# A Modified Fast Marching Method

Per-Erik Danielsson and Qingfen Lin

Computer Vision Laboratory,
Dept. of Electrical Engineering,
Linköping University, 581 83 Sweden
ped, qingfen@isy.liu.se

**Abstract.** In most, if not all fast marching methods published hitherto, the input cost function and the output arrival time are sampled on exactly the same grid. But since the input data samples are differences of the output samples we found it natural to separate the input and output grid half a sampling unit in all coordinates (two or three). We also employ 8-neighborhood (26-neighborhood in the 3D-case) in the basic updating step of the algorithm. Some simple numerical experiments verify that the modified method improves the accuracy considerably. However, we also feel the modified method leads itself more naturally to image processing applications like tracking and segmentation.

## 1 Introduction

The *minimum-cost path* problem has been considered in different research communities such as robotics, geometric optics, geographic information system, wire routing. One common form of the problem, which has been discussed extensively in the graph theory literature, is to find the least expensive path on a finite graph where different weights (costs) are assigned to every arc linking two vertices.

In image processing applications of this kind, the cost function $\tau(\mathbf{x})$ is given as samples, typically on a 2D or 3D Cartesian grid. The goal is to find a path that minimizes the cumulative traveling time from a given point $A$ to some destination point $B$, which means that we interpret the cost as a delay. The accumulated cost $u(\mathbf{x})$, which may be interpreted as arrival time, is computed along a propagating wave front growing outwards from $A$. When the point $B$ has been reached by the wave front, the path can be found by gradient descent methods in the generated 2D- or 3D-function $u(\mathbf{x})$ . In general, the initial state may consist of a set of points instead of the single point $A$.

The solution to the minimum-cost path problem is in principle given by the Eikonal equation

$$\|\nabla U\| = \tau. \tag{1}$$

An exact solution is usually impossible to find, the problem has to be solved numerically. Fast marching, first proposed in [7] and later named and rederived in [1], is an efficient numerical method applicable to this problem [6]. The strategy

J. Bigun and T. Gustavsson (Eds.): SCIA 2003, LNCS 2749, pp. 1154–1161, 2003.

of fast marching is to introduce an order in the selection of the grid points, in a way similar to Dijkstra's method [2]. This order is based on the causality condition, i.e., the arrival time $u$ at any point depends only on neighbors that have earlier (smaller) arrival times.

The fast marching method can be summarized as follows. There are three sets of points in the 2D- or 3D-space, namely, *Accepted, Trial*, and *Far*. In the initial stage, we label the starting points *Accepted*, the neighbors *Trial*, and all the other points *Far*. The arrival time $u$ for the starting points in the *Accepted* set are set to zero, and the time to reach the neighboring points are computed and recorded. The arrival time at all the other points are set to infinity. Then, given the cost $\tau$, the arrival time to all other points are to be computed using the following two-step loop.

1. Find the point $\mathbf{x}_m$ that has the smallest arrival time in the *Trial* set and move it to the *Accepted* set.
2. Update the arrival times at neighbors of $\mathbf{x}_m$ and move those in the *Far* set to the *Trial* set.

Thus, the fast marching method is picking up points one-by-one from the *Trial* set while gradually expanding the *Accepted* set and diminishing the *Far* set. Multi-pass scans sweeping back and forth over the whole image, which are typical for many distance transforms, have no place in fast matching. It should be noted however, that distance transforms also can be executed using an expanding and propagating wave front, see e.g. [5]. Actually, in case the input function is constantly one, the result $u$ computed by the fast marching should be equivalent to the Euclidean distance to the starting point.

## 2 A modified fast marching method

### 2.1 Shifted input-output grids

In most, if not all fast marching method published hitherto, the cost function $\tau$ and the arrival time $u$ are both defined at the very same grid points as shown in Fig. 1 (a) and (b). When updating the arrival time value $u_C$ from the northwest, the cost $\tau_C$ is used in the quadrant $ABC$. In the same area, if the updating is done from the northeast to the point $B$, the cost used to compute $u_B$ will be $\tau_B$. In fact, the cost defined at a point may be used in any of the four quadrants around it depending on from which direction the front approaches. The influence areas of cost $\tau$ therefore overlap as shown in Fig. 1 (a) and (b). The actual cost becomes dependent on the marching direction.

