On Path Planning: Adaptation to the Environment using Fast Marching

Javier V. Gómez*, César Arismendi†, Santiago Garrido† and Luis Moreno †
Systems and Automation Department
Carlos III University of Madrid, Leganés, 28914, Spain
* Email: jvgomez@pa.uc3m.es
† Email: carismen, sgarrido, moreno@ing.uc3m.es

***Abstract*—This paper presents a novel methodology for fast path planning based on an offline predefined skeleton of the environment by means of the Fast Marching Square method. The FM^2 skeleton concept is introduced whose intuition is a set which contains the shortest paths (in terms of time) of a given environment. This way, the path planning algorithm is adapted to the environment where a robot will navigate. An algorithm to obtain this skeleton is detailed in order to be able to use it for path planning purposes. When planning paths over the skeleton, the results show that the time reduction is about 50% while the characteristics of the FM^2 method remain: completeness, safeness, smoothness and absence of local minima. This paper also shows that the proposed method performs well in more than 2 dimensions.**

I. Introduction

Motion planning problems have been an active research area, playing a fundamental role in many applications involving robotics, aerospace, automotive, defense, medical equipments, nuclear power plant construction and decommissioning among others. An important goal for path planning methods is to be able to generate the path considering a wide range of constraints. Most times its intention is to work out the optimal safe trajectory in the proper time.

The Fast Marching Method is a complete algorithm which has been proved to work out the path generation problem with outstanding performance, obtaining the fastest path between two points [1]. Moreover, it has been complemented employing Voronoi diagrams and similar techniques.

The Voronoi concept has been extensively used in diverse fields for over three centuries. In his *Traité de la Lumiére* published in 1644, Descartes uses diagrams similar to Voronoi to show the disposition of matter in the solar system and its environment. Voronoi diagrams have been appearing since 1970, as published in the surveys by Aurenhammer [2] and Okabe [3]. Both present various algorithms, applications, and generalizations of Voronoi diagrams. The Voronoi diagram and the Fast Marching method have been already mixed in different ways: planning paths over the Voronoi diagram [4] and the so-called Voronoi Fast Marching (VFM) [5]. A similar approach is presented in [6] where the Fast Marching Square (FM^2) method is detailed.

The FM^2 method is a method that uses an algorithm somewhat similar to those based on potential fields, but does not exhibit the typical problems of the potential-field-based methods [7]: (1) Trapping due to local minima, (2) No passage between closely spaced obstacles, and (3) Limit cycles. The algorithm uses a two-part approach. The first part builds a "velocities map" of the environment that represents the admissible velocity at discrete cells in the workspace, better known as slowness map. The second part computes a smooth, safe and time efficient path using 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 [6].

Even thought the generated path by FM^2 is safe, it is not always necessary to displace as far as possible from the walls and obstacles, as distance may be safe enough to navigate. Considering this reason a further improvement is made to FM^2: a saturated variation of slowness map is implemented; when the first path of the algorithm is completed, the slowness map is first scaled and then saturated giving more freedom of mobility to the robot.

In this paper, we get a fast path planning method based on FM^2. We present an algorithm which creates random paths for a given map using the FM^2 path planning method. The main idea of the proposed algorithm is to preprocess an already know map in order to get the *common* possible paths for that map, combining them when a new path is required. The key idea is that in a real robot application, the environment will be known (or obtained through SLAM). This environment can be dynamic but its main parts will remain, such as floor, doors, walls, and so on. Then the path planning method *adapts* to the environment creating some kind of road map which allows the FM^2 to spend the shortest possible time in path planning.

This paper is structured as follows. In Section II, we will briefly describe the Fast Marching Method, and review some related concepts. Furthermore, path planning application using the FM^2 will also be presented. In Section III, we introduce a novel concept called Fast Marching Square Skeleton. Path planning over the FM^2 Skeleton is presented in Section IV. We extend and generalize this framework to d-dimensions in Section V, and Section VI concludes this paper.

II. Fast Marching and Path Planning

The Fast Marching method is a level set method, proposed by Osher and Sethian [8], [1] to solve the Eikonal equation

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

which describes the motion of a frontwave propagating in a non-homogeneus media where the speed F does not have to be the same everywhere, but it is always non-negative. $T(x)$ is called the arrival function which computes the time the wave will take to reach a point x.

