Fully Isotropic Fast Marching Methods on Cartesian Grids

Vikram Appia and Anthony Yezzi

Georgia Institute of Technology, GA, USA

Abstract. The existing Fast Marching methods which are used to solve the Eikonal equation use a locally continuous model to estimate the accumulated cost, but a discontinuous (discretized) model for the traveling cost around each grid point. Because the accumulated cost and the traveling (local) cost are treated differently, the estimate of the accumulated cost at any point will vary based on the direction of the arriving front. Instead we propose to estimate the traveling cost at each grid point based on a locally continuous model, where we will interpolate the traveling cost along the direction of the propagating front. We further choose an interpolation scheme that is not biased by the direction of the front. Thus making the fast marching process truly isotropic. We show the significance of removing the directional bias in the computation of the cost in certain applications of fast marching method. We also compare the accuracy and computation times of our proposed methods with the existing state of the art fast marching techniques to demonstrate the superiority of our method.

Keywords: Fast Marching Methods, Isotropic Fast Marching, Segmentation, Tracking, FMM, Eikonal Equation, minimal cost path.

1 Introduction

A large number of computer vision applications such as segmentation, tracking, optimal path planning *etc*. use the minimal cost path approach. The Fast Marching Method which is widely used to solve the minimal path problem was first introduced by Sethian [1,10] and Tsitsiklis [11]. Cohen and Kimmel [4,5] later noticed that the minimal cost problem satisfies the Eikonal equation,

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

For the Eikonal equation 1 defined on a Cartesian Grid, $\tau(x)$ would be the traveling cost at a given grid point and $u(x)$, the accumulated cost. Since we solve the Eikonal equation numerically on Cartesian Grids, it is impossible to find the exact solution. Some modifications have been suggested in [6,7] to improve the accuracy of the Fast Marching method. Authors in [6,8,9,11] also suggest using an 8-connected neighbor scheme to improve accuracy. All these techniques use a locally continuous model to estimate the accumulated cost, but assume the traveling cost to be constant (discretized) around each grid point. Only [6] interpolates τ by shifting it to the center of the grid with a nearest neighbor interpolation, but it still assumes a discretized shifted grid for τ. In this paper we propose to use a locally continuous model to estimate τ as well.

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part VI, LNCS 6316, pp. 71–83, 2010.

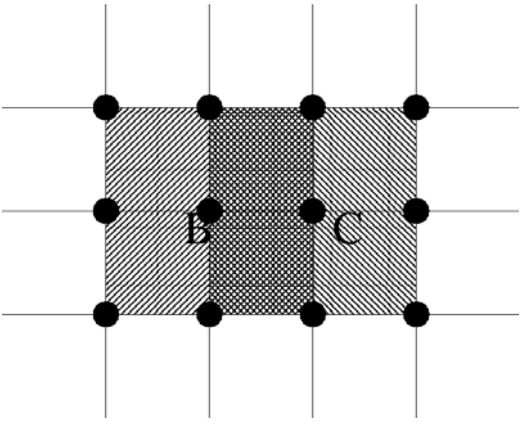


Fig. 1. Overlap in the influence areas of τ_B and τ_C

For the geometry shown in Figure 1, the Fast Marching Method uses linear approximation to compute the accumulated cost at the point C, but it uses a constant traveling cost τ_C for each of the four grid cells containing the point C. The influence area of the cost function given at a grid point will include all the four quadrants around it. Thus, there is an overlap in the areas of influence of the grid points B and C. This means the value of u_C will vary depending on the direction from which the front is arriving. Ideally, for isotropic fast marching, the accumulated cost should be independent of the direction of the arriving front. For the image shown in Figure 2, we use the traveling cost, $\tau(x) = I(x)$, where I(x) is the intensity at each pixel. The accumulated cost in traveling from point A to B should be equal to the cost in traveling from B to A. But, due to the dependence on the direction of marching, there will be a difference in the accumulated costs. Figure 2 compares the minimal path obtained using back propagation from end point B to the source point A with the minimal path obtained by reversing the direction of front propagation. The difference in the two paths highlights the error caused by the directional dependence of the Fast Marching method.

