An Efficient Solution to the Eikonal Equation on Parametric Manifolds

Alon Spira Ron Kimmel
Department of Computer Science
Technion–Israel Institute of Technology
Technion City, Haifa 32000, Israel
{salon,ron}@cs.technion.ac.il

February 23, 2004

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. It is further used for two applications: image enhancement and face recognition.

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 ϕ 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 [12] and by Tsitsiklis [20]. Sethian's fast marching method was extended by Kimmel and Sethian [7] 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 [7], Voronoi diagrams, and offsets [8, 9] on triangulated manifolds. Sethian and Vladimirsky [15] 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 ϕ. As an example, they demonstrate the solution of the Eikonal equation for manifolds which are function graphs. Also Tsai et al. [19] solved the equation on function graphs, but they used an iterative sweeping method. A similar sweeping approach was previously used by Danielson [3] to compute Euclidean distance maps on flat domains with regular grids. Mémoli and Sapiro [10] 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 [1] . 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"

[1] An early version of the paper was presented at the INTERPHASE 2003 meeting at the Newton Insitute during the 2003 Programme Computational Challenges in PDEs [17].

procedure of Kimmel and Sethian or for finding the"near front" as done by Sethian and Vladimirsky. The proposed method solves the equation in one sweep of the numerical grid as opposed to Tsai et al.'s method, which is an iterative approach that performs several sweeps in every iteration. Furthermore, the number of iterations required for the convergence of their method depends on the anisotropy of the equation. The presented method is first order accurate as that of Kimmel and Sethian, but may be extended to higher orders by using Sethian and Vladimirsky's higher order directional derivative approximations [14]. 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

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

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. Section 7 demonstrates applications of the numerical scheme in image processing and computer vision. The conclusions appear in Section 8.

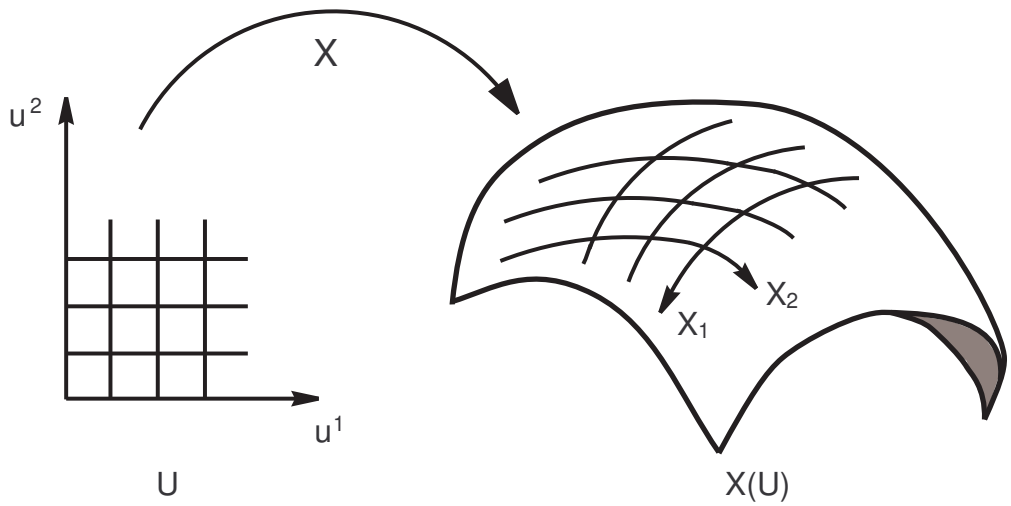

Figure 1: The orthogonal grid on the parameterization plane is transformed into a non-orthogonal one on the manifold.

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 [12] 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.

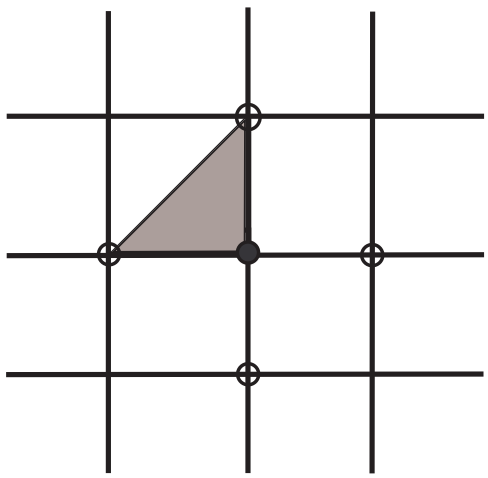

Figure 2: The numerical stencil for the orthogonal fast marching algorithm is a right triangle.

