A Comparative Study of Implementing Fast Marching Method and A* Search for Mobile Robot Path Planning in Grid Environment: Effect of Map Resolution

Chia Hsun Chiang
Institute of Information Science Academia Sinica
fetishist@gmail.com

Po Jui Chiang
Department of Electrical Engineering, I-Shou University
d90941006@ntu.edu.tw

Jerry Chien-Chih Fei
Department of Computer Science, University of Toronto
fei.jerry@gmail.com

Jin Sin Liu
Institute of Information Science Academia Sinica
liu@iis.sinica.edu.tw

***Abstract*- Theoretically both of Fast Marching Method (FMM) and A* SEARCH have the same time complexity of $O(N \log N)$, where N is the number of grid points in a map, and guarantee to find an optimal solution if this path exists. This paper examines the effect of alteration of resolution of map on the performance of path planning by A* and FMM on 2D rectangular grid environments. Our observations on the basis of comparative simulations in three test environments at different resolutions are that: A* is more focused and thus faster to generate a polygonal line (continuous but not smooth) path, while FMM generates a better curvature-control, generally smoother and shorter path that is a numerically consistent approximation to the true shortest continuous path as the resolution of map is finer and finer. The difference of path quality for these two methods is more significant as the resolution is finer.**

I. INTRODUCTION

Point-to-point path planning of autonomous mobile robot, defined as finding a collision-free path linking a given start configuration to a goal configuration, has been extensively explored in last two decades. Many different methods achieving varying degrees of success in a variety of conditions/criteria of motion and environments have been developed (see e.g. [3], [24]). More recently, Fast Marching Method (FMM) introduced by Sethian [2], [4], [17]-[19] has received more attentions and has been applied to path planning problem in mobile robots or UAV (underwater autonomous vehicles) and many other research fields [5]-[7], [9]-[12], [15]-[16], [27]-[28]. Before performing the FMM, we could preprocess the environment by utilizing the distance transform to compute the Euclidean distance transform of a given binary map image. For a given distance metric, the distance transform of an image produces a distance map of the same size. Each pixel in the image would be assigned a number that is the distance between that pixel and the nearest nonzero pixel of the image, then the distance map, similar to potential map, can be used as cost function for the Eikonal equation to avoid obstacle. The work of distance transform could be accomplished by Fast-Marching Method, but the Closest Point Transform (CPT) developed by Sean Mauch would achieve the same goal more efficiently [13, 14]. Many distance transform approaches involves the concept of Voronoi diagram [21].

Theoretically both of FMM and A* SEARCH have the same time complexity of $O(N \log N)$, where N is the number of grid points in a map. Both are efficient to plan a shortest path in grid environment, where the path is represented as an (finite) ordered sequence of adjacent nodes corresponding to a piecewise linear curve connecting start and goal that minimizes the accumulated cost from start to goal found by the employed algorithm. In addition, both algorithms are complete [24], i.e. guarantee to find an optimal path if the path exists or report that no such path exist. Previous comparisons of FMM and A* for path planning on the basis of single resolution are: A* is more efficient [23], and path generated by FMM is smoother [25]. In addition, [10] pointed out that graph search algorithms are not consistent, while level set methods are consistent, i.e. as the grid is refined, the obtained solution converges to the true continuous solution. Through mobile robot path planning simulations in test grid environments at different resolutions, a path is generated by FMM or A* at each map resolution. The aim of this paper based on observations on the genrerated paths by these two methods is two-fold: on one hand, to verify the statements drawn from a single resolution of map in the existing literature, and on the other hand, to reveal their merits according to their real performance measured by some metric (for example, the number of points visited), and the differences between the quality (e.g. smoothness, overall path length, convergence etc.) of the path generated by these two methods at different resolutions, although they all generate paths that are polygonal lines. More importantly, the effect of resolution of environment grid map, possiibly built from different devices (e.g. camera, laser scanner, satellite imaginery) [26] in regards to the precision with which the paths must be generated, on path qualities is investigated.

