Geometric Filters, Diffusion Flows, and Kernels in Image Processing

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

[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

1 Introduction

Diffusion flows are processes applied to digital images in order to enhance or simplify them. These flows are usually implemented by appropriate discretizations of partial differential equations *partial differential equations* (PDEs). Iteratively applying these discretizations named also *numerical schemes* to an image, results in a series of images with decreasing detail, see Fig. 1. Using a suitable flow, one can enhance important image features such as edges and objects while filtering the image from undesired noise. This can be done not only to gray level and color images, but also to textures, movies, volumetric medical images, etc.
Diffusion flows are important members of the family of methods for image processing, computer vision, and computer graphics based on the numerical solution of PDEs. Other members of the family include active contours/surfaces for image segmentation, reconstruction of 3-dimensional scenes from their shading or stereo images, graphic visualization of natural phenomena, and many others. This family of methods has many advantages among them theoretical origin due to derivation from a minimization of (usually geometric) cost functions, efficiency and robustness.

2 Diffusion Flows and Geometric filters

Diffusion processes are widely spread in many areas of Physics. Naturally, they found their way to the field of image processing. At first, only linear diffusion was used, but gradually also nonlinear diffusions were introduced and geometry-based filters proposed. This section reviews the development of these methods from the early days till the recent present.

Fig. 1. A diffusion flow of a color image. The original image is on the top left

2.1 The Heat Equation

The simplest diffusion is the one generated by the 2-dimensional heat equation

$$I_t = \Delta I,$$

with $I(x, y)$ the 2-dimensional data, I_t its partial derivative according to time, and Δ the Laplacian operator $(\partial_{xx} + \partial_{yy})$. This equation depicts, for instance, the temporal change in the heat profile of a metal sheet. In our case $I(x, y)$ gives the gray level values of the image.
The heat equation was the first diffusion process applied to images [44]. It was mainly used to create a *scale space* for an image, meaning a 3-dimensional volume with a scale coordinate t added to the spatial coordinates x and y. At the origin of t we have the original image as initial condition, and as we advance along t we get smoother versions of it. The idea behind scale space is that important features of the original image should survive the change of scale and therefore all the scale space of the image should be used to detect

these features. Later on, the heat equation was suggested for filtering noise corrupting the image [9]. After applying the heat equation for a short duration, the noise which is of fine resolution would disappear.
The heat equation as a diffusion flow generating a scale space has an important attribute which is its linearity. It is therefore also referred to as *linear diffusion.* However, it damages the edges of objects in images and does not preserve connected components, see Fig. 2. This simple example was presented in the introduction of the first papers collection on this topic [37].

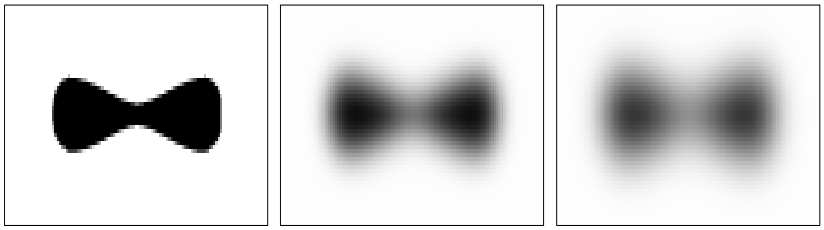

Fig. 2. The heat equation damages edges and separates connected components

2.2 The Geometric Heat Equation

New flows were suggested to overcome the problem of the change in the number of connected components. One such flow, first introduced by Alvarez, Guichard, Lions and Morel [1] in the context of invariant image processing, is the level set curvature flow. *Level set* curves are another way to describe the structure of a gray level image. Given an image $I(x, y)$, its level set curves are defined as $C(h) = \{(x, y) : I(x, y) = h\}$. See Fig. 3 for the level curves of the images in Fig. 2. The interior of a closed contour can be considered as a component, and the number of components somehow indicates the complexity of the image [3].

