IEEE Robotics and Automation Magazine (2013). 20(4), 111-120.
DOI: http://dx.doi.org/10.1109/MRA.2013.2248309

The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories

Alberto Valero-Gomez, Robotics Lab, Universidad Carlos III de Madrid, Spain. E-mail: alberto.valero@uc3m.es.

Javier V. Gomez, Robotics Lab, Universidad Carlos III de Madrid, Spain. E-mail: jvgomez@ing.uc3m.es.

Santiago Garrido, Robotics Lab, Universidad Carlos III de Madrid, Spain. E-mail: sgarrido@ing.uc3m.es.

Luis Moreno, Robotics Lab, Universidad Carlos III de Madrid, Spain. E-mail: moreno@ing.uc3m.es.

This article provides a comprehensive view of the novel fast marching (FM) methods we developed for robot path planning. We recall some of the methods developed in recent years and present two improvements upon them: the saturated FM square (FM2) and an heuristic optimization called the FM2 star (FM2*) method. The saturated varia-tion of the existing satu-rated FM2 provides safe paths that avoid unnec-essarily long trajectories (like those computed using the Voronoi diagram). FM2* considerably reduces the computation time. As a result, these methods provide not only a trajectory but also an associated control speed for the robot at each point of the trajectory. The proposed methods are complete; if there is a valid trajectory, it will always be found and will always be optimal in estimated completion time.

Path-Planning Algorithms

Path planning is a well-known problem with a well-under-stood mathematical basis. It already has many approaches that successfully provide acceptable solutions for the desired task. The characteristics of the paths pro-vided by the different existing algo-rithms change if the path planning is focused on video games, artificial intelligence, mobile robots, unmanned aerial vehicles, and so on.

The path-planning objective has changed since the first approaches were proposed. Initially, the goal was to create an algo-rithm capable of finding a path from an initial point to a final point with ensured completeness (i.e., the algorithm would find a path if it existed). As this objective has largely been solved (i.e., Dijkstra algorithm) and computational capacity has computational capacity has increased exponentially, the objective has become more objective is to find the shortest or fastest path while maintaining safety constraints. new methods are also expected to provide smooth, humanlike paths. here are many different path-planning lgorithms proposed. LaValle [1] proposes a classification into two big groups, depending on the way the information is discretized. combinatorial planning constructs structures that capture all the information needed in path planning [2], [3] and sampling-based planning incre-mentally searches in the space for a solution using a collision-detection algorithm [3]. In the first group, the most widespread methods are those based on road maps, which mainly obtain precalculated short paths from a road map and create the path by taking the needed sections of the map.

In the second group, there are very different options. For example, rapidly exploring random trees [4] provide a fast solution based on creating random branches from an initial point. The collision-free branches are stored, and new branches are created iteratively until the goal point is reached. Another option is to model the environment in an ccupancy gridmap and apply search algorithms, such as the Dijkstra algo-rithm or A* algorithm, using each cell as a node. There are also potential field-based approaches [5], in which the robot is treated as a particle under the influence of an artificial potential field. The main problem with these methods is that they can have local minima. If the path falls in a local min-ima, a correct path will not be found even if it exists.

This article focuses on these potential field-based algorithms (sampling-based algorithms). These methods are based on creating artificial potential fields from the sampled information through sensors and obtaining the path from these fields. The FM method can be applied to create the potential fields and obtain artificial local-minima-free fields, thereby solving one of the most important drawbacks of these path-planning methods. We apply the FM method successfully by combining it with a Voronoi diagram [6] and applying the method iteratively [7].

The FM Method

The FM method is a particular case of level set methods initially developed by Osher and Sethian [8]. It is an efficient computational numerical algorithm for tracking and modeling the motion of a physical wave interface (front), denoted as Γ. This method has been applied to different research fields, including computer graphics, medical imaging, computational fluid dynamics, image processing, computation of trajectories, etc. [9]–[11].

The FM method can be understood intuitively by considering the expansion of a wave. If a stone is thrown into a pond, a wavefront is originated, and this wave expands with a circle shape around the point where the stone fell. In this example, the fluid is always water, thus the wave expansion velocity is always the same, and that is why the wavefront is circular. However, if we repeat this experiment by mixing water and oil, we would observe that the wave expands at different speeds in each medium. As a result, the wavefront will not be circular anymore. If we consider another point in the fluid (a target point), the wavefront will arrive to that point after a certain time. The path that the wavefront has followed from the origin to the target point will be the shortest path in time, considering that the traveling speed along the path is the expansion velocity of the wavefront (which differs depending on the fluid).

In the FM method, the wavefront is called the interface. The interface can be a flat curve [two-dimensional (2-D)] or a three-dimensional (3-D) surface, but the mathematical model can be generalized to n dimensions. The FM method calculates the time T that a wave needs to reach every point in the space. The wave can originate from more than one point, and each source point originates one wave. Source points have an associated time $T = 0$.

In the context of the FM method, we assume that the front Γ evolves by motion in the normal direction. The speed F does not have to be the same everywhere, but it is always nonnegative. At a given point, the motion of the front is described by the equation known as the Eikonal equation (as given by Osher and Sethian [8])

$$1 = F(x)\left|\nabla T(x)\right|,$$

where x is the position, $F(x)$ is the expansion speed of the wave at that position, and $T(x)$ is the time the wave interface requires to reach x.

The magnitude of the gradient of the arrival function $T(x)$ is inversely proportional to the velocity

$$\frac{1}{F} = \left|\nabla T\right|.$$

The full mathematical development of this equation and its discrete solution can be found in [12], [8].

It is important to highlight a property of the wave's expansion. The $T(x)$ function originated by a wave that grows from one single point presents only a global minima at the source and no local minima. As $F > 0$ the wave only grows (expansion), and hence, points farther from the source have greater T. A local minima would imply that a point has a T value lower than a neighboring point that is nearer to the source, which is impossible, as this neighbor must have been reached by the wave sooner.

In the following section, an algorithm to solve the Eikonal equation over a gridmap is presented in detail.

Implementation

Equation 5 (see "Discrete Eikonal Equation Solution") can be solved iteratively over a gridmap. To do so, the cells of the gridmap must be labeled as one of the following types.

