# An Efficient Solution to the Eikonal Equation on Parametric Manifolds

A. Spira and R. Kimmel

**Abstract**

We present an efficient solution to the Eikonal equation on parametric manifolds, based on the fast marching approach. This method overcomes the problem of a non-orthogonal coordinate system on the manifold by creating an appropriate numerical stencil. The method is tested numerically and demonstrated by calculating distances on various parametric manifolds.

## 1 Introduction

The viscosity solution $\phi(x, y)$ of the Eikonal equation

$$\|\nabla\phi\| = F, \tag{1}$$

is a weighted distance map from a set of initial points, where the values of $\phi$ are given. The weights are given by the scalar positive function $F(x, y)$. Efficient solutions to the Eikonal equation on the plane parameterized by a regular (orthogonal) numerical grid were introduced by Sethian [6] and by Tsitsiklis [10]. Sethian's fast marching method was extended by Kimmel and Sethian [1] to the solution of the Eikonal equation on triangulated manifolds

$$\|\nabla_M\phi\| = F, \tag{2}$$

with $M$ the manifold and $\nabla_M\phi$ the gradient on the manifold. This extension enables a fast calculation of geodesic paths [1], Voronoi diagrams, and offsets

[2, 3] on triangulated manifolds. Sethian and Vladimirsky [9] presented Ordered Upwind Methods (OUM) for static Hamilton-Jacobi equations. These methods enable the solution of equations where the directions of the characteristics are different from those of the gradients of $\phi$. As an example, they demonstrate the solution of the Eikonal equation for manifolds which are function graphs. Mémoli and Sapiro [4] calculated distances on implicit manifolds by using orthogonal fast marching in a thin offset band surrounding the manifold.

We present here an efficient solution to the Eikonal equation on parametric manifolds, based on the fast marching approach. A parametric manifold consists of a parameterization plane $U = \{u^1, u^2\} \in \mathbb{R}^2$, which is mapped by $X : \mathbb{R}^2 \to \mathbb{R}^N$ to the parametric manifold $X(U) = \{x^1(u^1, u^2), x^2(u^1, u^2), \ldots, x^N(u^1, u^2)\} \in \mathbb{R}^N$. In this method the calculations are done on the 2-dimensional uniform Cartesian grid of the parameterization plane and not on the manifold like in Kimmel and Sethian's method or in $\mathbb{R}^N$ according to Mémoli and Sapiro. The numerical stencil at each grid point is calculated directly from the metric and there is no need for the "unfolding" procedure of Kimmel and Sethian or for finding the"near front" as done by Sethian and Vladimirsky. The presented method is first order accurate as that of Kimmel and Sethian, but can be extended to higher orders by using Sethian and Vladimirsky's higher order directional derivative approximations [8]. The error of Mémoli and Sapiro's method is $o\left(\sqrt{h}\right)$.

The derivatives of $X$ with respect to $u^i$ are defined as $X_i \triangleq \frac{\partial X}{\partial u^i}$, and they constitute a non-orthogonal coordinate system on the parametric manifold. See Figure 1. The distance element on the manifold is

$$ds = \sqrt{g_{ij} du^i du^j}, \tag{3}$$

where we use Einstein's summation convention, and the metric tensor of the manifold $g_{ij}$ is calculated by

$$(g_{ij}) = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix} = \begin{pmatrix} X_1 \cdot X_1 & X_1 \cdot X_2 \\ X_2 \cdot X_1 & X_2 \cdot X_2 \end{pmatrix}. \tag{4}$$