In this paper we propose two methods to overcome the above-mentioned shortcomings. The first method uses a linear/bilinear model locally to estimate τ along the direction of the propagating front within each grid cell. Here we use a continuous model to estimate τ and also take the direction of arrival into consideration. We also discuss how the scheme can be made truly isotropic by removing any bias due to the marching direction. We call this method the Interpolated Fast Marching Method and it is discussed in detail in Section 2. In the second method we calculate u on an upsampled grid. In upsampling the grid, τ in the neighborhood of each grid point becomes constant, which eliminates the need to estimate τ using a continuous model. We will use the value of τ from the direction of arriving front. The upsampled version of the 4 and 8-connected neighbor schemes are discussed in Section 3. Finally, in Section 4 we describe a few numerical experiments conducted to highlight the significance of making the fast marching method independent of direction and we test the accuracy of the proposed methods.

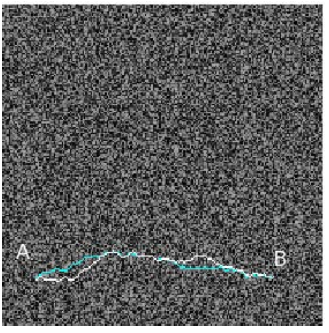


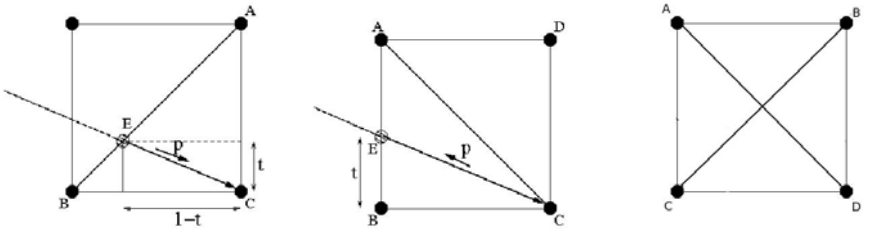
Fig. 2. Image with random noise

2 Interpolated Fast Marching Method

For interpolated Fast Marching scheme we will assume τ to be continuous around each grid point and use linear/bilinear interpolation to estimate the value of the local traveling cost within each grid cell. Here we will derive the equations for the linear and bilinear Interpolated Fast Marching schemes. To estimate the traveling cost in a grid cell, the bilinear scheme will use the value of τ from all the grid points for a given quadrant. Since only 2 neighbors are used in each quadrant to calculate u in a 4-connected neighbor scheme, we only discuss the 8-connected neighbor scheme with bilinear interpolation.

2.1 Linear Interpolation

4-Connected Neighbors Scheme. Consider a front arriving at the grid point C from the quadrant AB and intersecting $\overline{AB}$ at E as shown in Figure 3(a). We will use the linear interpolation of the local traveling cost along the path $\vec{EC}$ to compute u_C. Thus the accumulated cost at C will be,



(a) 4-Connected Neighbors Scheme (b) 8-Connected Neighbors Scheme (c) Isotropic triangulation of a Grid Cell

Fig. 3. Triangulation of Grid cells

$$u_C = \min_{0\le t\le 1}\Big\{u_B(1-t) + u_A t + \int_0^1 \tau(p)\sqrt{t^2+(1-t)^2}dp\Big\}. \tag{2}$$

Substituting, $\tau(p) = \tau_C + (\tau_A - \tau_C)p(1-t) + (\tau_B - \tau_C)pt$, $0 \le p \le 1$, in (2) we get,

$$u_C = \min_{0\le t\le 1}\Big\{u_B(1-t) + u_A t + \sqrt{t^2+(1-t)^2}\left(\frac{\tau_A+\tau_C}{2} + \frac{\tau_B-\tau_A}{2}t\right)\Big\}. \tag{3}$$

We get the necessary optimality condition to obtain the minimum of u_C by solving $\frac{du_C}{dt} = 0$, which yields,

$$\begin{aligned} &u_A - u_B + \sqrt{t^2+(1-t)^2}\left(\frac{\tau_B-\tau_A}{2}t\right) \\ &+ \frac{2t-1}{\sqrt{t^2+(1-t)^2}}\left(\frac{\tau_A+\tau_C}{2} + \frac{\tau_B-\tau_A}{2}t\right) = 0. \end{aligned} \tag{4}$$

8-Connected Neighbors Scheme. The geometry for 8-connected neighbors is shown in Figure 3(b). Using linear interpolation to estimate the local traveling cost along $\vec{EC}$, the accumulated cost, u_C, will be,

$$u_C = \min_{0\le t\le 1}\Big\{u_B(1-t) + u_A t + \int_0^1 \tau(p)\sqrt{1+t^2}dp\Big\}. \tag{5}$$