To overcome this unwanted effect, we propose to define the cost function and compute the arrival time at half-grid offset positions. As in Fig. 1 (c), the input cost function is now defined in the center of a grid cell, i.e. a rectangle bounded by four neighboring grid points. For simplicity, nearest neighbor interpolation is used and the cost is therefore uniform inside each grid cell. The output $u$-values are computed at the grid points as before. If an application requires that input

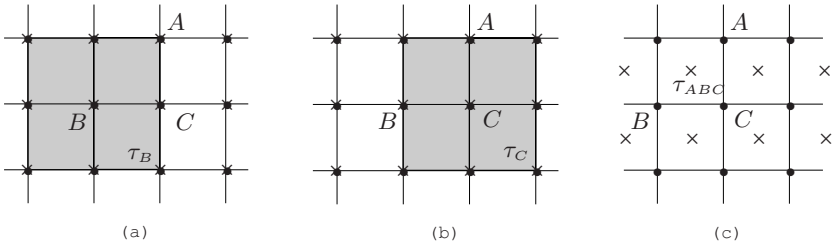


**Fig. 1.** (a, b) In the original fast marching method the cost (×) and the arrival time (•) are defined at the same grid points. The influence areas of $\tau_B$ in (a) and $\tau_C$ in (b) overlap. (c) Alternative sampling pattern where cost (×) is defined inside the grid cell and arrival time (•) is computed at the grid point.

function $\tau$ and output function $u$ must have a common grid, we suggest that one either resamples the input function before or resamples the output after the marching process to obtain the sampling pattern of Fig. 1 (c).

Under the new sampling condition, the causality condition still holds. Thus, the main loop of the fast marching is used as usual and the computation of the arrival time is done by

$$u_{AB}(C) = \min_{0 \le t \le 1} (t u_A + (1-t) u_B + \sqrt{t^2 + (1-t)^2}\tau_{ABC}) \tag{2}$$

for the point $C$ in Fig. 1 (c) in the quadrant $ABC$. The parameter $t$ decides where the propagation is from and $\tau_{ABC}$ is the cost defined inside the grid cell surrounded by grid points $A, B$ and $C$. We name the modified fast marching method the shifted-grid (SG) fast marching method.

### 2.2 The 2D case with 8-connected neighbors

Another modification we made to the original fast marching is to include all the 8-connected neighboring points instead of the 4-connected points when computing the $u$-values. The use of eight neighbors has also been suggested in [3] and [7], although not in connection with shifted input-output grids. The geometry using 8-connected neighbors is shown in Fig. 2 (a). To update the arrival time $u_C$ from the octant given by $AB$ is to solve

$$u_{AB}(C) = \min_{0 \le t \le 1} (t u_A + (1-t) u_B + \sqrt{1 + t^2} \cdot \tau_{ABC}). \tag{3}$$

This minimization can be solved explicitly. The closed form solution is

$$u_{AB}(C) = \begin{cases} u_B + \tau_{ABC} & \text{if } u_B \le u_A, \\ u_A + \sqrt{2} \cdot \tau_{ABC} & \text{if } \tau_{ABC} \le \sqrt{2} \cdot (u_B - u_A), \\ u_B + \sqrt{\tau_{ABC}^2 - (u_B - u_A)^2} & \text{Otherwise.} \end{cases} \tag{4}$$

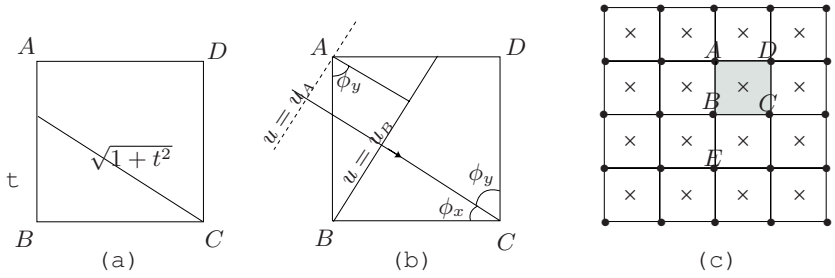


**Fig. 2.** (a) The arrival time at point $C$ is to be computed from the 8-connected neighboring points $A$ and $B$. (b) Alternative interpretation of (a) given by the direction cosines. (c) A larger neighborhood around point $C$.

where $A$ and $B$ are diagonal neighbors, $A$ is diagonal neighbor to $C$ and $B$ is 4-connected to $C$ as in Fig. 2.