The $T(x)$ function is originated by a wave that grows from one single point (global minimum) at the source. As $F > 0$ the wave only grows, and hence, points farther from the source have greater T. A local minimum would involve that a point has a lesser T value than a neighbour point which is closer to the source, which is impossible, as this neighbour must have been reached by the wave before.

A. Application to Path Planning

The frontwave expansion speed F can be directly assigned from the values of the environment modeled in a gridmap, where for each cell 0 means obstacles and collision-free space is labeled as 1. To obtain a path from a point p to a point q a wave is expanded from q until it reaches p. Due to wave expansion properties, the wavefront will follow the shortest time path between p and q. Even more, considering the propagation speed as constant, the trajectory will also be the shortest in terms of distance. The consequence is that the T function will be local-minima-free, with one global minimum at the goal point. Applying gradient descent over the T function from the goal point, the initial point will be reached. Fast Marching ensures that the path obtained is unique and complete.

Figure 1 shows the result of applying FM to find a trajectory. The calculated path, although it is the shortest in length, runs too close to obstacles so it might not be a safe path. This causes the robot to go slowly when it is close to obstacles in order to avoid collisions. Therefore the path might also not be the shortest in time. In the following paragraphs we are introducing two other different methods based on Fast Marching which solve this problem: Fast Marching over the Voronoi Diagram and Fast Marching Square (FM^2).

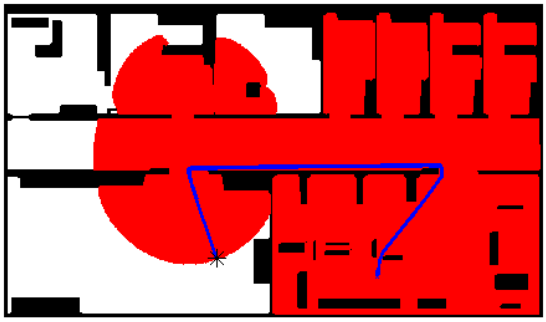

Fig. 1. Binary map and the path obtained using Fast Marching. The frontwave is also drawed (red).

B. Path Planning over the Voronoi Diagram

Since the Voronoi Diagram concept is simple and intuitive, it can be applied to a huge number of applications. Path planning is one of those. 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 which belong to the frontier among regions.

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 decreasing the time spent in the search [4]. The main property of this diagram is that it provides the safest possible path (in terms of distance to obstacles) between two points due to its definition. Results are shown in figure 2.

C. FM2 Path Planning Method

In the previous sections, 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 [6]. As pixels get far from the obstacles, the computed T value is greater. This generates a new map that can be seen as a *slowness map*. If we consider the T value as a proportional measure to the maximum allowed speed of the robot at each point, we can appreciate that speeds are lower when a pixel is close to 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 to the obstacles.

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 slowness map at x, y, we will have that the expansion speed depends on the position. As the slowness map provides the

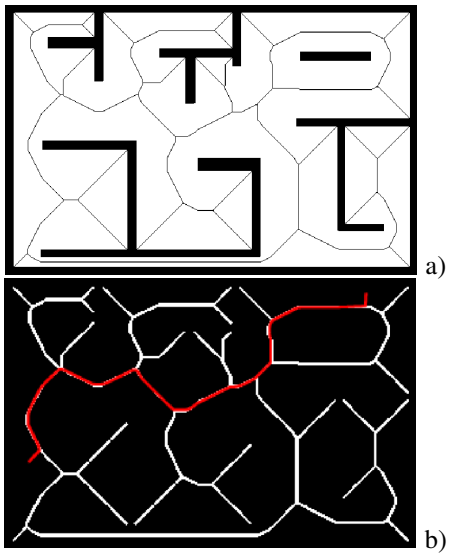

Fig. 2. a) Binary map and its Voronoi Diagram. b) Path obtained using this diagram.

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.

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, a saturated variation of the slowness map can be employed. 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, that is the distance from the closest obstacle at which the maximum speed can be reached.

Figure 3 shows the performance of this algorithm.

