Fast Marching Methods in Path Planning

Alberto Valero-Gomez, Javier V. Gomez, Santiago Garrido, Luis Moreno

***Abstract*—This paper gives a detailed overview of fast marching methods for path planning. The paper recalls some of the methods developed by the authors of this work in the recent years and presents two new methods: the saturated fast marching square (FM2), and an heuristic modification called fast marching star (FM2*). The saturated variation of the existing FM2 provides safe paths which avoid unnecessary long trajectories to keep far from obstacles (like in the Voronoi Diagram computed trajectories). The FM2* reduces considerably the computation time. As a result the method provides not only a trajectory, but also an associated control speed for the robot at each point of the trajectory. This path is optimal in completion time.**

***Index Terms*—Path Planning, Mobile Robots, Fast Marching, Heuristic**

I. Introduction

The path planning is a well-known problem, with a well-understood mathematical basis, which has already lots of approaches which successfully provide good enough solutions, but always depending on the final application. The characteristics of the paths provided by the different existing algorithms change if the path planning is focused on video games artificial intelligence, mobile robots, UAVs, etc.

The path planning objective has changed since the first approaches were proposed. In the beginning the objective was to create an algorithm able to find a path from a initial point to a goal point ensuring completeness (the algorithm would find a path if it exists). Since this objective has been largely solved (i. e. Dijkstra algorithm) and the computational capacity has increased exponentially the objective has become eager: the current objective is to find the shortest path, or the fastest one, even maintaining safety constraints. These solutions are also expected to provide smooth, human-like paths.

There are many different path planning algorithms proposed. LaValle proposes a classification into 2 big groups depending on the way the information is discretized [1]: *Combinatorial Planning*, which constructs structures which capture all the information needed in path planning [2], [3] and *Sampling-Based Planning*, which incrementally search in the space for a solution using a collision detection algorithm [3]. In the first group the most widespread methods are those based in road maps which mainly consist on obtaining precalculated short paths from the map (road map) and creating the path by taking the sections of the road map needed (referencias). In the second group, there are very different options: rapidly exploring random trees (RRTs) [4] provide a fast solution based on creating randomly branches from a initial point, those branches which are collision-free are stored and new branches are created iteratively since the goal point is reached. Other option is to model the environment in a occupancy gridmap and apply search algorithms such Dijkstra or A* using each cell as nodes. Also, there are potential fields based approaches [5] in which treats the robot as a particle under the influence of a artificial potential field.

Alberto Valero-Gomez, Javier V. Gomez, Santiago Garrido and Luis Moreno are with the Robotics Laboratory of Universidad Carlos III de Madrid, Spain. alberto.valero@uc3m.es

This paper focuses on these potential field based algorithms (sampling-based algorithm). These methods are based on creating artificial potential fields from the sampled information through sensors and obtaining the path from the characteristics of these fields. The main problem of these methods are that they can have local minima which provokes the path to fall in those minima not being able to find a path even if it exists. The Fast Marching Method can be applied to create the potential fields and to obtain artificial local minima free fields, solving one of the most important drawback of these path planning methods. The authors have applied the Fast Marching method successfully combining in with a Voronoi diagram [6] or applying the Fast Marching method iteratively [7].

This document is organized as follows: in section II introduces the Fast Marching Method, its formulation, implementation and first approaches to path planning solving. In section IV the Fast Marching Square methods are outlined and a new variation is introduced: Fast Marching Square - Star Method. Next, in the section V the results of the new proposed method are shown in comparison with the Fast Marching Square Method. Finally, in section VI the conclusions are extracted and the future work is indicated.

II. The Fast Marching Method

The Fast Marching Method is a particular case of Level Set Methods, initially developed by Osher and Sethian [8], as an efficient computational numerical algorithm for tracking and modeling the motion of a physical wave interface (front) 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], [10], [11]. The interface can be a flat curve in 2D, or a surface in 3D (but the mathematical model can be generalized to n dimensions). The fast marching method calculates the time T that a wave needs to reach every point of the space. The wave can be originated from more than one point, each source point originates one wave. Source points have an associated time $T = 0$.

In the context of the Fast Marching 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 non-negative. 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]):

$$1 = F(x)|\nabla T(x)|$$

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

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

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

Because the front can only expand ($F > 0$), the arrival time T is single valued. Sethian proposed a discrete solution for the Eikonal Equation [12]. In 2D the area is discretized using a gridmap. We denotate i, j the row i and column j of the gridmap, that corresponds to a point $p(x_i, y_j)$ in the real world. The discretization of the gradient ∇T according to [8] drives to the following equation:

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

In [12] Sethian proposes a simpler but less accurate solution for Eq. 1, expressed as follows:

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

where

$$\begin{array}{l} D_{ij}^{-x} = \frac{T_{i,j} - T_{i-1,j}}{\triangle x} \\ D_{ij}^{+x} = \frac{T_{i+1,j} - T_{i,j}}{\triangle x} \\ D_{ij}^{-y} = \frac{T_{i,j} - T_{i,j-1}}{\triangle y} \\ D_{ij}^{+1} = \frac{T_{i,j+1} - T_{i,j}}{\triangle y} \end{array} \quad (3)$$

and $\triangle x$ and $\triangle y$ are the grid spacing in the x and y directions. Substituting Eq. 3 in Eq. 2 and making

$$\begin{array}{l} T = T_{i,j} \\ T_1 = min(T_{i-1,j}, T_{i+1,j}) \\ T_2 = min(T_{i,j-1}, T_{i,j+1}) \end{array} \quad (4)$$

we can rewrite the Eikonal Equation, for a discrete 2D space as:

$$max\left(\frac{T - T_1}{\triangle x}, 0\right)^2 + max\left(\frac{T - T_2}{\triangle y}, 0\right)^2 = \frac{1}{F_{i,j}^2} \quad (5)$$