- *Unknown*: Cells whose T value is not yet known (the wavefront has not reached the cell).
- *Narrow Band*: Candidate cells to be part of the front wave in the next iteration. They are assigned a T value that can still change in future iterations of the algorithm.
- *Frozen*: Cells that have already been passed over by the wave and, hence, their T value is fixed.

The algorithm has three stages: initialization, main loop, and finalization.

Initialization

The algorithm starts by setting $T = 0$ in the cell or cells in which the wave originates. These cells are labeled as

Discrete Eikonal Equation Solution

The mathematical development of the equation modeling a wave expansion is as follows:

$$\frac{1}{F} = |\nabla T|.$$

As the front can only expand ($F > 0$), the arrival time T is single valued. Osher and Sethian [8] proposed a discrete solution for the Eikonal equation. In 2-D, the area is discretized using a gridmap. We denotate the row i and column j of the gridmap, which corresponds to a point $p(x_i, y_i)$ in the real world. The discretization of the gradient ∇T according to [8] leads to the following equation:

$$\left\{\begin{matrix}\max(D_{ij}^{-x}T,0)^2 + \min(D_{ij}^{+x}T,0)^2 + \\ \max(D_{ij}^{-y}T,0)^2 + \min(D_{ij}^{+y}T,0)^2 + \end{matrix}\right\} = \frac{1}{F_{ij}^2}. \quad \text{(S1)}$$

In [8], Osher and Sethian propose a simpler but less accurate solution for (S1), expressed as follows:

$$\left\{\begin{matrix}\max(D_{ij}^{-x}T, -D_{ij}^{+x},0)^2 + \\ \max(D_{ij}^{-y}T, -D_{ij}^{+y},0)^2 \end{matrix}\right\} = \frac{1}{F_{ij}^2}, \quad \text{(S2)}$$

where

$$\begin{aligned} D_{ij}^{-x} &= \frac{T_{i,j} - T_{i-1,j}}{\Delta x}, \\ D_{ij}^{+x} &= \frac{T_{i+1,j} - T_{i,j}}{\Delta x}, \\ D_{ij}^{-y} &= \frac{T_{i,j} - T_{i,j-1}}{\Delta y}, \\ D_{ij}^{+1} &= \frac{T_{i,j+1} - T_{i,j}}{\Delta y}, \end{aligned} \quad \text{(S3)}$$

and Δx and Δy are the grid spacing in the x and y directions. Substituting (S3) in (S2) and letting

$$\begin{aligned} T &= T_{i,j}, \\ T_1 &= \min(T_{i-1,j}, T_{i+1,j}), \\ T_2 &= \min(T_{i,j-1}, T_{i,j+1}), \end{aligned} \quad \text{(S4)}$$

we can rewrite the Eikonal equation for a discrete 2-D space as

$$\max\left(\frac{T - T_1}{\Delta x}, 0\right)^2 + \max\left(\frac{T - T_2}{\Delta y}, 0\right)^2 = \frac{1}{F_{i,j}^2}. \quad \text{(S5)}$$

As we are assuming that the speed of the front is positive ($F > 0$) T, must be greater than T_1 and T_2 whenever the front wave has not already passed over the coordinates *i, j*.

We can solve (S5) in three steps. First, we solve the following quadratic:

$$\left(\frac{T - T_1}{\Delta x}\right)^2 + \left(\frac{T - T_2}{\Delta y}\right)^2 = \frac{1}{F_{i,j}^2}. \quad \text{(S6)}$$

If $T > T_1$ and $T > T_2$ [taking the greater value of T when solving (S6)] the obtained value is the correct solution for (S5). Otherwise, if $T < T_1$ or $T < T_1$, from (S5) the corresponding member of $(T - T_1/\Delta x, 0)$ is 0, and hence, (S5) is reduced to

$$\left(\frac{T - T_1}{\Delta x}\right) = \frac{1}{F_{i,j}} \quad \text{(S7)}$$

$$\left(\frac{T - T_2}{\Delta y}\right) = \frac{1}{F_{i,j}}, \quad \text{(S8)}$$

depending on the final value of T.

frozen. Afterward, it labels all their Manhattan neighbors as narrow band, computing T (5) for each of them.

Main Loop

In each iteration the algorithm will solve the Eikonal equation (5) for the Manhattan neighbors (that are not yet frozen) of the narrow band cell with the lesser T value. This cell is then labeled as frozen. The narrow band maintains an ordered list of its cells. Cells are ordered by T value from lowest to highest.

Finalization

When all the cells are frozen (i.e., the narrow band is empty), the algorithm finishes. We can see the process in Figures 1 and 2. In Figure 1, the wave originates from

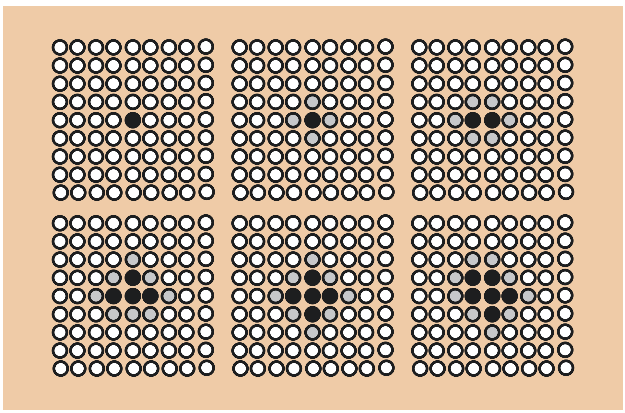

Figure 1. Iterative wave expansion with one source point.

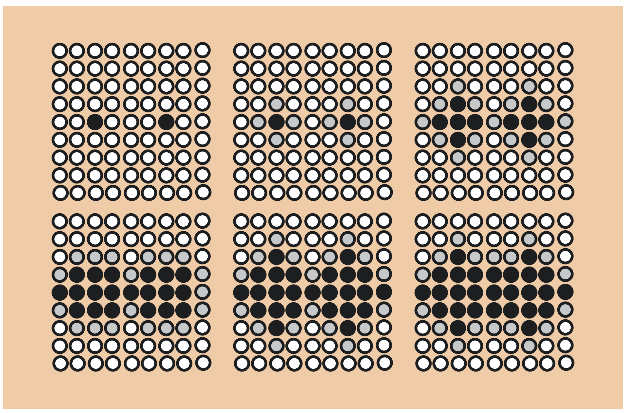

Figure 2. Iterative wave expansion with two source points.

