FAST MARCHING AND TRAVELTIME IMAGES: 8-NEIGHBOURHOOD APPROACH

Regis Dargent[1], Olivier Lavialle[1,3], Bruno Orsoni[2], Pierre Baylou[1], Pierre Melchior[2]
[1]Equipe Signal Image – LAP UMR-CNRS, Université Bordeaux I - ENSEIRB,
Av. Du Dr Schweitzer BP. 99, 33402 Talence Cedex France Tel :+33 556 846 185
[2]Equipe Crone - LAP UMR-CNRS Université Bordeaux I - ENSEIRB,
351, Crs de la Liberation, 33405 Talence Cedex France Tel :+33 556 846 607
[3]ENITA de Bordeaux, 33175 Gradignan Cedex France
dargent@tsi.u-bordeaux.fr

ABSTRACT

We develop two approaches for performing accurate traveltime in a 2D lattice. These methods are extensions of the Fast Marching method and consist in considering the 8-connected neighbors to compute the solution of the Eikonal equation. The interest of our approach is demonstrated by comparing the results obtained with the classical Fast Marching algorithm and with our methods in the case of a simple approximation of the Euclidian distance.

1. INTRODUCTION

Accurate and fast traveltime computation is an important task in image processing. It consists in finding the time needed to reach a pixel starting from one or several initial locations (called germs).
A particular case occurs when the time needed to cross a pixel is the same everywhere in the image. Thus, finding the traveltime is equivalent to compute the Euclidian distance to the closest "germ". The problem of using such a method is the complexity of the algorithm. Moreover, it is very difficult to take into account different speeds of travel between pixels.
A solution consists in simulating the evolution of an interface with a speed normal to itself. This approach leads to well known methods like Level Set methods [5] and more recently, Fast Marching methods [6]. Applications of such methods in image processing concern for instance image filtering and enhancement [1] or Active contour models [2][3]. More generally, these methods are also used in many other fields like path planning in robotics [4].

Fast Marching method consists in solving the nonlinear Eikonal equations:

$$|\nabla u(x)| = F(x) \text{ in } \Omega, \quad \mathrm{F(x)} > 0 \tag{1}$$

where Ω is a domain in R^n. $u(x)$ represents the time of arrival to point x, whereas $F(x)>0$ is the time necessary to cross the pixel x.

Typically, Ω corresponds to a 2D or 3D grid, with spacing Δh. In this case, we can approximate each component of the gradient ∇u with the finite difference:

$$u_i = \frac{\Delta u}{\Delta h} \quad \text{along the i}^{\text{th}}\text{ - axis.} \tag{2}$$

In fact, this approximation can be evaluated at first order by two different ways, using the positive or the negative part of the axis:

$$\begin{aligned} u_i^+ &\approx \frac{u(x)-u(x+h_i)}{\Delta h} \\ u_i^- &\approx \frac{u(x)-u(x-h_i)}{\Delta h} \end{aligned} \tag{3}$$

where $h_i=(0,\ 0,\ \ldots,\ 0,\ \Delta h,\ 0,\ \ldots,\ 0)$, the Δh term corresponding to the i$^{\text{th}}$ component.
As the calculus of the traveltime is realized in a growing way, we must have:

$$\begin{cases} u(x) \geq u(x+h_i) \\ u(x) \geq u(x-h_i) \end{cases} \text{, and then : } \begin{matrix} u_i^+ \geq 0 \\ u_i^- \geq 0 \end{matrix} \tag{4}$$

that is, the points used to calculate $u(x)$ must have already been calculated. Then we can precise the definition:

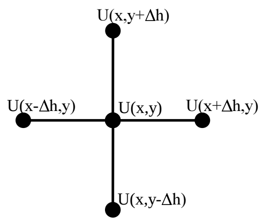
$$\begin{aligned} u_i^+ &\approx Max\left(\frac{u(x)-u(x+h_i)}{\Delta h}, 0\right) \\ u_i^- &\approx Max\left(\frac{u(x)-u(x-h_i)}{\Delta h}, 0\right) \end{aligned} \tag{5}$$