In Section II, we outline A* SEARCH and FMM for shortest path planning. Section III presents the experimental results on both methods for altering the resolution of grid map. In Section IV, we make emperically comparisons of the results followed by discussions of some possible ways of improving FMM.

978-1-4244-1953-1/07/$25.00 c2007 IEEE.

II. REVIEW OF A* SEARCH AND FAST MARCHING METHOD

To begin, a grid map (a two dimensional orthogonal grid of nodes) representing the flat terrain that the mobile robot will move is given. The nodes are either safe or forbidden and a start, a goal located in the map is given. A path of a point mobile robot from start to goal that minimizes a cost is to be found. In the following, we outline two search –based methods: A* and FMM that could search a shortest path, defined by an (finite) ordered sequence of nodes, linking the given pair of start and goal.

A. A SEARCH Algorithm [1]*

The idea of A* SEARCH is that each node is associated with a cost function

$$f(n) = g(n) + h(n),$$

where $g(n)$ is the cost from initial node to the current node and $h(n)$ is an estimated cost from the current node to the goal node. A* SEARCH generates and processes the successor nodes in a certain way. Whenever it looks for the next node to process, it employs heuristic function $h(n)$ trying to choose the lowest cost node to process.The following algorithm summarizes A* SEARCH:

A* Algorithm:

1. Put the start node s on a list called OPEN and compute $f(s)$.
2. If OPEN is empty, exit with failure; otherwise continue.
3. Remove from OPEN that node whose f value is smallest and put it on a list called CLOSED. Call this node n. Resolve ties for minimal f values arbitrarily, but always in favor of any goal node.
4. If n is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise continue.
5. Expand node n, generating all of its successors. If there are no successors, go immediately to 2. For each successor n_i, compute $f(n_i)$.
6. Associate with the successors not already on either OPEN or CLOSED the f values just computed. Put these nodes on OPEN and direct pointers from them back to n.
7. Associate with those successors that were already on OPEN or CLOSED the smaller of the f values just computed and their previous f values. Put on OPEN those successors on CLOSED whose f values were thus lowered, and redirect to n the pointers from all nodes whose f values were lowered.
8. Go to 2.

A* SEARCH operates in a way that it maintains two sets of nodes - OPEN and CLOSED. The OPEN list keeps track of the nodes that will be examined during path planning while CLOSED list holds the ones which will not be examined. Initially, OPEN list contains the initial node and CLOSED list stores the nodes that are unreachable (ex: obstacles in path planning problems). Each node also maintains a pointer to its parent node which is used for retrieving optimal solution in the end. A* SEARCH has a main loop that repeatedly gets the node with smallest $f(n)$ cost from the OPEN list until the goal node is obtained or cannot be reached. For each newly generated successor node, if it is already in OPEN or CLOSED list and the copy there has an equal or lower $f(n)$ estimate, then we can discard this newly generated node and move on. Otherwise we will remove the one with higher cost from the lists, and update $f(n)$, $g(n)$, and $h(n)$ of parent node of this newly generated node. Then store this node into OPEN list and return to the main loop. Note that $g(n)$ is calculated as the distance (cost) from previous node to the current node plus the distance already calculated before arriving to the previous node.

The A* search algorithm relies heavily on heuristic function. An appropriate heuristic function determines whether the algorithm can execute efficiently and accurately. In order to find an optimal solution, the heuristic must be admissible. To be admissible, the heuristic function must never over-estimate the cost from one node to the goal node.

B. Fast Marching Method (FMM)

Isotropic Eikonal equation is the following first order PDE

$$\begin{aligned} &\|\nabla u(x)\| = \tau(x), x \in \Omega \\ &u(x) = q(x), x \in \partial\Omega \end{aligned} \quad (1)$$

where τ is the speed (or cost) function that is a function of local position, Ω is the domain, $\partial\Omega$ its boundary. As implicated in the robotic navigation, the formulation can be interpreted as isotropic front propagation or isotropic min-time optimal trajectory problems. In the control-theoretic context, the characteristic lines of equation (1) can be interpreted as the optimal trajectories.

