How to deal with difficulty and uncertainty in the Outdoor Trajectory Planning with Fast Marching

S. Garrido, L. Moreno, and J. V. Gomez

Carlos III University of Madrid, Spain
{sgarrido,moreno,jvgomez}@ing.uc3m.es
http://roboticslab.uc3m.es

Abstract. This paper presents an interesting technique for finding the trajectory of an outdoor robot. This technique applies Fast Marching to a 3D surface terrain represented by a triangular mesh in order to calculate a smooth trajectory between two points. The method uses a triangular mesh instead of a square one because this kind of grid adapts better to 3D surfaces. The novelty of this approach is that, in the first step of the method, the algorithm calculates a weight matrix W that can represents difficulty, viscosity, refraction index or incertitude based on the information extracted from the 3D surface characteristics and the sensor data of the robot. Within the bestowed experiments these features are the height, the spherical variance, the gradient of the surface and the incertitude in the position of other objects or robots and also the incertitude in the map because some portions of the map can't be measured directly by the robot. This matrix is used to limit the propagation speed of the Fast Marching wave in order to find the best path depending on the task requirements, e.g., the trajectory with least energy consumption, the fastest path, the most plain terrain or the safest path. The method also gives the robot's maximum admisible speed, which depends on the wave front propagation velocity. The results presented in this paper show that it is possible to model the path characteristics as desired, by varying this matrix W. Moreover, as it is shown in the experimental part, this method is also useful for calculating paths for climbing robots in complex purely 3D environments. At the end of the paper, it is shown that this method can also be used for robot avoidance when two robots with opposite trajectories approach each other, knowing each others position.

1 Introduction

The objective of a path planner for a mobile robot operating in environments with unknown obstacles (dynamic or not), is to calculate collision-free trajectories with the best possible characteristics such as smoothness. In outdoor environments, the safety, the gradient or the roughness of the trajectory are also important.

The main objective of this paper is the development of technologies for obtaining mobile robots capable of complex tasks that demand a high degree of

autonomy and capacity of collaboration in the presence of human beings. One of the main applications of our work is the use of mobile robots in dangerous missions where the environment can be risky for humans (e.g., rescue missions, etc.). In this case, an unmanned autonomous vehicle (UAV) is sent in advance to obtain the images of the outdoor environment. Using the motion planning method proposed, the best trajectory to reach the goal is obtained over these images, and this trajectory is sent to the mobile robot. In the case that there are changes in the environment, the path can be recalculated using the same method. Another application would be, for example, that the images of the environment were obtained in real time using a laser scanner situated on the robot. Both situations will be considered in the experimental section.

If the Fast Marching (FM) techniques are used directly in Path Planning methods to calculate trajectories from one point to another. Nevertheless, following this strategy, the generated trajectories are not guaranteed to be safe and smooth. As explained in [1], the shortest geometrical path obtained can be unsafe, since it can touch the corners, walls, and obstacles, as shown in Figure 1. Because of these problems it is necessary to apply the FM method over a weight matrix that represents difficulty, nearness to obstacles, viscosity or refraction index.

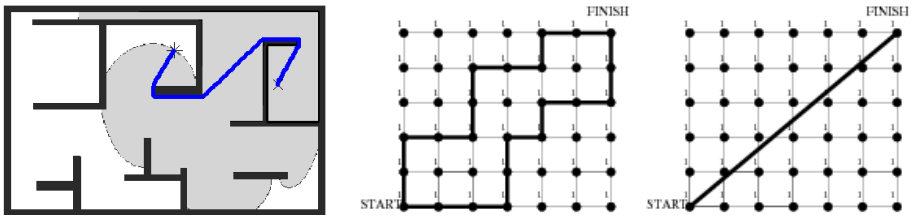

Fig. 1. Trajectory calculated by the Fast Marching method directly and comparison showing that Dijkstra's method gives multiple short paths (left image), whilst the Fast Marching method gives the optimal diagonal path (right image).

Moreover, in most works, these methods are used in $2D$ environments (indoor environments), whilst the method presented here adds another dimension to the problem (outdoor environments) and manipulates the 3D Fast Marching algorithm in order to modify the trajectories obtained to get a better and safer trajectory.