III. FM2 Skeleton

As shown in the previous section, the Fast Marching method is very versatile in path planning tasks. The main idea of this paper is to merge the low computational complexity of the method shown in II-B and the FM^2 method properties: safeness, smoothness, completeness, reliability and the absence of local minima, which is the most important characteristic.

To achieve this goal a new concept is included, which we have called FM^2 *skeleton*: a set of random paths, obtained by means of FM^2, distributed throughout the map. This set creates a collision-free, smooth roadmap.

A. Creation of a FM2 skeleton

The objective is to create a skeleton which have *branches* in every zone of the map. The next points outline an algorithm to obtain a FM^2 skeleton which satisfies this requirement:

- Model the environment as an occupancy grid map where the walls and obstacles are modeled with 0 (black) and the clear space with 1 (white).
- Distribute uniformly an n number of random points throughout the whole map. Erase those points which fall in zones where there are obstacles or walls.
- Include *characteristic* points of the environment. Those points where the robot commonly operates such as doors, light switches, furniture, among others.

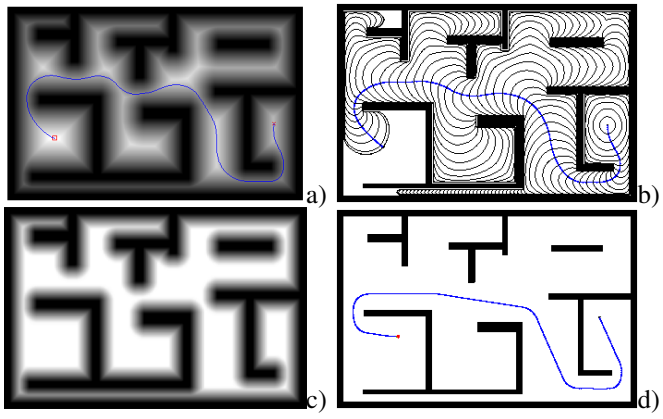

Fig. 3. a) Slowness map and the path obtained with FM^2. b) Wavefronts at each iteration. c) Saturated slowness map. d) Path obtained with the saturated version of FM^2.

Then, a loop begins whose steps are the following:

1) Select a point i.
2) Search a point j which Euclidean distance to the point i is higher than a value d_{min}. If no point is found, then select the farthest one.
3) Compute the path between the two points i,j according to the FM^2 method (specifically, we use the saturated version of the FM^2, at level sat).
4) Store the path with the other paths calculated in a binary map, called $\mathbf{W_p}$.

The loop ends when there are not any more couple points to obtain more paths. The resulting $\mathbf{W_p}$ can be considered as a FM^2 skeleton. The figure 4 a) shows the random points distribution and 4 a) displays the skeleton obtained for those points. As one can see, all the rooms of the environment are connected with smooth branches.

IV. Path Planning over the FM2 Skeleton

In section II-B the Fast Marching method has been applied over the Voronoi Diagram. Here, the same concept is going to be applied but using the FM^2 skeleton. To do this, several additional steps are needed in order to adapt the FM^2 skeleton:

- Dilate (with image processing techniques) the $\mathbf{W_p}$ map to obtain a thickened skeleton, ensuring the continuity of the skeleton.
- Compute the maximum between $\mathbf{W_p}$ and a value g_{min} close to 0 but always positive. This allows to expand the FM frontwave out of the skeleton if necessary.

