Log of the inverse of the Distance Transform and Fast Marching applied to Path Planning

Santiago Garrido, Luis Moreno, Dolores Blanco and Fernando Martin
Robotics Lab.
Carlos III University
Madrid, Spain
Email: {sgarrido, moreno, dblanco and fmartin}@ing.uc3m.es

***Abstract*— This paper presents a new Path Planning method based in the inverse of the Logarithm of the Distance Transform and in the Fast Marching Method. The Distance Transform of an image gives a grey scale that is darker near the obstacles and walls and more clear far from them and it is calculated via Voronoi Diagram. The Logarithm of the inverse of the Distance Transform imitates the repulsive electric potential from walls and obstacles. This method is very fast and reliable and the trajectories are similar to the human trajectories: smooth and not very close to obstacles and walls.**

I. INTRODUCTION

When we want to move a robot from a place to another place it is necessary a global map to calculate a global trajectory. Mobile robot path planning approaches can be divided into five classes [1]. Roadmap methods extract a network representation of the environment and then apply graph search algorithms to find a path. Exact cell decomposition methods construct non-overlapping regions that cover free space and encode cell connectivity in a graph. Approximate cell decomposition is similar, but cells are of predefined shape (e.g. rectangles) and do not exactly cover free space. Potential field methods differ from the other four in that they consider the robot as a point evolving under the influence of forces that attract it to the goal while pushing it from obstacles. Navigation functions are commonly considered a special case of potential fields.

In order to calculate the trajectory in the global map, this paper presents a new Path Planning method based in the Logarithm of the inverse of the Distance Transform and the Fast Marching Method. The Distance Transform of an image gives a grey scale that is darker near the obstacles and walls and more clear far from them and is calculated via Voronoi Diagram. The Logarithm of the inverse of the Distance Transform imitates the repulsive electric potential in 2D from walls and obstacles. This potential impels the robot to follow a trajectory far from obstacles.

The Fast Marching Method has been applied to Path Planning [2], and their trajectories are of minimal distance, but they are not very safe because the path is too close to obstacles and what is more important, the path is not smooth enough.

In order to improve the safety of the trajectories calculated by the Fast Marching Method, it is possible to give two solutions:

First possibility, in order to avoid unrealistic trajectories, produced when the areas are narrower than the robot, the segments with distances to the obstacles and walls less than the size of the robot need to be removed from the Voronoi diagram previous to the Distance Transform.

Second possibility, used in this work, it is to dilate the objects and walls in a security distance that assure that the robot doesn't collide and doesn't accept passages narrower than the robot size.

The last step is to calculate the trajectory in the image generated by the Logarithm of the inverse of the Distance Transform using the Fast Marching Method. Then, the path obtained verify the smooth and safety considerations required for mobile robot path planning.

The advantages of this method are the easy implementation, the speed of the method and the quality of the trajectories. The method works in 2D and 3D, and it can be used at a local scale operating with sensor information instead of using an a priori map (sensor based planning).

II. INTRODUCTION TO THE DISTANCE TRANSFORM

The distance transform is a useful tool in digital picture processing. It has found a wide range of uses in image analysis, pattern recognition, and robotics.

In a binary image, a pixel is referred to as feature (or background) if its value is one (or zero).

For a given distance metric, the distance transform of an image produces a distance map of the same size. For each pixel inside the objects in the binary image, the corresponding pixel in the distance map has a value equal to the minimum distance to the background.

Similarly, we can also define a distance map for the background. Generally, we can define a zero-distance set in the distance map. The zero-distance set may correspond to the background or the objects in the binary image depending on the applications. From now on we'll say that a distance map is created in the complement of the zero distance set. Different metrics have been defined for the discrete plane, for example, city block distance and chessboard distance. These distances deviate quite substantially from the Euclidean distance. Even though they can be used in certain applications, the usual ideal is Euclidean metric. Octagonal distance is a combination of the two distances mentioned above. Still, relative errors between the octagonal distance and the Euclidean distance are about 10% and large absolute errors occur for large distances.