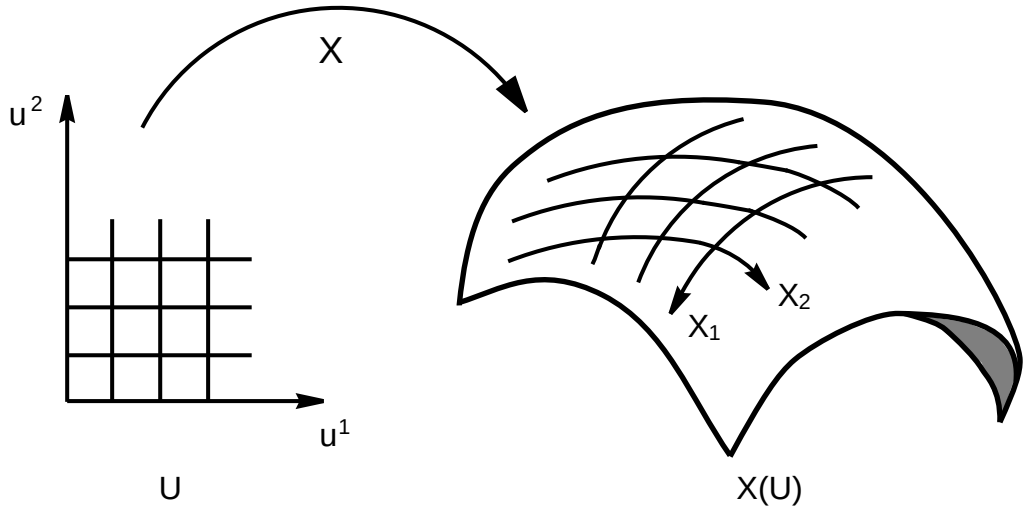


Figure 1: The orthogonal grid on the parameterization plane is transformed into a non-orthogonal one on the manifold.

This paper is organized as follows. The second section describes the non-orthogonality of the coordinate system on the manifold and the resulting problem. Section 3 introduces the construction of a numerical stencil which overcomes this problem. Section 4 presents the numerical scheme, and Section 5 the marching method for solving the Eikonal equation on the manifold. The performance and accuracy of the numerical scheme is tested in Section 6. The conclusions appear in Section 7.

# 2 The Non-Orthogonal Coordinate System on the Manifold

The power of the fast marching algorithm lies in its ability to solve the Eikonal equation in one sweep without iterations. The algorithm takes advantage of the upwind nature of the Eikonal equation in order to update the value of each grid point by a number of times bounded by the number of its

neighbors. We would like to devise a similar algorithm for Equation (2).

The orthogonal fast marching algorithm [6] solves the Eikonal equation for an orthogonal coordinate system. In this case, the numerical stencil for the update of a grid point consists of one or two points out of its four neighbors. The first point is one of the top/bottom pair and the second is one of the left/right pair. The two grid points in the stencil, together with the updated grid point, compose the vertices of a right triangle. See Figure 2.

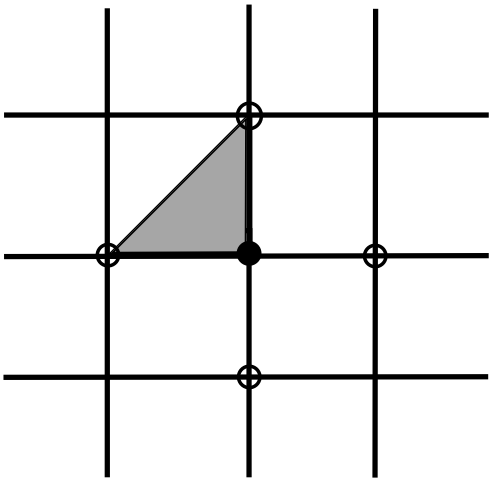

Figure 2: The numerical stencil for the orthogonal fast marching algorithm is a right triangle.

This is not the case for manifolds with $g_{12} \neq 0$, where we get a non-orthogonal coordinate system on the manifold, see Figure 3. The resulting triangles are not right triangles. Each grid point is a origin of two acute angles and two obtuse angles. If a grid point is updated by a stencil with an obtuse angle, a problem may arise. Depending on the direction of the advancing 'update front', the value of one of the points of the stencil might not be set in time and cannot be used properly for supporting the updated vertex. There is a similar problem with fast marching on triangulated domains which contain obtuse angles [1].

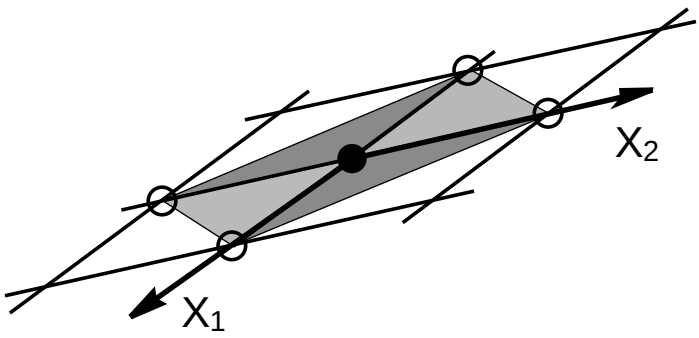


Figure 3: Two acute angles and two obtuse angles for a non-orthogonal coordinate system on the manifold.

