Efficient Beltrami Flow using a Short Time Kernel

A. Spira[1], R. Kimmel[1], and N. Sochen[2]

[1] Department of Computer Science, Technion, Israel
{salon,ron}@cs.technion.ac.il
[2] Department of Applied Mathematics, University of Tel-Aviv, Israel
sochen@math.tau.ac.il

Abstract. We introduce a short time kernel for the Beltrami image enhancing flow. The flow is implemented by 'convolving' the image with a space dependent kernel in a similar fashion to the implementation of the heat equation by a convolution with a gaussian kernel. The expression for the kernel shows, yet again, the connection between the Beltrami flow and the Bilateral filter. The kernel is calculated by measuring distances on the image manifold by an efficient variation of the fast marching method. The kernel, thus obtained, can be used for arbitrary large time steps in order to produce adaptive smoothing and/or a new scale-space. We apply it to gray scale and color images to demonstrate its flow like behavior.

1 Introduction

The Beltrami flow [4, 12] is a powerful tool for image enhancement. Its good visual effect results from de-noising the image while keeping the edges intact. The flow originates from minimizing the area of the 2-dimensional Riemannian image manifold embedded in $\mathbb{R}^N$, where $N = 3$ for gray scale images and $N = 5$ for color images.

A short time kernel has been presented for 1D non-linear diffusion in [10] and an approximation for the 2D Beltrami operator in [9]. These kernels enable the implementation of the flows by 'convolving' the signals with the kernels, similar to the implementation of the heat equation by a convolution with a gaussian kernel. This implementation replaces the conventional method of solving the first variation as a gradient descent PDE process by the appropriate numerical schemes. One of the main advantages of this approach is the ability to select an arbitrary time step for the kernel.

In order to compute the short time kernel we need to calculate distances on the image manifold. Measuring distances on manifolds has been done before for triangulated manifolds [5], graphs of functions [8], and implicit manifolds [6]. Here, we propose a new variation of the fast marching method for calculating the distances, especially suited for image manifolds.

This paper is organized as follows. The first section describes the Beltrami flow for gray scale and color images. In Section 2 the derivation of the short

time kernel is presented. Section 3 reviews our new contribution for calculating geodesic distances on the image manifold, which is required for the implementation of the short time kernel. The simulations and results are in Section 4 and the conclusions in Section 5.

2 The Beltrami Flow

In the Beltrami framework the image is regarded as the embedding $X : U \to \mathbb{R}^N$, with U the 2-dimensional image manifold and $\mathbb{R}^N$ the space-feature manifold. For gray scale images

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

where u^1, u^2 are the space coordinates and I is the intensity component. The metric h_{ij} of the space-feature manifold is

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

where β is the relative scale between the space coordinates and the intensity component. This is an Euclidean space-feature manifold. Non-Euclidean manifolds were addressed in [11, 13]. The metric elements g_{ij} of the image manifold U are derived from the metric elements h_{ij} and the embedding X by the pullback procedure

$$G = (g_{ij}) = \begin{pmatrix} 1 + \beta^2 I_1^2 & \beta^2 I_1 I_2 \\ \beta^2 I_1 I_2 & 1 + \beta^2 I_2^2 \end{pmatrix}, \tag{3}$$

where $I_i \triangleq \frac{\partial I}{\partial u^i}$.

For color images

$$X(u^1, u^2) = \{u^1, u^2, I^1(u^1, u^2), I^2(u^1, u^2), I^3(u^1, u^2)\}, \tag{4}$$

where I^1, I^2, I^3 are the three color components (for instance red, green and blue for the RGB color space). The metric h_{ij} of the space-feature manifold is

$$H = (h_{ij}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \beta^2 & 0 & 0 \\ 0 & 0 & 0 & \beta^2 & 0 \\ 0 & 0 & 0 & 0 & \beta^2 \end{pmatrix}, \tag{5}$$

and the metric of the image manifold is

$$G = (g_{ij}) = \begin{pmatrix} 1 + \beta^2 \sum_a (I_1^a)^2 & \beta^2 \sum_a I_1^a I_2^a \\ \beta^2 \sum_a I_1^a I_2^a & 1 + \beta^2 \sum_a (I_2^a)^2 \end{pmatrix}. \tag{6}$$