Better approximations to the Euclidean metric are proposed by Borgefors [3], [4], but errors are still proportional to distances. Euclidean distance transform on the Cartesian discrete plane was proposed by Danielsson [5] in 1980. This transform produces a distance map in which each pixel is a vector of two positive integer components. A four-point sequential Euclidean distance mapping algorithm and an eight-point sequential algorithm are described and analysed in detail in [5]. Parallel Euclidean distance transform algorithms are also discussed [5]. For the eight-point sequential algorithm, errors of less than 0.09 pixel units may occur at a few very rare locations due to the complex geometry of the objects. For most practical applications, this algorithm produces an error-free distance map. A modified version of the "ordinary" Euclidean distance transform, namely, the signed Euclidean distance transform is also briefly mentioned in [5]. A discussion on the signed Euclidean distance transform and its parallel algorithm can also be found in [6].

Clearly, the distance transform is closely related to the Voronoi diagram. The Voronoi diagram concept is involved in many distance transform approaches either explicitly or implicitly.

For any topologically discrete set S of points in Euclidean space and for almost any point x, there is one point of S to which x is closer than x is to any other point of S. The word "almost" is occasioned by the fact that a point x may be equally close to two or more points of S. If S contains only two points, a, and b, then the set of all points equidistant from a and b is a hyperplane, i.e. an affine subspace of codimension 1. That hyperplane is the boundary between the set of all points closer to a than to b, and the set of all points closer to b than to a.

In general, the set of all points closer to a point c of S than to any other point of S is the interior of a (in some cases unbounded) convex polytope called the **Dirichlet domain** or **Voronoi cell** for c. The set of such polytopes tesselates the whole space, and is the **Voronoi tessellation** corresponding to the set S. If the dimension of the space is only 2, then it is easy to draw pictures of Voronoi tessellations, and in that case they are sometimes called **Voronoi diagrams**.

The Distance Transform computes the Euclidean distance transform of the binary image. For each pixel in the image, the distance transform assigns a number that is the distance between that pixel and the nearest nonzero pixel of the image. The Distance Transform can have any dimension.

For two-dimensional the Distance Transform uses the second algorithm described in the article of Breu [7]. For higher dimensional Euclidean distance transforms, the distance transform uses a nearest-neighbor search on an optimized kd-tree, as described by Friedman [8].

The algorithm used in this work is based in the second algorithm work was done by Breu et al. [7]. In this work, the special properties of the Euclidean metric are exploited. They designed two linear-time algorithms based on Voronoi transforms where the second algorithm could have been improved if they had used the result of the previous row to reduce the set of possible candidates. It is an $O(mn)$ algorithm, where the image size is mxn.

III. Introduction to the Level Set Method and the Fast Marching Method

The level set method was devised by Osher and Sethian as a simple and versatile method for computing and analyzing the motion of the interface in two or three dimensions. The goal is to compute and analyze the subsequent motion of the interface under a velocity field. This velocity can depend on position, time, the geometry of the interface and the external physics. The interface is captured for later time as the zero level set of a smooth (at least Lipschitz continuous) function. Topological merging and breaking are well defined and easily performed.

The original level set idea of Osher and Sethian (see Osher [9]) for tracking the evolution of an initial front γ_0 as it propagates in a direction normal to itself with a given speed function V . The main idea is to match the one-parameter family of fronts $\{\gamma_t\}_{t\geq 0}$, where γ_t, is the position of the front at time t, with a one-parameter family of moving surfaces in such a way that the zero level set of the surface always yields the moving front. To determine the front propagation, we then need to find and solve a partial differential equation for the motion of the evolving surface. To be more precise, let γ_0 be an initial front in R^d, $d \geq 2$ and assume that the so-called *level set function* $\phi : \mathrm{R}^d \times \mathrm{R}_+ \to \mathrm{R}$ is such that at time $t \geq 0$ the zero level set of ϕ is the front γ_t. We further assume that $\phi(\boldsymbol{x}; 0) = \pm d(\boldsymbol{x})$; where $d(\boldsymbol{x})$ is the distance from $\boldsymbol{x}$ to the curve γ_0. We use plus sign if $\boldsymbol{x}$ is inside 0 and minus if $\boldsymbol{x}$ is outside. Let each level set of ϕ along its gradient field with speed V . This speed function should match the desired speed function for the zero level set of ϕ. Now consider the motion of, e.g., the level set

$$\left\{\boldsymbol{x} \in \mathrm{R}^d : \phi(\boldsymbol{x}; t) = 0\right\}. \tag{1}$$