The key feature of Eikonal equation (1) is that their characteristic lines coincide with the gradient lines of the viscosity solution $u(x)$; this allows the construction of single-pass Fast Marching Method (FMM) [2], [4], [17]-[19] that solves the Eikonal equation in a stable and consistent manner. Under 4-connectivity 2D grid condition, using the first order finite difference upwind scheme to approximate the Eikonal equation (1) in continuous domain yields a quadratic equation of u to estimate actual (geodesic) distances in discrete domain:

$$\begin{aligned} &\left(\max\left\{u - U_{i-1,j}, u - U_{i+1,j}, 0\right\}\right)^2 + \left(\max\left\{u - U_{i,j-1}, u - U_{i,j+1}, 0\right\}\right)^2 \\ &= \tau_{i,j}^2 \end{aligned} \quad (2)$$

where we assume the Cartesian grid with unit grid spacing; $U_{x,y}, \tau_{x,y}$ denotes the distance and speed at grid coordinate (x, y), respectively. If the discriminant of quadratic equation (2) is larger than zero, we solve the larger u solution of the quadratic equation; else we set the value of u with the following equation instead:

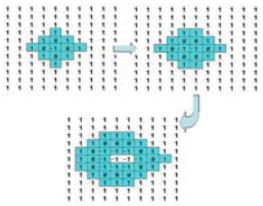

Fig. 1. Illustration of Narrow Band Expansion in FMM. The blocks with blue shadow are inside the narrow band computation mask. 1: *Far*, distance has not yet been computed. 0: *Trial*, distance is being computed but not set. -1: *Alive*, distance has been computed.

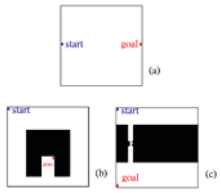

Fig. 2 Test environments. (a) Free space. (b) Map with one concavity obstacle. (c) Narrow passage

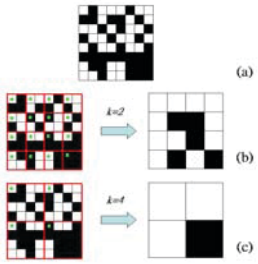

Fig. 3. Alter the resolution of a map. (a) The original map graph. (b) Map with spacing 2 sampling. (c) Map with spacing 4 sampling

$$u = \tau_{ij} + \min\{U_{i-1,j}, U_{i+1,j}, U_{i,j-1}, U_{i,j+1}\} \qquad (3)$$

In the following, we present the algorithm to incarnate the front propagation of FMM.

FMM Algorithm:

Definition. The nodes in the grid map are classified into three categories

1 *Alive (or Known)* is the set of all grid points at which the distance value u has been reached and will not be changed;

(a) *Trial* (or *Near*, *Narrow band*) is the set of next grid points to be examined/readjusted and for which an estimate U of u has been computed using equation (2) or (3) only from *Alive* points. U may be changed later;

2 *Far* is the set of all other grid points for which U is not yet computed;

Initialize

4 Alive points: Let Alive be the set of starting grid point p_0. Set

$$U(p_0) = u(p_0) = 0$$

5 Narrow Band points: Let Narrow Band be the set of all grid points neighboring to p_0 with initial values $U(p) = \tau(p)$;

6 Far Away points: Let Far Away be the set of the rest of the grid points; set their value to ∞;

Marching Forwards Loop

1 Let $p = (i_{\min}, j_{\min})$ be the Trial point with the smallest distance U;

2 Move it from the *Trial* set to the *Alive* set (i.e. $U_{i_{\min}, j_{\min}} = u_{i_{\min}, j_{\min}}$ is frozen);

3 For each nearest neighbor (i, j) (4-connectivity in 2D) of $(i_{\min}, j_{\min})$:

(b) If (i, j) is Far Away, add it to the Trial set and compute a first estimate U of u using equation (2) or (3);

(c) If (i, j) is Trial, update the distance $U_{i,j}$ using equation (2) or (3);