# 3 Splitting Obtuse Angles

Our solution to obtuse angles is similar to that of [1] with the exception that there is no need for the "unfolding" step. We perform a pre-processing stage for the grid, in which we split every obtuse triangle into two acute ones, see Figure 4. The split is performed by adding an additional edge, connecting the updated grid point with a non-neighboring grid point. The distant grid point becomes part of the numerical stencil.

The need for splitting is determined according to the angle between the non-orthogonal axes at the grid point. It is calculated by

$$\cos(\alpha) = \left( \frac{X_1 \cdot X_2}{\|X_1\| \|X_2\|} \right) = \frac{g_{12}}{\sqrt{g_{11} g_{22}}}. \tag{5}$$

If $\cos(\alpha) = 0$, the axes are perpendicular, and no splitting is required. If $\cos(\alpha) < 0$, the two angles with an angle of $\alpha$ should be split. Otherwise, the two other angles should be split. The denominator of Equation (5) is always positive, so we need only check the sign of the numerator $g_{12}$.

In order to split an angle, we should connect the updated grid point with another point, located $m$ grid points from the point in the direction of $X_1$ and $n$ grid points in the direction of $X_2$ ($m$ and $n$ may be negative). The point provides a good numerical support, if the obtuse angle is split into two

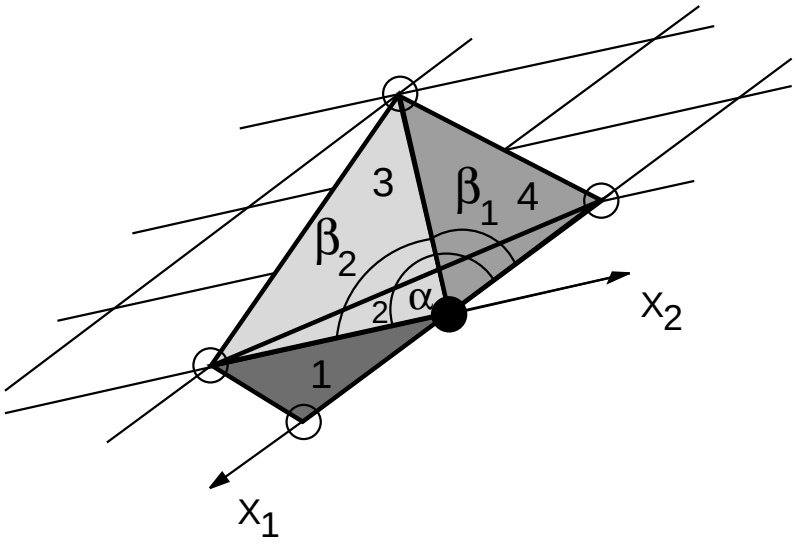


Figure 4: The numerical stencil for the non-orthogonal coordinate system. Triangle 1 gives good numerical support to the black grid point, but triangle 2 includes an obtuse angle. It is replaced by triangle 3 and triangle 4.

acute ones. For $\cos(\alpha) < 0$ this is the case if

$$\cos(\beta_1) = \left( \frac{X_1 \cdot (mX_1 + nX_2)}{\|X_1\| \|mX_1 + nX_2\|} \right) = \frac{mg_{11} + ng_{12}}{\sqrt{g_{11}(m^2 g_{11} + 2mng_{12} + n^2 g_{22})}} > 0, \tag{6}$$

and

$$\cos(\beta_2) = \left( \frac{X_2 \cdot (mX_1 + nX_2)}{\|X_2\| \|mX_1 + nX_2\|} \right) = \frac{mg_{12} + ng_{22}}{\sqrt{g_{22}(m^2 g_{11} + 2mng_{12} + n^2 g_{22})}} > 0. \tag{7}$$

Also here, it is enough to check the sign of the numerators. For $\cos(\alpha) > 0$, the equation for $\cos(\beta_2)$ changes its sign and the constraints are

$$mg_{11} + ng_{12} > 0, \tag{8}$$

and

$$mg_{12} + ng_{22} < 0. \tag{9}$$

Equations (6,7,8,9) give together the condition

$$\left|\frac{g_{12}}{g_{11}}n\right| < m < \left|\frac{g_{22}}{g_{12}}n\right|, \tag{10}$$

and we would like to find the minimal $m$ and $n$ that satisfy this condition. We define $P = \frac{|g_{12}|}{g_{11}}$ and $Q = \frac{g_{22}}{|g_{12}|}$. The problem is solved by the following algorithm