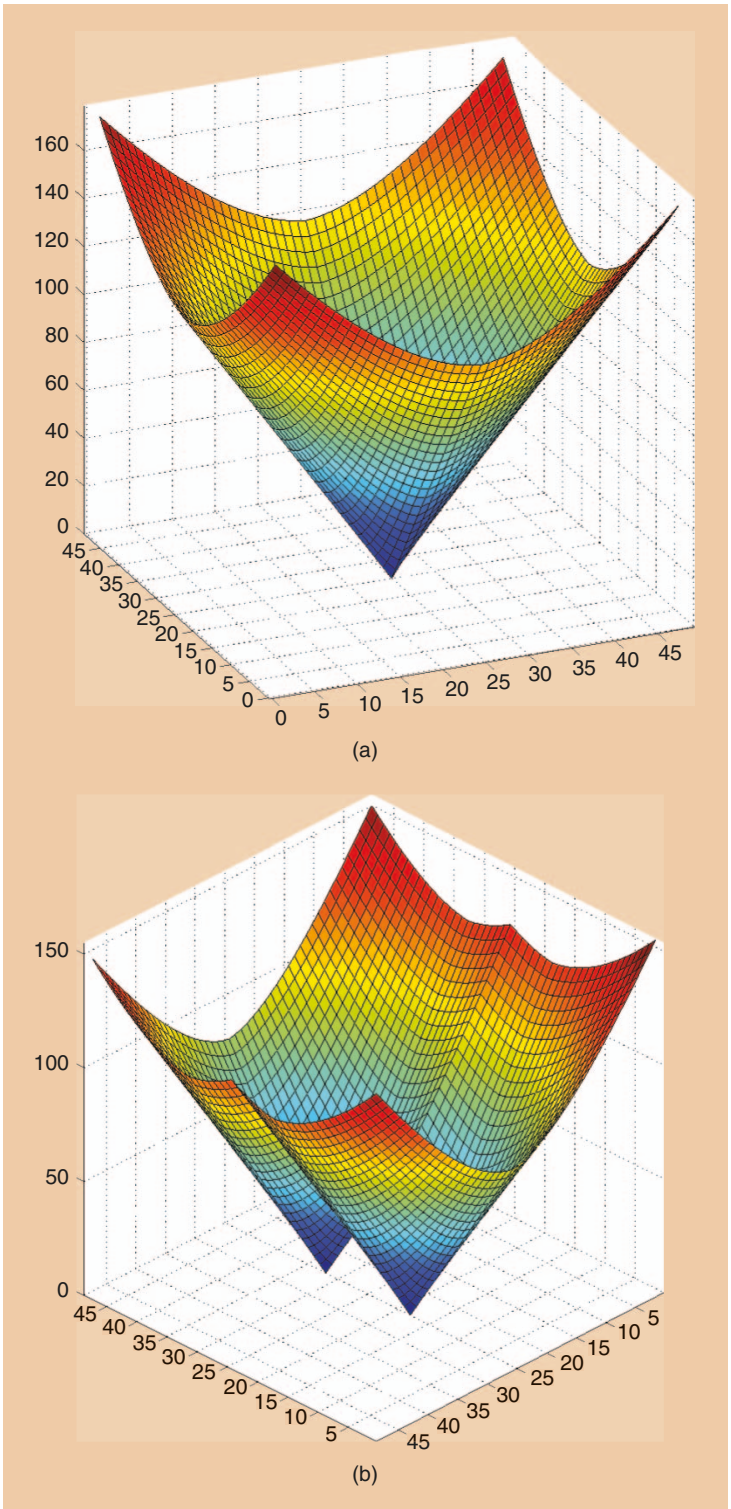

Figure 3. FM method applied over a 50 × 50 gridmap. (a) One source point. (b) Two source points.

one point. In Figure 2, there are two wave sources. Black points are frozen, and their T value will not change. Gray points are the narrow band, whereas white ones are unknown. In Figure 1, we can clearly see that the waves grow concentrically to the source. In Figure 2, they join as the waves develop individually and grow together. The iterative process expands cells at the same rate that the physical wave grows, as the cells with lower T are expanded first.

If we consider T as the third dimension over the z axis, the result of completing the wave expansion of Figures 1 and 2 results in Figure 3(a) and (b), respectively. Since $F > 0$, as we move away from the sources, the time T required to reach the point is greater (higher on the z axis). With a single source, there is only one minima (located at the source). With more than one source, we have a global minima at each one with $T = 0$. Algorithm 1 shows the pseudocode of the process.

FM and Path Planning

Let us consider a binary gridmap in which obstacles are valued as zero and free space as one. These values can be taken as the wave expansion speed F over the gridmap. At obstacles, the wave expansion speed is zero, as the wave cannot go through obstacles, and in free space, wave expansion speed is constant and equal to one. If we want to compute the path between two points p_0 and p_1 we could expand a wave from p_0 until it reaches p_1 (or vice versa). Because of the wave expansion properties, the path that the wave interface has followed from the target to the source point will always be the shortest trajectory in time. As the wave expansion speed is constant, this trajectory is also the shortest solution in distance. The wave originates from the target point, hence, the computed $T(x)$ field will have only one global minima at the target point. Hence, following the maximum gradient direction from the initial point, we will reach the target point, obtaining the trajectory. This solution is unique, and the algorithm is complete.

Figure 4 shows an example. We want to trace the shortest path from p_0 to p_1. Figure 4(b) shows the wave expansion in gray scale (the wave is originated at the initial point marked with a red cross). The farther the interface is from the initial point, the lighter the map becomes. Once the interface Γ has reached the target point p_1 (shown in blue), the algorithm stops expanding.

The resulting gridmap stores at any pixel the time T required by the wavefront to reach that pixel. The isocurves join together all the points that are passed through at the same instant in time [Figure 4(d)]. These curves are the trace of the front wave. If we compute the maximum gradient direction at any point of the gridmap, we obtain the normal direction to the isocurve, that is, the direction the curve followed while expanding. The maximum gradient direction is computed by applying the Sobel operator over the gridmap.