Finally, the choice of u_i^+ or u_i^- comes from the idea that to get the better precision, $u(x)$ has to be calculated from the points that are the closest to the “germs”, i.e. the minimum of $u(x+h_i)$ and $u(x-h_i)$. So we get the formula:

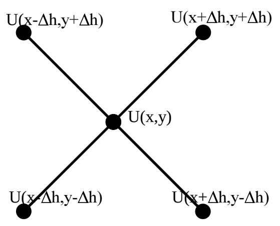
$$u_i \approx Max\left(\frac{u(x)-u(x+h_i)}{\Delta h},\frac{u(x)-u(x-h_i)}{\Delta h},0\right) \quad (6)$$

By this way, we can calculate the traveltime to a point x from the knowledge of the traveltimes of the neighbors of x.

Practically, the computation of the entire image is based on the use of a list of points, *trial*, sorted by growing traveltime. This list represents the wavefront of the propagation of the information. Initially, all the points of the image are set to have an infinite traveltime, unless the germs, which get a traveltime of 0, and are inserted in *trial*. At each step of the algorithm, the head of *trial* (the point with the less traveltime, i.e. the first point reached by the wavefront in its expansion) is extracted, and the traveltime of all of its neighbors is computed. Fig. 1a shows the shape of the neighborhood. If necessary, the traveltime associated with these points is updated, and in this case, they are inserted in *trial* (which is, naturally, sorted anew). The algorithm stops when *trial* contains no more point, that is, when all the points of the image have get their final value.



(a) “Straight” neighborhood



(b) “Diagonal” neighborhood

Fig. 1 Neighborhood of *U(x, y)* in the case of a 2D grid

The key point of this method resides in the sorting of the *trial* list (usually with a heapsort), which reduces the complexity of the algorithm. For example, in the case of a three-dimensional grid with N points in each direction, we get a complexity in $O(N^3 logN)$.

Nevertheless, a drawback of the Fast Marching method is the bias in the estimation of the traveltime: it is perfect in the direction of the axes, but relatively weak in the 45° and 135° directions. This can be shown in the case of a uniform speed of travel on the image. Then the desired solution for each pixel is the Euclidian distance to the closest germ. Fig. 2 shows the difference between the image of Euclidian distance and the one obtained with the Fast Marching method, with a unique germ in the center of the image.

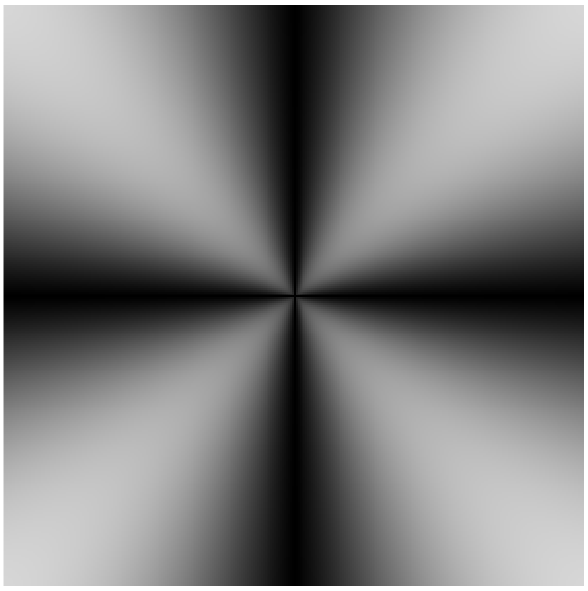


Fig. 2 Estimation error.

In this paper, we develop two methods to improve the precision of the results, while keeping the global structure of the algorithm. The section 2 presents the principle of the two methods, while the section 3 shows some experimental results and comparisons with the original method.

