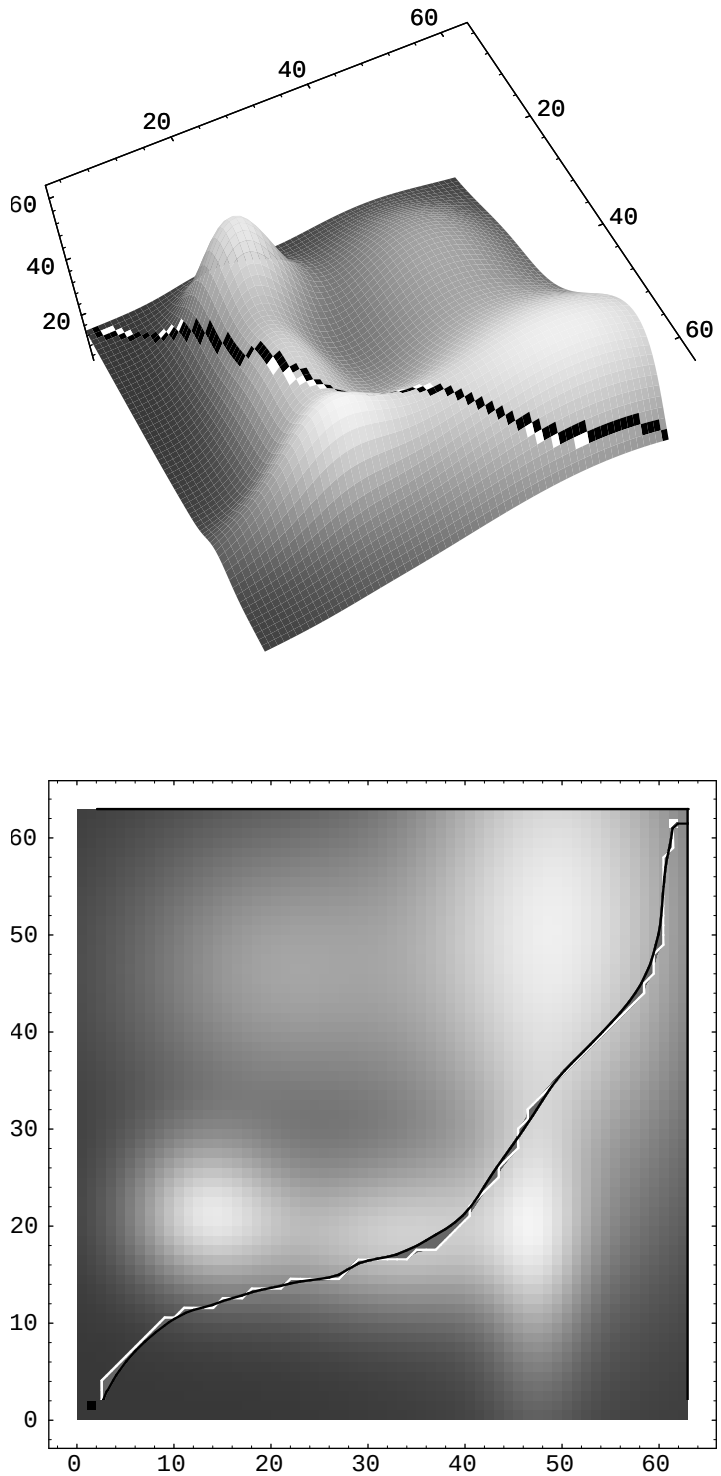

Figure 7.7: The white estimated path is refined to the black geodesic path on a complicated surface.

Chapter 8

Multi–Valued Distance Maps for Motion Planning

This chapter 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 [74, 75]. 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.

8.1 Chapter Introduction

Searching for shortest paths on surfaces with stationary obstacles is a classical problem. Depending on the surface representation it has been studied using computational geometry [125, 131, 144, 98], differential geometry and hybrid techniques [83, 39] (see also Chapters 6 and 7. 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* [59, 60]. The moving planar obstacle can then be represented as a three dimensional structure in the Cspace as shown in Figure 8.1.

This chapter considers the problem of finding the time-optimal path between two points on a surface (not necessarily planar) in the presence of moving obstacles. Relaxing the planarity requirement does not lead to a significant increase in the computational load. More important, we show that without making any approximations, 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}. 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.