Let $\boldsymbol{x}(t)$ be trajectory of a particle located at this level set so that

$$\phi(\boldsymbol{x}(t); t) = 0. \tag{2}$$

The particle speed $d\boldsymbol{x}/dt$ in the direction n normal to the level set is given by the speed function V, and hence

$$\frac{d\boldsymbol{x}}{dt} \cdot n = V. \tag{3}$$

where the normal vector n is given by

$$n = -\frac{\nabla\phi}{|\nabla\phi|}. \tag{4}$$

This is a vector pointing outwards, giving our initialization of n. By the chain rule

$$\frac{\partial\phi}{\partial t} + \frac{d\boldsymbol{x}}{dt} \cdot \nabla\phi = 0. \tag{5}$$

Therefore $\phi(\boldsymbol{x};t)$ satisfies the partial differential equation (the level set equation)

$$\frac{\partial \phi}{\partial t} - V\,|\nabla \phi| = 0, \tag{6}$$

and the initial condition

$$\phi(\boldsymbol{x}; t=0) = \pm d(\boldsymbol{x}). \tag{7}$$

This is called an Eulerian formulation of the front propagation problem because it is written in terms of a fixed coordinate system in the physical domain.

If the speed function V is either always positive or always negative, we can introduce a new variable (the arrival time function) $T(\boldsymbol{x})$ defined by $\phi(\boldsymbol{x}, T(\boldsymbol{x})) = 0$.

In other words, $T(\boldsymbol{x})$ is the time when $\phi(\boldsymbol{x};t) = 0$. If $\frac{d\boldsymbol{x}}{dt} \neq 0$, then T will satisfy the stationary Eikonal equation

$$V\,|\nabla T| = 1, \tag{8}$$

coupled with the boundary condition

$$T|_{d(\boldsymbol{x})=0} = 0. \tag{9}$$

The advantage of this formulation 8 is that we can solve it numerically by the fast marching method [2], which is precisely what we will do in this paper.

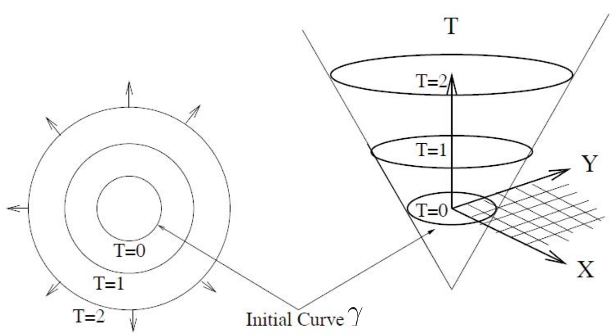

Fig. 1. Transformation of the front motion into boundary value problem.

Summing up, the central mathematical idea is to view the moving front γ_t as the zero level set of the higher-dimensional level set function $\phi(\boldsymbol{x};t)$. Depending on the form of the speed function V , the propagation of the level set function $\phi(\boldsymbol{x};t)$ is described by the initial problem for a nonlinear Hamilton-Jacobi type partial differential equation 5 of first or second order.

If $V > 0$ or $V < 0$, it is also possible to formulate the problem in terms of a time function $T(x)$ which solves a boundary value problem for a stationary Eikonal equation 8.

Fast Marching Methods are designed for problems in which the speed function never changes sign, so that the front is always moving forward or backward. This allows us to convert the problem to a stationary formulation, because the front crosses each grid point only once. This conversion to a stationary formulation, plus a whole bunch of numerical tricks, gives it its tremendous speed

Level Set Methods are designed for problems in which the speed function can be positive in some places are negative in others, so that the front can move forwards in some places and backwards in others. While significantly slower than Fast Marching Methods, embedding the problem in one higher dimension gives the method tremendous generality.