Note that only the Alive points are considered to solve Eikonal equation with upwind finite difference scheme. We examine neighbors of the point being examined and then select the suitable Alive ones. The movement of the front, opposite to the gradient, will point outward from the selected Alive point. As shown in the algorithm, FMM sweeps the front ahead in an upwind fashion by considering a set of points in narrow band around the existing front and march this narrow band forward, freezing the values of existing points and bringing new ones into the narrow band structure. (See Fig. 1 for an illustration of this process). The central idea of narrow band is to build an adaptive mesh around the propagation interface and perform computation only on these grid points. We make a tube containing all the points neighboring to the interface curve within some distance by calculating the signed distance function, points inside the frontier are with negative value and outside ones positive, and using that to select the points. The key is in the selection of which grid point in the narrow band to update with the help of heap operator. Using a min-heap structure for the Trial list, the algorithm computational complexity is $O(N \log N)$ where N is the number of grid points. An optimal path is generated by backtrack the points from goal to start via gradient descent by using the computed u values. The bend of the generated path would be affected by the gradient computation mask and resolution of the map, as we will see in simulations in Sec. 3.

III. Experiments

The test environments shown in Fig. 2 consist of a free environment, a concavity obstacle, and a narrow passage.

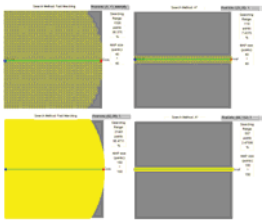

Fig. 4. (a)Case 1: FMM and A* generate identical path in free space

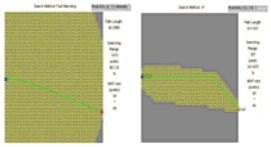

Fig. 4(b). Case 2: FMM (left) generates a shorter path than A* (right) in free space.

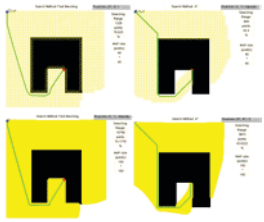

Fig. 5. (a)Simulation results of FMM and A* in environment with cavity

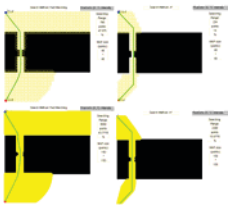

Fig. 5. (b) Simulation results of FMM and A* in narrow passage

All the maps are uniformly discretized into an evenly spaced 2D rectangular grid of nodes with binary values. The nodes of the grid represent samples of the continuous environment, in which 0 is assigned to nodes that obstacle occupies; 1 to free node. This constructs a binary representation of the flat terrain represented by $n \times n$ grid. Of course, grey level representing level of difficulty or danger of the grids surrounding the obstacles [25] or dangerous ground along the way can be assigned so that the robot can move along a path keeping away from the dangerous area. We assume the mobile robot, modeled as a point, can move along each of 8 directions in orthogonal grid environments. In the following experiments performed by A*, whenever a mobile robot moving toward up, down, left, or right the cost is 1 and going diagonally the cost is $\sqrt{2}$ and the heuristic used in A* is the Euclidean distance between the current node and goal node. Although this might not be the true distance between the two nodes, it never overestimates the actual distance since it is the shortest possible route.

To account for each environment whose map may be built from various devices that have different ranges and resolution, we vary the resolution of map and run the A* and FMM again. To vary the resolution of a map from coarser to

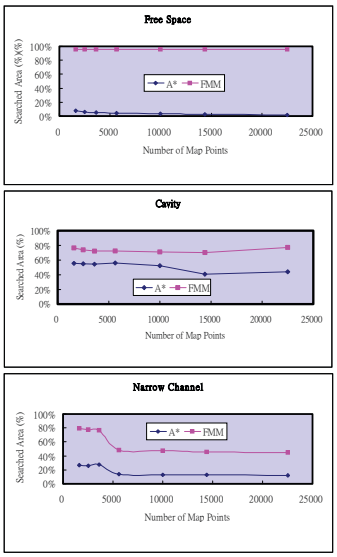

Fig.6. The coverage of FMM and A* in test environments at different resolutions.