- If $P \geq 1$, $p = P - \lfloor P \rfloor$ and $q = Q - \lfloor P \rfloor$. Else, $p = P$ and $q = Q$.
- Start with $n = 1$.
- $P_n = p \cdot n$, $Q_n = q \cdot n$.
- If $\lceil P_n \rceil < Q_n$, then $m = \lceil P_n \rceil$. Else, set $n = n + 1$ and return to the previous step.
- If $P \geq 1$, $m = m + \lfloor P \rfloor \cdot n$.
- If $\cos(\alpha) > 0$, then $n = -n$.

If we define $L = \lceil 1/(Q-P) \rceil = \lceil \frac{|g_{12}|g_{11}}{g} \rceil$, with $g = \det(g_{ij}) = g_{11}g_{22} - g_{12}^2$, then $|n|$ is bounded by $L$, because for $n = L$ we have $Q_n - P_n > 1$, and there will be an $m$ that complies to the condition in Equation (10). We could use binary search and the bound $L$ to get a complexity of $O(\log L)$ for this algorithm, but because the bound is not a tight one, we use the algorithm as is. It should be noted that $g_{ij}$ and therefore $L$ are parameterization dependent. If we have a parameterization with regions where $X_1$ and $X_2$ are almost parallel, the resulting $m$ and $n$ might be large, affecting the accuracy and efficiency of the numerical scheme.

# 4 The Numerical Scheme

Once the pre-processing stage is over, we have a suitable numerical stencil for each grid point and we can solve the Eikonal equation numerically. The stencil is composed of the vertices of an acute angle, see Figure 5, where the vertex

$C$ is updated according to the vertices $A$ and $B$. If the triangle was originally acute, we have $a = \sqrt{g_{11}}$, $b = \sqrt{g_{22}}$ and $\theta = \alpha$. If it is a triangle created by splitting, we have $a = \sqrt{g_{11}}$ or $a = \sqrt{g_{22}}$, $b = \sqrt{m^2 g_{11} + 2mng_{12} + n^2 g_{22}}$ and $\theta = \beta_1$ or $\theta = \beta_2$. Next, we want to find $t$ such that $\frac{t-u}{h} = F$.

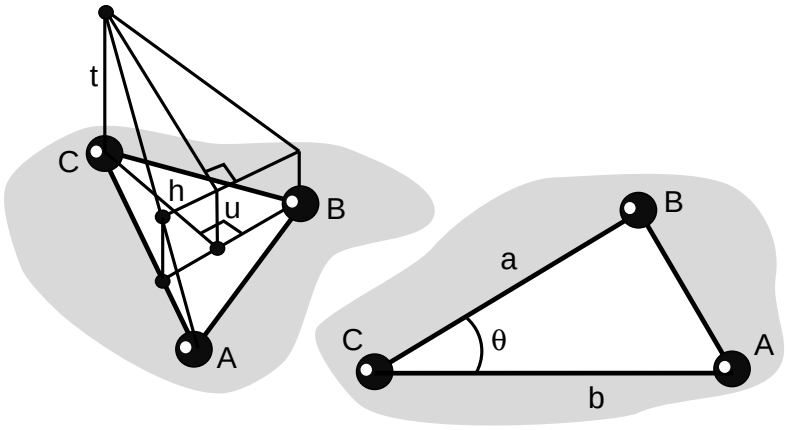


Figure 5: Two views of the numerical stencil.

The numerical scheme according to [1] is

- $u = \phi(B) - \phi(A)$.
- Solve the quadratic equation

$$(a^2+b^2-2ab\cos\theta)t^2+2bu(a\cos\theta-b)t+b^2(u^2-F^2a^2\sin^2\theta)=0. \quad (11)$$

- If $u < t$ and $a\cos\theta < \frac{b(t-u)}{t} < \frac{a}{\cos\theta}$, then $\phi(C) = \min\{\phi(C), t + \phi(A)\}$. Else, $\phi(C) = \min\{\phi(C), bF + \phi(A), aF + \phi(B)\}$.

# 5 Marching on Manifolds

After the pre-processing stage, the Eikonal equation is solved by the following algorithm [7].

**Initialization**:

- The initial points are defined as *Accepted* and given their initial values.
- All the other grid points are defined as *Far* and given the value infinity.