The new method proposed in this paper consists of several phases. First, as previously introduced, a $2\frac{1}{2}$ or 3D cloud of points of the surface is needed. This cloud can proceed from the robot laser, from a previous map or from a mixture of the two. Then, a triangular mesh is constructed over the image, which allows us to generate realistic surfaces due to the capabilities of the triangles to fit the characteristics of the map better than a square grid. Once the grid is constructed, the method extracts some information from the environment to obtain the height

and to calculate the gradient and the spherical variance, which gives information about the roughness of the surface. Then, it combines these data with the robot limitations to generate a weight matrix W. This matrix can be view as a difficulty or viscosity map which is situated on the 3D surface. Once the matrix is ready, the method applies the FM algorithm over this modified surface (the grid + matrix W) to generate the path.

If matrix W is not used, the trajectory obtained will be just the length of the shortest path between the two points, i.e., the geodesic distance. Applying matrix W, the proposed method gives a path which considers the features of the surface and the limitations of the robot. Moreover, it also gives us information about the speed of the robot based on the FM wave propagation speed [1, 2].

The results presented correspond first to the application of this method over outdoor images (3D) taken in advance. The fast execution of this method allows the updating of the given path as the images of the environment are updated. Therefore, this could be considered as an on-line path planning method. Second, the method is also used over a 3D environment constructed (in real time) by the laser mounted on a mobile robot. In both approaches, we prove that, using the proposed method, it is possible to generate smooth and safe plans in outdoor environments.

The remainder of the paper is organised as follows. Section 2 introduces an explanation about the FM method and how this algorithm can be implemented on triangulated meshes. The following section, section 3, introduces the viscosity matrix W and how it is formed. Section 4 presents some results obtained by simulation. Finally, the main conclusions of this paper are summarised in section 5.

2 The Eikonal Equation and the Fast Marching Planning Method

In robotics, the path planner of the mobile robot must drive it in a smooth and safe way to the goal point. In nature, there are phenomena that work in the same way: the electromagnetic waves. If at the goal point there is an antenna that emits an electromagnetic wave, then the robot could drive himself to the destination following the waves to the source. The idea of the electromagnetic wave is especially interesting because the potential have all the good properties desired for the trajectory, such as smoothness (that is, C^∞) and the absence of local minima.

The first arrival of the wave front expansion can be described by the Eikonal equation. The Eikonal (from the Greek 'eikon', which means 'image') is the phase function in a situation for which the phaseand amplitude are slowly varying functions of the position. Constantvalues of the Eikonal (level surfaces or level sets) represent surfaces of constant phase, orwave fronts. The normals to these surfaces are rays (the paths ofenergy flux); thus, the Eikonal equation provides a method for 'raytracing' in a medium of slowly varying index of refraction.

One way to characterise the position of a front in expansion is to compute the arrival time T, in which the front reaches each point of the interface. It is evident that, for one dimension, we can obtain the equation of the arrival function T in an easy way, simply considering the fact that the distance x is the product of the speed F and the time T.

$$x = F \cdot T \tag{1}$$

The spatial derivative of the solution function becomes the gradient

$$1 = F\frac{dT}{dx} \tag{2}$$

and therefore, the magnitude of the gradient of the arrival function $T(x)$ is inversely proportional to the speed:

$$\frac{1}{F} = |\nabla T| \tag{3}$$

For multiple dimensions, the same concept is valid because the gradient is orthogonal to the level sets of the arrival function $T(x)$.In this way, we can characterise the movement of the front as the solution of a boundary conditions problem. If speed F depends only on the position, then equation (3) can be reformulated as the Eikonal equation:

$$|\nabla T|\, F = 1. \tag{4}$$

The FM method is a numerical algorithm for solving the Eikonal equation, originally on a rectangular orthogonal mesh introduced by Sethian in 1996 [3]. The FM method is an $O(n)$ algorithm, as demonstrated in [4], where n is the total number of grid points. The algorithm relies on an upwind finite difference approximation to the gradient and a resulting causality relationship that lends itself to a Dijkstra-like programming approach.

The FM methods are used for problems in which the speed function never changes of sign, and so the wave front is always moving forward (there are no reflections, interferences, or diffractions). This allows us to transform the problem into a stationary formulation, because the wave front crosses each grid point only once time. This conversion into a stationary problem, offers a tremendous speed.

The wave propagation given by the Fast Marching Method gives us a distance that is the Geodesic distance measured with the metric given by the W matrix, which is completely different from the Euclidean Distance.

Since its introduction, the FM approach has been applied with success to a wide variety of problems that arise in geometry, computer vision, and manufacturing processes(see [5] for details). Varied advances have been made to the original technique, including the adaptive narrowband methodology [**?**] and the FM method for solving the static Eikonal equation [3].