More intuitively, we can deduce the result using the direction cosines (see Fig. 2 (b)). A planar wave is assumed to reach point $C$ from the line segment $AB$. The direction of arrival is perpendicular to the wave front. The time differences are then

$$u_B - u_A = \tau_{ABC} \cos \phi_y, \quad u_C - u_B = \tau_{ABC} \cos \phi_x.$$

where $\cos^2 \phi_x + \cos^2 \phi_y = 1$.

The causality condition still holds under the new geometry as proven in [4] using the Kuhn-Tucker condition. The main loop of fast marching can thus be applied. To implement the 8-neighbor shifted-grid fast marching method efficiently, we use the following observations. First, when a *Trial* point becomes accepted, we only need to recompute the arrival times at its neighbors from the octants that include the newly accepted point. For example, assume $B$ is the newly accepted point and the arrival times at its neighbors are to be computed (Fig. 2 (c)). At point $C$, only computations from line segments $AB$ and $EB$ are necessary. The smallest value among $u_{AB}(C)$, $u_{EB}(C)$ and the old $u_C$ will be assigned to $C$.

Furthermore, we note that updating a diagonal neighbor such as point $D$ from $B$ in Fig. 2 is unnecessary. The reason is as follows. If $D$ is actually dependent on $B$, that is the wave is coming from either $BA$ or $BC$, one of the inequalities $u_B < u_A < u_D$ and $u_B < u_C < u_D$ must hold. Thus the correct $u$-value will be assigned to $D$ later on when either $A$ or $C$ becomes accepted. An efficient updating step is given in Algorithm 1.

### 2.3 The 3D case

In the 3D case, the shifted input-output grid corresponds to the well known body-centered grid. The total set of grid-points can be divided into two half-sets so that each grid point has 8 neighbors, all of which belong to the other

**Algorithm 1** Update neighbors of $\mathbf{x}_m$, shifted grid, 8-neighbor scheme

– Definition: *Near* set is defined for every *Trial* point $\mathbf{x}$ as

$$NS(\mathbf{x}) = \{\mathbf{x}_j\mathbf{x}_k \mid \text{4-connected points to each other,}$$
$$\text{one is diagonal neighbor to } \mathbf{x},$$
$$\text{the other one is 4-connected neighbor to } \mathbf{x}\}.$$

– For each non-accepted 4-connected point $\mathbf{x}$ of $\mathbf{x}_m$,
- If $\mathbf{x}$ is *Far*, move it to the *Trial* set;
- $u(\mathbf{x}) = \min\{u(\mathbf{x}), \min_{\mathbf{x}_j\mathbf{x}_m \in NS(\mathbf{x})}(u_{\mathbf{x}_j\mathbf{x}_m}(\mathbf{x}))\}$ with $u_{\mathbf{x}_j\mathbf{x}_m}(\mathbf{x})$ computed by (4);

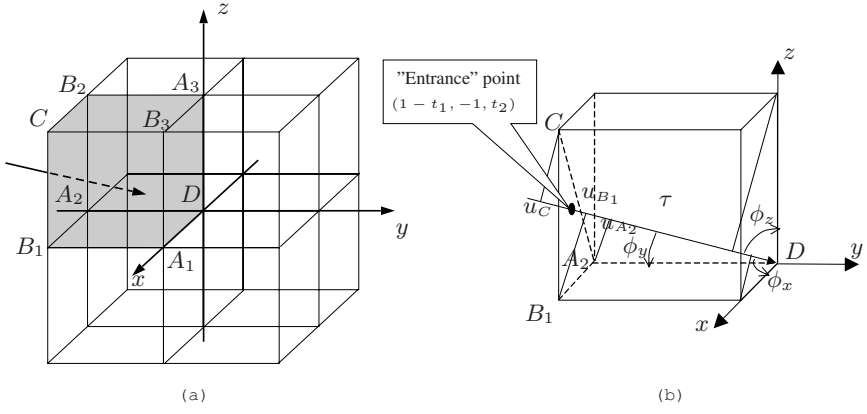


**Fig. 3.** (a) 3D Cartesian grid used in the fast marching method. The arrival time at point $D$ is to be computed from the direction given by the arrow. (b) The arrival time at point $D$ to be computed from 26-connected neighbors.

half-set. Consequently, an input cost value is located in the middle of a grid cell which is a cube bounded by eight neighboring grid points. In the 3D case, we might consider the possibilities to compute the arrival time at point D in Fig. 3 from three 6-connected neighbors, three 18-connected neighbors, or three 26-connected neighbors. For the sake of brevity, we will only discuss the 26-neighbor case. A treatment of the other two cases can be found in [4].