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 the 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 [7].

3 Splitting Obtuse Angles

Our solution to obtuse angles is similar to that of [7] 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

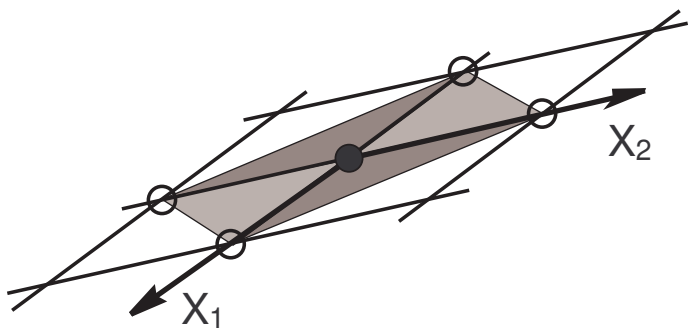

Figure 3: Two acute angles and two obtuse angles for a non-orthogonal coordinate system on the manifold.

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

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

and

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

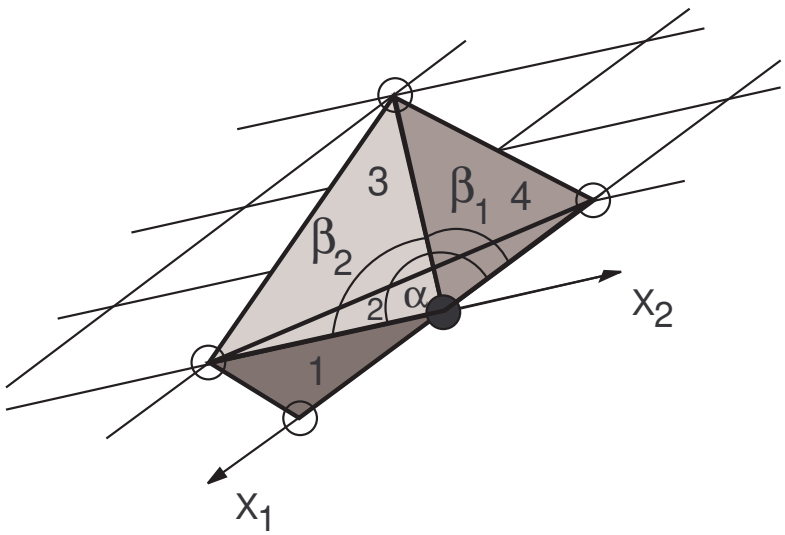

Figure 4: The numerical stencil for the non-orthogonal coordinate system. Triangle 1 gives good numerical support to the black grid point, but triangle 2 includes an obtuse angle. It is replaced by triangle 3 and triangle 4.

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 $\frac{t-u}{h} = F$.

The numerical scheme according to [7] is

- $u = \phi(B) - \phi(A)$.

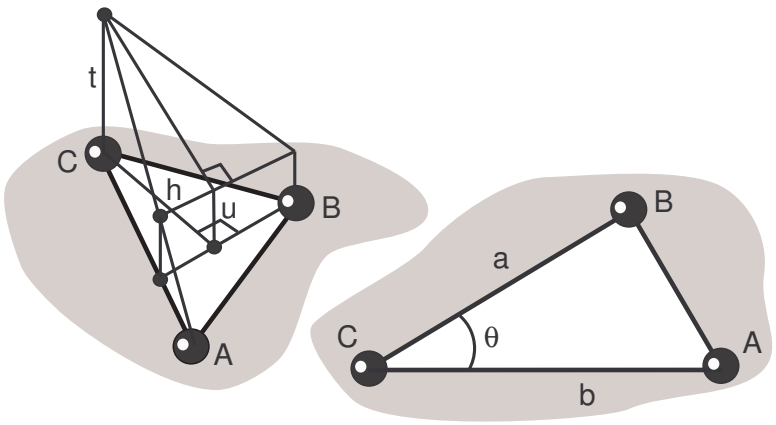

Figure 5: Two views of the numerical stencil.