Substituting, $\tau(p) = \tau_C + (\tau_B - \tau_C)p + (\tau_A - \tau_B)pt$, $0 \le p \le 1$, in (5) we get,

$$u_C = \min_{0\le t\le 1}\Big\{u_A t + u_B(1-t) + \sqrt{1+t^2}\left(\frac{\tau_B+\tau_C}{2} + \frac{\tau_A-\tau_B}{2}t\right)\Big\}. \tag{6}$$

Again the minimizer of u_C can be obtained by solving $\frac{du_C}{dt} = 0$. Thus we have,

$$u_A - u_B + \sqrt{1+t^2}\left(\frac{\tau_A-\tau_B}{2}\right) + \frac{t}{\sqrt{1+t^2}}\left(\frac{\tau_B+\tau_C}{2} + \frac{\tau_A-\tau_B}{2}t\right) = 0. \tag{7}$$

Isotropic Linear Interpolation Scheme. Figure 3(a) and 3(b) show the triangulation of a grid cell for the 4 and 8 neighbor schemes respectively. Depending on the front direction one of the quadrant/octant will be chosen to estimate the accumulated cost. But this will induce a directional bias. To overcome this directional bias, we will have to consider all possible triangulations shown in Figure 3(c). In effect the accumulated cost across a grid cell must be the minimum of the solutions obtained using the 4 and 8 neighbor schemes. This would make the scheme completely unbiased to direction and we call this scheme the Iso-Linear scheme.

2.2 Bilinear Interpolation

8-Connected Neighbors Scheme. The bilinear interpolation to estimate the local traveling cost along $\vec{EC}$ is given by,

$$\tau(p) = \tau_A(p)(pt) + \tau_B(p)(1-pt) + \tau_C(1-p)(1-pt) + \tau_D(1-p)(pt).$$

It is inherently independent of any directional bias within a grid cell. Substituting, this value of $\tau(p)$ for $0 \leq p \leq 1$, in (5) we get,

$$u_C = \min_{0\leq t\leq 1} \left\{ u_A t + u_B(1-t) + \sqrt{1+t^2} \left(\frac{\tau_B + \tau_C}{2} + \frac{\tau_A - \tau_B}{3} t + \frac{\tau_D - \tau_C}{6} t \right) \right\}. \tag{8}$$

We will again solve $\frac{du_C}{dt} = 0$, which yields,

$$\begin{aligned} &u_A - u_B + \sqrt{1+t^2} \left(\frac{\tau_A - \tau_B}{3} + \frac{\tau_D - \tau_C}{6} \right) \\ &+ \frac{t}{\sqrt{1+t^2}} \left(\frac{\tau_B + \tau_C}{2} + \frac{\tau_A - \tau_B}{3} t + \frac{\tau_D - \tau_C}{6} t \right) = 0. \end{aligned} \tag{9}$$

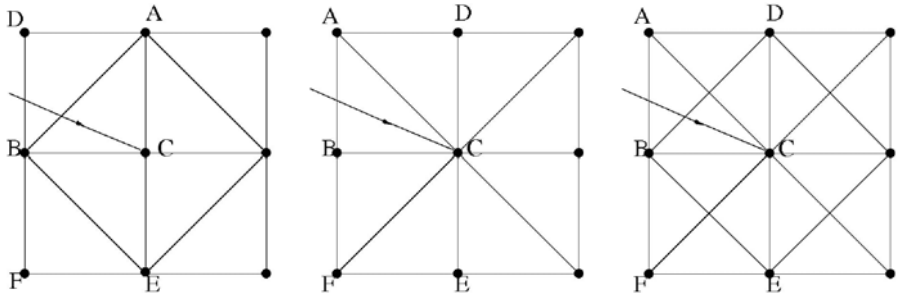
Algebraic manipulations on (4), (7) and (9) will yield quartic equations. We used the Ferrari and Newton methods to solve these quartic equations. We compared the solutions from both techniques and found that they generate equally accurate solutions. Since Newton's method has a quadratic convergence, three iterations were sufficient for convergence. Fixing the number of iterations in each update step also ensures that we have the same computation complexity in each update. This makes the technique suitable to implement on hardware. The solution to Newton's method has fewer (logical and mathematical) operations in comparison to finding the Ferrari (analytic) solution; hence using Newton's method is computationally efficient. We compare the computation times of the two methods on a 500x500 grid in the Table 1. Here we call the 4 and 8-connected neighbor linear Interpolated Fast Marching schemes, Linear-4 and Linear-8 respectively and the 8-connected neighbor bilinear Interpolated Fast Marching scheme, Bilinear-8. The computation times were measured on a laptop with a 1.73 GHz Processor.