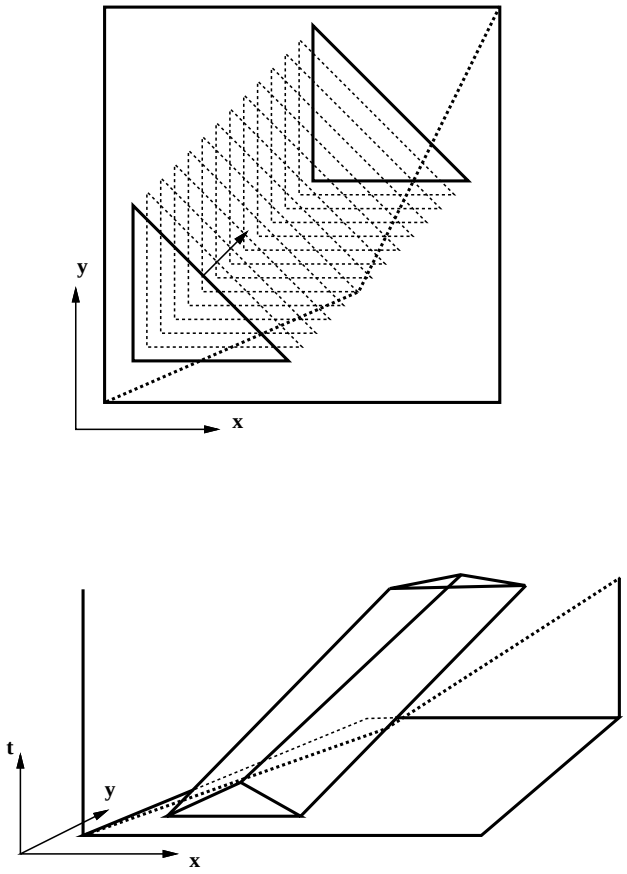

Figure 8.1: A triangular obstacle moving on the plane is represented by a prism in the Cspace. Searching for a path between two given points on the plane amounts to searching for a $3D$ path in the Cspace. This path should connect the two vertical lines in the Cspace that represent the starting and ending 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 [1].

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 [46] *"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.

8.2 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 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 $\mathcal{M}_{\mathbf{p}}((x,y),m) : \Omega \times \mathcal{E}_{max} \rightarrow \mathbb{R}^+$, where $\Omega \in \mathbb{R}^2$ 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 Figure 8.2.

In terms of the prairie fire model [8], 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 $\mathcal{S} \in \mathbb{R}^3$, where $\mathcal{S}$ is given by the function $z : \Omega \in \mathbb{R}^2 \rightarrow \mathbb{R}$, 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)