As we are assuming that the speed of the front is positive ($F > 0$) T must be greater than $T_1$and T_2whenever the front wave has not already passed over the coordinates i, j. Consequently Eq. 5 can be solved as the following quadratic:

$$\left(\frac{T - T_1}{\triangle x}\right)^2 + \left(\frac{T - T_2}{\triangle y}\right)^2 = \frac{1}{F_{i,j}^2} \quad (6)$$

Whenever $T > T_1$and $T > T_2$ (we always take the greater value of T when solving Eq. 6). If $T < T_1$or $T < T_2$, from Eq. 5 the corresponding member is 0, and hence Eq. 5 is reduced to:

$$\left(\frac{T - T_1}{\triangle x}\right) = \frac{1}{F_{i,j}} \quad (7)$$

if T resulted to be smaller than $T_1$when solving 6, or

$$\left(\frac{T - T_2}{\triangle y}\right) = \frac{1}{F_{i,j}} \quad (8)$$

if T resulted to be smaller than $T_2$when solving 6.

It is important, for the understanding of this paper, to highlight a property of the waves 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 involve that a point has a T value lesser than a neighbor point which is nearer to the source, which is impossible, as this neighbor must have been reached by the wave before.

In the following section we are presenting in detail an algorithm for solving the Eikonal Equation over a gridmap.

A. Implementation

Eq. 5 can be solved iteratively over a gridmap. For doing so, the cells of the gridmap must be labeled of one of the following types:

- *Unknown*: Cells whose T value is not known yet (the wave front 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. These stages are described bellow.

a) Initialization.: The algorithm starts by setting $T = 0$ in the cell or cells that originate the wave. These cells are labeled as *frozen*. Afterward it labels all their Manhattan neighbors as *narrow band*, computing T (Eq. 5) for each of them.

b) Main Loop: In each iteration the algorithm will solve the Eikonal Equation (Eq. 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 increasing T value (first cells have lesser T values).

c) Finalization: When all the cells are *frozen* (the *narrow band* is empty) the algorithm finishes.

We can see the process in Figures II-A0c and II-A0c. In Fig. II-A0c the wave is originated from one point. In II-A0c there are two wave sources. Black points are *frozen*, and their T value will not change. Gray points are the *narrow band*, while white ones are *unknown*. As it can be appreciated the waves grow concentric to the source. In Fig. II-A0c they join, and the waves develop themselves growing together. The iterative process expands cells in the same order that the physical wave grows, as cells with less T are expanded first, that is, if two cells have a different arrival time, the cell that is reached before by the front wave is expanded first.

If we consider T as the third dimension over the z axis, the result of completing the wave expansion of Figures II-A0c and II-A0c results in Figures 3a and 3b respectively. As it

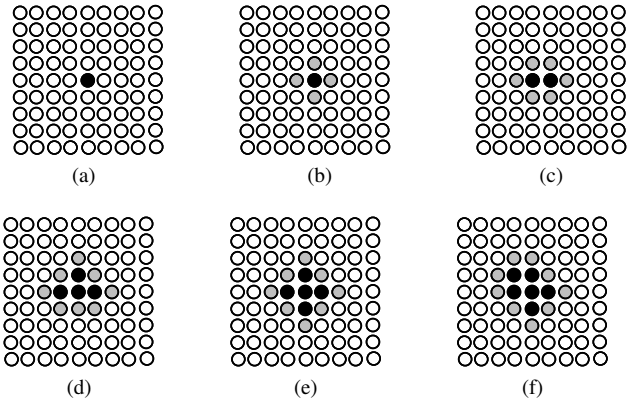


Figure 1. Iterative Wave Expansion with 1 Source Point

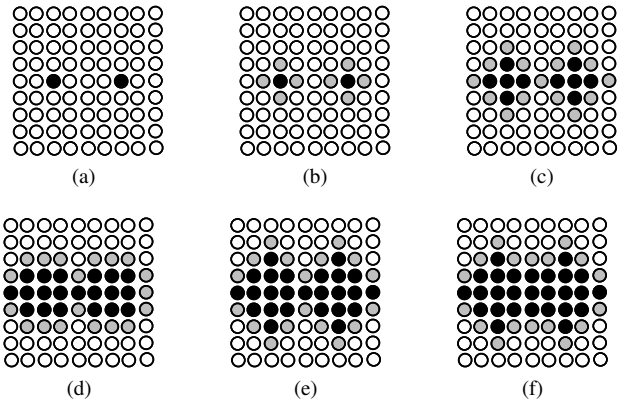
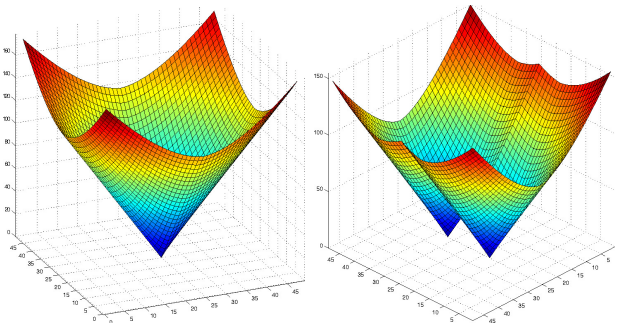