The Beltrami flow can be obtained by minimizing the area of the image manifold

$$S = \iint \sqrt{g} du_1 du_2, \tag{7}$$

with respect to the embedding, where $g = \det(G) = g_{11}g_{22} - g_{12}^2$. The corresponding Euler-Lagrange equations as a gradient descent process are

$$X_t^a = -g^{-\frac{1}{2}} h^{ab} \frac{\delta S}{\delta X^b} = g^{-\frac{1}{2}} \partial_i (g^{\frac{1}{2}} g^{ij} \partial_j X^a), \tag{8}$$

with g^{ij} the contravariant metric of the image manifold (the inverse of the metric tensor g_{ij}) and using Einstein's summation convention. In a matricial form it reads

$$X_t^a = \underbrace{\frac{1}{\sqrt{g}} \mathrm{Div}\left(\sqrt{g} G^{-1} \nabla X^a\right)}_{\Delta_g X^a} \tag{9}$$

where Δ_g is the Laplace-Beltrami operator which is the extension of the Laplacian to manifolds.

For gray scale images we get

$$I_t = \Delta_g I. \tag{10}$$

For color images we get for each color component

$$I_t^i = \Delta_g I^i. \tag{11}$$

We introduce in the next sections the kernel method for solving these coupled and highly non-linear partial differential equations.

3 A Short Time Kernel for the Beltrami Flow

It can be shown that applying the heat equation

$$I_t = \Delta I \tag{12}$$

to the 2-dimensional data $I(u^1, u^2, t_0)$ for the duration t is equivalent to convolving the data with a Gaussian kernel

$$\begin{aligned} I(u^1, u^2, t_0 + t) &= \int I(\tilde{u}^1, \tilde{u}^2, t_0) K(|u^1 - \tilde{u}^1|, |u^2 - \tilde{u}^2|; t) d\tilde{u}^1 d\tilde{u}^2 \\ &= I(u^1, u^2, t_0) * K(u^1, u^2; t) \ , \end{aligned} \tag{13}$$

where the kernel is given by

$$K(u^1, u^2; t) = \frac{1}{4\pi t} \exp\left(-\frac{(u^1)^2 + (u^2)^2}{4t}\right) \ . \tag{14}$$

An iterative implementation of the PDE is replaced, in this approach, by a one step filter.

In this section we extend this result to the Beltrami flow. Because of the non-linearity of this flow (the Beltrami operator depends on the data I), a global (in time) kernel is impossible. We therefore develop a short time kernel that if used iteratively, has an equivalent effect to that of the Beltrami flow. We replace Equation (13) with

$$I^i(u^1,u^2,t_0+t)=\int I^i(\tilde{u}^1,\tilde{u}^2,t_0)K(u^1,u^2,\tilde{u}^1,\tilde{u}^2;t)d\tilde{u}^1d\tilde{u}^2\ , \tag{15}$$

which we denote by

$$I^i(u^1,u^2,t_0+t)=I^i(u^1,u^2,t_0)*_g K(u^1,u^2;t), \tag{16}$$

This is not a convolution in the strict sense, because K does not depend on the differences $u^i-\tilde{u}^i$. In general, the coordinates u^i are arbitrary local coordinates on the manifold. These coordinates are not a geometric object and the difference between coordinates, therefore, has no intrinsic meaning. We will justify our definition of a convolution on a manifold after we develop the explicit form of the kernel. The general form of K is

$$K(u^1,u^2;t)=\frac{H(u^1,u^2;t)}{t}\exp\left(-\frac{\psi^2(u^1,u^2)}{t}\right), \tag{17}$$

where we take, without lose of generality, $(\tilde{u}^1,\tilde{u}^2)=(0,0)$ and omit from K the notation of dependency on these coordinates. It will be re-instated later on by fixing the integration constants. Note, that ψ does not depend on t at all, while H is a regular function of t and can be expanded as a Taylor series $H(x,y,t)=\sum_{n=0}^{\infty}H_n(x,y)t^n$. In order to find K, we use the fact that it should satisfy Equation (11). Therefore,

$$K_t=\Delta_g K. \tag{18}$$

The left hand side of the equation is

$$\begin{aligned}K_t&=\partial_t\left(\frac{H}{t}\exp\left(-\frac{\psi^2}{t}\right)\right)\\&=\left(\frac{H_1}{t}-\frac{H_0}{t^2}-\frac{H_1}{t}+\frac{H_0\psi^2}{t^3}+\frac{H_1\psi^2}{t^2}+O(\frac{1}{t})\right)\exp\left(-\frac{\psi^2}{t}\right)\\&=\left(\frac{H_0\psi^2}{t^3}+\frac{H_1\psi^2-H_0}{t^2}+O(\frac{1}{t})\right)\exp\left(-\frac{\psi^2}{t}\right)\end{aligned} \tag{19}$$