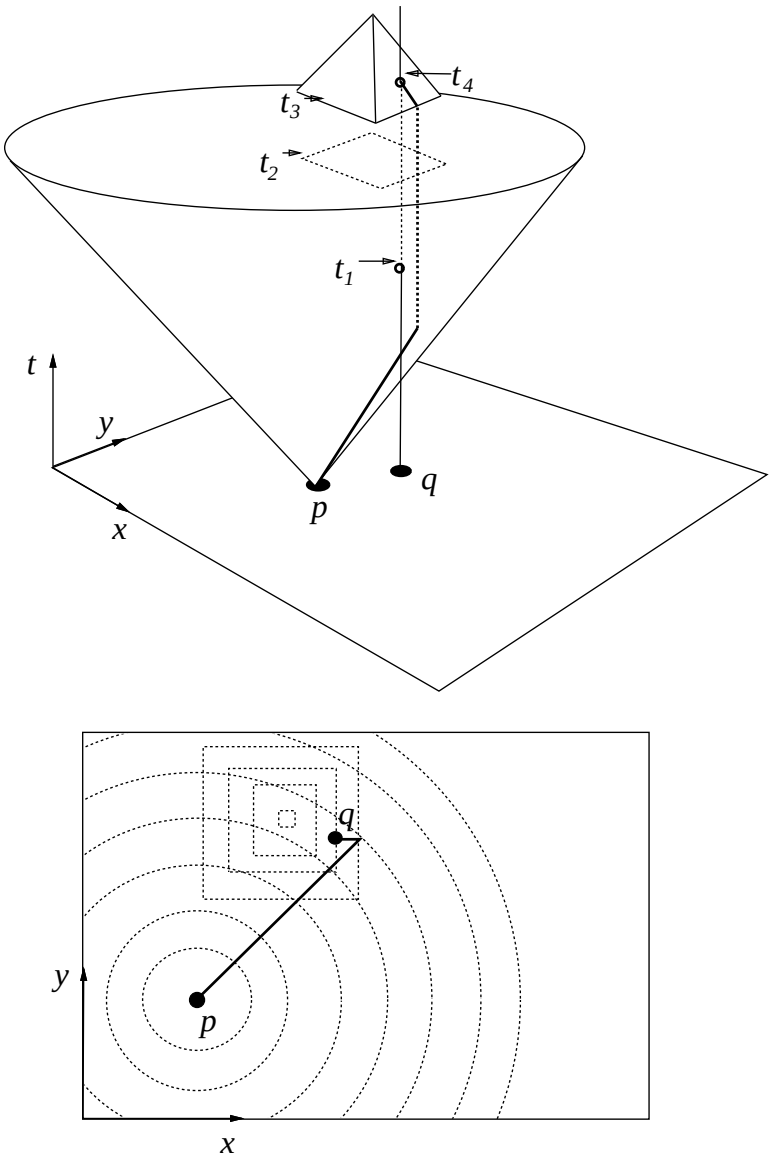

Figure 8.2: A robot is located at a point $\mathbf{p}$ on a plane at $t = 0$. It should travel from $\mathbf{p}$ to another point $\mathbf{q}$ and stay there. At t_2 an obstacle lands on the plane in a square that includes point $\mathbf{q}$ and disappears at t_3. From t_2 to t_3 the robot cannot stay 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 $\mathbf{q}$, $\mathcal{M}_{\mathbf{p}}(\mathbf{q}, 1)$ 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.

time the wavefront's projection passes through it. With moving obstacles, consider a point $\mathbf{p} \in \Omega$ through which the wave front 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 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 8.3.

Define $\mathcal{X}(\mathbf{p})$ 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

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

The new "configuration space" over which it is sufficient to perform the search is $\Omega \times \mathcal{E}_{max}$. 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.

8.3 Algorithm Development

In this section the differential equation that describes the propagation of equal distance contours on a given surface is determined, based on the results from Chapters 6 and 7. 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 first derivatives 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). See Chapter 2 and [43, 120].

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 $\mathcal{S} \in \mathbb{R}^3$ (that can be described by a function $\mathcal{S} = \{(x, y, z(x, y)) | (x, y) \in \Omega \in \mathbb{R}^2\}$), let the $3D$ equal distance contour at distance t from $\mathbf{p}$ be defined as

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

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 $\mathcal{S}$. As proved in Chapter 6, given a $3D$ parametric curve $\alpha(u)$ on $\mathcal{S}$, the equal distance contour propagation rule is