2.1 Algorithm Implementation on an triangular mesh

The FM method applies to phenomena that can be described as a wave front propagating normal to itself with a speed function $F = F(x)$. The main idea is

to methodically construct the solution using only upwind values (the so called entropy condition). Let $T(x)$ be the solution surface at which the curve crosses the point x; then, it satisfies $|\nabla T|F = 1$, the Eikonal equation.

In order to understand how fast marching works, imagine an imprudent visitor that leaves unextinguished fire at some location in a natural reserve. The flame quickly becomes a forest fire, which expands outwards. Fire consumes the reached trees so the fire always propagates forward. We can record the fire front position at different points in time. It appears that the fire traverses the route having the smallest propagation time (and hence, the shortest length if the velocity is constant).In optics and acoustics this fact is known as Fermats principle or, in a more general form, the least action principle. In plain language, Fermats principle states that light traveling between two points always chooses the quickest path. Snells law of refraction follows directly from this principle.

It is necessary to know that the propagation happens from smaller to bigger values of T. The algorithm classifies the points of the triangular mesh into three sets: black, red and green, because our interface propagates like of a forest fire. Black points are points where the arrival time has been computed and is not going to change in the future. Green points are points that haven't been processed yet, for which the arrival time have not been computed up to now (corresponding with live trees). Red points are those belonging to the propagating wave front, which can be considered as an interface between the black and the green regions of the triangular mesh. In our forest fire example, red points correspond with trees that are currently in flames. Initially, only the source x_0 is marked as black and all points adjacent to it, are marked as red. The remaining points are marked as green. At each iteration, the red with the smallest value of T(x) is put into the black set. This T(x) value is calculated using the black points in triangles sharing it. The updated adjacent points are tagged as red. The process continues until all points become black.

The numerical basis of the fast marching method and its foremost difference with Dijkstras algorithm resides in the update procedure. While in Dijkstras algorithm the path is restricted to the graph edges, and a graph vertex was updated each time from an adjacent vertex, in fast marching, because the path can pass through the triangular faces of the mesh, a vertex has to be updated from a triangle, requiring two supporting vertices. We assume that the update step is applied to a triangle (x_1, x_2, x_3), where x_1 is the red point with the smallest arrival time $T_1 = T(x_1)$, x_2 is a point for which some arrival time approximation $T_2 = T(x_2)$ is available, and x_3 is the red or green point, whose arrival time approximation $T_3 = T(x_3)$ is that the triangle lies in the plane with $x_3 = 0$. In essence, given that the front reaches x_1 at time T_1 and x_2 at time T_2, the update step has to estimate the time when the front arrives to x_3.

3 Matrix W : the difficulty or viscosity map

As expressed in section 1, the direct application of the FM method still has some problems. The foremost one that usually arises in mobile robotics is that

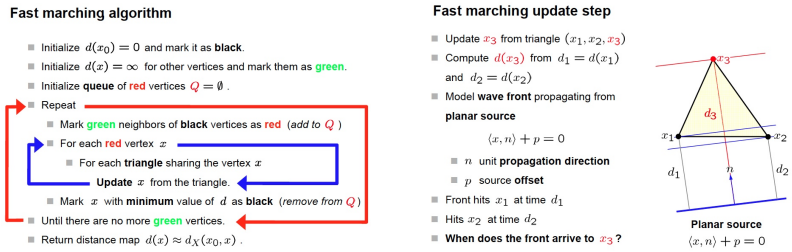

Fig. 2. Scheme of Fast Marching propagation.

optimal motion plans may bring robots too close to obstacles, and this can be dangerous.

In our approach, this problem has been solved in the same way as nature does: the electromagnetic waves, as light, have a propagation speed that depends on the media, that is the slowness or refraction index of the front wave propagation of a medium. In our case, the refraction index is defined by the viscosity map. This map can also be interpreted as difficulty or uncertainty map.

The proposed technique is based on the FM method, changing the speed of the wave front using a potential surface generated from the3D environment characteristics and the robot limitations. By doing so, the method changes the time when the front reaches each point and when the generated trajectory is calculated. This trajectory is not going to be the simple geodesic, but it is going to be modified according to the robot and task needs. To be able to modify this speed, the proposed method creates a weight matrix W, which is currently built based on three main characteristics of the 3D surface: the *spherical variance*, the *saturated gradient*, and the *height*. Some other characteristics can be added to the method and it will build a different potential surface.