Figure 2. Iterative Wave Expansion with 2 Source Points

was supposed to happen, as $F > 0$, when we get far from the sources the time T required to reach the point is greater (higher on the z axis). It can be appreciated that with one single source, there is only one minima at the source. With more than one source we have a minima at each source with $T = 0$.



(a) One source point (b) Two source points

Figure 3. Fast Marching Method applied over a 50x50 Gridmap

The algorithm is shown in Alg. 1.

input : A gridmap G of size $m \times n$
input : The set of cells Ori where the wave is originated
output: The gridmap G with the T value set for all cells

Initialization;
foreach $g_{ij} \in Ori$ **do**
  $g_{ij}.T \leftarrow 0$;
  $g_{ij}.state \leftarrow$ FROZEN;
  foreach $g_{kl} \in g_{ij}.neighbours$ **do**
    if $g_{kl} =$ *FROZEN* **then** skip; **else**
      $g_{kl}.T \leftarrow solveEikonal(g_{kl})$;
      if $g_{kl}.state =$ *NARROW BAND* **then**
      $narrow\_band.update\_position(g\_kl)$;
      if $g_{kl}.state =$ *UNKNOWN* **then**
        $g_{kl}.state \leftarrow$ NARROW BAND;
        $narrow\_band.insert\_in\_position(g_{kl})$;
      end
    end
  end
  Iterations;
  while $narrow\_band$ *NOT EMPTY* **do**
    $g_{ij} \leftarrow narrow\_band.pop_first()$;
    foreach $g_{kl} \in g_{ij}.neighbours$ **do**
      if $g_{kl} =$ *FROZEN* **then** skip; **else**
        $g_{kl}.T \leftarrow solveEikonal(g_{kl})$;
        if $g_{kl}.state =$ *NARROW BAND* **then**
        $narrow\_band.update\_position(g\_kl)$;
        if $g_{kl}.state =$ *UNKNOWN* **then**
          $g_{kl}.state \leftarrow$ NARROW BAND;
          $narrow\_band.insert\_in\_position(g_{kl})$;
        end
      end
    end
  end
end

Algorithm 1: Fast Marching Algorithm

III. Fast Marching and Path Planning

Let us consider a binary gridmap, in which obstacles are valued as 0, and free space as 1. These values can be taken as the wave expansion speed F over the gridmap. At obstacles, wave expansion speed is 0, as the wave cannot go through obstacles, and on free space, wave expansion speed is constant and equals to 1. If we want to compute the path between two points p_0 and p_1 we could expand a wave from p_1 until it reaches p_0. Due to the wave expansion properties, the path that has followed the wave interface from the target to the source point will be always the shortest trajectory in time. As the wave expansion speed is constant, this trajectory is also the shortest solution in distance. The wave is originated from the target point, hence, the computed $T(x)$ field will have only one 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 complete.

Fig. 4 shows an example. We want to trace the shortest path from p_o to p_1. Fig. 4b shows the wave expansion in gray scale, the further the interface is from the target, the clearer the map becomes. Once the interface Γ has reached the initial point p_0 the algorithm stops expanding.

The resulting gridmap stores at any pixel the time T required by the front wave to reach that pixel. The isocurves join together all the points that have been passed through at the same instant of time (Fig. 4d). 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 has followed when expanding. The maximum gradient direction is computed applying the Sobel operator over the gridmap.

$$grad_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \star L$$

$$grad_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \star L$$

For tracing the path between p_o and p_1 we just need to follow the maximum gradient direction starting at p_0. The path is computed iteratively. $grad_{ix}$ and $grad_{iy}$ are computed at every point p_i. From p_i is computed p_{i+1} and successively until arriving to p_1. As $p_1$is located at the global minima it is always reached (whenever there is path).

$$mod_i = \sqrt{grad_{ix}^2 + grad_{iy}^2}$$
$$alpha_i = \arctan(\frac{grad_{iy}}{grad_{ix}})$$
$$p_{(i+1)x} = p_{ix} + mod_i \cdot \cos(alpha_i)$$
$$p_{(i+1)y} = p_{iy} + mod_i \cdot \sin(alpha_i)$$

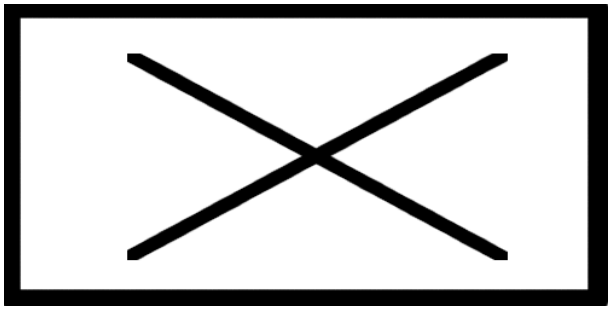
In Figs. 4c and 4d we appreciate that the created field has just one global minima at p_1, and hence the solution is unique and complete.