Because of the nonlinear nature of the governing partial differential equation 5 or 8, solutions are not smooth enough to satisfy this equation in the classical sense (the level set function and the time function are typically only Lipschitz). Furthermore, generalized solutions, i.e., Lipschitz continuous functions satisfying the equations almost everywhere, are not uniquely determined by their data and additional selection criteria (entropy conditions) are needed to pick out the (physically) correct generalized solutions. The correct mathematical framework in which to treat Hamilton-Jacobi type equations is provided by the notion of viscosity solutions (see Crandall [10], [11]).

After its introduction, the level set approach has been successfully applied to a wide collection of problems that arise in geometry, mechanics, computer vision, and manufacturing processes, see Sethian [12] for details. Numerous advances have been made to the original technique, including the adaptive narrow band methodology (see Adalsteinsson and Sethian [13]) and the fast marching method for solving the static Eikonal equation (see Sethian [14], [12]). For further details and summaries of level set and fast marching techniques for numerical purposes (see Sethian [12]). The complexity of the Fast Marching Method is $O(m \times n)$, where the dimensions of the image are $(m \times n)$ (see Yatziv [15]).

IV. Implementation of the Method

This method starts with the calculation of the Logarithm of the inverse of the Distance Transform of the 2D a priori map of the environment (or the inverse of the Distance Transform in case of 3D maps). Each white point of the initial image (which represents free cells in the map) is associated to a level of grey that is the logarithm of the inverse of the 2D distance to nearest obstacles (or the inverse of the Distance Transform in 3D). As a result of this process, it is obtained a kind of potential proportional to the distance to the nearest obstacles to each cell. Zero potential indicates that a given cell is part of an obstacle and maxima potential cells corresponds to cells located in the Voronoi diagrams (which are the cells located equidistant to the obstacles).

This function introduces a potential similar to a repulsive electric potential (in 2D), that can be expressed by

$$\phi = c_1 \log(r) + c_2. \tag{10}$$

If $n > 2$, the potential is

$$\phi = \frac{c_3}{r^{n-2}} + c_4. \tag{11}$$

More precisely, the equation of the electric potential ϕ in a region Ω free of charge with border $\partial\Omega$ and constant potential

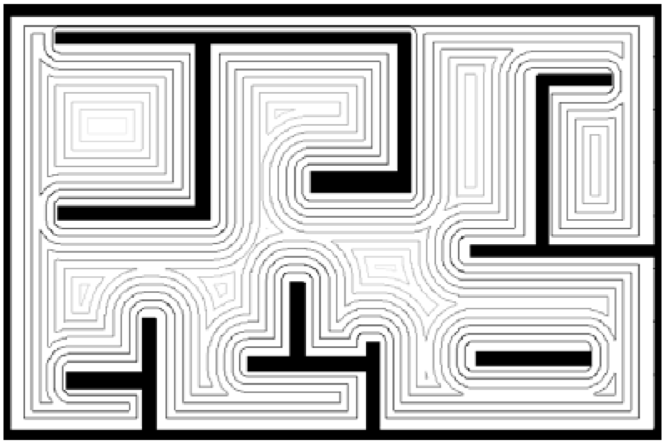

Fig. 2. Potential of the Logarithm of the inverse Distance Transform.

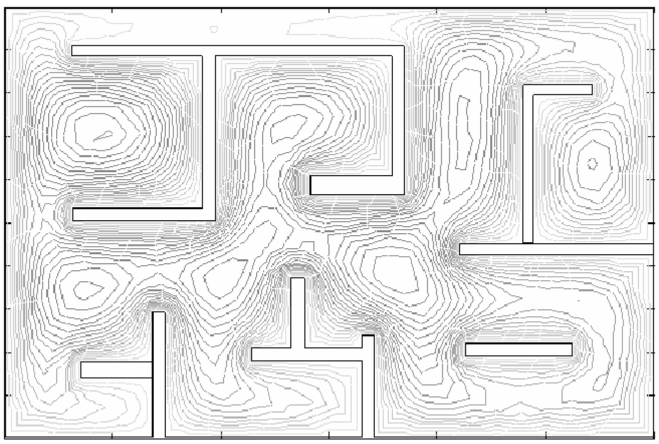

Fig. 3. Potential of Poisson's equation.