To trace the path between p_1 and p_0, we just need to follow the maximum gradient direction starting at p_1, (target point). The path is computed iteratively. grad_{ix} and grade_{iy} are computed at every point, $p_i \cdot p_{i+1}$ is computed from p_i, and this process is repeated successively until p_0 is reached. As p_0 is located at the global minima, it is always reached whenever there is a valid path.

In Figure 4(c) and (d), we can see that the created field has just one global minima at p_1 and, hence, the solution is unique.

FM over Voronoi Diagram, FM^2, and FM^{2*}

The path calculated in the section "FM and Path Planning" is the shortest in length, but it might not be safe because of its proximity to obstacles. For this reason, it is also not the shortest in time, as the robot must move slowly when it is close to the obstacles to avoid collisions or risky movements. The usual solution is to expand obstacles before calculating the path. Nonetheless, when a robot moves through a door, we would like it to pass at an equal distance from both walls, and the same holds for corners, etc. In the following section, we introduce three different methods based on FM: FM over the Voronoi diagram, the FM^2 method for path planning, and an heuristic modification of FM^2 called the FM^{2*} method. We will show how FM^{2*} considerably reduces the computation time while providing the same trajectory.

FM over Voronoi Diagram

As the Voronoi diagram concept is simple and intuitive, it can be applied to a large number of applications, including path planning. Given a finite set of objects in a space, all locations in that space are associated with the closest member of the object set. Thus, the space is partitioned into a set of regions, called Voronoi regions. The generalized Voronoi diagram is the set of the points that belong to the frontier between regions.

Algorithm 1. ***FM Algorithm***

```
1  input: A gridmap G of size m × n
2  input: The set of cells Ori where the wave is
          originated
3  output: The gridmap G with the T value set for
           all cells
4      Initialization;
5  foreach g_ij ∈ Ori do
6      g_ij.T ← 0;
7      g_ij.state ← FROZEN;
8      foreach g_kl ∈ g_ij.neighbours do
9          if g_kl = FROZEN then skip; else
10             g_kl.T ← solveEikonal(g_kl);
11             if g_kl.state = NARROW BAND then
12             narrow_band.update_position(g_kl);
13             if g_kl.state = UNKNOWN then
14                 g_kl.state← NARROW BAND;
15                 narrow_band.insert_in_position(g_kl);
16             end
17         end
18     end
19     Iterations;
20     while narrow_band NOT EMPTY do
21         g_kl ← narrow_band.popfirst();
22         foreach g_kl ∈ g_ij.neighbours do
23             if g_kl = FROZEN then skip; else
24                 g_kl.T ← solveEikonal(g_kl);
25                 if g_kl.state = NARROW BAND then
26                 narrow_band.update_position(g_kl);
27                 if g_kl.state = UNKNOWN then
28                     g_kl.state← NARROW BAND;
29                     narrow_band.insert_in_position(g_kl);
30                 end
31             end
32         end
33     end
34 end
```

The Voronoi diagram is used as a way to obtain a roadmap of the map. The FM method is then used to search for a path over the Voronoi diagram. Applying FM over the Voronoi roadmap decreases the time spent in the search with respect to other existing methods. The main steps of this method are the following.

1) *Map preprocessing*: The map is turned into a binary grid map, in which the obstacles are black (value 0) and the clear space is white (value 1). The obstacles and walls are enlarged to ensure that the robot will not collide with walls or obstacles and will not accept passages narrower than its size. Also, an averaging filter is applied with the objective of erasing small hairs of the diagram.
2) *Voronoi diagram*: The diagram is obtained using morphological image-processing techniques (concretely the methods proposed by Breu [13] in 2-D and Gagvani [14] in 3-D). The diagram is dilatated, obtaining a thickened Voronoi diagram wherein to apply the FM method.
3) *FM:* The FM method is used to create a potential field that represents the propagation of a wave (from the goal point), to which time is added as the third axis in 2-D or the fourth in 3-D (see Figure 3).
4) *Path extraction*: The gradient method is applied to the previously obtained potential from the current position of the robot with the goal point as the final point.

Figure 5 shows the main aspects of this algorithm when it is applied to the map shown in Figure 4(a).

FM^2

Let us take as evidence a gridmap in which obstacles are labeled as zero and free space as one. We can apply the

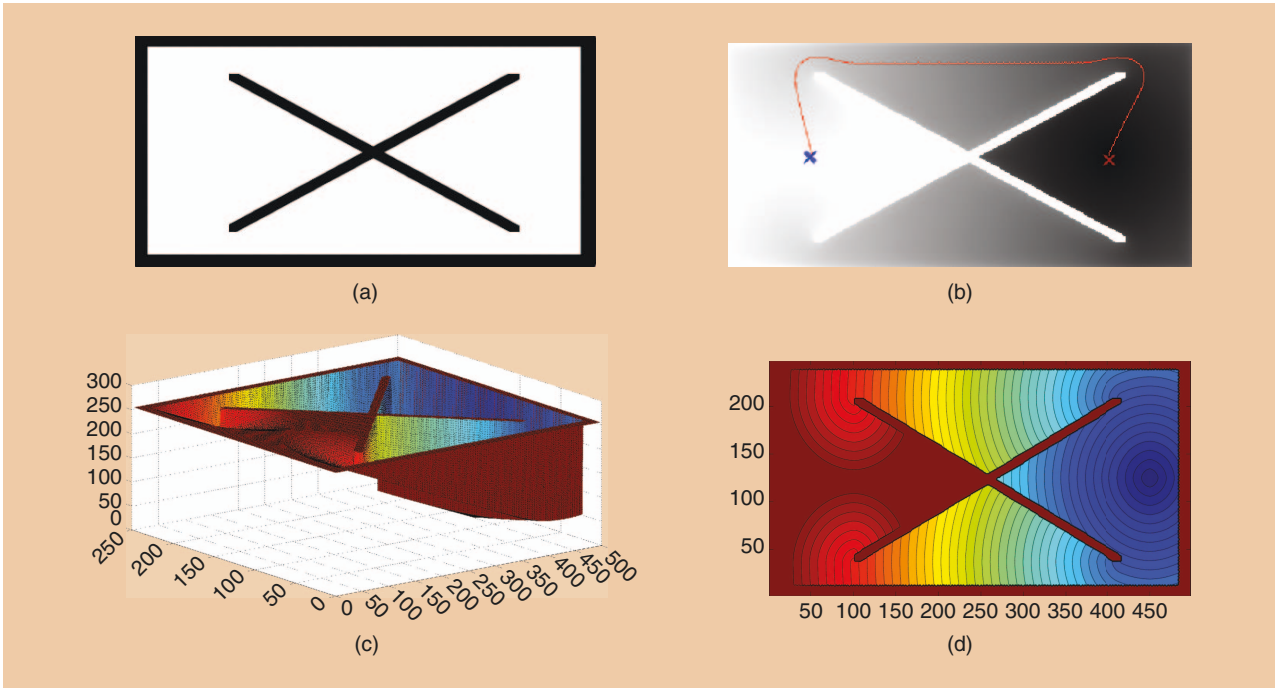

Figure 4. FM and path planning. (a) Original map. (b) Path and wave expansion in gray scale. (c) FM field: One unique global minimum. (d) FM field: Isometric curves.