- 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 [13].

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 [13].

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 $z = 3x + 2y$, with the initial point at $(x = 0.5, y = 0.5)$. In this figure and the ones to follow, lower values are assigned brighter shades of gray and black curves are used to indicate the level curves. 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 $z = 0.5 \sin(4\pi x) \sin(4\pi y)$ with the same initial point. The proposed algorithm can work also for parametric manifolds that are not function graphs. In Figure 8 the algorithm is implemented for the sphere $\{x = \cos(\theta)\cos(\phi), y = sin(\theta)\cos(\phi), z = \sin(\phi)\}$. In this case, The initial points construct a square on the parameterization plane $\{u^1, u^2\} = \{\theta, \phi\}$. The range of the parameters in the figure is $-54° < \theta, \phi < 54°$. In Figure 9 the algorithm is implemented on the tilted plane $z = 2x + 2y$. This time, F on the right hand side of the Eikonal equation (2) changes abruptly on the parameterization plane . Its value is $F = 10$ for $x > 0.5, y > 0.5$ and $F = 1$ otherwise. This figure shows that the numerical scheme can handle a sharply changing F.

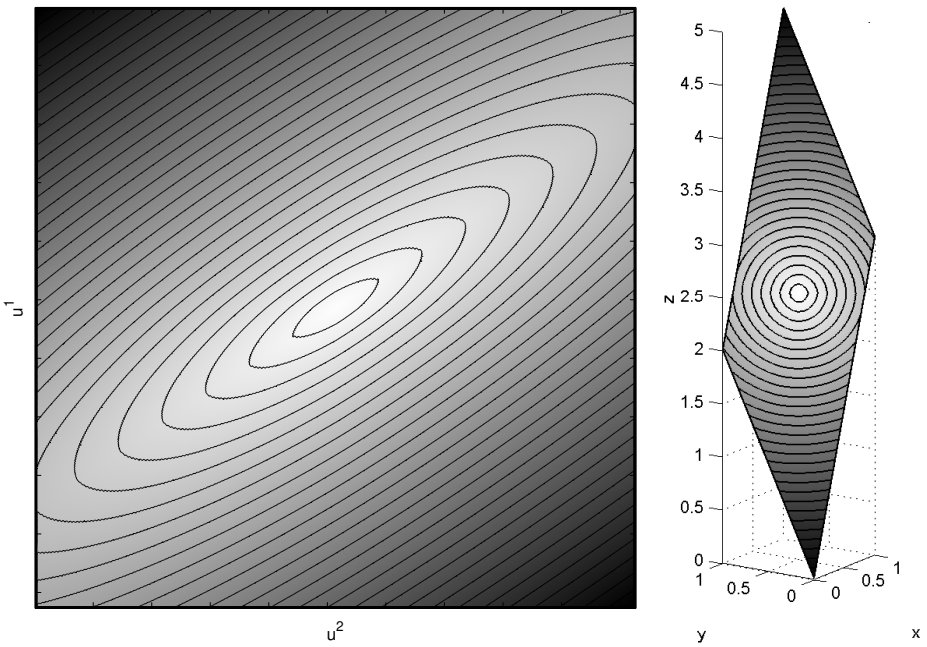

Figure 6: Fast marching on the manifold $z = 3x + 2y$. Left: implemented on the parameterization plane. Right: projected on the manifold.