Table 1. Comparison of computation times

	Linear-4	Linear-8	Bilinear-8
Analytic (Ferrari)	1.51s	2.83s	3.23s
Newton's Method	0.51s	0.52s	0.65s

2.3 Marching Forward Loop

We will still follow the main loop as explained in the basic Fast Marching method [10]. But, when a $trial$ point is $accepted$ in the min heap structure we will compute the value of u from both the quadrants/octants which include the newly $accepted$ point and replace the newly calculated u with the minimum of the two solutions and the existing value of u (if the point is marked as $trial$).



(a) 4-Connected Neighbors Scheme (b) 8-Connected Neighbors Scheme (c) Isotropic Fast Marching Scheme

Fig. 4. B is the newly $accepted$ grid point and u_C is to be computed

Consider the example in Figure 4(a) where B is the newly $accepted$ point and the accumulated cost at neighbor C is to be computed. As opposed to the basic fast marching technique, u_C does not solely depend on u_A,u_B,u_E and the local traveling cost, τ_C, but it also depends on the costs at all the other 8-connected neighbors. Thus, using the quadrant containing the minimum of u_A and u_E will not necessarily guarantee the minimum solution to (3). Hence we have to consider both the quadrants that contain B. If the front also arrives at C from the other two quadrants, they will be considered when the corresponding neighbors become $accepted$. The same argument can be extended to the 8-connected neighbor case shown in Figure 4(b). Here we only need to calculate u_C from the two octants containing $\overline{AB}$ and $\overline{FB}$ once point B is $accepted$. For the front arriving at point C as shown in Figure 2(c), we will consider the possibilities of the front arriving from $\overline{AB}$,$\overline{BD}$ and $\overline{DA}$.

We depart from the traditional Fast Marching method only in the update procedure for the accumulated cost, but follow the same main (outer) loop. Thus the parallel algorithm explained in Bronstein et al.[2], can be extended for the implementation on hardware.

3 Upsampled Fast Marching Method

Figure 5 shows that there is no overlap in the influence areas of τ on the upsampled grid. Here the solid circles are the grid points from the original grid. Since the traveling cost is constant in each grid cell, there is no directional bias in the calculation of u. We will compute u on the upsampled grid and then downsample the output on the original grid.

3.1 4-Connected Neighbors Scheme

In the upsampled grid, τ is constant in each quadrant around a grid point. Again the constant traveling cost within each grid cell makes this scheme isotropic. Depending on the direction of the front we will choose the value of τ in calculating u. For example, if

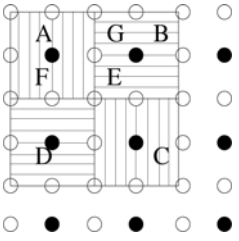


Fig. 5. No overlap in the influence areas of τ_A, τ_B, $\tau_C$and τ_D

the front arrives at E from the north-west then we would use τ_A (Figure 5). At the point G we would use τ_A for a front arriving from the west and τ_B for a front arriving from the east. We would use τ_A to calculate u_A irrespective of the direction of the arriving front. Since the value of τ is constant along the direction of the front at a sub-pixel level, it is not necessary to assume a locally continuous model in interpolating τ. Thus, the accumulated cost at E with the front arriving from the north-west would be,

$$u_E = \min_{0 \le t \le 0.5} \left\{ u_F t + u_G(0.5 - t) + \tau_A \sqrt{t^2 + (0.5 - t)^2} \right\} \tag{10}$$

This minimization leads to the closed form solution,

$$u_E = \begin{cases} \frac{(u_F + u_G + \sqrt{\delta})}{2} & \text{if } \delta \ge 0 \\ \min(u_F, u_G) + \frac{\tau_A}{2} & \text{otherwise} \end{cases}$$

where, $\delta = \frac{\tau_A^2}{2} - (u_F - u_G)^2$.

3.2 8-Connected Neighbors Scheme

As in the case with 4-connected neighbors, τ is constant in each octant around a grid point in the upsampled grid. We note that there will be exactly one point in each octant that corresponds to a point in the original grid. We will use the corresponding value of τ to compute u.