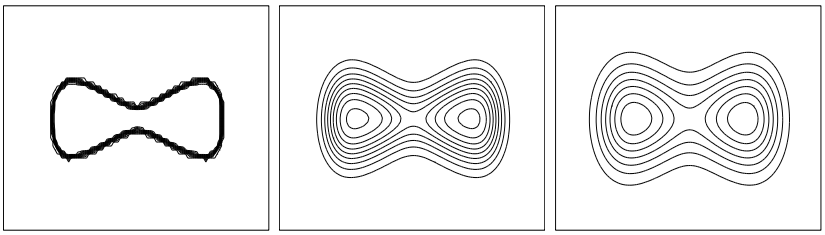

Fig. 3. Level set curves of the images in Fig. 2

The idea was to use the powerful Grayson theorem [10] for curve evolution via its curvature. The theorem states that the curvature flow

$$C_t = \kappa \mathbf{n},$$

with κ the curvature of the closed planar curve C and $\mathbf{n}$ the unit normal vector to the curve, results in the convergence of the curve to a point.
Using the Osher-Sethian [21] level set formulation the whole image could be propagated via the curvature flow equation. That is, each and every level set of the original image could be propagated by its curvature flow, and all this process could be described by a single evolution equation for the whole image given by

$$I_t = \operatorname{div}\left(\frac{\nabla I}{|\nabla I|}\right)|\nabla I|.$$

This process is possible due to the Evans-Spruck [7] confirmation that as embedding of such propagating curves is preserved, the level set formulation is indeed valid for the curvature flow. One nice property of this flow is that connected components remain connected until they disappear. Moreover, this flow is invariant to Euclidean transformations in the image plane.
Next, came the interesting question of what could be said about more complicated transformations. In [1] the authors also introduced the equi-affine invariant flow given by

$$I_t = \left(\operatorname{div}\left(\frac{\nabla I}{|\nabla I|}\right)\right)^{1/3}|\nabla I|. \tag{1}$$

Again, the connection to curve evolution was presented at the same time by Sapiro in his PhD thesis [26]. First, the curvature flow can be equivalently written by

$$C_t = C_{ss},$$

where $C(s) = \{x(s), y(s)\}$, and s is the Euclidean arc length parameterization. This is why it is also known as the *geometric heat equation* . Using similar writing for the equi-affine flow, that is,

$$C_t = C_{vv},$$

where v is the equi-affine arc length $dv = \kappa^{1/3} ds$, the resulting geometric flow can be written by

$$C_t = \kappa^{1/3}\mathbf{n}.$$

This equation known as the *affine heat equation* enjoys some of the nice properties of Grayson's theorem, like preservation of embedding of the propagating contours. It is thus directly related to Eq. (1) again via the Osher-Sethian level set formulation. These beautiful relations and geometric properties started a new era in the image processing and analysis field. For example, when smoothing stereo images we would better use the affine heat equation, and not the geometric heat equation that would distort the geometric structure relating the two images. Applications of these operators include computation of geometric signatures [12, 8], and extensions of these ideas deal with problems like geometric scale space for images painted on surfaces [13, 33].

2.3 Isotropic Nonlinear Diffusion

At the other end researchers started to explore the field of variational principles and geometry in image processing. That is, define an integral measure that somehow captures the norm of the image. For example, the TV norm was a popular selection proposed in [25]. The TV (stands for total variation) is defined by

$$\iint |\nabla I| dxdy,$$

for which the Euler-Lagrange equation is given by

$$\operatorname{div}\left(\frac{\nabla I}{|\nabla I|}\right) = 0.$$

That is, the level set curvature should be equal to zero. This geometric connection should not come as a surprise, since by the co-area equation we have that

$$\iint |\nabla I| dxdy = \iint_{\Omega} ds dh$$

where s is the arc length parameter of each and every level set contour, and h is a parameter running over the image intensities I. The zero curvature is indeed the result of minimizing the arc length of all level set contours in the image.
The methods used to denoise an image based on the TV norm usually apply the Euler-Lagrange as a gradient descent via a PDE of the form