2. IMPROVEMENTS ON THE FAST MARCHING

The Fast Marching leads to a lack of precision along the two diagonals. The main idea of the two methods exposed here is the use of the “diagonal neighborhood” to correct the estimation of the traveltime.

The first method, developed for the 2D case is explained in subsection 2.1. It simply improves the quality of the estimation on the two diagonals, while keeping the precision along the horizontal and vertical axes. The second one, presented in subsection 2.2, implements a somehow different, but better, way to calculate the traveltime.

2.1. Double 4-neighbourhood (4-N) Fast Marching

The first way of using the whole 8-neighbourhood (8-N) is based on the observation that the points of the “diagonal” neighborhood also form a grid, with axes rotated by 45°, and with a spacing of $\Delta h\sqrt{2}$.

The main idea was then to calculate the traveltime to a point using both, but separately, the “straight” and the “diagonal” neighborhood, as shown in Fig. 3. So we get two possible values for the traveltime, $U_{straight}$ and $U_{diagonal}$.

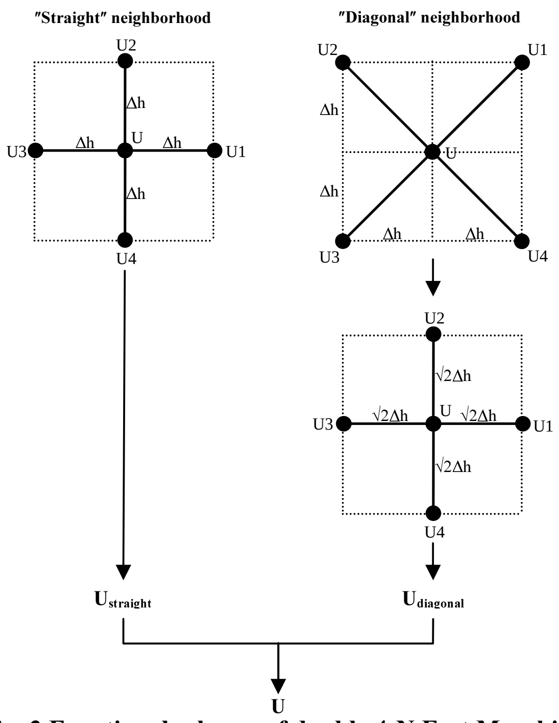


Fig. 3 Functional schema of double 4-N Fast Marching

The choice of the best candidate is based on the fact that the standard Fast Marching always overestimates the traveltime. The value which will be the closest to the truth is then the minimum of $U_{straight}$ and $U_{diagonal}$.

2.2. 8-V Fast Marching

In the Fast Marching method described in section 1, the computation of *u(x)* is obtained by selecting the best neighbor on each axis (i.e. the pixel with the minimal traveltime). (6) can be rewritten:

$$\text{if} \quad \begin{cases} u(x) - u(x+h_i) \geq 0 \\ \text{or} \\ u(x) - u(x-h_i) \geq 0 \end{cases},$$

$$u_i \approx \frac{u(x) - Min\left(u(x+h_i), u(x-h_i)\right)}{\Delta h} \qquad (7)$$

Otherwise,

$$u_i \approx 0$$

The double 4-neighbourhood method proposed in the previous subsection is based on the same principle as it is only an extension taking into account the diagonal axis.

Here, we propose to use the whole 8-neighbourhood to compute an approximation of the gradient based on the general formula:

$$\forall \vec{h}, \nabla u(x).\vec{h} = D_{\vec{h}} u(x) \approx u(x+\vec{h}) - u(x), \qquad (8)$$

where $D_{\vec{h}} u(x)$ is the derivative of *u* at point *x*, in the direction of a vector $\vec{h}$.

To solve (8) we need to obtain the value of *n+1* unknowns (*u(x)* and the *n* components of *∇u(x)*). (1) provides us with one equation, thus we have to introduce *n* other equations by considering the values of the neighbors of *x*.