3.1 Spherical variance

The spherical variance [**?**] finds the roughness of a surface to determine if it is crossable or not. In [7] a method to calculate the roughness degree is presented. This method is based on the normal vector deviation in each point of the surface. The spherical variance is obtained from the orientation variation of the normal vector in each point. The study uses the following reasoning:

- In a uniform terrain (low roughness), the normal vectors in a surface will be approximately parallel and, for this reason, they will present a low dispersion.
- On the other hand, in an uneven terrain (high roughness) the normal vectors will present great dispersion due to changes in their orientation.

  The spherical variance is obtained as follows:

  1. Given a set of n normal vectors to a surface, defined by their three components $\overrightarrow{N_i} = (x_i, y_i, z_i)$, the module of the sum vector R is calculated by:

$$R = \sqrt{\left(\sum_{i=0}^{n} x_i\right)^2 + \left(\sum_{i=0}^{n} y_i\right)^2 + \left(\sum_{i=0}^{n} z_i\right)^2} \tag{5}$$

2. Next, the mean value is normalized by dividing the module R between the number of data n, so the value of the result is within $[0, 1]$. In this way, we have
$$\frac{R}{n} \in [0, 1] \tag{6}$$
3. Finally, the spherical variance ω is defined as the complementary of the previous result.
$$\omega = 1 - \frac{R}{n} \tag{7}$$

When $\omega = 1$, there exists a maximum dispersion that can be considered as the maximum roughness degree, and when $\omega = 0$, a full alignment exists and the terrain will be completely flat.

3.2 Saturated Gradient

The gradient of a scalar field is a vector field which points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change. The gradient of f is defined to be the vector field whose components are the partial derivatives of f. That is:

$$\nabla f = (\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n}) \tag{8}$$

The saturated gradient consists of giving a limit value to the gradient of each point over the 3D surface. It means that, if the gradient value exceeds that limit, the point will not be included in the list of accessible points determined by the robot limitations. The gradient depends on the robot capabilities; the maximum inclination that the robot is able to cross will be the limit value for the saturated gradient.

3.3 Construction of Matrix $\boldsymbol{W}$

As previously explained, with this matrix the algorithm can modify the path that the robot is going to follow across the 3D surface. The way the matrix modifies the path is by giving a viscosity value for each point on the surface. It means that the propagation speed of the front end of the FM wave is modified.Hence, the time when the wave reaches each point will depend on that viscosity. We can add as many characteristics as we need to get different paths. These characteristics will modify the viscosity at each point.

The saturated gradient, the spherical variance, and the height are three matrices G, Sv, and H with the same size as the vertex matrix (the 3D mesh).The

value of each vertex of the 3D grid will be determined by the calculated gradient, spherical variance, and the height of each point.

To build matrix W we give a weight factor to each surface characteristic and we can determine which one is the most important depending on the task requirements.

The values of the three matrices vary from 0 to 1, so the values of matrix W are also within this range. The components of matrix W with a value of 0 will be points in the *vertex* matrix with maximum speed. Hence, these are points which the robot can cross without any problem and at its maximum speed. The components of W with a value of 1 will be points in the *vertex* with a minimum speed. This means that the robot will not be able to pass across them.

$$W = a_1 \cdot G + a_2 \cdot Sv + a_3 \cdot H \tag{9}$$

where:

$$\sum_i a_i = 1 \tag{10}$$

After matrix W is generated, the method runs the FM algorithm over the modified mesh (3D mesh + matrix W) to calculate the best trajectory. With the FM method the path found will be the less time path in the W metrics. In the normal FM evolution,this path will be the shortest because all the points in the surface will have the same 'speed' for the front propagation, i.e. it is the geodesic. With matrix W, the proposed method changes that 'speed', since this matrix gives information about the difficulty to pass through each point of the surface. The trajectory will be modified depending on the surface conditions and characteristics and according to the robot limitations. Since the method modifies the 'speed' of the Fast Marching wave, it gives not only the best trajectory, but also the speed to control the robot.

4 Algorithm Simulations

4.1 Test on data taken in advance

As previously stated, in the proposed method we need terrain data that can be a previous image of the environment, an elevation map, laser data or a mixture of all. In relation to the outdoor environment reconstruction, there are many ways to build an environment and represent it as a $2\frac{1}{2}$ or 3D surface.