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

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

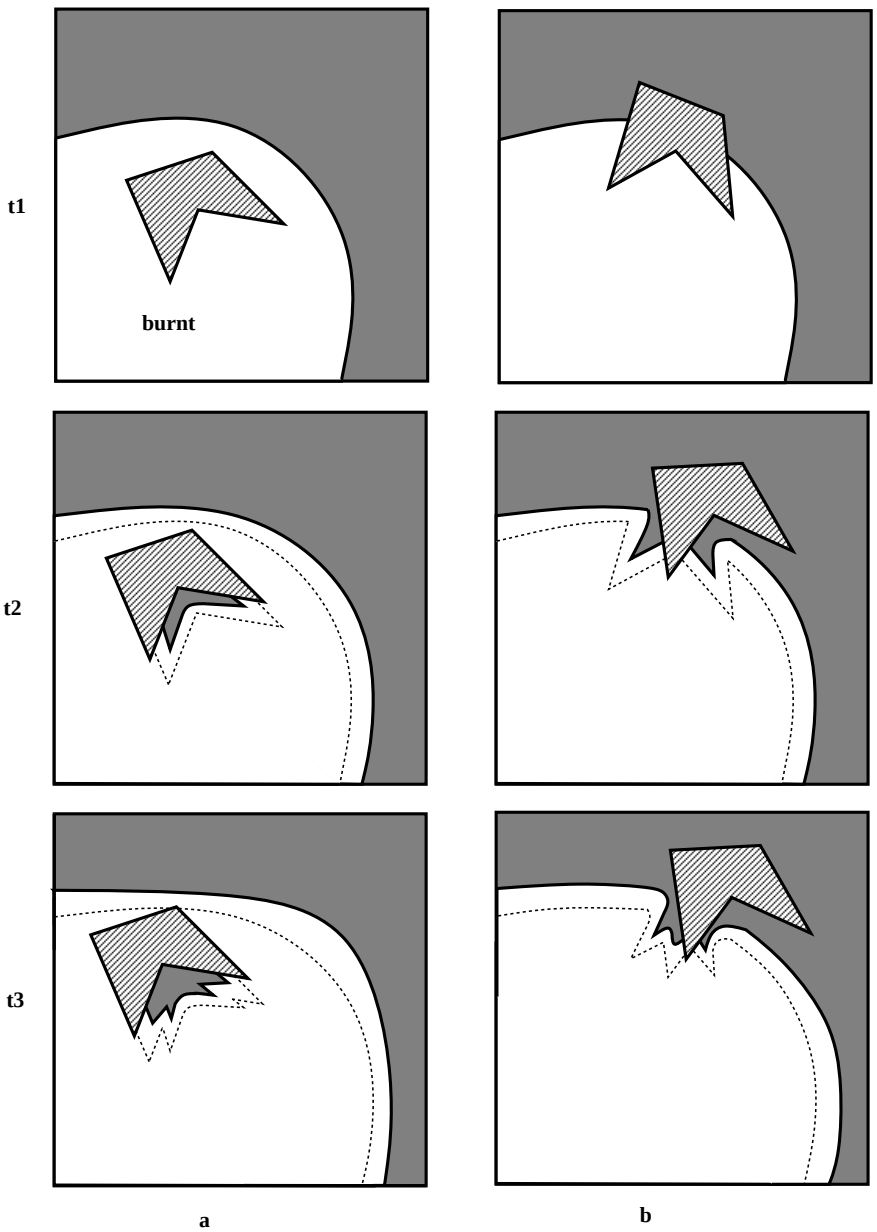

Figure 8.3: In column **a** an obstacle (indicated in gray) is moving in a "burnt" (white) area. The fire front and the obstacle's position are displayed at three successive times. Observe that the fire front at t_2 is the obstacle's boundary and an offset of the fire front and the obstacle's boundary at t_1. Column **b** displays an obstacle moving along the propagating fire front. The obstacle's position in time deforms the wave front shape as it stops the front from propagating at t_1. At t_2 the wave front is further deformed by the current position of the obstacle. The outline of the obstacle is impressed into the wave front.

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,

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

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.

As shown in Chapter 6, the planar normal component of the projected velocity of the evolving equal distance contour is

$$V_N = \langle \pi \circ (N \times \mathcal{T}^{\alpha}), \mathcal{N} \rangle.$$

Writing the normal to the planar curve in terms of its components $\mathcal{N} = (n_1, n_2)$, and the surface normal as $N = (-p,-q,1)/\sqrt{1+p^2+q^2}$, where $p \equiv \partial z(x,y)/\partial x$ and $q \equiv \partial z(x,y)/\partial y$, it is straightforward [63, 64] 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

$$\frac{\partial}{\partial t}\mathcal{C}(t) = V_N \mathcal{N}$$

given

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

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 [63, 64, 65] (as shown in Chapter 6). 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 $\beta(t) : S^1 \rightarrow \mathbb{R}^3$ denote the continuous $3D$ curve representing the optimal path. Starting at the destination point, the direction of backtracking on the surface is given by

$$\vec{\eta} = N \times \mathcal{T}^{\alpha},$$

where N is the surface normal and $\mathcal{T}^{\alpha}$ is the tangent unit vector to the equal distance contour. 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

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

Using these terms

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

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. Suppose for example that the surface and the distance map are defined on a rectangular grid. Denote $z_{ij} \equiv z(i\Delta x, j\Delta y)$, and similarly for the distance map $\mathcal{M}_{ij}^{m} = \mathcal{M}(i\Delta x, j\Delta y, m)$. A possible definition of the neighborhood of a point (i, j) is $\mathcal{W}_{ij} = \{(i, j+1), (i-1, j+1), (i-1, j), (i-1, j-1), (i, j-1), (i+1, j-1), (i+1, j), (i+1, j+1)\}$, see Figure 8.4.

The optimal path is 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

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

subject to the two following constraints

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

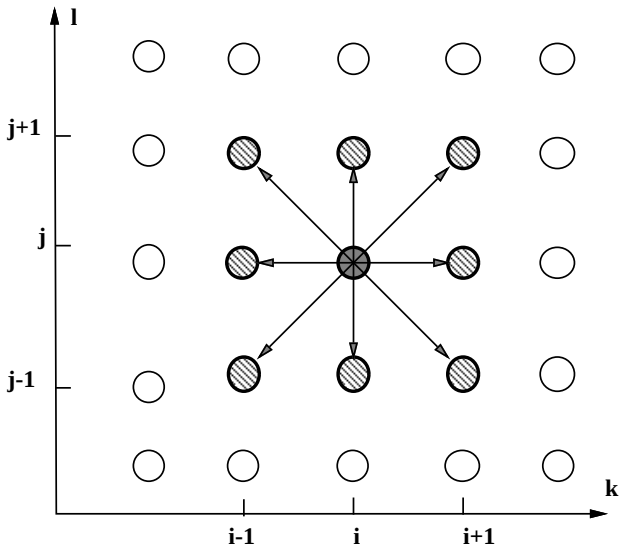

Figure 8.4: The neighborhood $\mathcal{W}_{ij}$, of the grid point (i, j).

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. 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 (for a planar surface $\delta \leq 0.5sin(\pi/8)$.) 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 $\mathcal{V}$).

Angular errors that are accumulated by the back propagation procedure can be corrected using geodesic curvature flow [77, 28] to shorten the generated curve into a geodesic (between the moving obstacles), as shown in Chapter 7.

8.3.1 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 $F : \Omega \in \mathbb{R}^2 \rightarrow [0, 1]$, 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

$$\frac{\partial \mathcal{C}}{\partial t} = F(x,y)V_N(p,q,\mathcal{N})\mathcal{N}.$$

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

$$\frac{\partial \mathcal{C}}{\partial t} = F(x,y)V_N(t,p,q,\mathcal{N})\mathcal{N}.$$

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

$$\frac{\partial \mathcal{C}}{\partial t} = F(x,y,t)V_N(t,p,q,\mathcal{N})\mathcal{N}.$$

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.