By following the procedure described in Section 2.3, we calculate u only from the two octants that contain the newly *accepted* point. If F is the newly *accepted* point, we will calculate u_E in the octants containing $\overline{FA}$ and $\overline{FD}$ (Figure 5). The solution will be the minimum of the two values obtained. Thus, for a front arriving from north-west, the accumulated cost at E will be,

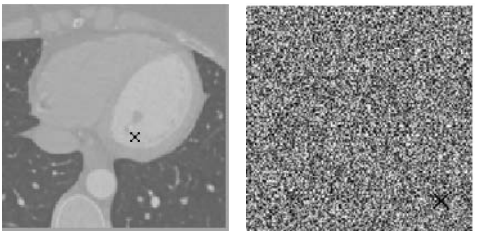
$$u_E = \min_{0 \le t \le 0.5} \left\{ u_A t + u_F(0.5 - t) + \tau_A \sqrt{0.5 + t^2} \right\} \tag{11}$$

giving the closed form solution,

$$u_E = \begin{cases} u_F + \frac{\tau_A}{2} & \text{if } u_F \leq u_A \\ u_A + \sqrt{2}\frac{\tau_A}{2} & \text{if } \tau_A \leq 2\sqrt{2}(u_F - u_A) \\ u_F + \frac{\sqrt{\tau_A^2 - 4(u_F - u_A)^2}}{2} & \text{otherwise} \end{cases}$$

4 Numerical Experiments

We conducted a few experiments to compare the proposed methods to the basic Fast Marching Method (FMM) [10], Tsitsiklis scheme [11], Shifted-Grid Fast Marching (SGFM) [6] and Multi-stencil Fast Marching (MSFM) [7]. We also compare the upsampled 4 and 8-connected neighbor Fast Marching schemes with the upsampled version of the SGFM scheme (upSG).

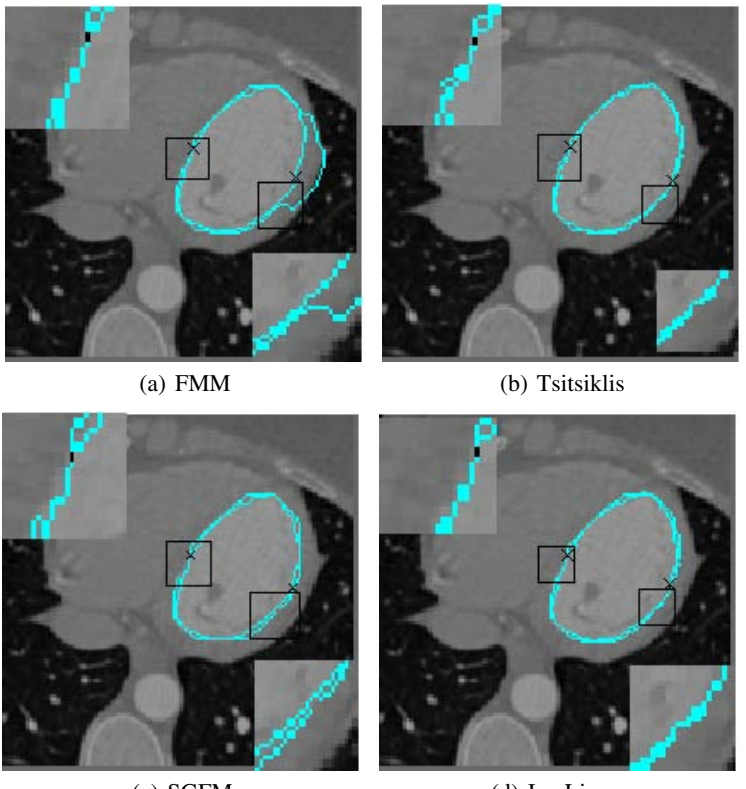


(a) Cardiac Data (b) Random noise

Fig. 6. Test Images

In the first experiment we pick a random point, marked by the 'x' in the images shown in Figure 6, and compute u at every point of the image. We then compute the total cost in propagating a front from each point of the image back to the point marked by the 'x'. We take the average of the difference (error) across the entire image. The numerical values are listed in the Table 2, under the column labeled Average Back Propagation Error (ABPE). We used the cost function, $\tau(x) = \frac{1}{1+|\nabla I|^2}$ for the cardiac image and $\tau(x) = I(x)$ for the random noise image.