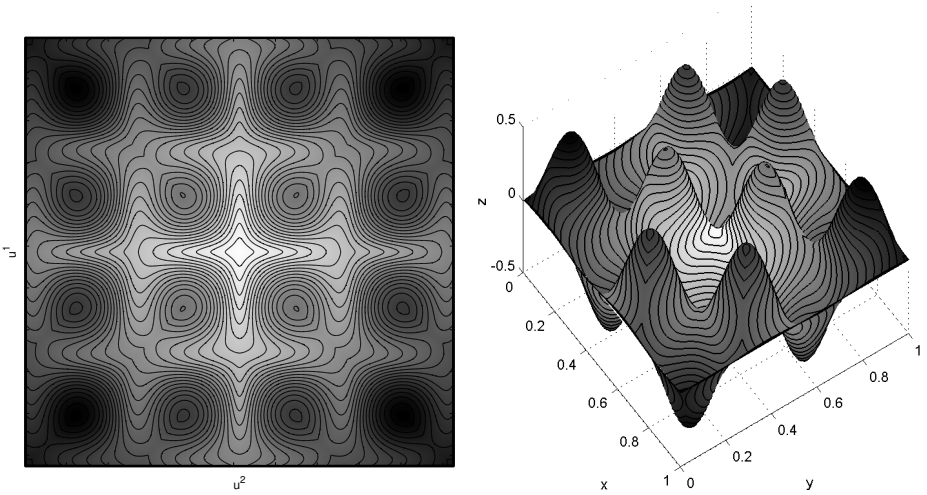

Figure 7: Fast marching on the manifold $z = 0.5\sin(4\pi x)\sin(4\pi y)$. Left: implemented on the parameterization plane. Right: projected on the manifold.

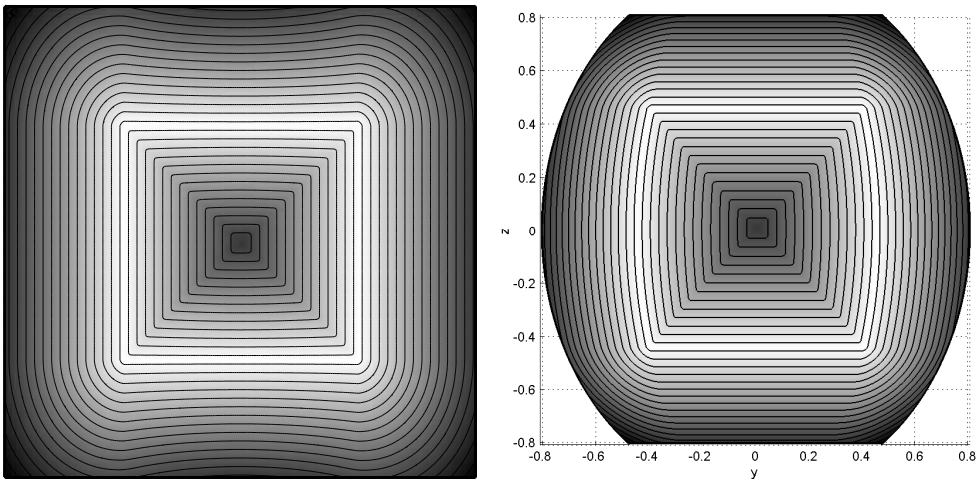

Figure 8: Fast marching on a sphere. Left: implemented on the parameterization plane. Right: projected on the manifold.