Since the method is working in 3D, there are 3 matrices, one for each coordinate X, Y, and Z. X and Y are the coordinates for a plain surface and Z is the height of each point. In order to create a triangular mesh, the algorithm reads the data from the bitmap file to create these three matrices X, Y, and Z and then, it builds a plain mesh based on X and Y coordinates. The algorithm generates a Delaunay triangulation in 3D[**?**].

After the mesh is created, the algorithm is able to extract the needed data, the vertices and the faces of the triangles, from the matrices. Using these values, the algorithm is able to model the 3D surface.

Next, in this section several paths over the surface already presented will be obtained between the same initial and final points. The initial and the final points have been situated over the top of the two sides of the mountain range. Those paths are obtained by varying the values of the weight factors a_i of matrix W.

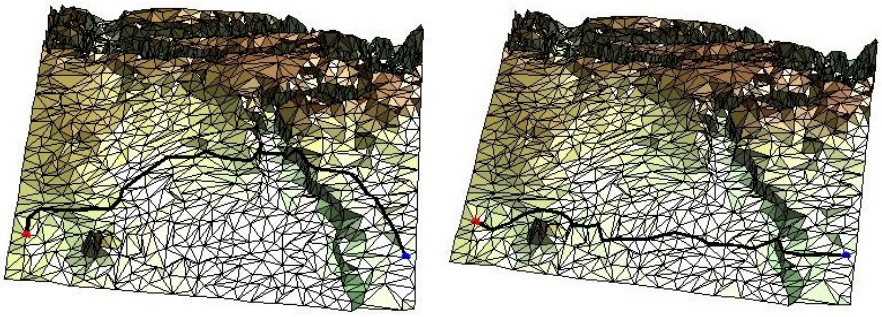

Fig. 3. Path calculated when $W = A$ and when $W = G$ in a mars map

In the case that $W = A$, this implies that the difficulty of the path will be determined by the height of every point of the mesh. In Figure 3, the path obtained when the height is penalised, without considering the roughness of the surface or its inclination, is presented. As can be observed, the calculated path will try to reach the final point passing trough the deepest part of the map.

On the other hand, if we decide to calculate the path penalising just the inclination of the surface, then the viscosity matrix is defined as $W = G$. In this case, as shown in Figure 3, the path will follow the parts with smallest slope.

Finally, the general idea proposed in this paper is the possibility of combining the different matrices in order to obtain a path that considers the height A, the roughness Sv, and the inclination G of the surface, among others. In the previous figures, it can be observed that, for the selected initial and final points, the height matrix favour that the path goes all the way trying to avoid the highest parts of it. On the other hand, the gradient matrix G favours the path with smallest slope. Therefore, we can select the values of each weight factor a_i in order to consider the limitations or features of the robot used.

Figure 4 shows a view of the path obtained when $W = Sv$. As can be observed the result is an intermediate path. Fig. 4 shows a view of the path obtained when $W = 0.20 * A + 0.40 * Sv + 0.40 * G$.

Moreover, the values of the weight factors a_i can be changed if the robot to be used is different or modified. It is also important to note that the trajectories calculated are a tentative path for the robot. The path can be modified online by modelling the environment with the robot sensors and recalculating the trajectory in a local area.

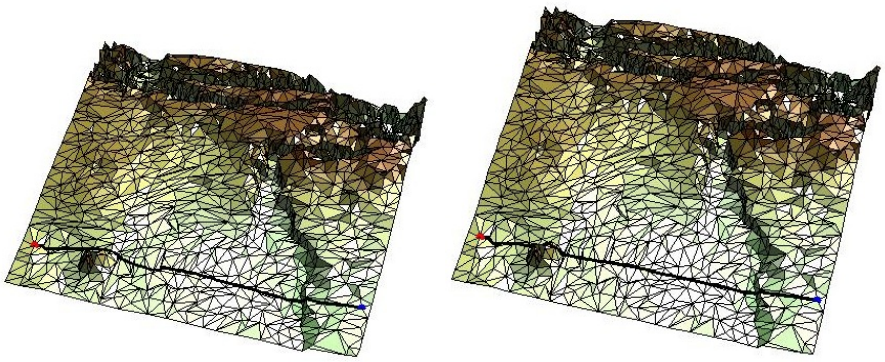

Fig. 4. Path calculated using $W = Sv$ and when $W = 0.20 * A + 0.40 * Sv + 0.40 * G$ in a mars map

4.2 Introduction of the uncertainty in the slowness matrix W

When there is a certain uncertainty the robot has to modify the trajectory or the velocity. For example, in the case of robot in Mars, if the robot doesn't have enough information of part of the trajectory, because it hasn't visual data of that part, could be better to change the trajectory to zones the robot can visualise.