at obstacles (Dirichlet conditions) is the Laplace equation:

$$\nabla^2 \phi = 0. \tag{12}$$

The solutions of this equation are called Harmonic Functions. In case of spherical symmetry, the solution depends only on r(distance from the origin). The general expression of the Laplace's equation in polar coordinates can be written as

$$\nabla^2 \phi = \phi_{rr} + \frac{n-1}{r}\phi_r + angular\,terms, \tag{13}$$

where ϕ_{rr} is the second partial derivative of ϕ with respect to r and ϕ_r is the first partial derivative with respect to r. Since $\phi = \phi(r)$, the angular terms become zero. Then the Laplace equation becomes

$$\phi_{rr} + \frac{n-1}{r}\phi_r = 0; \tag{14}$$

or

$$\frac{\phi_{rr}}{\phi_r} + \frac{n-1}{r} = 0. \tag{15}$$

Integrating once, we obtain

$$\phi_r = \frac{c_1}{r^{n-1}} \tag{16}$$

For n=2, the solution is

$$\phi = c_1 \log(r) + c_2 = c_1' \log(1/r) + c_2' \tag{17}$$

If $n > 2$, the solution is

$$\phi = \frac{c_3}{r^{n-2}} + c_4 \tag{18}$$

This result connect the potential defined previously (with the logarithm of the inverse of the Distance Transform) to the Harmonic Functions, which are the solutions of the Laplace's equation. The difference with other Harmonic Functions based methods is that, in those methods, the resolution of Laplace's equation requires a repulsive potential (done by the Dirichlet, Neumann or Robin contour conditions) of walls and obstacles, and an attractive potential done by the objective (which is another contour condition). In the case of Dirichlet conditions, with constant high level potential for obstacles and walls and constant low level for the objective, the trajectories obtained are the shortest trajectories along potential surface (which is the minimal area surface that connect high level potential points and low level potential points). This surface is an Harmonic Function. These shortest trajectories are not the shortest trajectories on 2D environment map.

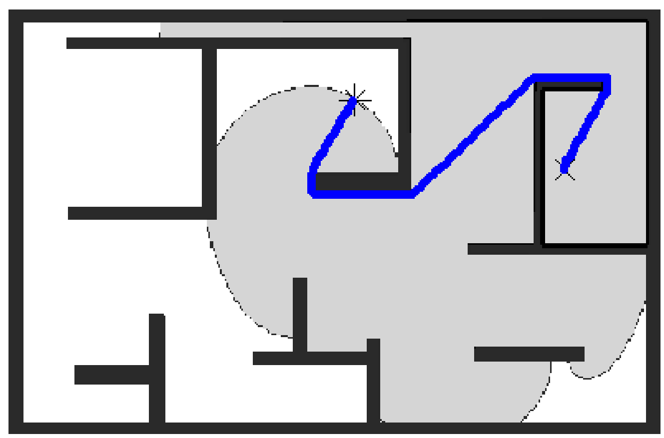

Fig. 4. Trajectory calculated with Fast Marching without the Logarithm Distance Transform.

As can be seen in equation 17, the Log of the inverse of the Distance Transform can be calculated as $\phi = c_1 \log(r) + c_2$, where c_1 is a negative constant.

In a second step, the technique proposed here uses Level Set Method (Fast Marching) to calculate the shortest trajectories in the potential surface defined by logarithm of the inverse of the Distance Transform. The calculated trajectory is the geodesic in the potential surface, i.e. with a viscous distance. This viscosity is done by the grey level. If the Level Set Method were used directly on the environment map, we would obtain the shortest geometrical trajectory (fig. 4), but the trajectory is not safe nor smooth.

This method is connected with the Poisson's equation,

$$\nabla^2 \phi = f(x) \tag{19}$$

Fig. 5. Potential of the Logarithm of the inverse of Distance Transform.