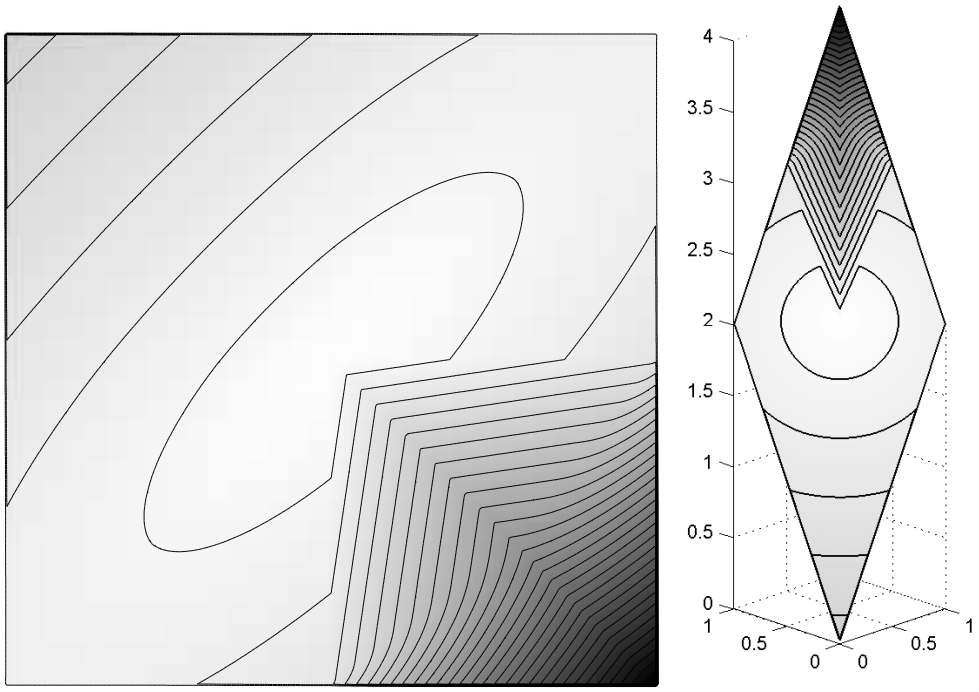

Figure 9: Fast marching with a non-constant F. Left: implemented on the parameterization plane. Right: projected on the manifold.

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_∞ 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 [11]

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

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_∞^n:	0.4425	0.2702	0.1746	0.0977	0.0277	0.0101
r_∞^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 Applications in Image Processing and Computer Vision

The solution to the Eikonal equation on parametric manifolds has many applications. In this section we demonstrate its use in the areas of image processing and computer vision. The first application consists of the acceleration of the image enhancing Beltrami filter [6, 16] by using a short time kernel [18]. Calculating the kernel requires the solution to the Eikonal equation on the image manifold. The second application is the implementation

of face recognition [1] by geometric invariants [4, 5] without reconstruction of the facial surface [2]. In this case a signature of the face is computed from geodesic distances between points on the facial manifold. The geodesic distances are calculated from the surface metric using our method.

7.1 A Short Time Kernel for the Beltrami Filter

The Beltrami filter [6, 16] results from the minimization of the area of the 2-dimensional Riemannian image manifold U embedded in the space-feature manifold $\mathbb{R}^N$, where $N = 3$ for gray scale images and $N = 5$ for color images. For gray scale images we have

$$X(u^1, u^2) = \{u^1, u^2, I(u^1, u^2)\}, \tag{13}$$

where u^1, u^2 are the space coordinates and I is the intensity component. We use a Euclidean space-feature manifold with the metric h_{ij} given by

$$(h_{ij}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \beta^2 \end{pmatrix}, \tag{14}$$

where β is the relative scale between the space coordinates and the intensity component. The metric g_{ij} of the image manifold is derived by the pullback procedure

$$(g_{ij}) = \begin{pmatrix} 1 + \beta^2 I_1^2 & \beta^2 I_1 I_2 \\ \beta^2 I_1 I_2 & 1 + \beta^2 I_2^2 \end{pmatrix}, \tag{15}$$

where $I_i \triangleq \frac{\partial I}{\partial u^i}$. A similar derivation is applicable for color images.

The Beltrami filter results from minimizing the area of the image manifold

$$S = \iint \sqrt{g} du_1 du_2, \tag{16}$$

with respect to the embedding. The corresponding Euler-Lagrange equations as a gradient descent process are

$$X_t^a = -g^{-\frac{1}{2}} h^{ab} \frac{\delta S}{\delta X^b} = g^{-\frac{1}{2}} \partial_i (g^{\frac{1}{2}} g^{ij} \partial_j X^a), \tag{17}$$

with g^{ij} the contravariant metric of the image manifold. For gray scale images we get

$$I_t = g^{-\frac{1}{2}}\partial_\mu(g^{\frac{1}{2}}g^{\mu\nu}\partial_\nu I) = \Delta_M I. \tag{18}$$

Using the short time kernel [18], we replace the partial differential equation (18) with a convolution-like process

$$I(u^1, u^2, t_0 + t) = \iint I(\tilde{u}^1, \tilde{u}^2, t_0) K(u^1, u^2, \tilde{u}^1, \tilde{u}^2; t) d\tilde{u}^1 d\tilde{u}^2 \ , \tag{19}$$

where

$$K(u^1, u^2, \tilde{u}^1, \tilde{u}^2; t) \ = \ \frac{H_0}{t} \exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)} ds\right)^2}{4t} \right), \tag{20}$$

and H_0 is taken such that integration over the kernel equals one. The operand of the exponent in this equation includes the geodesic distance between the filtered pixel and its neighboring pixels. Its calculation necessitates the solution to the Eikonal equation on the image manifold.

Figure 10 shows the result of applying the short time kernel Beltrami filter to a gray scale image. In this case $\beta = 3$, the time step taken was $t = 0.5$, and only grid points with a kernel value above 0.01 were used for the filtering. The time difference between the images is 1.