A. FM over Voronoi Diagram

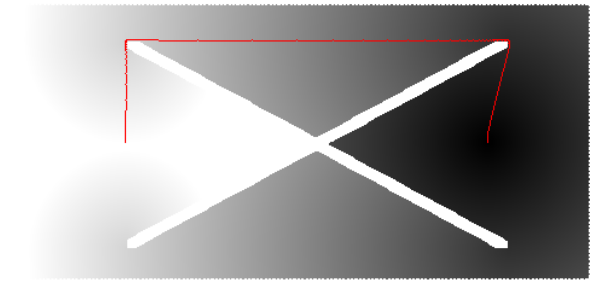
The path calculated previously, even if the shortest in length, might not be a safe path as it gets close to the obstacles. This aspect also causes that it is not the shortest in time, as the robot has to go slowly when it is close to the obstacles (in order to avoid collisions or risky movements). The usual solution is to expand obstacles before calculating the path. Nonetheless, when a robot moves through a door, we would like to pass through the middle and not touching the wall, the same holds for corners, etc. A variation method consists of calculating before had the Voronoi Diagram and then compute the FM method over it. As the Voronoi Diagram stays far from obstacles we providing safe paths.

The Voronoi Diagram is used as a way to obtain a roadmap of the map. Then the Fast Marching method is used to search the path over the Voronoi Diagram. The main steps of this method are the followings:

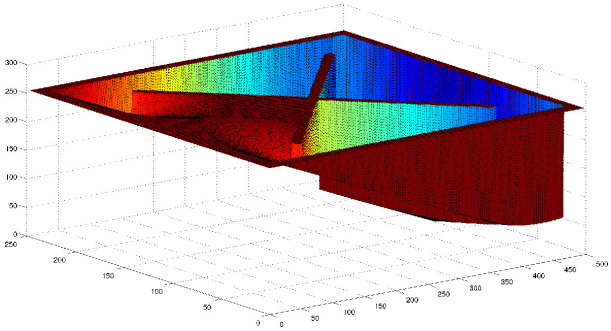
1) *Map preprocessing.* The map is turned into a binary grid map, where the obstacles are black (value 0) and the clear space is white (value 1). The obstacles and walls are enlarged in order to ensure the robot will neither collide with walls and obstacles nor accept passages narrower than its size. Also, an averaging filter is applied with the objective of erase small *hairs* of the diagram.



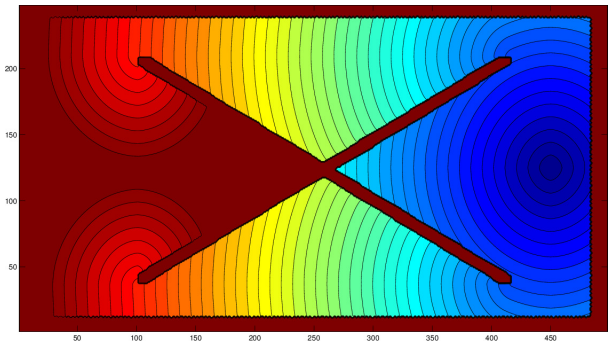
(a) Original Map



(b) Path and Wave Expansion in Gray Scale



(c) Fast Marching Field. One Unique Global Minimum



(d) Fast Marching Field. Isometric Curves

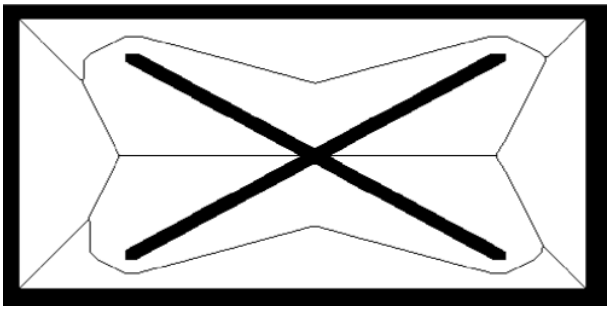
Figure 4. Fast Marching and Path Planning

2) *Voronoi Diagram.* The diagram is obtained using morphological image processing techniques (concretely the methods proposed by Breu in 2D [13]) as shown in Fig. 5a. A dilatation of the diagram is done getting a thickened Voronoi Diagram (Fig. 5b) wherein to apply the Fast Marching method.
3) *Fast Marching.* The Fast Marching method is used to expand a wave from the target to the initial point on

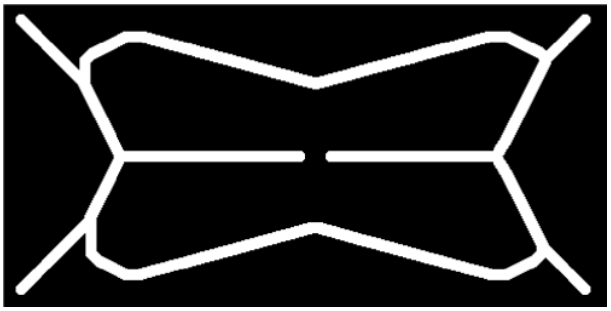
the thickened voronoi diagram. As a result we have a potential field over the thickened voronoi diagram (Fig. 5c).