$$I_t = \operatorname{div}\left(\frac{\nabla I}{|\nabla I|}\right).$$

Again, the corresponding flow of the image level sets can be written as

$$C_t = \frac{1}{|\nabla I|}\kappa \mathbf{n},$$

[14]. This is nothing but a selective curvature flow, where the flow is enhanced at smooth regions, and suppressed near the image edges (where the image gradient is high), so that these important features are preserved.
Another popular filter proposed at the same time is the Perona-Malik [23] *anisotropic diffusion* . Unlike its name, the filter is an inhomogeneous yet locally isotropic flow given by

$$I_t = \operatorname{div}\left(f(|\nabla I|)\nabla I\right).$$

We see that setting $f(s) = s^{-1}$ we are back with the TV flow, while other selections lead to other filters.
f is the *diffusivity* function and its role is to control the amount of diffusion according to the gradient of the image. At image edges, where $|\nabla I|$ is large,

the diffusion should be minimal, and vice versa at the interior of objects. To accomplish that, f should be monotonically decreasing. A popular choice for f is

$$f(|\nabla I|) = \frac{1}{1 + |\nabla I|^2/\lambda^2}.$$

2.4 Anisotropic Nonlinear Diffusion

Gabor [9, 2, 20, 16] was probably the first to consider anisotropic diffusion by smoothing along the edge and inverting the heat operator and thereby generating an unstable enhancing process across the edge. If we write the gradient direction as $\xi = \nabla I/|\nabla I|$ and η as the orthogonal direction, see Fig. 4, $I_t = I_{\eta\eta}$ is nothing but the curvature flow. Gabor proposed to use one iteration of a discretization of the equation

$$I_t = I_{\eta\eta} - \epsilon I_{\xi\xi},$$

where ϵ determines the amount of inverse diffusion. This simple and nice formulation for image enhancement (which can not be easily extracted from a variational principle) was re-discovered many times along the evolution of the image processing field.

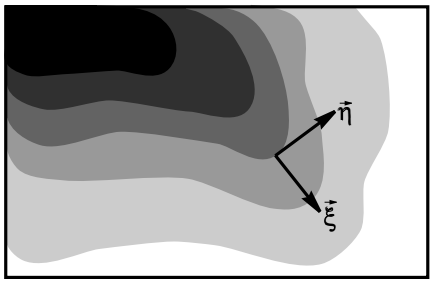

Fig. 4. The gradient direction and the tangent direction of the image level sets

A recent interesting anisotropic differential filter for image analysis is Weickert's [42] edge direction sensitive flow. Weickert's idea was to plug a 2×2 symmetric positive definite matrix instead of the scalar function $f(s)$ of the Perona-Malik flow. The orthonormal eigenvectors of the matrix are selected according to the image gradient direction

$$v_1 \parallel \nabla I, \qquad v_2 \perp \nabla I$$

and their corresponding eigenvalues are taken such that

$$\lim_{|\nabla I| \to \infty} \frac{\lambda_1(|\nabla I|)}{\lambda_2(|\nabla I|)} = 0.$$

This way, the smoothing is mostly along the edges and not across them.

2.5 Mean Curvature Flow

Many times the way we represent objects defines the way(s) in which we can manipulate them. Images, for example, were represented traditionally as a matrix of numbers. Many image processing techniques followed this representation. Recently, a more geometric point of view emerged. An image is regarded and represented as a surface. In fact, the graph of the intensity function for gray-valued images is a two-dimensional surface. One may think of it as embedded in $\mathbb{R}^3$ with coordinates x, y and I. Once described in this way it is natural to ask geometric questions such as about the curvature of the surface at a given point. We may also envisage processes that alter the geometric properties of the surface. Noting that noise is represented in the image as points (or small regions) of high curvature, it is natural to give a smoother version of the image by reducing points with high curvature. One way to achieve this goal is to define an evolution equation that depends on the curvature. We move, at each instant, the image surface in the direction of the normal to the surface. Note that this is the only direction that changes the *shape* of the image. Movement along the other two directions simply causes a reparameterization that does not change the image's gray-value content. The amount of change at each point is proportional to the mean curvature in that point. Denoting the mean curvature H, and the normal to the surface $\mathbf{N}$, we find the following PDE