7.2 Face Signature without Reconstruction of the Facial Surface

Face recognition by geometric invariants [1], which is an application of the bending invariant canonical forms [4, 5], relies on a signature of the face computed from geodesic distances between points on the facial manifold. This use of geodesic distances makes this method highly robust to isometric deformations of the face, such as those resulting from facial expressions.

The face manifold is given by

$$X(u^1, u^2) = \{u^1, u^2, z(u^1, u^2)\}, \tag{21}$$

Figure 10: Application of the short time kernel Beltrami filter to a gray scale image. The original image is in the top left. The order of the images is from top to bottom and left to right.

and its metric can be acquired by photometric stereo, which requires at least three images of the face using independent illumination directions. Assuming a Lambertian reflection model, the generated images are

$$I^i(u^1, u^2) = \rho(u^1, u^2) N(u^1, u^2) \cdot L^i, \tag{22}$$

where $\rho(u^1, u^2)$ is the albedo at each point, $N(u^1, u^2)$ is the normal to the facial surface and L^i is the illumination direction for the i image. The normal is given by

$$N(u^1, u^2) = \frac{\{-z_1(u^1, u^2), -z_2(u^1, u^2), 1\}}{\sqrt{1 + \|\nabla z(u^1, u^2)\|^2}}, \tag{23}$$

where $z_i \triangleq \frac{\partial z}{\partial u^i}$.

Given at least three images with independent illumination directions, $\nabla z(u^1, u^2)$ can be extracted by Least Squares. The facial surface metric is

$$(g_{ij}) = \begin{pmatrix} 1 + z_1^2 & z_1 z_2 \\ z_1 z_2 & 1 + z_2^2 \end{pmatrix}, \tag{24}$$

and it enables the calculation of geodesic distances on the facial surface by our method for the solution of the Eikonal equation on such manifolds. Figure 11 shows distance maps calculated by our method from the metric of a face manifold. Note that the manifold itself $\{u^1, u^2, z(u^1, u^2)\}$ needs not be reconstructed.

The calculation of geodesic distances enables the use of the bending invariant canonical forms framework [4, 5] to produce a face signature. In this framework, the face manifold is sampled and a matrix of the geodesic distances between the points is produced. Using multidimensional scaling (MDS) the points are embedded in $\mathbb{R}^3$, where they tend to form a 2-dimensional function. The face signature is then constructed from this function. Because the geodesic distances are invariant to isometric deformations, this signature is robust to such deformations, which are frequent in face manifolds. For details on the quality and characteristics of this face signature see [1, 2].