4) *Path extraction.* The gradient method is applied to the previously obtained potential from the current position of the robot and the goal point as the final point as shown in Fig. 5d.

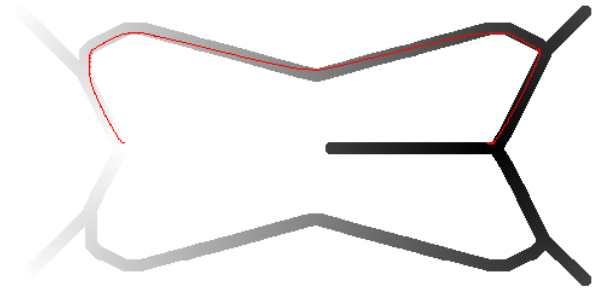
Fig.5 shows the main aspects of this algorithm when it is applied to the map shown in Fig.4a.



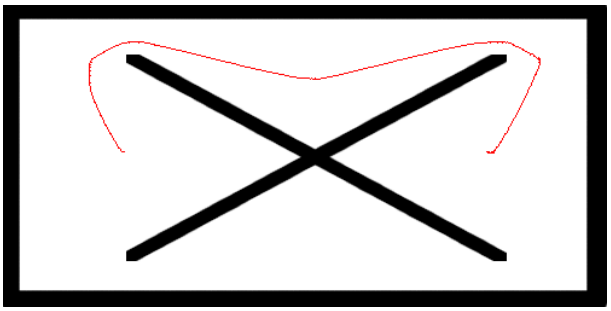
(a) Voronoi Diagram Over the Initial Map



(b) Thickened Voronoi Diagram



(c) Fast Marching and Path over Voronoi Diagram



(d) Path

Figure 5. Fast Marching over the Voronoi Diagram

IV. FM-Square and FM-Square-Star

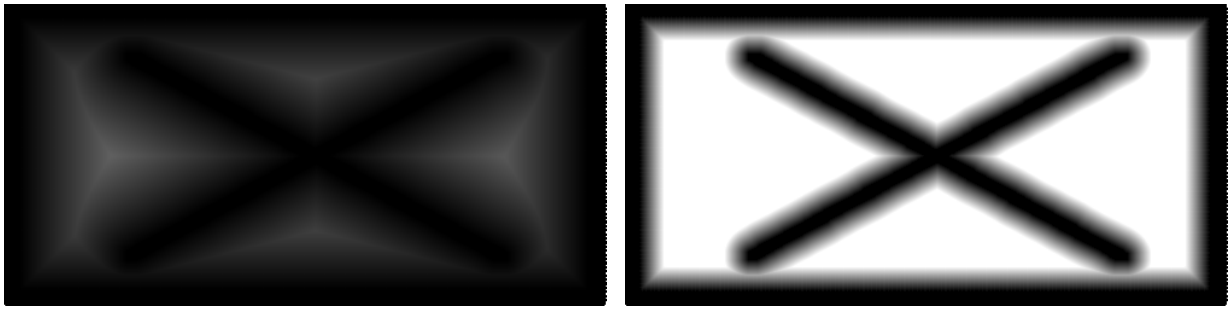
As an alternative to the simple Fast Marching Method and the Voronoi Diagram Method for computing trajectories, we are introducing three different methods based exclusively on Fast Marching: 1) the Fast Marching Square (FM2), 2) the saturated FM2 variation, and 3) an heuristic modification of it, called Fast Marching Square Star (FM2*) method. We will show how the FM2* reduces the computation time wrt. the FM2 while providing the same trajectory.

A. FM2: Fast Marching Square

If we take an evidence gridmap in which obstacles are labeled as 0 and free space as 1. We can apply the Fast Marching Method to this map being all the obstacles a wave source. In the previous section, there was just one wave source, at the target point. Here all the obstacles are a source of the wave, and hence, several waves are being expanded at the same time. The map resulting of applying this wave expansion to the map depicted in Fig. 4a can be seen in Fig. 6a. We have called the resulting map *fast marching gridmap* (FMGridMap); it represents a potential field of the original map. As pixels get far from the obstacles, the computed T value is greater. This map can be seen as a *slowness map*. If we consider the T value as a measure proportional to the maximum allowed speed of the robot at each point, we can appreciate that speeds are lower when the pixel is close to the obstacles, and greater far from them. In fact, 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 have then the *slowness map*, that provides a safe speed for the robot at any point of the environment. In Fig. 6c we can appreciate the speeds profile. In the image is clear that speeds become greater far from the obstacles.

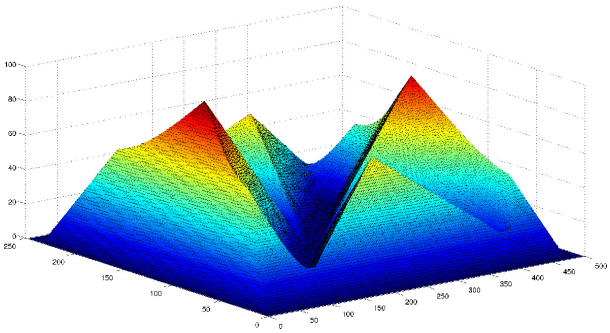
We could calculate now the path as we did in Section III but instead of taking a constant value for the expansion speed F, we use the speed given by the *slowness map*. Now, if we expand a wave from one point of the gridmap, considering that the expansion speed $F(x, y) = T(x, y)$, being $F(x, y)$ the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

Previously to the FM2 method, the authors of this work developed a similar method called Voronoi Fast Marching (VFM) [14]. The intuition of this method is the same as FM2: get a first potential (slowness or viscosity potential) and propagate a wave over this potential creating a second potential in which the path is obtained. 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.