For the right hand side of Equation (18) we calculate

$$K_i=\frac{\partial K}{\partial u^i}=\left(\frac{H_{0i}}{t}-\frac{2H_0\psi\psi_i}{t^2}-\frac{2H_1\psi\psi_i}{t}+O(1)\right)\exp\left(-\frac{\psi^2}{t}\right)\ . \tag{20}$$

The second derivative is calculated similarly. The first two leading terms that multiply the exponential are

$$\frac{4H_0\psi^2\psi_i\psi_j}{t^3} - \frac{2(H_{0i} - 2H_1\psi\psi_i)\psi\psi_j + \partial_j(2H_0\psi\psi_i)}{t^2} . \tag{21}$$

Putting everything together, we get for the leading order

$$\Delta_g K = \frac{4}{t^3} g^{ij} \left(\psi^2\psi_i\psi_j + O(t)\right) K. \tag{22}$$

Equating the leading terms in Equations (19) and (18), yields

$$g^{ij}\psi_i\psi_j = \|\nabla_g\psi\|^2 = \frac{1}{4}, \tag{23}$$

with ∇_g the extension of the gradient to the manifold. This is the Eikonal equation on the manifold, and its viscosity solution is a geodesic distance map ψ on the manifold. An efficient solution of the Eikonal equation for image manifolds is given in the next section. The H_n coefficients, which depend on the spatial variables, are solutions of the PDEs that are obtained by equating the coefficients of powers of t. It is not too difficult to be convinced that H_0 is a constant (see [10] for an example of such a computation).

The resulting short time kernel is thereby

$$\begin{aligned} K(u^1, u^2, \tilde{u}^1, \tilde{u}^2; t) &= \frac{H_0}{t} \exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)} ds\right)^2}{4t}\right) \\ &= \frac{H_0}{t} \exp\left(-\frac{d_g^2\left((u^1, u^2), (\tilde{u}^1, \tilde{u}^2)\right)}{4t}\right) , \end{aligned} \tag{24}$$

where ds is an arc-length element on the manifold, and $d_g(p_1, p_2)$ is the geodesic distance between two points, p_1 and p_2, on the manifold. Note that in the Euclidean space with Cartesian coordinate system $d_E(p_1, p_2) = |p_1 - p_2|$. The geodesic distance on manifolds is therefore the natural generalization of the difference between coordinates in the Euclidean space. It is natural then to define the convolution on a manifold by

$$I^i(u^1, u^2) *_g K(u^1, u^2; t) = \int I^i(\tilde{u}^1, \tilde{u}^2) K\left(d_g\left((u^1, u^2), (\tilde{u}^1, \tilde{u}^2)\right)\right) d\tilde{u}^1 d\tilde{u}^2 \tag{25}$$

The update step for the image is

$$I^i(u^1, u^2, t_0+t) = \frac{H_0}{t} \iint_{(\tilde{u}^1,\tilde{u}^2)\in N(u^1,u^2)} I^i(\tilde{u}^1, \tilde{u}^2, t_0) \exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)} ds\right)^2}{4t}\right) d\tilde{u}^1 d\tilde{u}^2, \tag{26}$$

with $N(u^1, u^2)$ the neighborhood of the point (u^1, u^2), where the value of the kernel is above a certain threshold. Because of the monotone nature of the fast marching algorithm used in the next section for the solution of the Eikonal equation, once a point is reached, where the value of the kernel is smaller than the threshold, the algorithm can stop and thereby naturally bound the numerical support of the kernel. The value of the kernel for the remaining points of the manifold would be negligible. Therefore, the Eikonal equation is solved only in a small neighborhood of each image point. H_0 is taken such that integration over the kernel in the neighborhood $N(u^1, u^2)$ of the point equals one.

The short time Beltrami kernel in Equation (24) is very similar to the Bilateral filter kernel [15, 3]. The difference between them is that the Beltrami kernel uses geodesic distances on the image manifold, while the Bilateral kernel uses Euclidean distances. The derivation of the Beltrami kernel shows that the Bilateral filter originates from image manifold area minimization. The Bilateral filter can actually be viewed as an Euclidean approximation of the Beltrami flow. Another connection between the Beltrami flow and the Bilateral filter appears in [2].