finer discretizations, the method we adopt is regular sampling with unique spacing. This is illustrated in Fig. 3. Fig. 3(a) is a 8×8 map. To lessen the amount of information grasped from the map, we sampling the grid points by every interval k. For example, in Fig. 3(b), we set $k = 2$, i.e. beginning from the upper left of the map, the value of each upper left corner point in the 2×2 red block will be sampled. If the value got is 0, then the value represent the red block is 0; similarly, if the value got is 1, then the value assigned to the red block is 1. Fig. 3(c) shows the case of $k = 4$. In implementation, to avoid the confusion of selecting the start and goal point as the resolution decreases, special characteristic points, such as corner or centroid, in a map, are preferred (See Fig. 2 for reference). For coarse resolution, the map size is reduced at the expense of growing number of forbidden nodes. This could eliminate some valid paths.

All tests are run on a PC with Intel Core 2 Duo 1.8GHz microprocessor. The simulation results for visualization of the generated path and the visited area for each method are plotted in Fig.4 for free space, where two cases in free space are simulated, and Fig. 5 for concavity obstacle and a narrow passage between initial and goal. For free space, two cases were run (see Fig.4). Case one is that (start, goal) pair lies on a level line, and Case two is that (start, goal) pair lies on an arbitrary line. For Case one (Fig.4(a)), the paths generated by two methods are identical straight line, while in Case two (Fig. 4(b)) FMM will run nearly diagonal; however, A* would go zigzag. For Fig 5(a) of the map with concavity, both FMM and A* avoid the obstacle and reach the overcurtained goal. In order to reach the goal in Fig. 2(c), the robot must make its way through the narrow passage (note that the map resolution for a successful search should be fine enough). Both FMM and A* generate a path through the narrow passage (see Fig. 5(b)), not trapped by the local minimum, which might happen in potential field method [3]. Fig. 6 shows the coverage of search

space, i.e. percentage of the number of visited points before the solution is found relative to total number of grid points as a measure of efficiency. Unlike running time, coverage is less dependent on the computer used in experiments but more on characteristicsof the method employed.

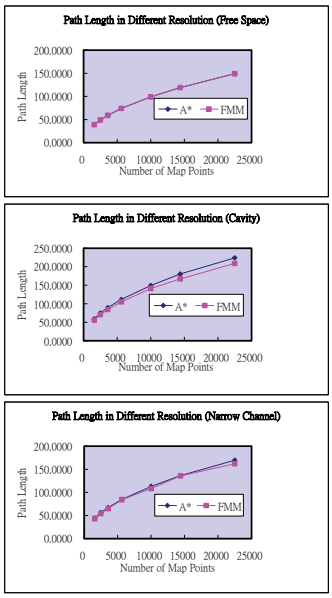

Fig. 7. The path length in different environments.

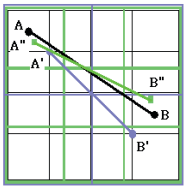

Fig.8.As the resolution goes from 4×4 (black), 3×3 (green) to 2×2 (purple), the path deviates from $\overline{AB}$, $\overline{A'B'}$, to $\overline{A''B''}$.

IV. Comparisons and Discussions

Comparing the paths generated by the grid planners FMM and A* in test environments uniformly discretized at different resolutions, a number of observations can be highlighted to reveal different aspects of the grid planners.

(a)*Efficiency and memory storage*

From Fig. 6, we could find that the searching range of A* SEARCH is smaller than FMM by 20~30 percent. In some extreme case (free map), the gap between them reaches nearly 90 percent. Generally speaking, the decreasing rate of searching range by A* is larger than that by FMM while the resolution increases. These evidences solidify that A* SEARCH is more focused consumes less memory. With the aid of heuristic, A* SEARCH concentrated its exploration around the optimal path. A*would be more efficient than FMM on point-to-point path planning problem [23].

(b) *Quality of generatedl path*

It is visually clear from simulation result of path planning in a map with obstacles (Fig. 5) at different resolutions that the paths produced by FMM are smoother, which was also observed in [15], [25]. We explain this phenomenon as follows. A* generates a path of polygonal line with slopes constrained by the 8 allowable directions, which is continuous but not curvature-continuous. FMM generates the optimal path by employing gradient descent to backtrack the nodes, whose direction is not constrained by the neighboring points around the point being processed to be the next candidate point. With this flexibility, FMM could have more path curvature control capability to comply with the kinematics of mobile robot. [16]-[17] presented a curvature-control FMM to control the curvature of generated path in advance.