**Iterations**:

1. *Far* 'neighbors' of *Accepted* points are defined as *Close*.
2. The values of the *Close* points are updated according to the numerical scheme.
3. The *Close* point with the minimal value becomes an *Accepted* point.
4. If there remain any *Far* points, return to step 1.

We used the term 'neighbors' above to describe grid points that belong to the same triangular numerical stencil. These points are not necessarily neighboring points on the original grid. We find these 'neighbors' during the pre-processing stage described in the previous section.

The complexity of the algorithm is upper bounded by $O(N \cdot \max(\log L, \log N))$, where $N$ is the number of points in the grid. The $\log N$ results from using a min-heap data structure for sorting the *Close* points [7].

# 6 Testing the Numerical Scheme

The algorithm was tested for parametric manifolds with non-orthogonal coordinate systems. In Figure 6 it is implemented on the tilted plane $z = 3x + 2y$. The correctness of the distance map is evident from the resulting level curves, which are concentric circles on the manifold. In Figure 7 the algorithm is implemented for the manifold $z = 0.5 \sin(4\pi x) \sin(4\pi y)$.

The accuracy of the algorithm is measured by running the algorithm on the manifold $z = 0.5\sin(4\pi x)\sin(4\pi y)$ with one initial point at $(x = 0.5, y = 0.5)$. Table 1 gives the estimated errors of the algorithm on various grid sizes and estimations of the order of accuracy. The normalized $L_2$ error at grid size $n^2$ is $e_2^n = \frac{\|v-u^n\|_2}{n^2}$, where $u^n$ is the result of the algorithm on a grid of size $n^2$ and $v$ is the correct solution. The $L_\infty$ error for this grid is $e_\infty^n = \|v - u^n\|_\infty$. Since $v$ is unknown, we estimate it by the result of the algorithm on a grid of size $1025^2$. Assuming that $v$ is of the form $v = u^n + Ch^r + O(h^{r+1})$, where $h = \frac{1}{n-1}$, the order of accuracy of the numerical scheme according to the $L_k$ norm at grid size $n^2$ can be estimated according to [5]

$$r_k^n = \log_2\left(\frac{e_k^n}{e_k^{2n}}\right). \tag{12}$$

| size: | $17^2$ | $33^2$ | $65^2$ | $129^2$ | $257^2$ | $513^2$ |
|---|---|---|---|---|---|---|
| $e_2^n$: | $6.2 \cdot 10^{-3}$ | $2.3 \cdot 10^{-3}$ | $8.4 \cdot 10^{-4}$ | $3.0 \cdot 10^{-4}$ | $4.1 \cdot 10^{-5}$ | $5.7 \cdot 10^{-6}$ |
| $r_2^n$: | 1.43 | 1.45 | 1.50 | 2.86 | 2.85 | |
| $e_\infty^n$: | 0.4425 | 0.2702 | 0.1746 | 0.0977 | 0.0277 | 0.0101 |
| $r_\infty^n$: | 0.71 | 0.63 | 0.84 | 1.82 | 1.46 | |

Table 1: The estimated errors and orders of accuracy of the algorithm as a function of grid size.

# 7 Conclusions

A new efficient method for solving the Eikonal equation on parametric manifolds was introduced. The method requires only the metric tensor at each grid point in order to determine the numerical stencil and execute the numerical scheme. This method enables a fast calculation of distances on manifolds, needed in many applications.