The Euclidean distance used in the Bilateral filter, while being easier to calculate, does not take into account the image intensity values between two image points. A point can have a relatively high kernel value, although it belongs to a different object than that of the filtered image point. The Beltrami kernel takes this effect into account and penalizes a point that belongs to a different 'connected component'. That is, it is not 'as blind' as the Bilateral filter to the spatial structure of the image.

4 Solving the Eikonal Equation on Image Manifolds

The image manifold is a parametric manifold, where the metric g_{ij} is given for every point. We present here an efficient solution for the Eikonal equation on parametric manifolds, based on the fast marching approach [7]. A more detailed description appears in [14].

The original fast marching algorithm [7] solves the Eikonal equation in an orthogonal coordinate system. In this case, the numerical support for the update of a grid point consists of one or two points out of its four neighbors. The first point is from the up/down pair and the second is from the left/right pair. The two selected grid points, together with the updated one, compose the vertices of a right triangle. See Figure 1.

This is not the case for image manifolds. There $g_{12} \neq 0$ and we get a non-orthogonal coordinate system on the manifold, see Figure 2. The resulting angles are not necessarily right angles. If a grid point is updated by a stencil which includes an obtuse angle, a problem may arise. The value of one of the points of the stencil might not be set in time and cannot be used. There is a similar problem with fast marching on triangulated domains which include obtuse angles [5].

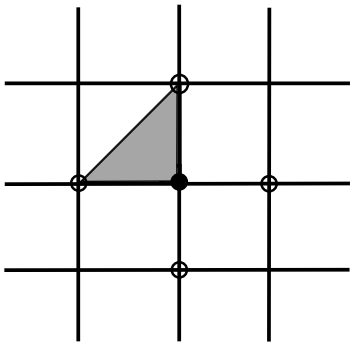

Fig. 1. The numerical support for the orthogonal fast marching algorithm.

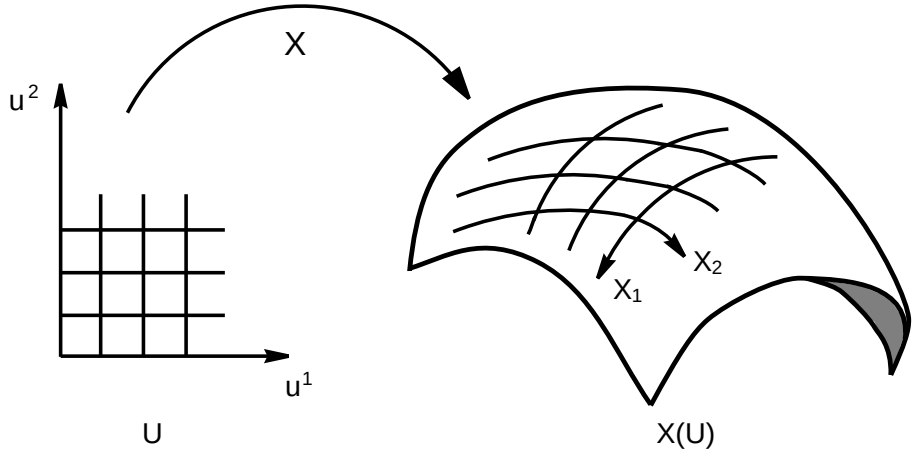

Fig. 2. The orthogonal grid on the parameterization plane is transformed into a non-orthogonal grid on the manifold.

Our solution is similar to that of [5]. We perform a pre-processing stage for the grid, in which we split every obtuse angle into two acute ones, see Figure 3. 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

$$\cos(\alpha) = \left(\frac{X_1 \cdot X_2}{\|X_1\|\|X_2\|}\right) = \frac{g_{12}}{\sqrt{g_{11}g_{22}}}. \tag{27}$$