8.3.2 The Eulerian formulation

The construction of the multi–valued distance map is a curve evolution process. We may therefore follow Chapters 2 and 6 and construct the Eulerian formulation. 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 8.5. As presented in Chapter 6, the numerical implementation [63, 64, 65] 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 [111, 108]. Using the normal component of the velocity in the Eulerian formulation equation yields the following evolution rule:

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

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 [63, 64, 65]. Every time step the moving obstacles force the ϕ function to become negative in areas covered by them.

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 [83, 99], this may be difficult in alternative approaches.

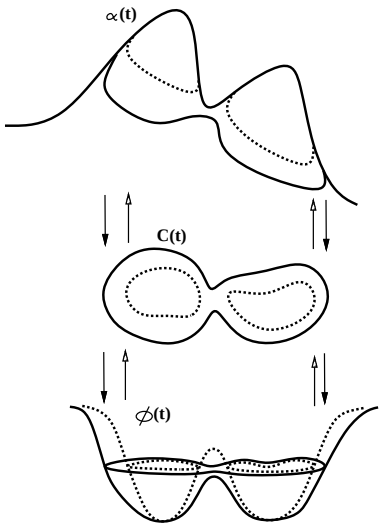

Figure 8.5: Propagating the $3D$ contour $\alpha(t)$ on the surface is done by first projecting the contour onto the plane to obtain $\mathcal{C}(t)$, and then, as suggested by the Eulerian formulation, to propagate its implicit representation $\phi(t)$. This formulation avoids difficulties due to topological changes: observe how a single connected contour propagates and splits gracefully into two separate contours.

8.3.3 Computational considerations

The order of the computational complexity of serial generation of the multi–valued distance map is $\mathcal{O}(\frac{L}{\Delta t}n^2)$, where L is the length of the optimal path, Δt is the time step in the numerical scheme and n^2 is the number of grid points (the size of the planar domain Ω). Observe that L corresponds to the first time the propagating contour reaches the destination point. The memory complexity is $\mathcal{O}(\mathcal{E}_{max} \cdot n^2)$, 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 $\mathcal{O}(\Delta t)$. It can be reduced by using higher order schemes [108, 111].

8.4 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 $(48, 48)$.

In the first example, Figure 8.6, 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 8.2.

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 the examples shown in Figures 8.8 to 8.9 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 8.6. 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 8.7, 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 $v = 1$).

In Figure 8.8, 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 8.9, 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.