In the 2D case, the idea is to select the "best" point x_0 in the 8-neighbourhood of the current point *x*. This best point is the one that would give the smallest value for *u(x)*, if it were the only point to use in the calculus. Eventually, to obtain additional information on the gradient, we take into account two other points, x_1 and x_2, which are the neighbors of both *x* and x_0, as shown in Fig. 4.

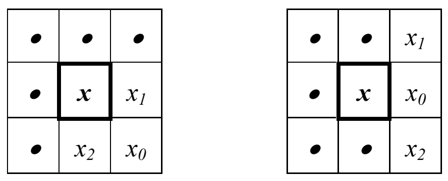


Fig. 4 Two possible repartitions of the interesting neighborhood of *x*

Practically, the gradient components are then defined by:

$$\nabla u(x) \approx \begin{pmatrix} \dfrac{u(x_0) - u(x)}{\|x_0 - x\|} \\ \dfrac{u(x_1) - u(x_2)}{\|x_1 - x_2\|} \end{pmatrix}, \qquad (9)$$

Finally the value of *u(x)* is obtained quite easily by the resolution of the equation (1).

3. EXPERIMENTAL RESULTS

We present here some comparative results obtained with the different methods. In order to evaluate the quality of the resulting images, we considered the 2D problem, with a uniform cost of displacement (i.e. $F(x) = C, \ \forall \mathrm{x}$). A unique germ was placed in the center of the traveltime images and the results of the computing have been compared to the theoretical result, which is the distance image obtained with the Euclidian distance.

The Fig.5 shows the difference between the Euclidian distance and the traveltime returned by the different Fast Marching algorithms. It highlights the directions in which the error is the most important. In particular, it shows how the calculus along the diagonal directions has been improved; Fig. 6 shows more precisely in which proportions the improvements vary according to the direction (the curve has been traced at a distance of 100 pixels of the germs).

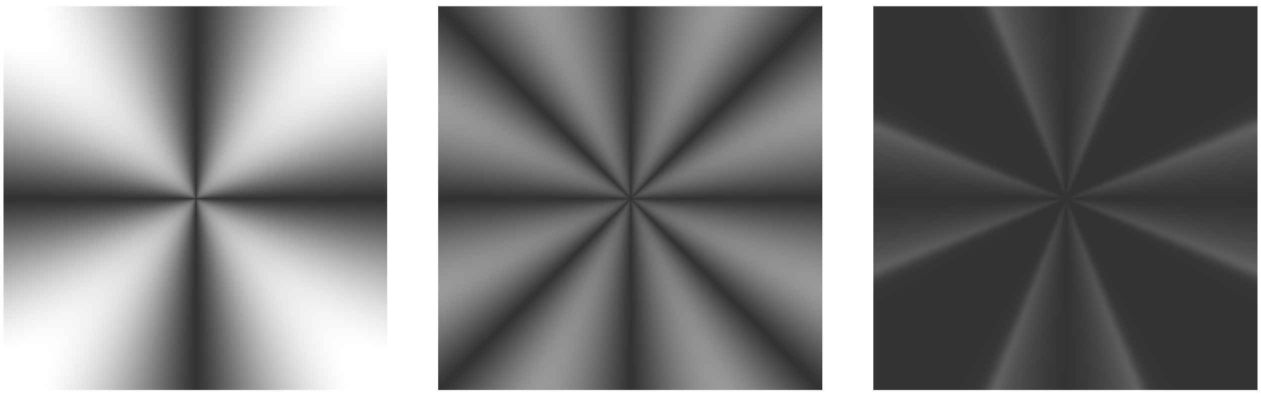


Fig. 5 Error variation according to the direction; black means no error, the same palette of gray levels is used for all images: Standard Fast Marching, (b) Double 4-N, (c) 8-N Fast Marching