FM method to this map, considering all the obstacles to be a wave source. In the section "FM over Voronoi Diagram," there was just one wave source (at the target point). In this case, all the obstacles are a source of the wave and, hence, several waves are being expanded at the same time. The map resulting from applying this wave expansion to the map shown in Figure 4(a) can be seen in Figure 6(a). We called the resulting map the *fast marching gridmap* (FMGridMap), which represents a potential field of the original map. As pixels get farther from the obstacles, the computed T value is greater. This map can be seen as a *slowness map*. We can consider the T value to be proportional to the maximum allowed speed of the robot at each point. It is evident from the figure that speeds are lower when the pixel is close to the obstacles and greater when far away from them. A robot whose speed at each point is given by the T value will never collide, as $T \to 0$ when approaching the obstacles. Making an appropriate scaling of the FMGridMap cell values to the robot allowed speeds, we obtain a slowness map, which provides a safe speed for the robot at any point in the environment. Figure 6(c) depicts the speed profile. In the image, it is clear that speeds become greater when the robot is far away from the obstacles.

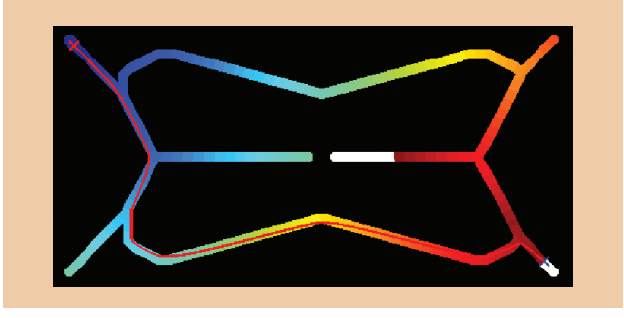

Figure 5. FM over thickened Voronoi diagram.

We can now calculate the path as we did in the section "FM and Path Planning," but instead of taking a constant value for the expansion speed F, we use the speed given by the slowness map. Now, if we expand a wave from one point of the gridmap, considering that the expansion speed $F(x,y) = T(x,y)$, being $F(x, y)$ the speed at point x, y and $T(x, y)$. The value of the FMGridMap at x, y, we will see that the expansion speed depends on the position, precisely the safe speed given by the slowness map. As the slowness map provides the maximum safe speed for the robot, the obtained trajectory is the fastest path (in time), assuming the robot moves at the maximum allowed speed at every point.

There is a previous version of the FM^2 method called Voronoi FM (VFM) [15]. The intuition of this method is the same as FM^2: derive a first potential (slowness or viscosity potential), and propagate a wave over this potential, creating a second potential from which the path is extracted. The main change is the way of obtaining the first potential. In VFM, the first potential is obtained using the extended Voronoi transform, which assigns a value to each cell of the grid map proportional to the Euclidean distance of that cell to the closest obstacle in the environment. The main advantage of FM^2 over VFM is that FM^2 is easier to implement, as both methods have a similar computational cost and the paths provided are the same.

FM^2: The Saturated Variation

In Figure 6(e), we can see that the computed path is not the logical/optimal trajectory we would expect. The FM^2-computed path, as it is presented, tries to keep the trajectory as far as possible from obstacles. This computed trajectory is similar to the path computed with the Voronoi diagram [16]. However, there are environments in which there is no need to follow a trajectory so far away from obstacles, as a smaller distance might be safe enough to navigate. To solve this, we implemented a saturated variation of the FMGridMap. When the first FM has been computed, the FMGridMap is first scaled and then saturated.

The map is scaled according to two configuration parameters:

- maximum allowed speed, which is the maximum control speed the robot may receive
- safe distance, which is the distance from the closest obstacle at which the maximum speed can be reached.

Finally, the map is saturated to the maximum allowed speed. After this scaling and saturation process, the