In Figure 7 we present the results of segmenting the left ventricles in a 2D cardiac slice. To segment the image we pick a point on the boundary of the object and compute the saddle points as described in [5]. From each saddle point we then obtain two minimal paths back to the initial point; these paths will give the segmentation of the object. The minimal paths were obtained using a sub-pixel level back propagation scheme. We then choose the saddle point which minimizes the Chan-Vese [3] energy of the obtained segmentation. Images in Figure 7 show the overlay of segmentation curves initialized with 2 different user given points on the boundary. We see that the segmentation curves are not consistent and they depend on the initialization. This is mainly due to the difference in the marching direction in each case and weak image features at certain locations. We highlight certain regions in these images to compare the segmentation obtained from the different methods.



(a) FMM (b) Tsitsiklis

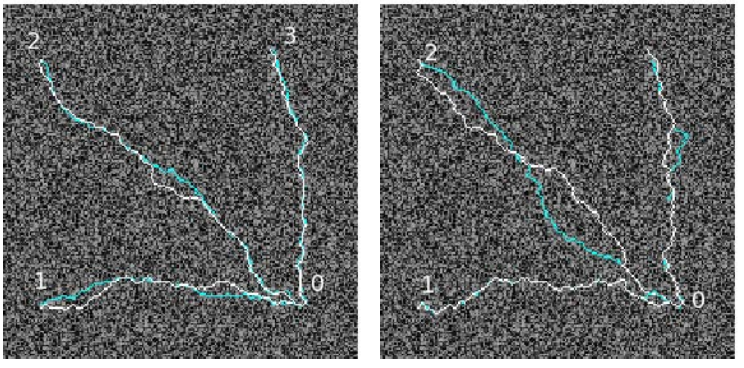
(c) SGFM (d) Iso-Linear

Fig. 7. A comparison of segmentation

In the images shown in Figure 8, we compare the minimal paths obtained in traveling from point '0' to points '1','2' and '3' with the corresponding paths obtained by reversing the direction. We see that using interpolated FMM gives consistent paths, even in the absence of any strong image feature. The results are in accordance to the Average Back Propagation Errors listed in Table 2. The ABPE for the Tsitsiklis scheme is the highest and accordingly the paths obtained with the Tsitsiklis scheme show a lot of variation. Although the SGFM shows lower average error there are variations in the obtained minimal paths. This is because the interpolation of the cost function in SGFM is equivalent to image smoothing for the τ ($\tau(x) = I(x)$) used in this example. This decreases the corresponding average error, but it also decreases the difference in the geodesic distances of the various paths. Thus with the change in the marching direction, the back propagation takes different paths between two given points.

In the next example we compare the accuracy of the various techniques for two cost functions on a 50x50 grid,

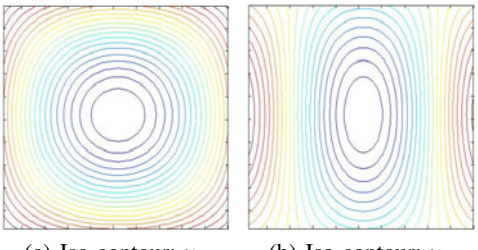
$$\tau_1(x,y) = 1/20\sqrt{(sin\tfrac{x}{20}cos\tfrac{y}{20})^2 + (cos\tfrac{x}{20}sin\tfrac{y}{20})^2},$$
$$\tau_2(x,y) = 1/10\sqrt{(sin\tfrac{x}{20}cos\tfrac{y}{10})^2 + (cos\tfrac{x}{20}sin\tfrac{y}{10})^2}.$$