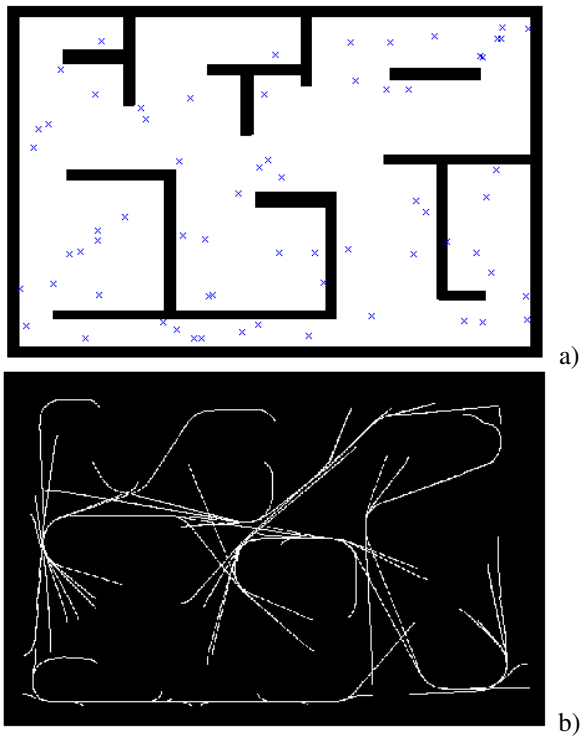

Fig. 4. a) Initial binary map with the set of n points randomly chosen but uniformly distributed. b) $\mathbf{W_p}$ map.

- The final map $\mathbf{W_{skeleton}}$ is calculated including the walls and obstacles (as black values) to the skeleton map $\mathbf{W_p}$.

These three steps can be summarized in the following formula:

$$\mathbf{W_{skeleton}} = min\left(\mathbf{W_0}, max(g_{min}, \mathbf{W_p} \oplus \mathbf{SE})\right) \quad (2)$$

where $\mathbf{W_0}$ is the initial binary map and the symbol $\oplus$ represents the dilation operation with the structuring element $\mathbf{SE}$, which shape can be a disk or square (in 2D) and its size depends on how thick the skeleton is wanted to be (large $\mathbf{SE}$ means more area covered by the skeleton but less time reduction when planning).

Then, the result is a map $\mathbf{W_{skeleton}}$ in which the skeleton has the highest gray level (1) and the rest of the free space have a low gray value (g_{min}). The reason to do this is that the initial and final points are not restricted to fall into the FM^2 skeleton when searching a path. If the points are not in it, the robot will take the shortest path to return into the skeleton where the wave expansion is much faster due to its higher gray level in the $\mathbf{W_{skeleton}}$ map.

Finally, to obtain a path over the FM^2 skeleton it is enough to apply the base FM method, expanding a wave from the goal point until it reaches the initial point. This will provide a funnel-shaped potential which represents the time the wave takes to expand. The last step is then to apply gradient descent over this potential to obtain the path.

This method can be employed as an offline unsupervised learning algorithm, where the robot is able to *predict* which paths are going to be executed more often. Thus precalculating and merging those paths in a skeleton form can be interpreted as a training process where the path planning system evolves to adapt to the environment constrains.

Figure 5 shows the final performance of the proposed algorithm. Although the generation of the diagram is not deterministic, the algorithm appears to predict where are the main zones of the map (corridors between rooms) and the branches of the skeleton go into each room. The paths provided by our method could be sometimes a little worse (a bit longer, with not very smooth curves) but the time reduction could be worthy in most of the applications. For this particular case, when simulating in a 628x412 pixels map the time elapsed with the proposed algorithm is 0.16 seconds, while it took 0.40 seconds to the standard FM^2 method. The generation of the skeleton (using the parameters described in the next section) took 13.24 seconds. In the worst case, the proposed method will last at most the same time as the standard FM^2 when the required path is quite new.

A. Setting the parameters

The proposed algorithm has a set of parameters which can change significantly the skeleton obtained. Following, how this parameters influence is described:

- *Number of points, n*: An equation has been experimentally obtained which assures a balanced distribution of points (in 2D):

$$n = \alpha(\sqrt{x} + \sqrt{y}) \quad (3)$$

where x and y are the dimensions of the map (columns and rows respectively) and α is a factor manually chosen.
- *Minimum distance between points, d_{min}*: Our experiments point out that a good equation d_{min} can be found by:

$$d_{min} = \beta\sqrt{(x^2 + y^2)} \quad (4)$$

where β is another factor to be set manually.
- *Minimum gray level, g_{min}*: Between 0 and 1.
- *Saturation gray level, sat*: Between 0 and 1.