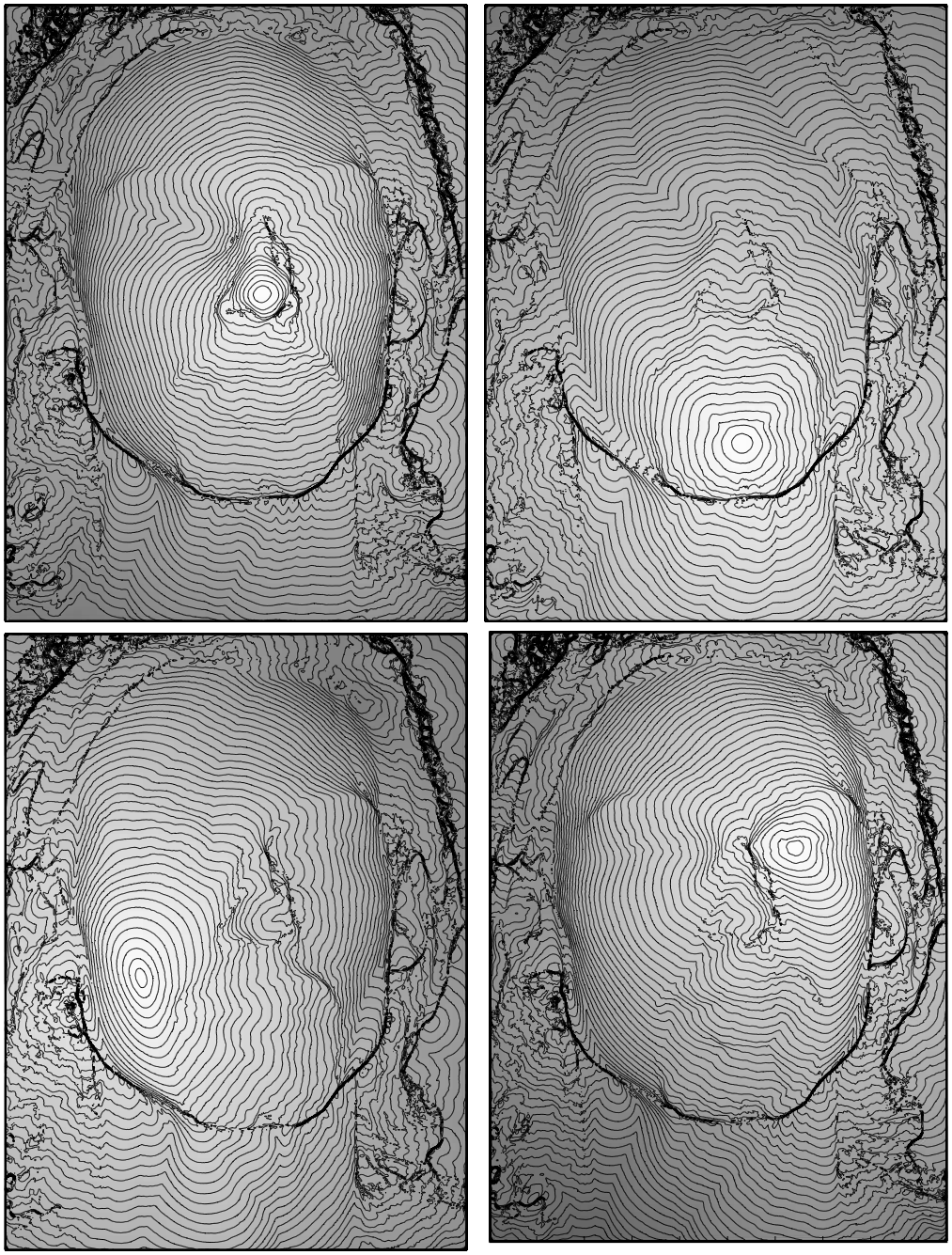

Figure 11: Distance maps of a face manifold from various source points. The level curves of the distance maps are drawn on the parameterization plane.

8 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, M. Bronstein and A. Bronstein for the facial manifold metric data, and the anonymous reviewers for their good comments.

References

[1] A. Bronstein, M. Bronstein, and R. Kimmel. Expression-invariant 3D face recognition. In *Proc. Audio and Video-based Biometric Person Authentication (AVBPA), Lecture Notes in Comp. Science 2688*, pages 62–69. Springer, 2003.

[2] A. Bronstein, M. Bronstein, R. Kimmel, and A. Spira. 3D face recognition without facial surface reconstruction. *submitted for publication*, March 2003.

[3] P. Danielson. Euclidean distance mapping. *Computer Graphics and Image Processing*, (14):227–248, 1980.

[4] A. Elad and R. Kimmel. Bending invariant representations for surfaces. In *Proc. of CVPR 2001*, Hawaii, December 2001.

[5] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[6] R. Kimmel, R. Malladi, and N. Sochen. Image processing via the Beltrami operator. In *Proc. of 3-rd Asian Conf. on Computer Vision*, Hong Kong, January 1998.

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

[8] 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.

[9] 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.

[10] 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.

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

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

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

[14] 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.

[15] 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.

[16] N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Trans. on Image Processing*, 7(3):310–318, 1998.

[17] A. Spira and R. Kimmel. An efficient solution to the Eikonal equation on parametric manifolds. In *INTERPHASE 2003 meeting, Isaac Newton Institute for Mathematical Sciences, 2003 Preprints, Preprint no. NI03045-CPD*, UK, June 2003.

[18] A. Spira, R. Kimmel, and N. Sochen. Efficient Beltrami flow using a short time kernel. In *Proc. of Scale Space 2003*, Isle of Skye, Scotland, June 2003.

[19] Y. Tsai, L. Cheng, S. Osher, and H. Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SIAM Journal on Numerical Analysis*, 41(2):673–694, 2003.

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