$$\mathbf{S}_t = H\mathbf{N}.$$

How should we understand this equation? how is it applied to images? In order to give an answer we go back to the representation of the image as a surface. The graph of the image embedded in $\mathbb{R}^3$ is represented as the trinari $(x, y, I(x, y))$. The two tangent vectors along the canonical coordinates x and y are given by $X_1 = (1, 0, I_x)$ and $X_2 = (0, 1, I_y)$. The normal vector is derived easily as orthogonal to X_1 and X_2. Its form is

$$\mathbf{N} = \frac{1}{\sqrt{1 + |\nabla I|^2}}(-I_x, -I_y, 1).$$

The mean curvature at each point is

$$H(x, y) = \frac{(1 + I_x)^2 I_{yy} - 2I_x I_y I_{xy} + (1 + I_y^2) I_{xx}}{(1 + I_x^2 + I_y^2)^{\frac{3}{2}}}.$$

It follows that the equation is

$$(x, y, I)_t^T = H(-I_x, -I_y, 1)^T \frac{1}{\sqrt{1 + |\nabla I|^2}}.$$

Since we work in a constant domain and a constant coordinate system, namely the Cartesian x and y coordinates the only change that actually takes place is the value of the gray value at each pixel. In order to have the required effect

while changing the gray values only, we change the gray value at each point such that its projection on the normal has exactly the magnitude of the mean curvature. A simple calculation shows that we need to multiply by a factor of $\sqrt{1+|\nabla I|^2}$, see Fig. 5.

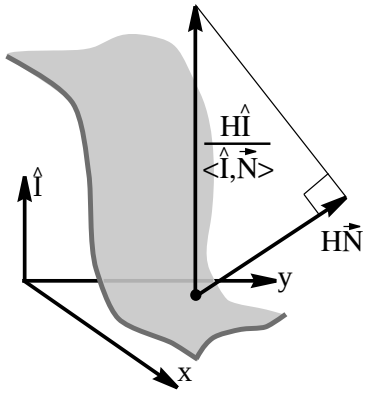

Fig. 5. The mean curvature flow for gray level images is accomplished by only changing the intensity component

The final equation is

$$I_t = \frac{(1+I_x)^2 I_{yy} - 2I_x I_y I_{xy} + (1+I_y^2) I_{xx}}{(1+I_x^2+I_y^2)}$$

2.6 Color Images

Color images are the canonical example of vector value images. The light that is reflected from a surface is described by the wavelength spectrum $R(\lambda) = S(\lambda)\rho(\lambda)$ where $S(\lambda)$ is the spectrum of the illumination and $\rho(\lambda)$ is the material reflectance property known as the albedo. Three filters are applied at each spatial point to the spectrum to produce the three channels $I^i = \int d\lambda R(\lambda) f^i(\lambda)$. These three channels are usually called Red, Green, and Blue (RGB) with respect to the regions in spectrum space where the filters extract most of their energy. The information is then encoded in three functions $R(x,y)$, $G(x,y)$ and $B(x,y)$.
There are several approaches in the denoising process of color, and other multi-channel images. The first and most simple and naive approach is to apply a denoising process to each channel separately. This approach ignores completely the correlation between the different channels. Since the channel edges are not necessarily aligned, an anisotropic channel by channel process may blur regions where only one channel has an edge. In case several strong edges in all channels exist with small offsets, artificial colors may appear.