**Acknowledgements**: We thank I. Blayvas for helpful suggestions.

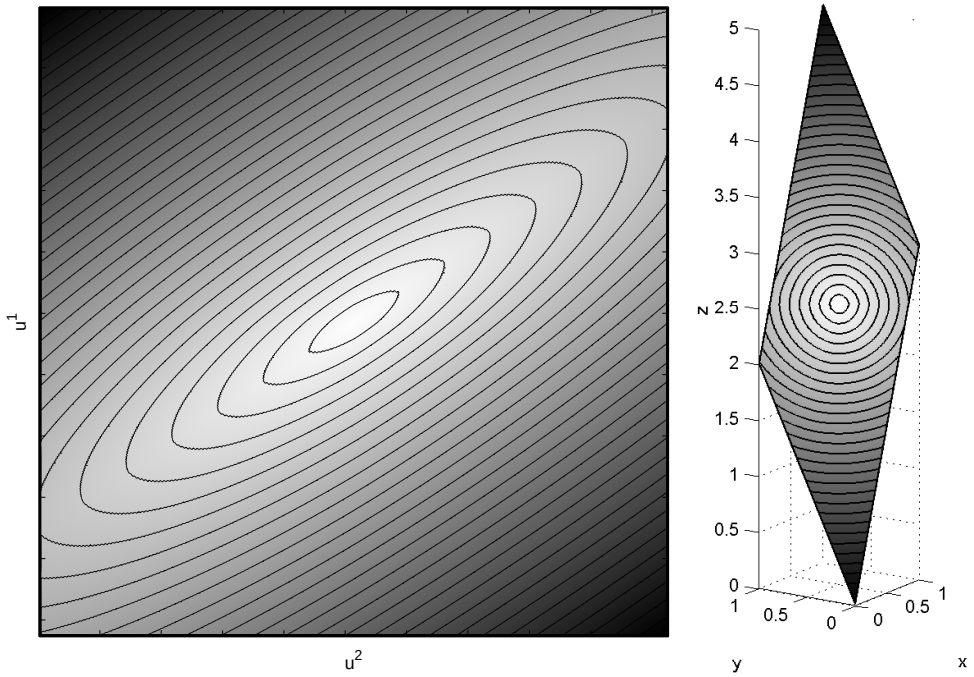


Figure 6: Fast marching on the manifold $z = 3x + 2y$. Left: implemented on the parameterization plane. Right: projected on the manifold.

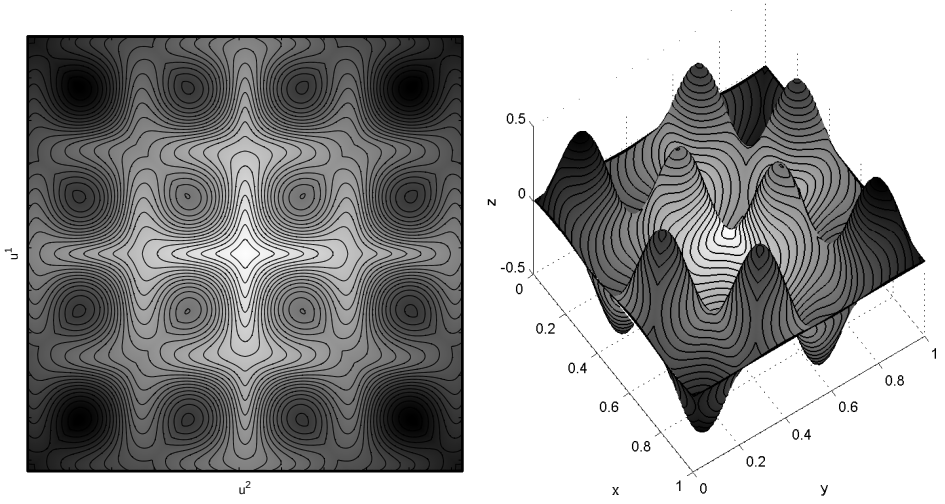


Figure 7: Fast marching on the manifold $z = 0.5\sin(4\pi x)\sin(4\pi y)$. Left: implemented on the parameterization plane. Right: projected on the manifold.

# References


[1] R. Kimmel and J. Sethian. Computing geodesic paths on manifolds. *Proceedings of National Academy of Sciences*, 95(15):8431–8435, July 1998.

[2] R. Kimmel and J. Sethian. Fast voronoi diagrams and offsets on triangulated surfaces. In *AFA Conference on Curves and Surfaces*, Saint-Malo, France, July 1999.

[3] R. Kimmel and J. Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14(3):237–244, May 2001.

[4] F. Mémoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *Journal of Computational Physics*, 173(2):730–764, 2001.

[5] S. Osher and J. Sethian. Fronts propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[6] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of National Academy of Sciences*, 93(4):1591–1595, 1996.

[7] J. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge university press, 1996.

[8] J. Sethian and A. Vladimirsky. Fast methods for the eikonal and related hamilton-jacobi equations on unstractured meshes. *Proc. Nat. Acad. Sci. USA*, 97:5699–5703, 2003.

[9] J. Sethian and A. Vladimirsky. Ordered upwind methods for static hamilton-jacobi equations: theory and applications. *SIAM J. on Numerical Analysis*, 41(1):325–363, 2003.

[10] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*, 40(9):1528–1538, 1995.