How can we introduce that uncertainty in the map in order to change the trajectory? Fortunately, the viscosity matrix W can also be understood as an uncertainty matrix, the grey degree can be understood as a measurement of the uncertainty.

For example, suppose that the robot has no data of the points lower than its altitude, in that case the shadow points are represented in the matrix W with values next to zero (velocity of the media). In Fig. 5 is shown the difference between the robot trajectories without and with unprecise data of the points lower than the robot's altitude. As can be seen in the figure on the right, the trajectory is modified to not go through the lower areas. In Fig. 6 is shown the difference between the difficulty-uncertainty W matrices when the robot has and hasn't data of the points lower than its altitude. As can be seen in the right figure the lower parts have a bluish colour due to a bigger uncertainty and lower values in the W matrix that correspond to lower media velocity. In Fig. 7 is shown the difference between the wave expansion D matrices when the robot has and hasn't data of the points lower than its altitude. As can be seen, in the figure on the right, the expansion of the wave is more directed to the zone with less incertitude.

In Fig. 8 is shown the difference in the paths when the gradient is not saturated and when it is.

In Fig. 9 are shown the saturated gradient and the spheric variance corresponding to the previous figures.

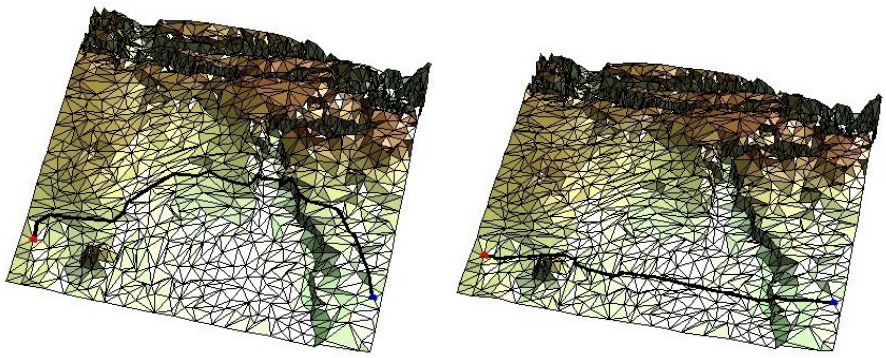

Fig. 5. Difference between the robot trajectories without and with unprecise data of the points lower than its altitude.

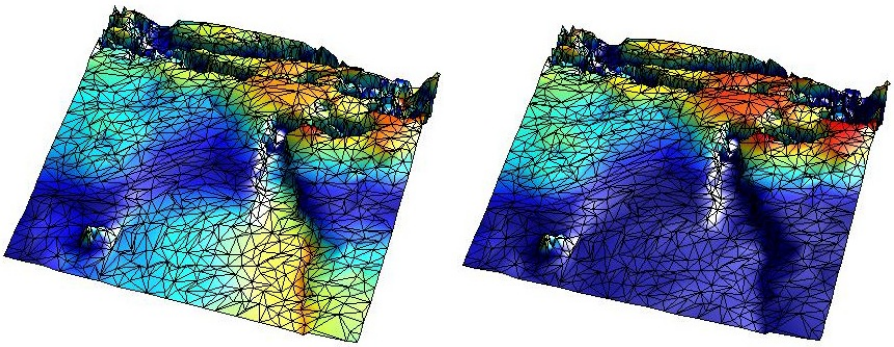

Fig. 6. Difference between the difficulty-uncertainty W matrices when the robot has and hasn't data of the points lower than its altitude.

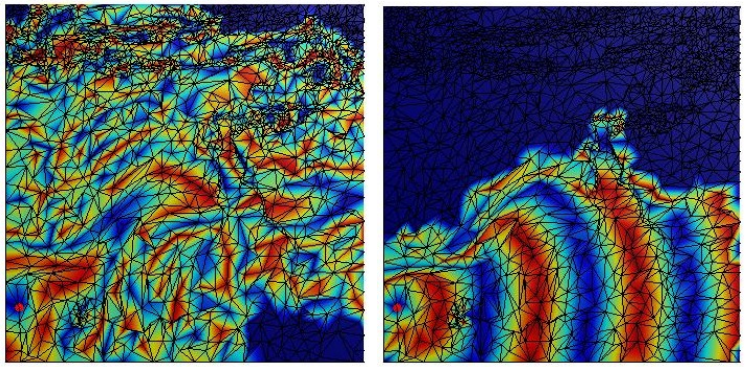

Fig. 7. Difference between the wave expansion D matrices when the robot has and hasn't data of the points lower than its altitude.