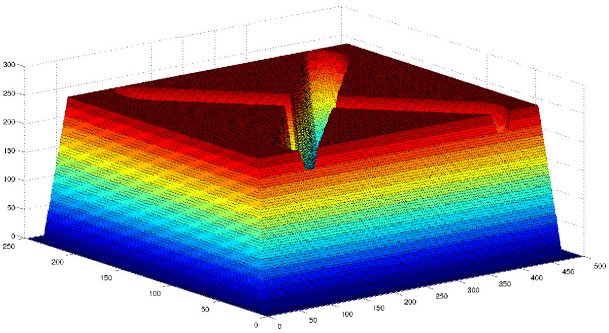


(a) FMGridMap

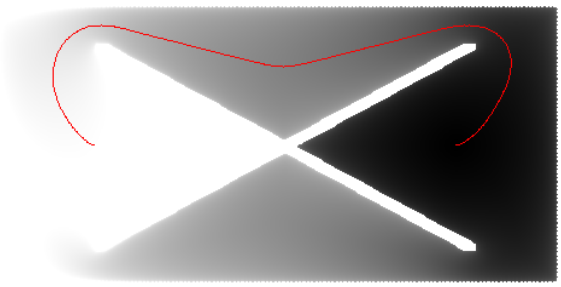
(b) Saturated FMGridMap



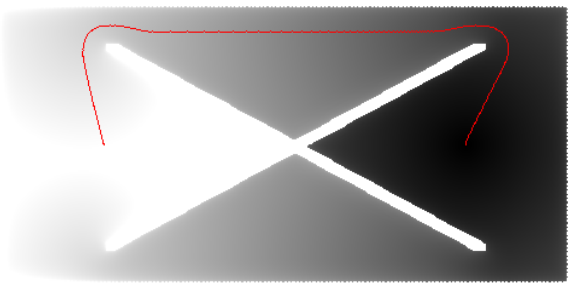
(c) 3D Representation



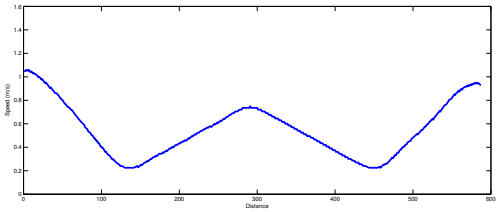
(d) 3D Representation



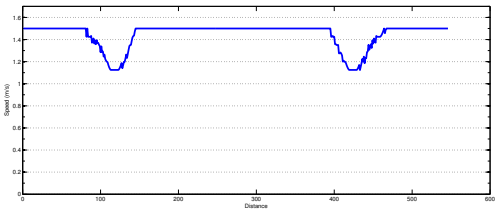
(e) Second Fast Marching and Computed Path



(f) Second Fast Marching and Path



(g) Speed Profile



(h) Speed Profile

Figure 6. FM2 and Saturated FM2

B. FM2: The Saturated Variation

In Fig. 6e we appreciate that the computed path is not the logical/optimal trajectory we would expect (and the same holds for Fig 5d). The FM2 computed path, as it has been presented, tries to keep the trajectory far from obstacles. This computed trajectory is similar to the path computed with the Voronoi diagram [15] and presented above.

But there are environments in which there is no need to follow the farther trajectory from obstacles, as distance may be safe enough to navigate. To solve this we implemented a saturated variation of the FMGridMap. When the first fast marching has been computed, the FMGridMap is first scaled and then saturated.

The scaling of the map is made 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.

At the end the map is saturated to the maximum allowed speed. After this scaling and saturation process the slowness map provides the maximum speed for all the points that are farther

than the safe distance from the obstacles and the control speed varying form 0 (at obstacles) to the maximum speed (at safe distance) for the rest of points.

Fig. 6b shows the saturated variation of Fig. 6e with the new computed trajectory. We can appreciate that now the path is as expected (Fig. 6b. In Fig. 6d the speed profile with the saturated area is shown.

C. The Heuristic FM2: Fast Marching Square - Star

Consider the map of Fig. 7a in which a trajectory among two points on a free space must be computed. Fig. 7b shows the FM2 wave expansion originated from the target point. As it can be appreciated the wave grows concentrically around the target point until it reaches the initial point. The FM2* method is an extension of the FM2 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 FM2* algorithm works in exactly the same way as the FM2 algorithm. The only difference is the function used to sort the *narrow band* queue. The FM2 sorts the *narrow band* cells in increasing T order, so that in each iteration, first element in the queue (lowest T) becomes *frozen* and it is expanded. In FM2* the algorithm uses the cost-to-come T, which is known, and the optimal cost-to-go, that is, the minimum time the robot would employ 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. In fact, if we take the optimal cost-to-go as 0, the FM2* algorithm is equivalent to the FM2 algorithm. If the estimation is greater than the real cost-to-go, the FM2* algorithm could take more computational steps than the FM2 to find the path and the path could be not the shortest.