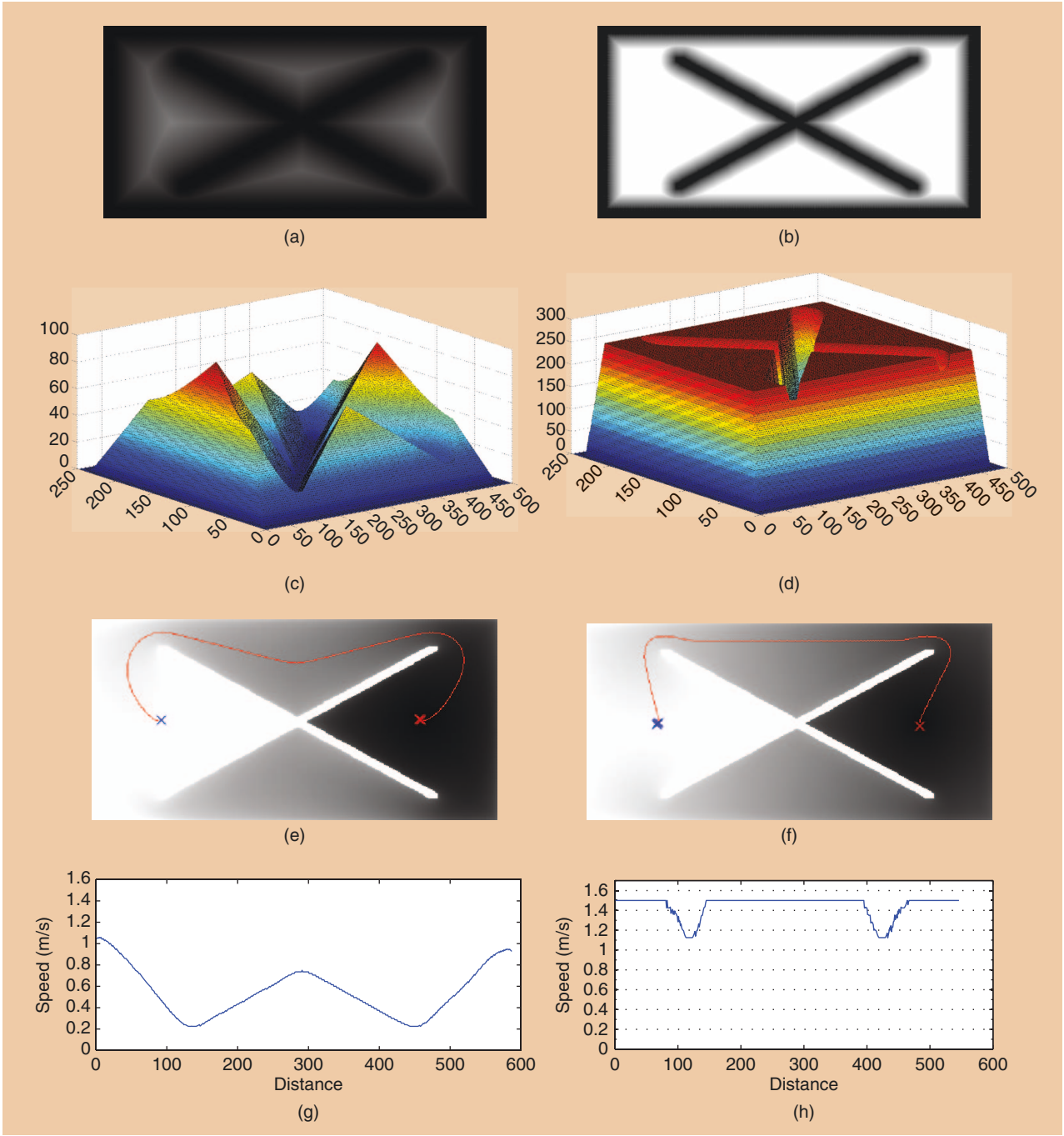

Figure 6. FM^2 and saturated FM^2. (a) FMGridMap. (b) Saturated FMGridMap. (c) 3-D representation. (d) 3-D representation. (e) Second FM and path. (f) Second FM and path. (g) Speed profile. (h) Speed profile.

slowness map provides the maximum speed for all the points that are farther than the safe distance from the obstacles and the control speed, which ranges from zero (at obstacles)

to the maximum speed (at safe distance), for the rest of the points.

Figure 6(b) shows the saturated variation of Figure 6(e) with the new computed trajectory. We can see that now the path is as expected. In Figure 6(d), the speed profile with the saturated area is shown. A video showing how the saturation value affects the computed path can be found with the digital edition of this article on IEEE Xplore.

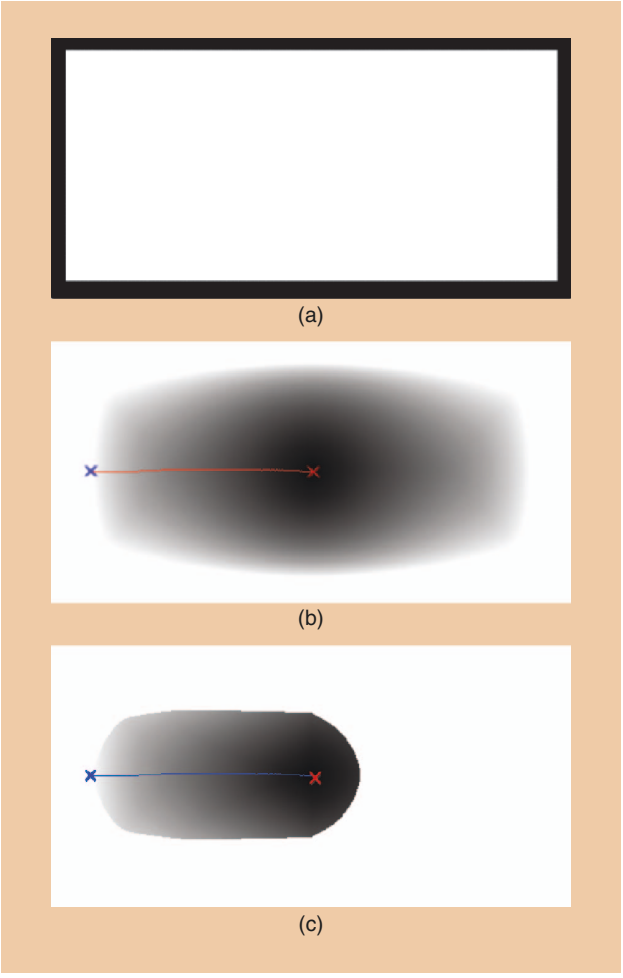

Figure 7. Comparison between FM^2 and FM^{2*}. (a) Original grid map. (b) Wave expansion and path with FM^2. (c) Wave expansion and path with FM^{2*}.

The Heuristic FM^2: FM^{2*}

Consider the map of Figure 7(a), in which a trajectory between two points in a free space must be computed. Figure 7(b) shows the FM^2 wave expansion that originated from the target point. As illustrated in the figure, the wave grows concentrically around the target point until it reaches the initial point. The FM^{2*} method is an extension of the FM^2 method. It tries to reduce the total number of expanded cells (wave expansion) by incorporating an heuristic estimate of the cost to get to the goal from a given point.

The FM^{2*} algorithm works in exactly the same way as the FM^2 algorithm. The only difference is the function used to sort the narrow band queue. FM^2 sorts the narrow band cells in increasing T order so that, in each iteration, the first element in the queue (lowest T) becomes frozen and expands. In FM^{2*}, the algorithm uses the cost-to-come T, which is known, and the optimal cost-to-go, that is, the minimum amount of time the robot would take to reach the target. This implies that the narrow band queue is sorted by estimates of the optimal cost from the given cell to the target. Whenever the optimal cost-to-go is an underestimate of the real cost-to-go, the algorithm will still work. If we take the optimal cost-to-go as zero, the FM^{2*} algorithm is equivalent to the FM^2 algorithm. If the estimation is greater than the real cost-to-go, the FM^{2*} algorithm might take more computational steps than FM^2 to find the path, and the path might not be the shortest one.