From Fig. 7 and Table, it is noted that the path generated by FMM is generically shorter than the ones generated by A* in each test environment. As the resolution of grid map increases, the path length also increases, and the difference between the path length generated by the two methods is more evident. Especially under the swerve case, FMM saves more travel distance than A*. Note that the manipulation of resolution decreasing will result in loss of some information contained in original map and cause an irreversible location deviation (See Fig. 8). The effect disrupts the comparison of path lengths in grid map at different resolutions.

(c)*Convergence*

From Fig. 5 (and other simulations not shown here), we could find that while the grid spacing (i.e. resolution) becomes finer and finer, the path generated by FMM becomes smoother and smoother, by contrast, the path generated of A* changes little. This evidence solidifies that while the resolution rises, FMM could generate solutions convergent to the continuous solution of Eikonal equation (1). This is explained as follows. FMM produces a numerically consistent approximation to solution of continuous Eikonal equation [4], [10]

TABLE PATH LENGTH COMPARISON

		Path Length in Free Environment		Path Length in Concavity		Path Length in narrow passage	
		A*	FMM	A*	FMM	A*	FMM
Map Size	40*40	39.0000	39.0000	59.0416	55.6032	44.7990	43.0915
	50*50	49.0000	49.0000	74.8701	70.5730	56.4558	54.3016
	60*60	59.0000	59.0000	89.5269	84.4587	67.1127	64.9045
	75*75	74.0000	74.0000	111.4260	105.0020	84.7696	84.2960
	100*100	99.0000	99.0000	149.1542	141.1820	112.9120	108.3350
	120*120	119.0000	119.0000	180.2961	166.8210	136.3970	135.6800
	150*150	149.0000	149.0000	223.9530	208.7100	169.5390	161.7970

$$u_x^2 + u_y^2 = c$$

Thus FMM will generate a true collision-free shortest path (i.e. geodesic) as the resolution of the map become finer and finer. On the contrary, as mentioned in [17], A* SEARCH

algorithm, like Dijkstra method, does not converge to the continuous solution even the resolution becomes finer and finer. Instead, it produces the solution to the following PDE

$$\text{Max}\left(\left|u_x\right|,\left|u_y\right|\right) = h * C$$

where h is the grid size. As h goes to zero, this does not converge to the solution of the continuous Eikonal equation (3).

(d) *Some possible ways to improve FMM*

These comparisons suggest that FMM is a useful method for high quality path planning. As a means of shrinking the searching area, one can attempt to add the heuristic function (as A* SEARCH) to FMM. However, the ordering breaks the monotone condition that is required by Fast Marching, so the errors would accumulate. Some literature mentioned that the error is tolerable [11, 12]. Other attempts to enhance the FMM are

(i) Triangulate the map instead of rectangular grid environment to lessen the size of work space and then apply FMM [20]. Facing this more complicated situation, FMM could not work well directly in that the gradient vector may not coincide with the orientation of information propagation.

(ii) According to [22], the time complexity of FMM could reach $O(N)$ by use of cyclic untidy priority queue data structure instead of heap queue in current implementation. The cost is to include mistakes result from fail to select the real minimum value.

ACKNOWLEDGMENT

This research was supported by National Science Council under contract NSC 96-2221-E-001-018-MY2.

REFERENCES

[1] N. Nilsson, Problem-solving methods in artificial intelligence, McGraw-Hill, New York, 1971.

[2] D. Adalsteinsson, and J.A.,Sethian, "A Fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, pp.269-277, 1995.

[3] Jean Claude Latombe, Robot motion planning, Kluwer Academic Publishers, 1991.

[4] J. A. Sethian, Level Set Methods, Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science, Cambridge University Press,1999.

[5] Santiago Garrido, Luis Moreno, Mohamed Abderrahim, Fernando Martin, "Path planning for mobile robot navigation using Voronoi diagram and fast marching", *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 9-15, 2006.