Assume that the wave front is arriving from the direction given by the arrow in Fig. 3 (a) and the cost is $\tau$ inside the shaded cube. Depending on surrounding arrival times, the arrival time at $D$ is to be computed as if the propagating front is entering the shaded cube via one of several triangular surfaces patches. These triangles have three vertices that are 6-connected, face-diagonal and body-diagonal neighbors to $D$, respectively. There are 48 such triangles in total. In Fig. 3 (a) and (b), $A_2B_1C$ is such a triangle. A line perpendicular to the planar wave front through point $D$ is given in Fig. 3 (b). This line is intercepted by

wave fronts with arrival times $u_{A_2}$, $u_{B_1}$, and $u_C$ as indicated. The time differences between points $D$, $A_2$, $B_1$, and $C$ should then satisfy

$$u_D - u_{A_2} = \tau \cos\phi_y, \quad u_{A_2} - u_{B_1} = \tau \cos\phi_x, \quad u_{B_1} - u_C = \tau \cos\phi_z, \tag{5}$$

where the direction cosines satisfy

$$\cos^2\phi_x + \cos^2\phi_y + \cos^2\phi_z = 1.$$

This gives $u_D = u_{A_2} + \sqrt{\tau^2 - (u_{A_2} - u_{B_1})^2 - (u_{B_1} - u_C)^2}$.

A more strict analytical treatment requires that we compute the arrival time at $D$ from $A_2B_1C$ as

$$u_{A_2B_1C}(D) = \min_{t_1,t_2}(t_1 u_{A_2} + t_2 u_C + (1 - t_1 - t_2)u_{B_1} + \sqrt{1 + (1 - t_1)^2 + t_2^2} \cdot \tau) \tag{6}$$

$$\text{s.t. } t_1 \geq 0, t_2 \geq 0 \text{ and } t_1 + t_2 \leq 1.$$

Such a minimization problem can be solved explicitly using the Kuhn-Tucker condition in a similar manner as done in [7]. If the minimum is not found inside the triangle $A_2B_1C$, the boundaries and the vertices will be searched. Details on solving (6) are given in [4].

The two-step loop of the fast marching method remains the same in the 3D case. In fact, the argument given in Section 2.2 for updating only the four nearest neighbors in Algorithm 1 still holds so that in the 3D case, only the six nearest neighbors have to be updated.

### 2.4 Numerical experiments

To compare the proposed shifted-grid fast marching methods with the original fast marching, we give the following numerical examples. The cost functions are chosen so that the analytical result of the arrival time $u_{analytic}$ can be found. Three types of cost functions are used, namely:

1. $\tau(\mathbf{x}) = h(|\mathbf{x} - \mathbf{x}_0|)$. The cost $\tau$ is defined as a function of the distance from the starting point $\mathbf{x}_0$.
2. $\tau(\mathbf{x}) = 1/(ax + by + c)$. The reciprocal of the cost function (speed) is a linear function.
3. $\tau(\mathbf{x}) = \sqrt{(\partial u/\partial x)^2 + (\partial u/\partial y)^2}$. The cost $\tau$ is computed from a continuous and differentiable $u$ function according to the Eikonal equation.

The three cost functions used in the following examples are

$$\tau_1(\mathbf{x}) = 1, \quad \tau_2(\mathbf{x}) = \frac{1}{x+1}, \quad \tau_3(\mathbf{x}) = \frac{1}{20}\sqrt{\left(\sin\frac{x}{20}\cos\frac{y}{20}\right)^2 + \left(\cos\frac{x}{20}\sin\frac{y}{20}\right)^2}.$$

The cost functions for the original and the shifted-grid fast marching methods are sampled at half-grid offset as in Fig. 1 (c), so that the output $u$-values are

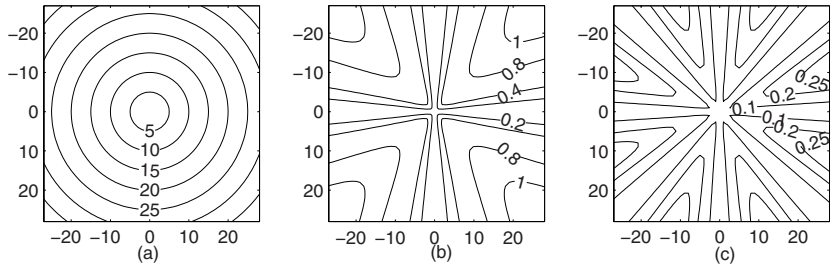


**Fig. 4.** Cost function $\tau_1(\mathbf{x}) = 1$. (a) Iso-curves of the correct $u$-value, a Euclidean distance map. Iso-curves of errors when using FM (b) and SGFM-8 (c).