In this problem, the optimal cost-to-go (optimal time to reach the target) would happen 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 be always greater than this computed value. So, the *narrow band* queue is ordered according to the value T^*.

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

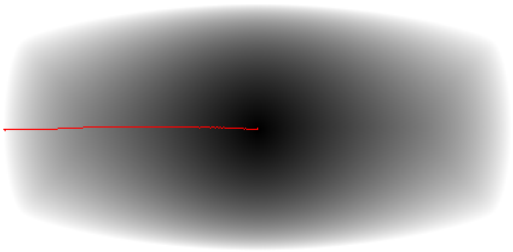
These two methods are analogous to Djikstra and A* in path-finding over graphs [3].

The wave expansion computed with FM2 and shown in Fig. 7b, computed with FM2* is shown in Fig. 7c.

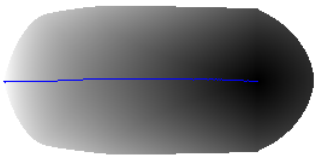
V. Results

Fig. 8 shows the resulting path from applying the FM2 and FM2* over a realistic map of 50 x 18 square meters. The gridmap has 500x180 pixels. Trajectories start by the center of the map and go the right and to left of the gridmap respectively. Fig. 8b shows the slowness map, computed with saturation, the maximum speed is $1.5m/s$ and the safe distance 2 meters, that is, 2 meters away from the obstacles, the control speed will be $1.5m/s$. Figs. 8c, 8e show the computed trajectory and the wave expansion field resulting of applying FM2. Figs. 8d and 8f show the computed trajectory and the wave expansion field resulting of applying FM2*. It can be appreciated that the three paths are the same, but the FM2* heuristic reduces the number of cells expanded. The computational times are:

(a) Original Grid Map



(b) Wave Expansion and Path with FM2



(c) Wave Expansion and Path with FM2*

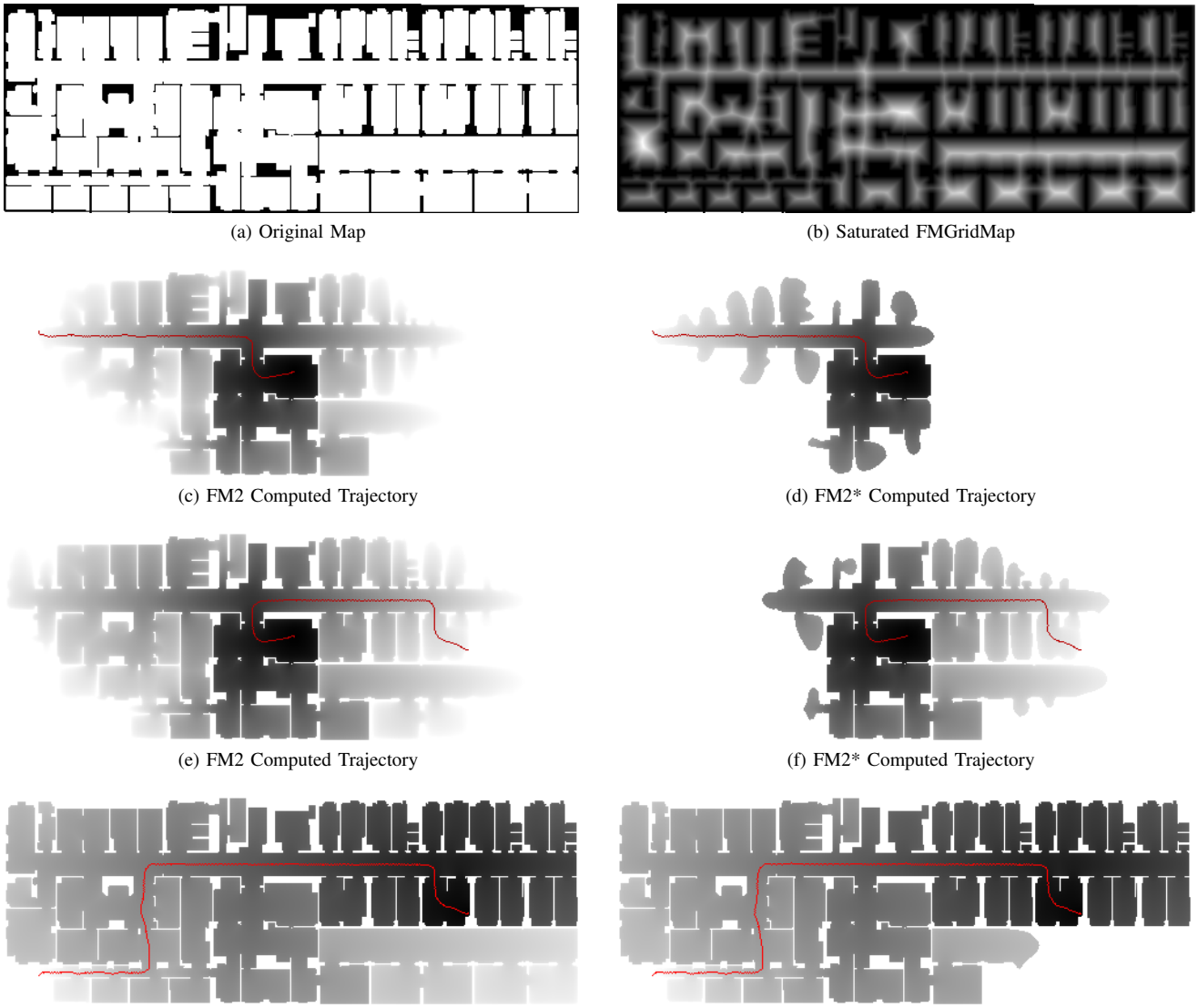
Figure 7. Comparison between FM2 and FM2*