In this problem, the optimal cost-to-go (optimal time to reach the target) would be achieved if the robot went straight forward at maximum speed. This cost-to-go is given by the Cartesian distance (minimum distance) divided by the maximum speed the robot can reach. We know that the real cost-to-go will always be greater than this computed value, so the narrow band queue is ordered according to the value T^*

$$T^* = T + \frac{\text{cartesian\_distance\_to\_target}}{\text{robot\_max\_speed}}.$$

These two methods are analogous to Djikstra and A* in path-finding over graphs [3]. The wave expansion computed with FM^2 is shown in Figure 7(b), and the wave expansion computed with FM^{2*} is shown in Figure 7(c).

Results

Figure 8 shows the path that results from applying FM^2 and FM^{2*} over a realistic map of 50 $\times$ 18 m. The gridmap has 500 $\times$ 180 pixels. Starting points are shown in red, and final points are shown in blue. Figure 8(b) shows the slowness map computed with saturation. The maximum speed is 1.5 m/s, and the safe distance is 2 m; that is, at 2 m from the obstacles, the control speed will be 1.5 m/s, and it will linearly decrease when approaching the obstacles. Figure 8(c)–(e) shows the computed trajectory and the wave expansion field resulting of applying FM^2. Figure 8(d) and (f) shows the computed trajectory and the

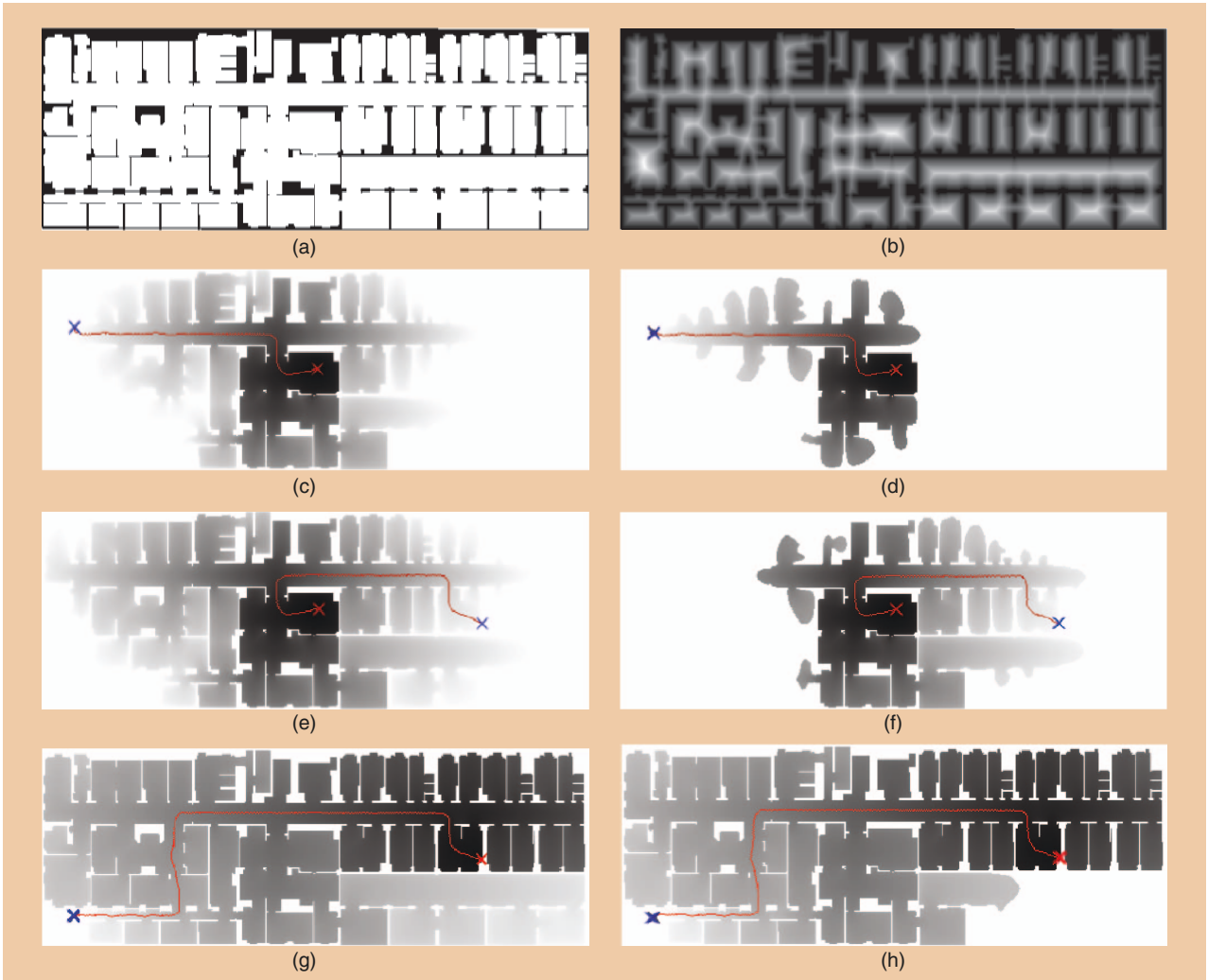

Figure 8. FM^2 versus $FM^{2\star}$. (a) Original map. (b) Saturated FMGridMap–slowness map. (c) FM^2-computed trajectory. (d) $FM^{2\star}$-computed trajectory. (e) FM^2-computed trajectory. (f) $FM^{2\star}$-computed trajectory. (g) FM^2-computed trajectory. (h) $FM^{2\star}$-computed trajectory.

wave expansion field resulting from applying $FM^{2\star}$. We can clearly see that the three paths are the same, but the $FM^{2\star}$ heuristic reduces the number of cells expanded. The computation times are listed in Table 1.

The third computed path has to expand over almost the entire map and, hence, computational times do not diverge too much. In the most favorable case of the first two trajectories, the path is computed over four times faster with $FM^{2\star}$.