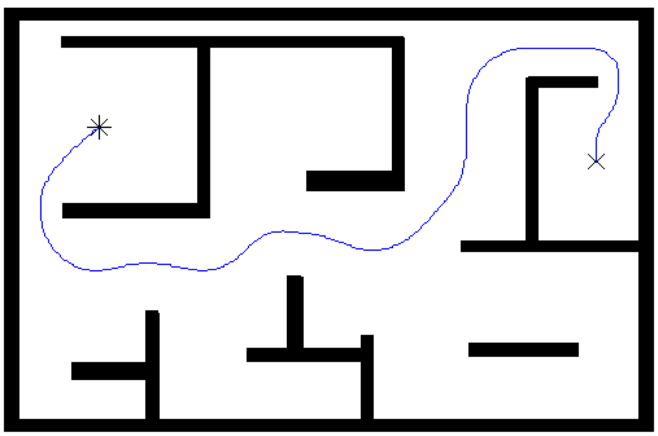

Fig. 6. Trajectory calculated with Fast Marching with the Logarithm Distance Transform.

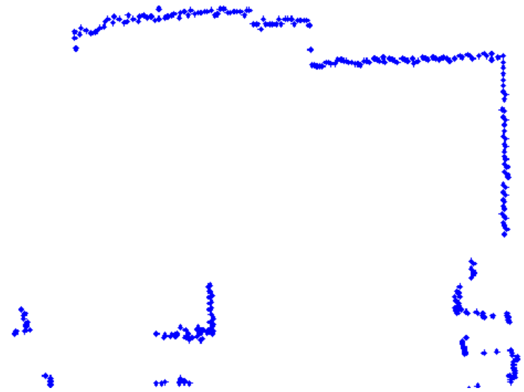

Fig. 7. Laser data read by the robot.

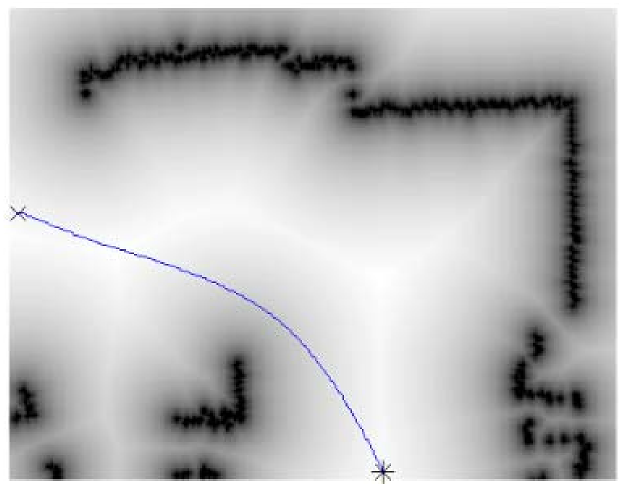

Fig. 8. Trajectory calculated with Fast Marching using the laser data (Local map).

that corresponds to the electric field due to a charge density $f(x)$ (in this case uniform) located in the free areas, or the auto-deformation due to its own weight of an elastic surface (latex type) with material density $f(x)$, fixed to obstacles and walls, as it can be seen in the figure 3.

The trajectories obtained tend to go by the Voronoi diagram but properly smoothed. In case of using the inverse of the Distance Transform, the trajectories calculated by Fast Marching are quite close to the Voronoi diagram. By using the Logarithm of the inverse of the Distance Transform, the trajectory is smoother but still close to the Voronoi diagram.

The potential created has local minima as shown in fig. 2 and 5, but the trajectories are not stuck in these points because the Fast Marching Method gives the trajectories that correspond to the propagation of a wave front faster in clearer regions and slower in the darker ones. The method proposed, can also be used for sensor based planning, working directly on a raw sensor image of the environment, as shown in figures 7 and 8.

V. Results

To illustrate the method possibilities, it has been used for planning a trajectory in a typical offices's indoor environment as shown in figures 9 and 10. The dimensions of the environment are 116x14 meters (the cell resolution is 12 cm). For this environment the first step (Log of inverse Distance Transform) takes 0.06 seconds in a Pentium 4 at 2.2Ghz, and the second step (Fast Marching) takes 0.20 seconds for a long trajectory.

The proposed method is highly efficient from a computational point of view because of the method operates directly over a 2D image map (without extracting adjacency maps), and due to the fact that Fast Marching complexity is $O(m \times n)$ and the Distance Transform is also of complexity $O(m \times n)$, where $m \times n$ is the number of cells in the environment map.