| $L_\infty$ Error | $\tau_1$ | $\tau_2$ | $\tau_3$ |
|---|---|---|---|
| FM: | 1.1290 | 0.2209 | 3.8709 |
| SGFM-4: | 1.1290 | 0.1863 | 3.0939 |
| SGFM-8: | 0.2754 | 0.1863 | 0.5245 |

| $L_2$ Error | $\tau_1$ | $\tau_2$ | $\tau_3$ |
|---|---|---|---|
| FM: | 0.4991 | 0.0240 | 7.9358 |
| SGFM-4: | 0.4991 | 0.0048 | 0.8526 |
| SGFM-8: | 0.0276 | 0.0004 | 0.0504 |

**Table 1.** $L_\infty$- and $L_2$- Errors of the arrival time computed with different fast marching method.

aligned and can be compared. Furthermore, since the analytical results of the $u$-values of these three types of cost functions are easy to compute, these results are used as the golden standard for comparison. In all three examples, there are $56 \times 56$ grid points with $\Delta x = \Delta y = 1$. The starting point is in the center of the images for $\tau_1$ and $\tau_3$ and in the middle of the top row for $\tau_3$. The three different fast marching methods used are:

**FM:** the original 4-neighbor fast marching,
**SGFM-4:** the shifted-grid 4-neighbor fast marching,
**SGFM-8:** the shifted-grid 8-neighbor fast marching.

Cost function $\tau_1$ transforms the problem to the common distance transform. The Euclidean distance map is shown in Fig. 4 (a). We also notice that SGFM-4 will give exactly the same result as FM in this case since the cost is homogeneous. The results of FM and SGFM-8 is shown in Fig. 4 (b) and (c) in the form of error images.

We measure the numerical errors as

$$L_\infty \text{ Error } = \max(|u - u_{analytic}|), \qquad L_2 \text{ Error } = \text{mean}|u - u_{analytic}|^2$$

and list the result in Table. 1.

From Fig. 4, Table. 1 and the result figures for $\tau_2$ and $\tau_3$ which are not shown, we observe that:

– The 4-neighbor fast marching methods generate small errors along the vertical and horizontal direction. Errors increase rapidly along the diagonal direction.

- The 8-neighbor fast marching method generates small errors along the vertical, horizontal and diagonal direction. Errors in the regions between these directions are large.
- The results given by the 4-neighbor shifted-grid fast marching method are more accurate than those given by the original fast marching for $\tau_2$ and $\tau_3$.
- The results given by the 8-neighbor fast marching are more accurate than those given by the 4-neighbor fast marching methods for all three examples.

## 3 Discussion

To summarize, the fast marching method we have proposed has two significant properties. The first and most important one is to shift the input and output grids half a sampling distance relative to each other. Hereby, we can understand the sampled input image in traditional signal processing terms, define a basis function to connect the sampled function with the underlying continuous one etc. But we also get rid of the feature in the original method that makes the cost dependent on the wave propagation direction in a rather unpredictable and data-dependent manner.

The second property follows naturally from the previous one, namely the possibility to employ not only the 4 (6) nearest neighbors but the full 8 (26) neighbors in the updating procedure. Numerical experiments show that this approach gives more accurate results.

On the other hand, the shifted-grid fast marching methods do require more computation. In the 2D case, they tend to double the amount of the computation of the original fast marching method. In Fig. 1 and Fig. 2, when point $C$ is updated from $B$, propagation delays via two quadrants have to be evaluated. In the original fast marching where the cost is defined at point $C$, only one such evaluation is necessary.

## References

1. D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *Journal of Computational Physics*, 118:269 – 277, 1995.
2. E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269 – 271, 1959.
3. Seongjai Kim. An $O(N)$ level set method for eikonal equations. *SIAM J. Sci. Comput.*, 22(6):2178 – 2193, 2001.
4. Qingfen Lin. *Enhancement, extraction, and visualization of 3D volume data.* PhD thesis, Linköping University, 2003.
5. Ingemar Ragnemalm. Neighborhoods for distance transformation using ordered propagation. *CVGIP - Image Understanding*, 56:161 – 166, 1992.
6. J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1999.
7. John N. Tsitsiklis. Efficient algorithm for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.