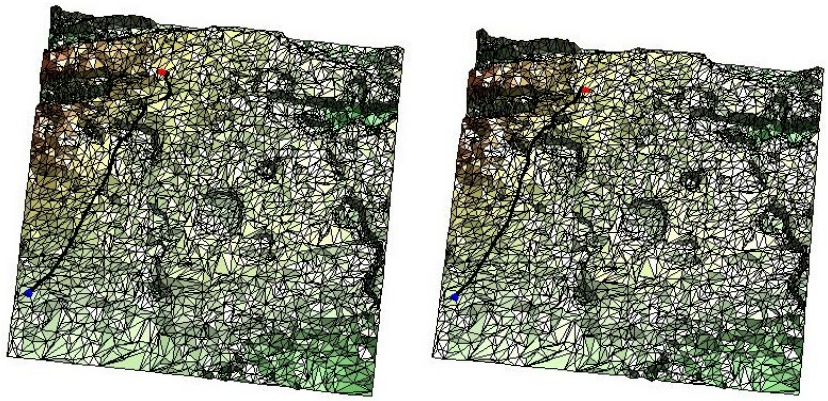

Fig. 8. Difference between the difference in the paths when the gradient is not saturated and when it is.

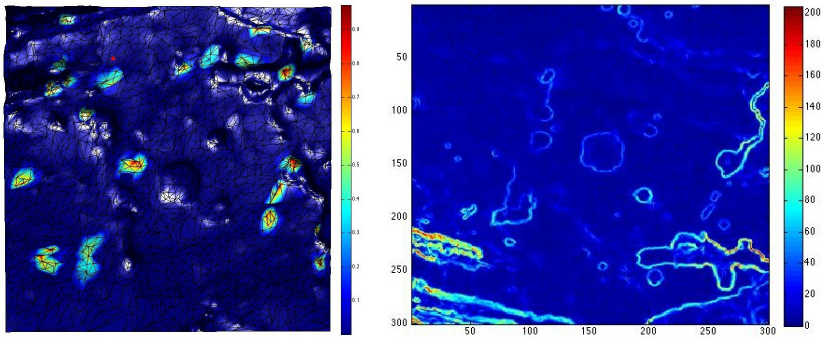

Fig. 9. Difference between the difficulty-uncertainty W matrices when the gradient is not saturated and it is.

4.3 Dynamic evolution of the paths of two robots when they sense each other's position

Another interesting application of the proposed method is the obtaining of the dynamic trajectories of two robots which navigate approaching each other, sensing their mutual position.

In order to get these trajectories, each robot maintains a map with the matrix of difficulty of the terrain. In each of these maps, it is necessary to add a small Gaussian to the position of the other robot in order to avoid possible collisions.

This bivariable Gaussian distribution is centered at the sensed position of the other robot with an incertitude given by a standard deviation of 10 in the direction of the trajectory (longer vector), and of 3 in the second direction (shorter vector, orthogonal to the longer vector). This Gaussian has been rescaled with a factor of 10 so that it can have the necessary importance in the difficulty map.

In the initialization phase, the algorithm calculates the general difficulty matrix or viscosity map W. Then, each robot maintains a copy of this map and, at each iteration, a Gaussian $G_i(k)$ is added to the sensed position of the other robot i. Afterwards, at each iteration, the trajectory of the robot i is calculated using the described Fast Method on the map $W + G_i(k)$, until the goal point is reached.

In the following figures the dynamic evolution of the trajectories of two robots is shown. The goal of each robot is represented by two red dots (one of them situated in the upper-right corner and the other one in the bottom-left corner), while the current position is represented by two blue ones. The starting point of each robot, as shown in Figure 10, is almost the same goal point of the other one. Therefore, the first planned paths cross each other in several points, provoking potential collisions. As the robots move towards their goals, they approach each other. In Figure 11 we observe the situation where both robots are approaching but they are not aware of the position of the other. In this case, a collision of both robots is expected. Later, when the robots are closer, they start sensing each other and, as can be seen in Figure 12, their paths run in parallel with no collision. Finally, in Figure 13 it is observed that each robot moves away from the other till they reach their goal points. As has just been explained, this robot avoidance is made thanks to the modification of the difficulty matrix W by adding a Gaussian to the sensed position of the robot.