A video showing a comparison of the wave expansion between the FM^2 and the $FM^{2\star}$ methods can be found with the digital edition of this article on IEEE *Xplore*. This video clearly shows how the FM^2 wave expands concentrically around the initial point, whereas, in the $FM^{2\star}$ expansion, the waves are directed toward the target point, reducing the number of cells expanded and thus the computation time.

Table 1: FM^2 and $FM^{2\star}$ computation times.

Trajectories	FM^2	$FM^{2\star}$
Figure 8(c) and (d)	0.130669	0.032623
Figure 8(e) and (f)	0.130914	0.031855
Figure 8(g) and (h)	0.189093	0.147036

Conclusions and Further Work

In this article, we presented the mathematical foundations of the FM method developed by Osher and Sethian [12]. We presented the algorithm that we have implemented to apply FM over a gridmap and demonstrated how the FM method can be applied to compute the visibility path between two points in a gridmap. This methodology is limited in that it provides the shortest path in distance, which leads to risk because of its closeness to obstacles. The FM^2 method was explained in detail. FM^2 and all its variations were developed by the authors of this article. FM^2 computed two wave expansions over the gridmap. The first expansion computed the FMGridMap, a slowness map that provided the maximum allowed speed of the robot at each point on the map.

This slowness map was used to compute the second expansion, from the target point to the initial point. As a result of the second expansion, the trajectory was computed using the maximum gradient direction. This solution provided both a path (way point) and the control speed at each point. As a result, this trajectory was safe and optimal in time. FM^2 computes paths that tend to navigate far from obstacles; however, this results in increased navi-gation completion time (increased path lengths), and it is not always neces-sary. To avoid this problem, a variation to FM^2 was presented, called the saturated FM^2. To reduce the computation time of FM^2, an heuristic FM^2, $FM^{2\star}$, was presented. The experimental analysis showed that the computation time was reduced up to four times with respect to FM^2, while provid-ing the same trajectory.

In the video attached to the digital edition, several applications of the FM methods for path planning are shown, including robot formations, multirobot reactive navigation, and road map calculation. A sequence of a robot using FM is also shown.

In a future article, we will focus on expanding $FM^{2\star}$ to more dimensions. Previous FM methods like FM^2 or VFM were applied successfully to higher-level problems, such as outdoor path planning [17], robot formation motion planning [18], exploration, and SLAM [19], but the computational complexity of these methods limited them to 2-D problems (although different options had been suggested to decrease the computational complexity when expanding to three or more dimensions). Using the faster $FM^{2\star}$, those high-level problems can be studied easily. Our results indicate that $FM^{2\star}$ is ideal for use in swarm robotics or in dynamic environments.

The source code in C++ of FM^2 and $FM^{2\star}$ and the recent developments of these algorithms can be retrieved from the subversion repository: http://svn.iearobotics.com/openmrl/. All the code stored there was distributed under the attribution–share alike–creative commons license.

References

[1] S. LaValle, "Motion planning," *IEEE Robot. Autom. Mag.*, vol. 18, no. 1, pp. 79–89, 2011.

[2] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 3rd ed. New York: Springer-Verlag, 2008.

[3] S. M. LaValle. (2006). *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press. [Online]. Available: http://planning.cs.uiuc.edu/

[4] J. J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. Robotics Automation, ICRA '00. IEEE Int. Conf.*, 2000, vol. 2, pp. 995–1001.

[5] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," in *Proc. Robots Unstructured Environments, 91 ICAR, 5th Int. Conf. Advanced Robotics*, 1991, vol. 2, pp. 1012–1017.

[6] S. Garrido, L. Moreno, and D. Blanco, "Voronoi diagram and fast marching applied to path planning," in *Proc. Robotics Automation, ICRA IEEE Int. Conf.*, May 2006, pp. 3049–3054.

[7] S. Garrido, L. Moreno, M. Abderrahim, and D. Blanco, "FM^2: A realtime sensor-based feedback controller for mobile robots," *Int. J. Robot. Autom.*, vol. 24, no. 1, pp. 48–65, 2009.

[8] S. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations," *J. Comput. Phys.*, vol. 79, no. 1, pp. 12–49, 1988.

[9] S. Jbabdi, P. Bellec, R. Toro, J. Daunizeau, M. Pélégrini-Issac, and H. Benali, "Accurate anisotropic fast marching for diffusion-based geodesic tractography," *Int. J. Biomed. Imaging.*, vol. 2008, p. 12, Jan. 2008.

[10] H. Li, Z. Xue, K. Cui, and S. T. C. Wong, "Diffusion tensor-based fast marching for modeling human brain connectivity network," *Comp. Med. Imag. Graph.*, vol. 35, no. 3, pp. 167–178, 2011.

[11] K. Yang, M. Li, Y. Liu, and C. Jiang, "Multi-points fast marching: A novel method for road extraction," in *Proc. 18th Int. Conf. Geoinformatics: GIScience in Change, Geoinformatics*, June 2010, pp. 1–5.

[12] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[13] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time Euclidean distance transform algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 5, pp. 529–533, 1995.

[14] N. Gagvani and D. Silver, "Parameter controlled skeletonization of three dimensional objects," Dept. Elect. and Comput. Eng., Rutgers Univ., Piscataway, NJ, Tech. Rep. CAIP-TR-216, 1997.

[15] S. Garrido, L. Moreno, D. Blanco, and M. I. Munoz, "Sensor-based global planning for mobile robot navigation," *Robotica*, vol. 25, no. 2, pp. 189–199, 2007.

[16] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Scituate, MA: Bradford, 2004.

[17] L. de Sanctis, S. Garrido, L. Moreno, and D. Blanco, "Outdoor motion planning using fast marching," in *Proc. CLAWAR*, Istanbul, Turkey, Sept. 2009.

[18] J. V. Gomez, S. Garrido, and L. Moreno, "Adaptive robot formations using fast marching square working under uncertainty conditions," in *IEEE Proc. Workshop Advanced Robotics Social Impacts*, Oct. 2011, pp. 68–71.

[19] S. Garrido, L. Moreno, and D. Blanco, "Exploration of 2D and 3D environments using Voronoi transform and Fast Marching method," *J. Intell. Robot. Syst.*, vol. 55, no. 1, pp. 55–80, 2009.