Table I shows the parameters configuration for many experiments, including a time comparison where t_s, t_{pp} and

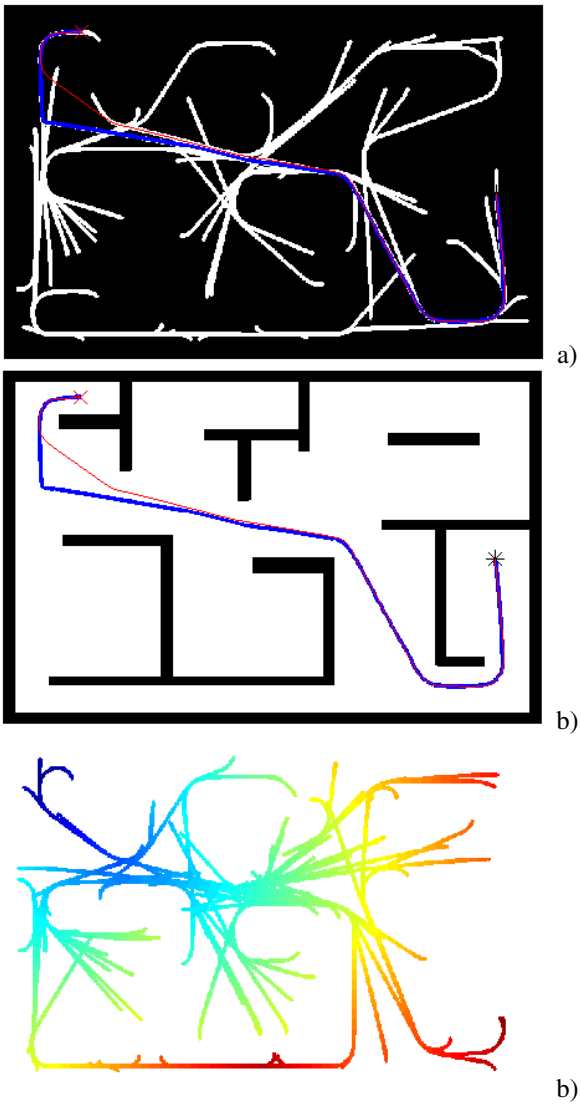

Fig. 5. a) Comparison of the paths obtained with the saturated variation of the FM^2 method (in red) and the proposed method (in blue) shown over the thickened skeleton. b) Same comparison but shown over the initial map. c) Potential obtained when propagating the wave through other skeleton of the same map. Blue means getting closer to the global minimum and red means that the time is increasing.

t_{FM^2} are the times elapsed when creating the initial skeleton, when planning over the skeleton and when planning with the standard FM^2 method, respectively. Also, figure 6 plots the results obtained in these experiments. The α and β values have been chosen experimentally with the objective of achieve enough points and with enough connectivity among them. Note that if α is too small the skeleton will have not enough branches to cover all the map. However, if it is set too large the skeleton will cover a wide area and the time reduction will not be that important. In the case of β, is a small value is chosen the skeleton will not have enough connectivity. Note that in the experiment (6) the environment is different. In this case the map is twice the size of the other experiments' map. The α has to change this is new map has more rooms, and with $\alpha = 2$ the probability of having a room without an skeleton branch is high.

The results outline that the time reduction for 2D planning use to be around 50% (depending on the parameters) obtaining a path as smooth and safe as with the base FM^2 method. The time consuming by the skeleton generation depends mostly on the parameters given to the algorithm but this is not critical since this process can be executed offline.

V. Extension to d-Dimensions

Many mobile robotics applications can be approximated by a 2-dimensional problem. But other ones simply cannot. The complexity of mobile manipulators or UAVs control does not accept bi-dimensional solutions. Therefore, most of the algorithms proposed which work very well in 2 dimensions often fail when increasing the dimensionality (due to the computational complexity).

Although it has been proved that the complexity of the Fast Marching method is $O(n)$ [9], it suffers the same problem when expanding to more than 2 dimensions, since the computational cost increases exponentially with the dimension of the problem. The proposed method in this paper assumes that the skeleton generation is done offline, and the path planning has to be done online. The fact of creating a skeleton by using FM^2 allows to search paths within the skeleton in d dimensions. Thus, the problem can be reduced to an *unidimensional* one, since the frontwave propagation is being guided by a *tube*.

To give a better intuition of this fact, we use the simile proposed by Greene to justify the string's theory [10]: let us suppose an ant moving in a cable. This cable exists in 3 dimensions but the ant only can move forwards/backwards and clockwise/counterclockwise around the cable. The movement of the ant can be perfectly defined with only two values (dimensions). Moreover, if we put the ant inside the cable, it can only move forwards or backwards (because moving clockwise or counterclockwise changes nothing when it is inside the cable) so the movement in 3 dimensions is restricted to only one degree of freedom. Also, if we look at a cable from far away it appears to be unidimensional (a line).