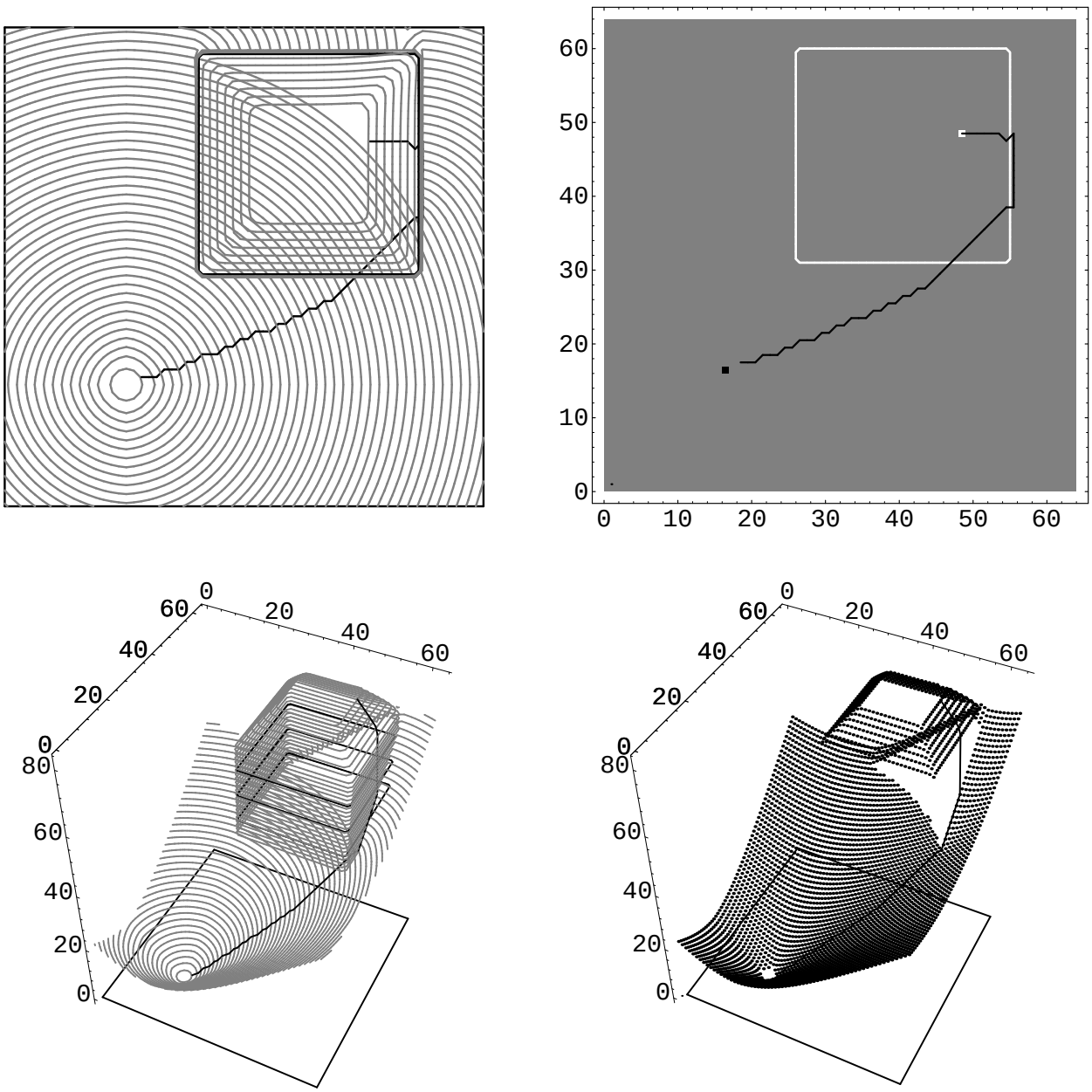

Figure 8.6: Time optimal path on a plane in the presence of a square obstacle that landed at $t = 44$ and disappeared at $t = 73$. **Upper left:** The gray curves are an overhead view of the equal distance contours. The black curves are the time optimal path and the obstacle boundary. **Upper right:** 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.