(a) FMM (b) Tsitsiklis

(c) SGFM (d) Iso-Linear

Fig. 8. A comparison of tracking



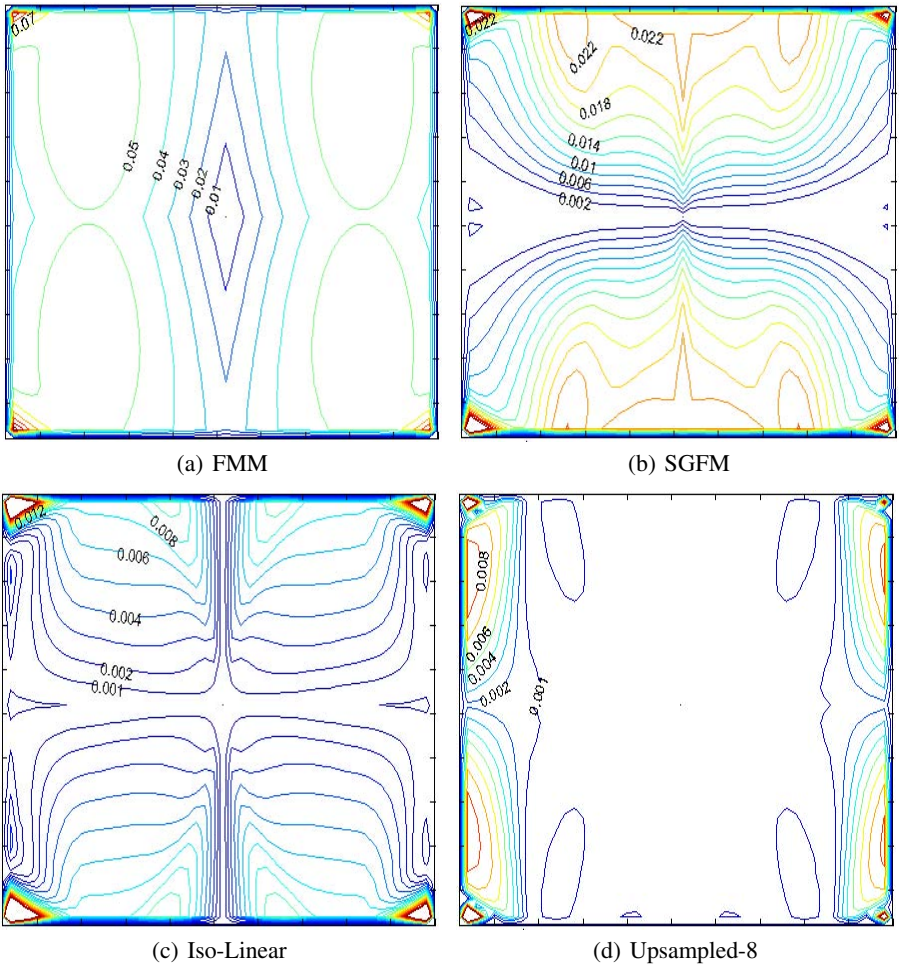
(a) Iso-contour: u_1 (b) Iso-contour: u_2

Fig. 9. Iso-contours

The iso-contours of $u_{analytic}$ are shown in Figure 9. The geodesics from the center $(26, 26)$ of the grid will be straight lines for τ_1 and curved for τ_2. Since, we have the analytic solution for these cost functions, we can compare the L_1, L_2 and L_∞ norms for each method.

Table 2. Error norms for τ_1 and τ_2, Average Back Propagation Errors and Computation times

	τ_1			τ_2			ABPE		Time
	L_1	L_2	L_∞	L_1	L_2	L_∞	I_1	I_2	(s)
FMM	$2.46\text{x}10^{-2}$	$6.73\text{x}10^{-4}$	0.0380	$4.37\text{x}10^{-2}$	$2.07\text{x}10^{-3}$	0.1060	0.0725	0.3901	0.27
Tsitsiklis	$2.14\text{x}10^{-2}$	$4.89\text{x}10^{-4}$	0.0281	$3.81\text{x}10^{-2}$	$1.57\text{x}10^{-3}$	0.0825	0.1007	0.4348	0.26
MSFM	$2.36\text{x}10^{-2}$	$6.07\text{x}10^{-4}$	0.0349	$4.23\text{x}10^{-2}$	$1.94\text{x}10^{-3}$	0.1007	0.0825	0.3572	0.29
SGFM	$2.33\text{x}10^{-3}$	$6.32\text{x}10^{-6}$	0.0051	$1.25\text{x}10^{-2}$	$2.14\text{x}10^{-4}$	0.0580	0.0022	0.0277	0.33
Linear4	$1.10\text{x}10^{-2}$	$1.71\text{x}10^{-4}$	0.0285	$1.69\text{x}10^{-2}$	$4.01\text{x}10^{-4}$	0.0875	0.0122	0.1036	0.51
Linear8	$2.25\text{x}10^{-3}$	$6.82\text{x}10^{-6}$	0.0046	$4.46\text{x}10^{-3}$	$3.43\text{x}10^{-5}$	0.0596	0.0028	0.0355	0.52
IsoLinear	$2.25\text{x}10^{-3}$	$6.82\text{x}10^{-6}$	0.0046	$4.03\text{x}10^{-3}$	$3.11\text{x}10^{-5}$	0.0596	0.0109	0.0911	0.91
Bilinear8	$2.74\text{x}10^{-3}$	$9.42\text{x}10^{-6}$	0.0052	$5.01\text{x}10^{-3}$	$4.10\text{x}10^{-5}$	0.0607	0.0028	0.0101	0.65
Up4	$1.79\text{x}10^{-3}$	$7.60\text{x}10^{-6}$	0.0101	$3.14\text{x}10^{-3}$	$2.89\text{x}10^{-5}$	0.0655	0.0451	0.1919	1.37
Up8	$2.99\text{x}10^{-4}$	$1.96\text{x}10^{-7}$	0.0014	$1.54\text{x}10^{-3}$	$7.81\text{x}10^{-6}$	0.0289	0.0011	0.0221	1.42
UpSG	$1.96\text{x}10^{-3}$	$4.15\text{x}10^{-6}$	0.0035	$1.20\text{x}10^{-2}$	$1.94\text{x}10^{-4}$	0.0566	0.0015	0.0141	1.42