If we apply this concept to path planning, the frontwave is expanded trough the d-dimensional skeleton the dimensionality can be considered as $d \simeq 1$. It is true that inside the tube the dimension is still d, but the *volume* of the tube is negligible in comparison with the volume of the rest of the space ($d = 3$) or hyperspace ($d > 3$).

The algorithm to employ is exactly the same as proposed in sections III-A and IV. It is only necessary to adapt the parameters described in section IV-A. Figure 7 shows an example in 3 dimensions, where the proposed method took 0.22 seconds, while the time elapsed with the standard FM^2 method was 0.76 seconds. The skeleton was generated in 15.12 seconds.

VI. Conclusions and Future Work

This paper presents a novel method for fast path planning without losing the safeness or smoothness in the obtained paths in comparison with the standard FM^2 path planning method. Also, the simulations carried out with Matlab show an important time reduction when calculating the path: 50% or even more.

It has been shown how the different parameters of the algorithm influence the skeleton and the time taken when calculating new paths.

Lastly, it has been proved that it is possible to extend the algorithm to more than 2 dimensions. Thus, the proposed algorithms is applicable not only in mobile robot applications but manipulation and UAVs among others.

We are aware that the algorithm has not been tested in real robot. However, from our point of view, the algorithm only influences on the path planning computation time, hence the

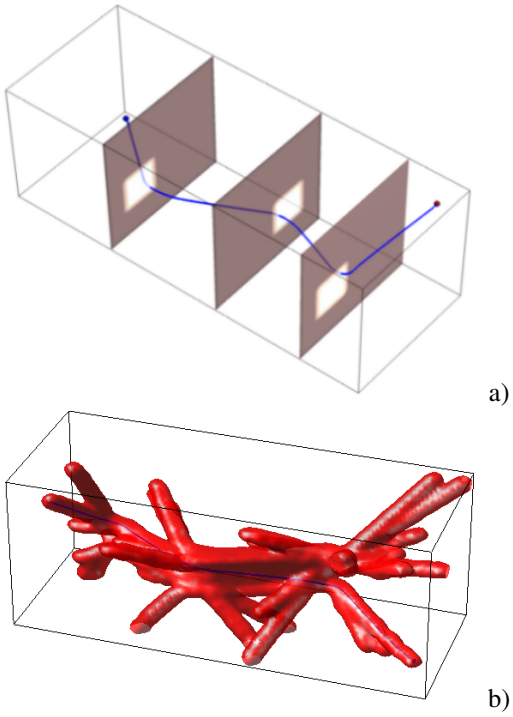

Fig. 7. Example of application of the algorithm in 3 dimensions. a) 3D initial map with FM^2 path. b) The skeleton and also the path obtained.

TABLE I
TIME RESULTS DEPENDING ON THE ALGORITHM PARAMETERS.

Test	α	n	β	d_{min} (grid cells)	g_{min}	sat	t_s (s)	t_{pp} (s)	t_{FM^2} (s)
1	2	76	0.3	158	0.001	0.3	3.34	0.10	0.20
2	**4**	**152**	0.3	158	0.001	0.3	6.82	0.11	0.20
3	2	76	**0.1**	**53**	0.001	0.3	5.82	0.07	0.20
4	2	76	0.3	158	**0.1**	0.3	3.94	0.21	0.21
5	2	76	0.3	158	0.001	**0.7**	4.21	0.09	0.20
6	4	188	0.3	250	0.001	0.3	23.24	0.17	0.46