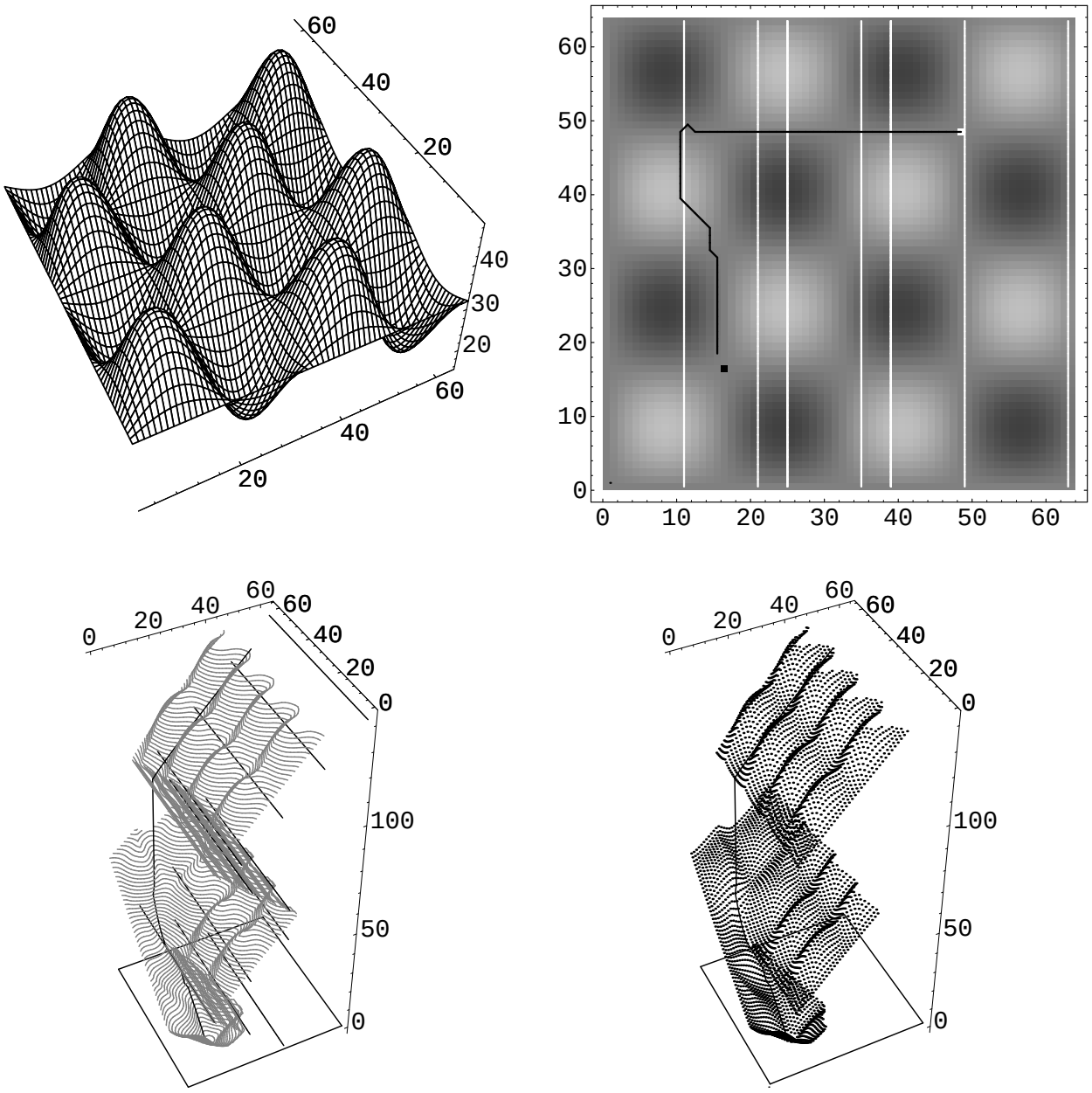

Figure 8.7: A single line obstacle is moving from $i = 40$ to $i = 10$, then backwards from $i = 10$ to $i = 50$, again to $i = 10$, and at last to the image frame boundary $i = 64$. The motion of the obstacle obviously creates multiple values in the distance map. **Upper left:** The 'egg-box' surface. **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.

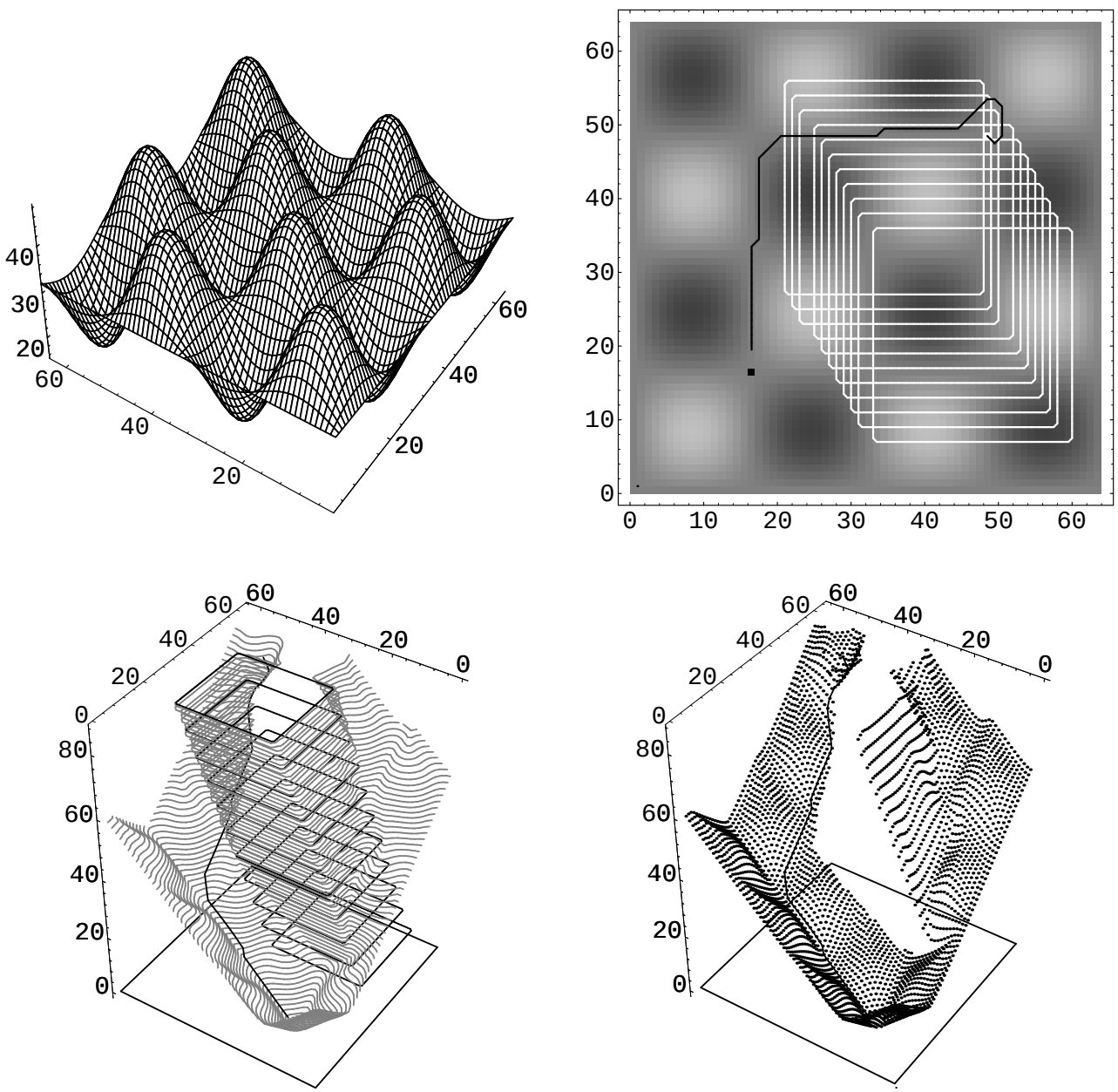

Figure 8.8: Time optimal path in presence of a moving square obstacle on an 'egg-box' surface. **Upper left:** The surface. **Upper right:** The time optimal path is the black curve. The white square is the boundary of the obstacle. The black point is the source and the white point is the destination. **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.