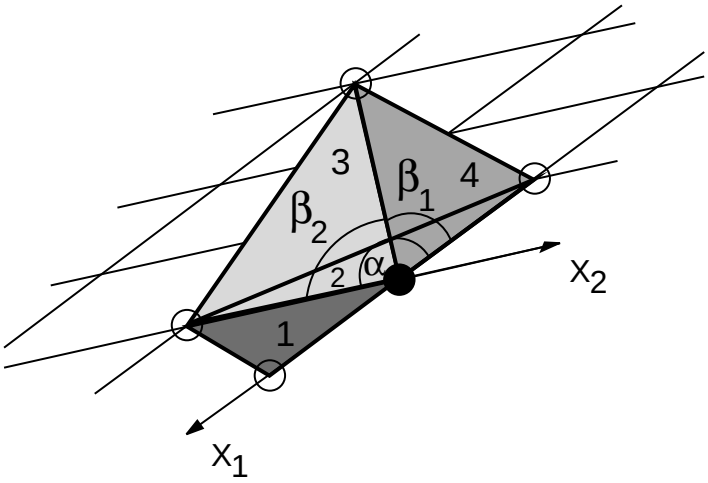

Fig. 3. The numerical support for the non-orthogonal coordinate system. Triangle 1 gives a proper numerical support, yet triangle 2 is obtuse. It is replaced by triangle 3 and triangle 4.

If $\cos(\alpha) = 0$, the axes are perpendicular, and no splitting is required. If $\cos(\alpha) < 0$, the angle α is obtuse and should be split. The denominator of Equation (27) 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 can be negative). The point is a proper supporting point, if the obtuse angle is split into two acute ones. For $\cos(\alpha) < 0$ this is the case if

$$\cos(\beta_1) = \left(\frac{X_1 \cdot (mX_1 + nX_2)}{\|X_1\|\|mX_1 + nX_2\|}\right) = \frac{mg_{11} + ng_{12}}{\sqrt{g_{11}\left(m^2 g_{11} + 2mng_{12} + n^2 g_{22}\right)}} > 0, \tag{28}$$

and

$$\cos(\beta_2) = \left(\frac{X_2 \cdot (mX_1 + nX_2)}{\|X_2\| \|mX_1 + nX_2\|} \right) = \frac{mg_{12} + ng_{22}}{\sqrt{g_{22}\left(m^2 g_{11} + 2mng_{12} + n^2 g_{22}\right)}} > 0. \tag{29}$$

Here it is enough to check the sign of the numerator. For $\cos(\alpha) > 0$, $\cos(\beta_2)$ changes its sign and the constraints are

$$mg_{11} + ng_{12} > 0, \tag{30}$$

and

$$mg_{12} + ng_{22} < 0. \tag{31}$$

This process is done for all grid points. Once the pre-processing stage is done, we have a suitable numerical stencil for each grid point and we can solve the Eikonal equation numerically. The numerical scheme used is similar to that of solving the Eikonal equation on triangulated manifolds [5] with the exception that there is no need to perform the unfolding step. The supporting grid points that split the obtuse angles can be found more efficiently. The required triangle edge lengths and angles are calculated according to the image metric g_{ij} at the grid point, see [14].

5 Simulations and Results

First, we demonstrate the performance of our algorithm for the solution of the Eikonal equation. The algorithm is tested for a parametric manifold with a non-orthogonal coordinate system. In Figure 4 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.

Next, we use the solution of the Eikonal equation to create the short time kernel for the Beltrami flow. Figure 5 shows the implementation of the Beltrami flow for a gray scale image using a short time kernel. 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. Similar results for color images appear in [1].

The use of pixels with a weight larger than 0.01 resulted in an average of 25 neighboring pixels that take part in the filtering of each image pixel. When the threshold is reached, the fast marching algorithm is stopped, and the calculation of the distance to unnecessary points is avoided. In order to make the fast marching algorithm even faster, we can bound in advance the neighborhood in which the Eikonal equation is solved. This way, the pre-processing stage of the algorithm, including the splitting of obtuse angles, is done only for relevant pixels. In Figure 5 the size of this neighborhood is 7×7.

In order to demonstrate the spatial structure of the kernel, we tested it on the synthetic image in Figure 6. At isotropic areas of the image, the kernel is isotropic and its weights are determined solely by the spatial distance from the