[6] Santiago Garrido, Luis Moreno, Dolores Blanco, "Voronoi diagram and fast marching applied to path planning", *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Oriando, Florida, May 2006.

[7] Santiago Garrido, Luis Moreno, Dolores Blanco, Fernando Martin, "Log of the inverse of the distance transform and fast marching applied to path planning," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 9-15, 2006.

[8] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences USA*, Vol. 93, pp 1591-1595, February 1996.

[9] R. Kimmel and J.A. Sethian, "Optimal algorithms for shape from shading and path planning," *Journal of Mathematical Imaging and Vision*, vol.14, pp.237-244, 2001.

[10] Laurent D. Cohen and Ron Kimmel, "Global minimum for active contour models: A minimal path approach", *International Journal of Computation Vision*, vol. 24, no.1, pp. 57-78, 1997.

[11] Gabriel Peyre and Laurent Cohen, "Heuristically driven front propagation for geodesic paths extraction", *Proceedings of VLSM'05*, Springer LNCS, pp.173-184, Oct. 2005.

[12] Gabriel Peyre and Laurent Cohen, "Landmark-based geodesic computation for heuristically driven path planning", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2229-2236, June 2006.

[13] Sean Mauch, "A Fast Algorithm for Computing the Closest Point and Distance Transform", Tech. Rept. CalTech, Dec. 2000

[14] Sean Mauch, "David Breen: A Fast Marching Method of Computing Closest Points", http://www.cco.caltech.edu/~sean/closestpoint/closept.html

[15] Clement Petres, Yan Pailhas, Pedro Patron, Yvan Petillot, Jonathan Evans, David Lane, "Path planning for autonomous underwater vehicles", *IEEE Trans. Robotics*, vol. 23, no. 2, pp. 331-341, Apr. 2007

[16] Clement Petres, Yan Pailhas, Yvan Petillot, David Lane, "Underwater path planning using fast marching algorithms," *Oceans-Europe 2005,* pp. 814-819, 2005.

[17] J.A. Sethian, "Fast algorithm for optimal control, anisotropic front propagation and multiple arrivals," *Proceedings of the International Congress Mathematicians,* Beijing vol. 3, 735--746, 2002.

[18] J.A. Sethian, and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations," *Proceedings of the National Academy of Sciences*, vol. 98, 11069-11074, 2001.

[19] R. Kimmel, J.A. Sethian, "Computing geodesic paths on manifolds," *Proceedings of the National Academy of Sciences*, pp. 8431-8435, 1998.

[20] Marcin Novotni, and Reinhard Klein, "Computing geodesic distances on triangular meshes," WSCG 2002, pp. 341-348.

[21] Mikhail J. Atallah, Algorithms and theory of computation handbook, CRC PRESS, Chap. 36, 1999

[22] Liron Yatziv, Alberto Bartesaghi, Guillermo Sapiro, "O(N) Implementation of the Fast Marching Algorithm", *Journal of Computational Physics* , vol. 212, pp. 393-399, 2006.

[23] Daniel Livingstone, Robert McDowell, "Fast marching and fast driving: combining off-line search and reactive A.I.," *GAME-ON* 2003.

[24] S. Lavalle, Planning algorithms, Cambridge University Press, New York, 2006.

[25] P. Melchior, B. Orsoni, O.Lavialle, A.Poty and A. Oustaloup, "Consideration of obstacle danger level in path planning using A* and fast-marching optimization: comparative study," *Signal Processing*, vol.11, pp.2387-2396, 2003.

[26] P. Tsiotras and E. Bakolas, "A hierarchical on-line path-planning scheme using wavelets," *European Control Conference*, Kos, Greece, July 2007.

[27] M.S. Hassouna, A.E. Abdel-Hakim and A. Farag, "Robust robotic path planning using level sets," *IEEE Int. Conf. Image Processing*, pp.473-476, 2005.

[28] R. Kimmel and J.A. Sethian, "Optimal algorithm for shape from shading and path planning," *J. Math. Imaging and Vision*, vol.14, no.3, pp.237-244, 2001.