The method provides smooth trajectories that can be used at low control levels without any additional smooth interpolation

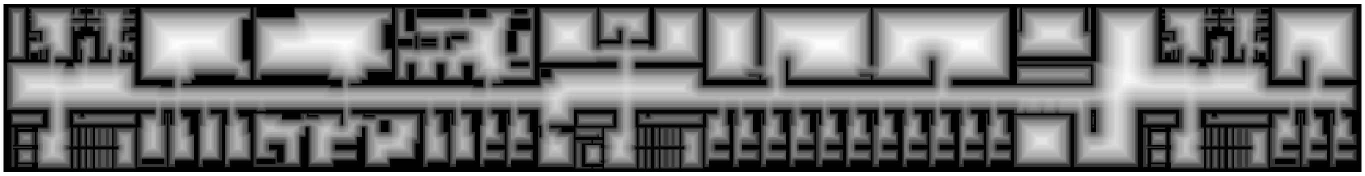

Fig. 9. Log of the Distance transform applied of the environment map of the Robotics Lab.

Fig. 10. Trajectory calculated to avoid obstacles in a cluttered environment with Fast Marching and the Logarithm Distance Transform (Global map).

process. The results are shown in fig 9 (Log of the inverse Distance Transform of the environment map of the Robotics Lab.) and fig 10 (the path obtained after applying the Fast Marching method to the previous potential image).

VI. Conclusion

The results obtained show that the Logarithm of inverse of the Distance Transform can be used to improve the results obtained with Fast Marching method applied to Path Planning, to obtain smooth and safe trajectories.

The algorithm complexity is $O(m \times n)$, where $m \times n$ is the number of cells in the environment map, which let us use the algorithm on line. Besides, the algorithm can be used directly with raw sensor data to implement a sensor based local path planning.

References

[1] J.-C. Latombe, *Robot motion planning*. Dordrecht, Netherlands: Kluwer Academic Publishers, 1991.

[2] J. A. Sethian, “A fast marching level set method for monotonically advancing fronts,” *Proc. Nat. Acad Sci. U.S.A.*, vol. 93, no. 4, pp. 1591–1595, 1996.

[3] G. Borgefors, “Distance transformations in arbitrary dimensions,” *Computer Vision, Graphics and Image Processing*, no. 27, pp. 321–345, 1984.

[4] G.Borgefors, “Distance transformations in digital images,” *CVGIP*, vol. 34, no. 3, pp. 344–371, 1986.

[5] P. E. Danielsson, “Euclidean distance mapping,” *Computer Graphics and Image Processing*, no. 14, pp. 227–248, 1980.

[6] H. Yamada, “Complete euclidean distance tranformation by parallel operation,” *Proc. of 7th Int. Conf. on Pattern Recognition*, pp. 69–71, 1984.

[7] K. D. W. M. Breu H., Gil J., “Linear time euclidean distance transform algorithms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 529–533, 1995.

[8] F. R. Friedman J. H., Bentley J.L., “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematics Software*, vol. 3, no. 3, pp. 209–226, September 1997.

[9] 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.

[10] P. L. L. M. G. Crandall, H. Ishii, “User’s guide to viscosity solutions of second order partial differential equations,” *Bull. Amer. Math. Soc.*, vol. 27, no. 1, pp. 1–67, 1992.

[11] P. L. L. M. G. Crandall, “Viscosity solutions of hamilton jacobi equations,” *Trans. Amer. Math. Soc.*, vol. 277, pp. 1–42, 1983.

[12] J. A. Sethian, “Theory, algorithms, and aplications of level set methods for propagating interfaces,” *Acta numerica*, pp. 309–395, 1996, cambridge Univ. Press.

[13] D. Adalsteinsson and J. Sethian, “A fast level set method for propagating interfaces,” *J. Comput. Phys.*, vol. 118, no. 2, pp. 269–277, 1995.

[14] J. Sethian, *Level set methods*. Cambridge University Press, 1996.

[15] G. S. L Yatziv, A Bartesaghi, “O (n) implementation of the fast marching algorithm,” *Journal of Computational Physics*, 2005.