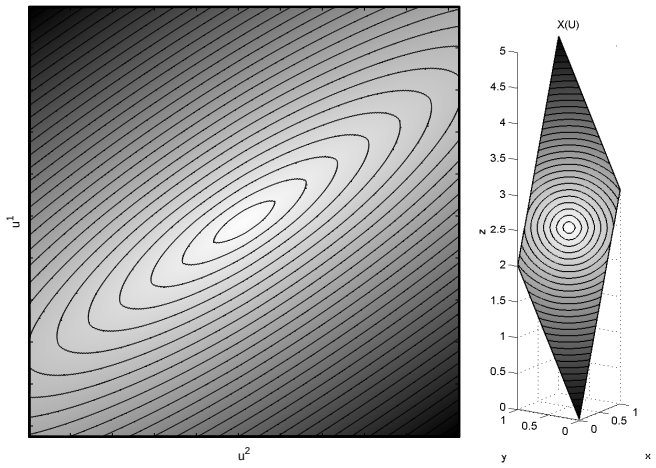

Fig. 4. Fast marching on the manifold $z = 3x + 2y$. Left: implemented on the parameterization plane. Right: projected on the manifold. Lower values are assigned brighter colors. The black curves are the level curves.

filtered pixel. Across edges the significant change in intensity is translated into a long geodesic distance, which results in negligent kernel weights on the other side of the edge. The filtered pixel is computed as an average of the pixels on the 'right' side of the edge.

6 Conclusions

A short time kernel was derived for the Beltrami image enhancing flow. Geodesic distances on the image manifold, which are required for the implementation of the kernel, were calculated in a new efficient way. From the theoretical stand point, a connection has been shown between the Beltrami flow and the Bilateral filter. The Bilateral filter is found to be a Euclidean approximation of the Beltrami flow. From a practical stand point, the kernel filter enables an arbitrary time step for a Beltrami-like adaptive smoothing, which is impossible for the explicit numerical schemes currently existing for color images.

References

1. http://www.cs.technion.ac.il/~ron/.
2. D. Barash. Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):844–847, June 2002.
3. M. Elad. On the bilateral filter and ways to improve it. *IEEE Transactions on Image Processing*, 11(10):1141–1151, October 2002.

Fig. 5. Beltrami flow using a short time kernel. The original image is in the top left. The order of the images is from top to bottom and left to right.

4. 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.
5. R. Kimmel and J. Sethian. Computing geodesic paths on manifolds. *Proceedings of National Academy of Sciences*, 95(15):8431–8435, July 1998.
6. 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.
7. J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of National Academy of Sciences*, 93(4):1591–1595, 1996.
8. J. Sethian and A. Vladimirsky. Ordered upwind methods for static hamilton-jacobi equations: theory and applications. Technical Report PAM-792(UCB), Center for Pure and Applied Mathematics, May 2001. submitted for publication to SIAM Journal on Numerical Analysis in July 2001.
9. N. Sochen. Stochastic processes in vision: From langevin to beltrami. In *Proc. of International Conference on Computer Vision*, Vancouver, Canada, July 2001.

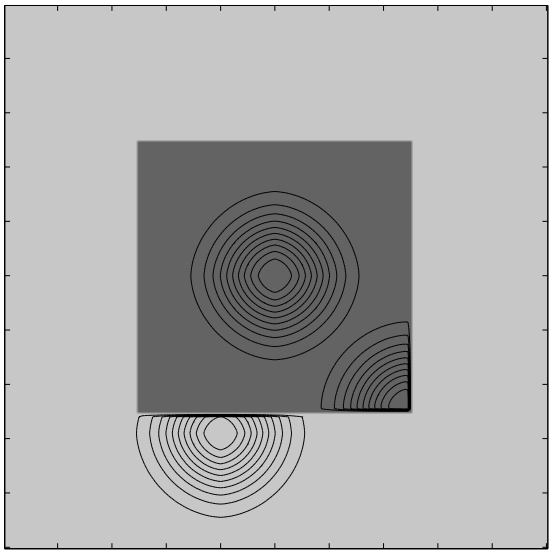

Fig. 6. Level curves of the kernel at various locations in a synthetic image.

10. N. Sochen, R. Kimmel, and A. Bruckstein. Diffusions and confusions in signal and image processing. *Journal of Mathematical Imaging and Vision*, 14(3):195–209, 2001.
11. N. Sochen, R. Kimmel, and R. Malladi. From high energy physics to low level vision. LBNL report LBNL-39243, UC Berkeley, August 1996.
12. N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Trans. on Image Processing*, 7(3):310–318, 1998.
13. N. Sochen and Y. Y. Zeevi. Representation of colored images by manifolds embedded in higher dimensional non-euclidean space. In *Proc. of ICIP98*, pages 166–170, Chicago, IL, January 1998.
14. A. Spira and R. Kimmel. An efficient solution to the eikonal equation on parametric manifolds. Submitted for publication, March 2003.
15. C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision*, Bombay, India, January 1998.