	Computation Time (seconds)	
Trajectories	FM2	FM2*
Figs. 8c and 8d	0.130669	0.032623
Figs. 8e and 8f	0.130914	0.031855
Figs. 8g and 8h	0.189093	0.147036

The third computed path has to expand almost all the map, and hence computational times do not diverge too much. In the most favorable case of the firs two trajectories, the time gaining is more than four times.

VI. Conclusions and Further Work

In this paper we have presented the mathematical foundations of the Fast Marching Method (FM) developed by Osher



(a) Original Map

(b) Saturated FMGridMap

(c) FM2 Computed Trajectory

(d) FM2* Computed Trajectory

(e) FM2 Computed Trajectory

(f) FM2* Computed Trajectory

(g) FM2 Computed Trajectory

(h) FM2* Computed Trajectory

Figure 8. FM2 vs. FM2*

and Sethian [12]. We have presented the algorithm that we have implemented for applying the FM over a gridmap. In Section III it has been presented how the FM method can be applied to compute the visibility path among two points in a gridmap. The limitations of this methodology lays in the fact that it provides the shortest path in distance, which leads to risk due to its closeness to the obstacles. In Section IV the Fast Marching Square Method (FM2) has been explained in detail. The FM2 and all its variations have been developed by the authors of this work. The FM2 computes two wave expansions over the gridmap. The first expansion computes the FMGridMap, a slowness map that provides the maximum allowed speed of the robot at each point of the map. This slowness map is used to compute the second expansion, form the target point to the initial point. As a result of the second expansion the trajectory is computed (using the maximum gradient direction). This trajectory does not only provides a way point but also the control speed at each point of the path, due to this, this trajectory is safe and optimal in time. The FM2 computes path that tends to go the far from obstacles, but this situation is not always necessary. A variation to the FM2 for avoiding this problem was presented, called the saturated FM2. To reduce the computation time of the FM2, a heuristic FM2, called FM2-Star (FM2*) was presented. In the experimental analysis it is shown that the computation time is reduced up to 4 times regarding the FM2, while providing the same trajectory.

The future work focuses on expand the FM2* to more dimensions. Previous Fast Marching Methods as FM2 or VFM have been applied successfully to higher level problems such as outdoor path planning [16], robot formations motion planning [17], exploration and SLAM [18], [19] but the computational complexity of these method limited them to 2D problems (although different options have been already suggested to decrease the computational complexity when expanding to 3 or more dimensions). Now, since the FM2* is

faster those high-level problems can be approached expecting computation times withing the problem solving framework. The results obtained encourage to use the FM2* also in swarm robotics or even in dynamic environments.

REFERENCES

[1] S. LaValle, "Motion planning," *Robotics Automation Magazine, IEEE*, 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. Springer-Verlag, 2008.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [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 *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, 2000, pp. 995–1001 vol.2.

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

[6] S. Garrido, L. Moreno, and D. Blanco, "Voronoi diagram and fast marching applied to path planning," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, may 2006, pp. 3049–3054.

[7] S. Garrido, L. Moreno, M. Abderrahim, and D. Blanco, "Fm 2: a real-time sensor-based feedback controller for mobile robots," *I. J. Robotics and Automation*, vol. 24, no. 1, 2009.

[8] S. Osher and J. A. Sethian, "Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations," *JOURNAL OF COMPUTATIONAL PHYSICS*, 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. Biomedical Imaging*, vol. 2008, 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. and 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 *The 18th International Conference on Geoinformatics: GIScience in Change, Geoinformatics 2010, Peking University, Beijing, China, June, 18-20, 2010*, 2010, pp. 1–5.

[12] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[13] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 529–533, 1995.

[14] S. Garrido, L. Moreno, D. Blanco, and M. l. Munoz, "Sensor-based global planning for mobile robot navigation," *Robotica*, vol. 25, pp. 189–199, 2007.

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

[16] L. de Sanctis, S. Garrido, L. Moreno, and D. Blanco, "Outdoor motion planning using fast marching," in *CLAWAR2009. Istambul, Turkey.*, September 2009.

[17] J. V. Gomez, S. Garrido, and L. Moreno, "Adaptive robot formations using fast marching square working under uncertainty conditions," in *IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO2011), San Francisco, CA, USA.*, October 2011.

[18] S. Garrido, L. Moreno, and D. Blanco, "Exploration of 2d and 3d environments using voronoi transform and fast marching method," *Journal of Intelligent and Robotic Systems*, vol. 55, no. 1, pp. 55–80, 2009.

[19] ——, "Exploration and mapping using the vfm motion planner," *IEEE T. Instrumentation and Measurement*, vol. 58, no. 8, pp. 2880–2892, 2009.