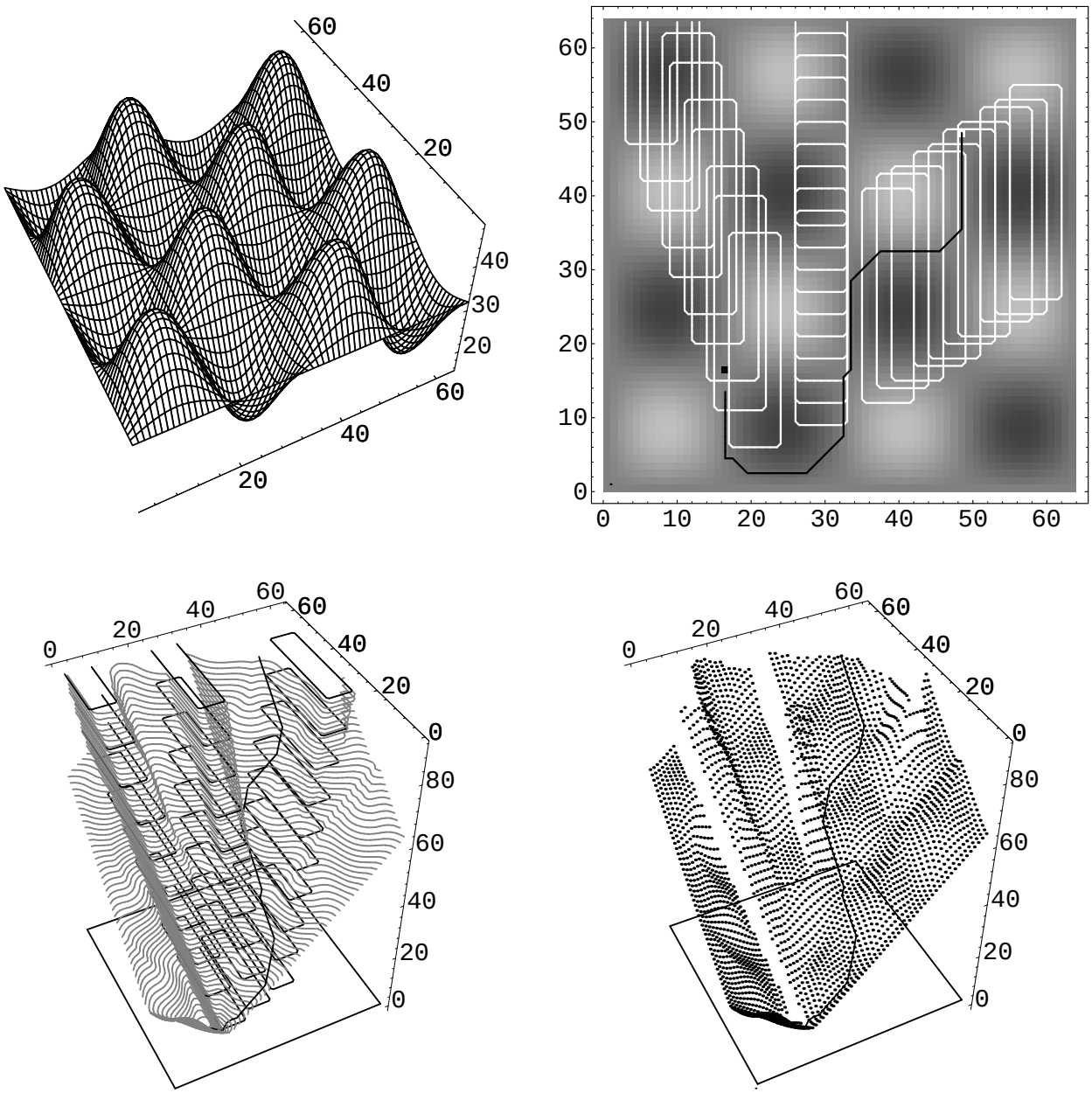

Figure 8.9: Time optimal path in presence of three moving obstacles on an 'egg-box' surface. **Upper left:** The surface. **Upper right:** The time optimal path is the black curve. The white square is the boundary of the obstacle. The black point is the source and the white point is the destination. **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.