(a) FMM (b) SGFM

(c) Iso-Linear (d) Upsampled-8

Fig. 10. Iso-contours of errors for τ_2

$$L_1 = \text{mean}(|u - u_{analytic}|),$$
$$L_2 = \text{mean}(|u - u_{analytic}|^2),$$
$$L_\infty = \max(|u - u_{analytic}|).$$

The numerical errors in using cost functions τ_1 and τ_2 are listed in Table 2. Notice that the error norms show significant improvement for the proposed methods, especially in the case with curved geodesics (τ_2). The iso-contours of the errors for τ_2 while using FMM, SGFM, Iso-Linear and up8 are shown in Figure 10.

We also enlist the computation times for each of these methods on a 500x500 grid in the last column of Table 2. All computation times were measured on a laptop with a 1.73 GHz Processor.

5 Conclusion

In this paper we present techniques to make the fast marching method independent of the marching direction and thus improve the accuracy of the Fast Marching Method. One approach interpolates the local traveling cost along the front and the other computes u on an upsampled grid. We also showed that combining the 8 and 4-connected neighbor schemes further reduces the inaccuracy by considering all possible directions of the arrival of the front. We have compared both our approaches to the existing Fast Marching techniques and we have shown a significant improvement over them. Although both our approaches have higher computation times, they can be implemented efficiently on hardware and they are practical solutions to eliminate the inaccuracies of existing techniques.

Acknowledgements

This work was partially supported by NIH grant R01-HL-085417, NSF grant CCF-0728911 as well as an EmTech seed grant.

References

1. Adalsteinsson, D., Sethian, J.A.: A fast level set method for propagating interfaces. Journal of Computational Physics 118, 269–277 (1994)
2. Bronstein, A.M., Bronstein, M.M., Devir, Y.S., Kimmel, R., Weber, O.: Parallel algorithms for approximation of distance maps on parametric surfaces (2007)
3. Chan, T., Vese, L.: An active contour model without edges. In: Nielsen, M., Johansen, P., Fogh Olsen, O., Weickert, J. (eds.) Scale-Space 1999. LNCS, vol. 1682, pp. 141–151. Springer, Heidelberg (1999)
4. Cohen, L., Kimmel, R.: Global minimum for active contour models: A minimal path approach. International Journal of Computer Vision 24, 57–78 (1997)
5. Cohen, L.D., Kimmel, R.: Global minimum for active contour models: A minimal path approach. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p. 666 (1996)
6. Danielsson, P.E., Lin, Q.: A modified fast marching method. In: SCIA, pp. 1154–1161 (2003)

7. Hassouna, M.S., Farag, A.A.: Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(9), 1563–1574 (2007)
8. Kim, S., Folie, D.: The group marching method: An o(n) level set eikonal solver
9. Polymenakos, L.C., Bertsekas, D.P., Tsitsiklis, J.N.: Implementation of efficient algorithms for globally optimal trajectories. IEEE Transactions on Automatic Control 43, 278–283 (1998)
10. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Cambridge University Press, Cambridge (1999)
11. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. IEEE Transactions on Automatic Control 40(9), 1528–1538 (1995)