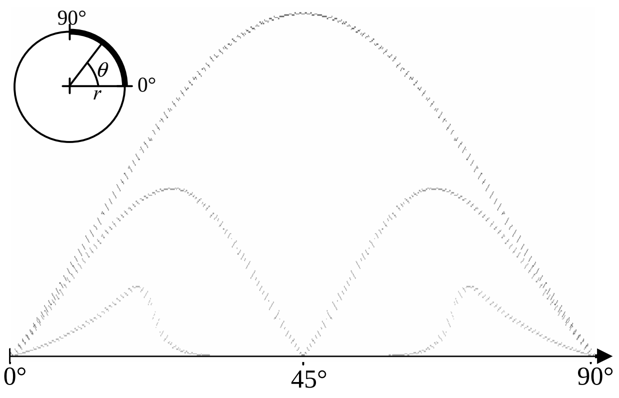


Fig. 6 Curve of error variation according to θ (*r=100*)

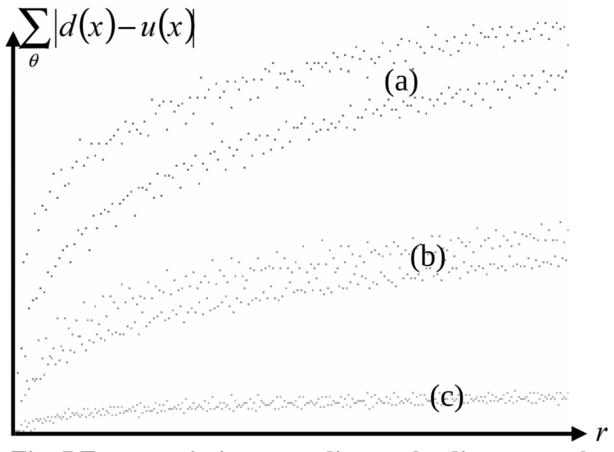


Fig. 7 Error variation according to the distance to the germs for three methods of Fast Marching: (a) Standard (b) double 4-N (c) 8-N

Another way to estimate the improvements given by the two methods consists in observing how the difference between the Euclidian distance and the Fast Marching evolves according to the distance to the germs. The resulting curve is given in Fig. 7.

4. CONCLUSIONS

The new approaches proposed in this paper significantly increase the performance of the Fast Marching algorithm in the places where it is the weakest, which are in the diagonal directions. To do this, they take into account not only the 4-neighbourhood of each point to compute, but its whole 8-neighbourhood.
Further works will be devoted to the 3D extension of our methods. As proposed by Sethian in a recent work [7], we will also extend the neighbourhood of the pixels to compute the traveltime with a greater precision.

4. REFERENCES

[1] L. Alvarez, P.L. Lions, J. M. Morel, ”Image selective smoothing and edge detection by non-linear diffusion”, *SIAM J Numerical Analysis*, 29, pp 845-867, 1992.
[2] L.D. Cohen, R. Kimmel "Global Minimum for Active Contour Models: A Minimal Path Approach", *Inter. Journal of Computer Vision*, Vol. 24, n°1, pp. 57-78, 1997.
[3] O. Lavialle, F. Angella, and P. Baylou, “Extension of the Minimal Path Searching for Structure Recovery", ICIP’99, Kobe, Japan, Vol. 4 pp. 405-409, 1999.
[4] P. Melchior, B. Orsoni, O. Lavialle, A. Oustaloup "The CRONE toolbox for Matlab: Fractional Path Planning Design in Robotics", *IEEE Intern Workshop on Robot and Human Communication*, 10, pp. 534-540, 2001.
[5] S. Osher, J.A. Sethian, “Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations”, *Journal of Comp. Physics*, 79, pp.12-49, 1988.
[6] J.A. Sethian, “Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science”, Cambridge University Press, Cambridge, UK, 1999.
[7] J.A. Sethian, “Fast Marching Methods”, *SIAM Review*, Vol. 41, No.2, pp. 199-235, 1999.