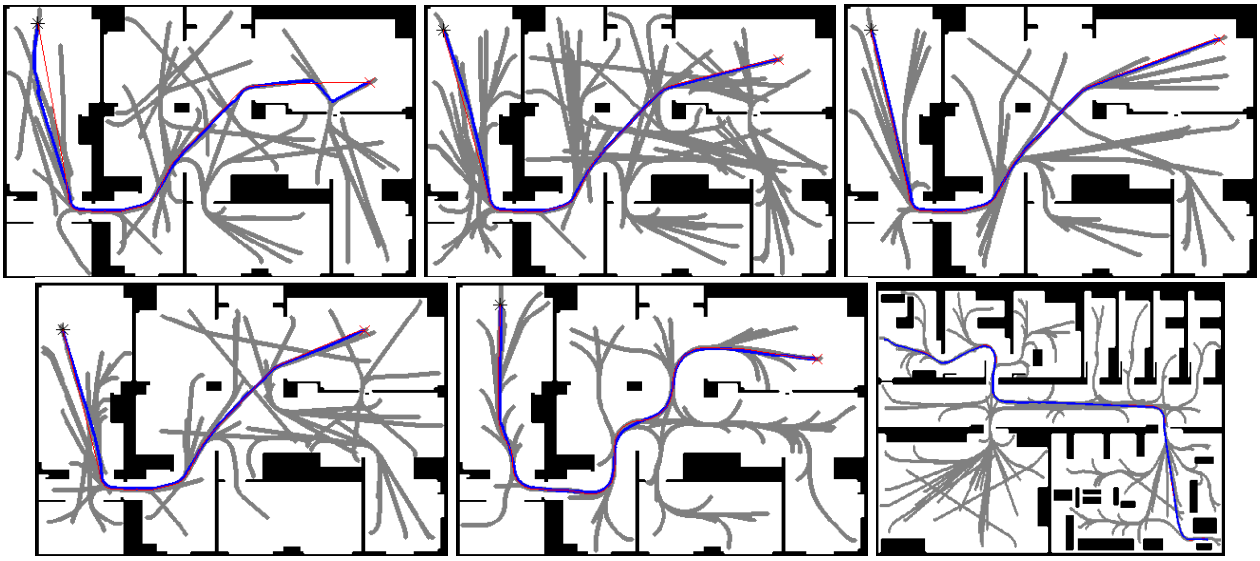

Fig. 6. Skeleton depending on the different parameters. From top left to bottom right, experiments 1 to 5 respectively. In most cases the path obtained using the FM^2 skeleton (blue) is very close to the one obtained with the standard FM^2 method (red).

implementation on a real robot adds nothing relevant to the paper. Also, we have focused on improving FM^2 algorithm so we understand that the comparison with other methods is well addressed in previous papers.

The future work is focused on removing the stochasticity of the algorithm. One of the main ideas is to automatically obtain the points to generate the FM^2 skeleton from the map characteristics such, for example, the center of the different rooms. Another interesting work is to convert the skeleton creation to an online algorithm, allowing it to evolve and adapt to environment changes.

ACKNOWLEDGMENT

This work is funded by the project number DPI2010-17772 founded by the Spanish Ministry of Science and Innovation.

REFERENCES

[1] J. A. Sethian, "A Fast Marching LevelSet Method for Monotonically Advancing Fronts," *Proceedings of the National Academy of Science*, vol. 93, no. 4, pp. 1591–1595, 1996.

[2] F. Aurenhammer, "Voronoi diagrams — a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, Sep. 1991.

[3] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi diagrams*. New York, NY: John Wiley & Sons, Inc., Nov. 1992.

[4] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching," in *International Conference on Intelligent Robots and Systems*, no. 9-15 Oct. 2006, 2006, pp. 2376–2381.

[5] S. Garrido, L. Moreno, and D. Blanco, "Sensor-based Global Planning for Mobile Robot Navigation," *Robotica*, vol. 25, pp. 189–199, 2007.

[6] S. Garrido, L. Moreno, D. Blanco, and F. Martin, "FM2: a Real Fast Marching Sensor-based Motion Planner," in *2007 IEEE/ASME international conference on Advanced intelligent mechatronics*, 2007, pp. 1–6.

[7] Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991, pp. 1398–1404.

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

[9] L. Yatziv, A. Bartesaghi, and G. Sapiro, "A Fast O(n) implementation of the Fast Marching Algorithm," *Journal of Computational Physics*, vol. 212, pp. 393–399, 2005.

[10] B. R. Greene, *The Elegant Universe: Superstrings, Hidden Dimensions, and the Quest for the Ultimate Theory*. Vintage Books, 1999.