5 Conclusions

The algorithm we have presented here is a new way to calculate trajectories for moving a robot over a 3D outdoor surface. One main point about the proposed method is that it can be used not only as a Path Planning method, but also to control the robot speed to keep it within a range given by the limit speed allowed over the 3D surface, taking into account environmental characteristics and task requirements reflected in matrix W, that can represent difficulty or incertitude. There are two values that can be attained from the results of the algorithm: the

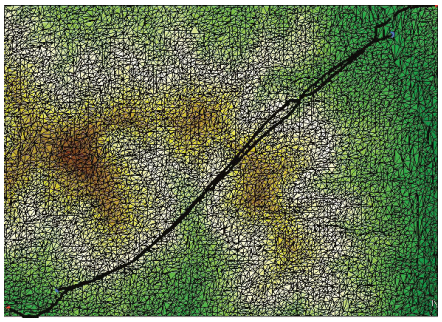

Fig. 10. Starting point. Both planned paths cross.

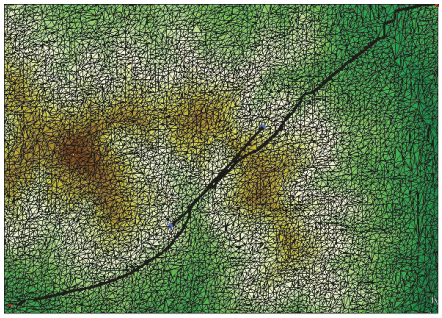

Fig. 11. Robots approaching with no view of each other.

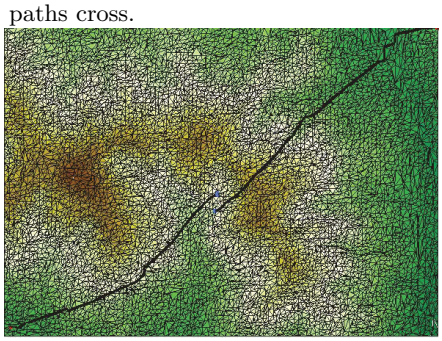

Fig. 12. Robots approaching sensing their mutual position.

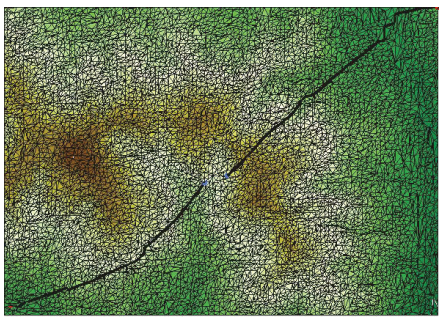

Fig. 13. Robots moving away from each other.

robot speed and the robot orientation. The speed is taken from the potential surface and the orientation can be taken from the next point in the trajectory that is going to be occupied by the robot. If the robot orientation and the next point where the robot is going to be are known, we can calculate the control law that has to be given to the robot in order to make it reach that next point. The most important thing about this algorithm is that it works in real time. It is really fast and give us the possibility to use it on-line to make decisions in order to avoid fixed or moving obstacles.

References

1. S. Garrido, L. Moreno, D. Blanco, Exploration of a cluttered environment using voronoi transform and fast marching method, Robotics and Autonomous Systems 56(12) (2008) 1069–1081.
2. S. Garrido, L. Moreno, M. Abderrahim, D. Blanco, Fm2: A real-time nsensor-based feedback controller for mobile robots, International Journal of Robotics and Automation 24(1) (2009) 3169–3192.
3. J. A. Sethian, Theory, algorithms, and aplications of level set methods for propagating interfaces, Acta numerica (1996) 309–395Cambridge Univ. Press.
4. L. Yatziv, A. Bartesaghi, G. Sapiro, A fast o(n) implementation of the fast marching algorithm, Journal of Computational Physics 212 (2005) 393–399.
5. J. Sethian, Level set methods, Cambridge University Press, 1996.
6. R. Kimmel, J. A. Sethian, Computing geodesic paths on manifolds, in: The National Academy of Sciences, Vol. 95, 1998.
7. C. Castejon, B. Boada, D. Blanco, L. Moreno, Traversable region modeling for outdoor navigation, Journal of Intelligent and Robotic Systems 43 (2-4).