We describe in this section a different approach where the color channels are correlated via the Di Zenzo metric [5]. More elaborate approaches that incorporate perceptual psychophysical data will be discussed below in the context of the Beltrami framework. Here we follow the approach of Sapiro and Ringach [27]. The Di Zenzo metric is defined in the color space . Its explicit form is

$$D = \begin{pmatrix} R_x^2 + G_x^2 + B_x^2 & R_x R_y + G_x G_y + B_x B_y \\ R_x R_y + G_x G_y + B_x B_y & R_y^2 + G_y^2 + B_y^2 \end{pmatrix},$$

where the subscripts x and y mean partial derivation. The elements can be written more simply with the Einstein summation convention: indices that appear twice are summed over. The elements are written as $D_{xx} = I_x^i I_x^i$ where the summation is over the index $i = 1, 2, 3$, and $I^1 = R,\ I^2 = G,\ I^3 = B$. In general $D_{\mu\nu} = I_\mu^i I_\nu^i$, where μ and ν take the values 1 and 2. They stand for x_μ and x_ν, where by convention $x_1 = x$ and $x_2 = y$.
The matrix D is real and symmetric and it can be diagonalized. Formally we can write $D = U^T \Lambda U$ where $\Lambda = \mathrm{diag}(\lambda_+, \lambda_-)$. The matrix U is composed of the eigenvectors that give the direction of maximal variation in color space and its perpendicular direction. The λ_i indicate the amount of change in each direction. Sapiro and Ringach suggest in their paper to construct an anisotropic process in the following manner:

$$I_t^i = \mathrm{div}\left(f(\lambda_+ + \lambda_-)\nabla I^i\right).$$

This equation can be derived as a gradient descent of a functional. It is simply $S[I^i] = \int \Psi(\lambda_+ + \lambda_-) dxdy$. A new analysis of this and many other approaches can be found in Tschumperlé's thesis [39].

2.7 The Beltrami Flow

In the Beltrami framework [15, 31] the image is regarded as an embedding of the image manifold in the space-feature manifold. In a more rigorous terms we describe the image as a *section* of a *fiber bundle*. The fiber bundle is composed of the spatial part, which is usually a rectangle in $\mathbb{R}^2$, called the base manifold, and the fiber that describes the feature space i.e. intensity, color, texture etc. A section of the fiber bundle is a choice of a specific feature from the feature space for every point in the base manifold. The feature space may be a linear space or a more complicated manifold. In the first case we call the section a vector field.
The most simple example is the gray-value image. Denote the embedding map by X. The explicit form of this map for gray level images is

$$X(u^1, u^2) = (u^1, u^2, I(u^1, u^2)),$$

where u^1, u^2 are the spatial coordinates and I is the intensity component, see Fig. 6. For color images the embedding map reads:

$$X(u^1, u^2) = (u^1, u^2, I^1(u^1, u^2), I^2(u^1, u^2), I^3(u^1, u^2)),$$

where I^1, I^2, I^3 are the three color components (for instance red, green and blue for the RGB color space).

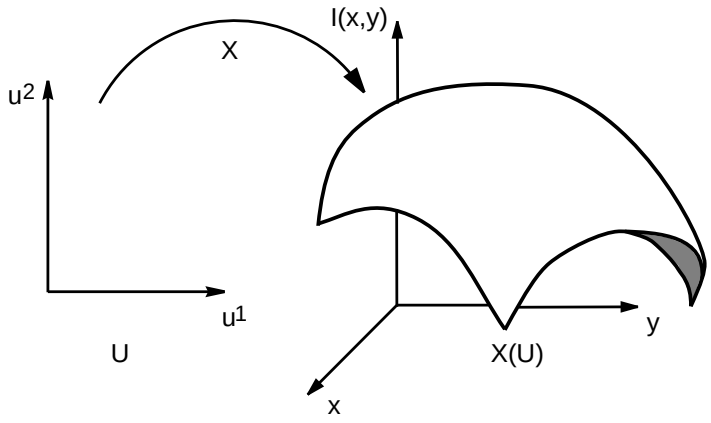

Fig. 6. A gray level image according to the Beltrami framework

The geometry of the image manifold, i.e. the section, is determined according to its metric tensor G, which is the result of the metric H chosen for the space-feature manifold, i.e. the fiber bundle . A natural choice for gray level images is a Euclidean space-feature manifold with the metric

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

where β is the relative scale between the space coordinates and the intensity component. The metric G of the image manifold is derived from the metric H and the embedding X by the pullback procedure

$$(\mathsf{G})_{ij} = \partial_i X^a \partial_j X^b h_{ab} \ .$$

Using the explicit form of the embedding map X and the metric of the fiber bundle H for gray level images, we can find the metric G:

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

where $I_i \triangleq \frac{\partial I}{\partial u^i}$.
The Euclidean metric H of the space-feature manifold for color images is

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

where the same scaling factor was chosen for the three color channels. The resulting image metric is

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

The Beltrami flow is obtained by minimizing the area of the image manifold

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

with respect to the intensity components, where $g = \det(\mathsf{G}) = g_{11}g_{22} - g_{12}^2$. The gradient descent process is given by the corresponding Euler-Lagrange equations

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

with g^{ij} the components of the contravariant metric of the image manifold G^{-1} (the inverse of the metric tensor G). The Christoffel symbols (also known as the Levi-Civita coefficients) Γ_{bc}^a are defined in terms of the fiber bundle metric H:

$$\Gamma_{bc}^a = \frac{1}{2} h^{ad} \left(\partial_b h_{dc} + \partial_c h_{bd} - \partial_d h_{bc} \right). \tag{2}$$

In matrix form it reads

$$X_t^a = \underbrace{\frac{1}{\sqrt{g}} \mathrm{div} \left(\sqrt{g} \mathsf{G}^{-1} \nabla X^a \right)}_{\Delta_g X^a} + Tr(\Gamma^a F),$$

where Γ^a is the matrix whose elements are $(\Gamma^a)_{ab} = \Gamma_{ab}^a$ and $F_{ab} = \partial_i X^a \partial_j X^b g^{ij}$. The symbol Δ_g is the Laplace-Beltrami operator which is the extension of the Laplacian to manifolds. The resulting diffusion flow for gray level images is

$$I_t = \Delta_g I = H\langle \hat{I}, \mathbf{N} \rangle,$$

i.e., the image surface moves according to the intensity component of the mean curvature flow, see Fig. 7. Because we chose a Euclidean feature space the Christoffel symbols are identically zero in this case. They vanish for color images as well. The diffusion equation for each color component reads

$$I_t^i = \Delta_g I^i. \tag{3}$$

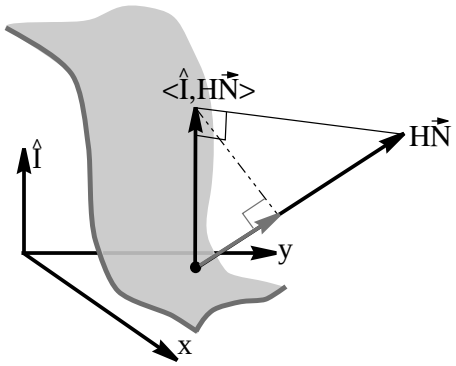

Fig. 7. In the Beltrami flow for gray level images the image surface moves according to the intensity component of the mean curvature flow. Geometrically, only the projection of this movement on the normal to the surface matters

The diffusion process in Fig. 1 is actually the Beltrami flow. Figure 8 contains a closeup of the images in Fig. 1, including the 2-dimensional manifolds of the red, green and blue color components. It is evident that the Beltrami flow filters out the noise while not only preserving the edges, but keeping their location in the three color components aligned.

3 Extending the Beltrami Framework

The basic idea of the Beltrami framework of treating the image as a manifold and enhancing it by minimizing its area can be extended in various ways. In this section the framework is extended to higher dimensional spaces (for texture, video and volumetric data), non-Euclidean feature spaces and other diffusion directions.

3.1 Texture, Video and Volumetric Data

We have discussed color for which researchers try to give a simple geometric interpretation, like an arc length that would capture our visual sensitivity to colors. Next, we claimed that in order to extract technology from such definitions we need to link the color arc length to another measure of distance in the image domain. This way we came up with the hybrid space idea.
Next comes the interesting question of what is texture and how should we treat it? Like color, we try to interpret texture as a region for which homogeneity is no longer determined by a single constat like color, but rather repeating patterns in the image domain. Again we need some sort of measure that defines a distance between different patterns. There are many ways to achieve this

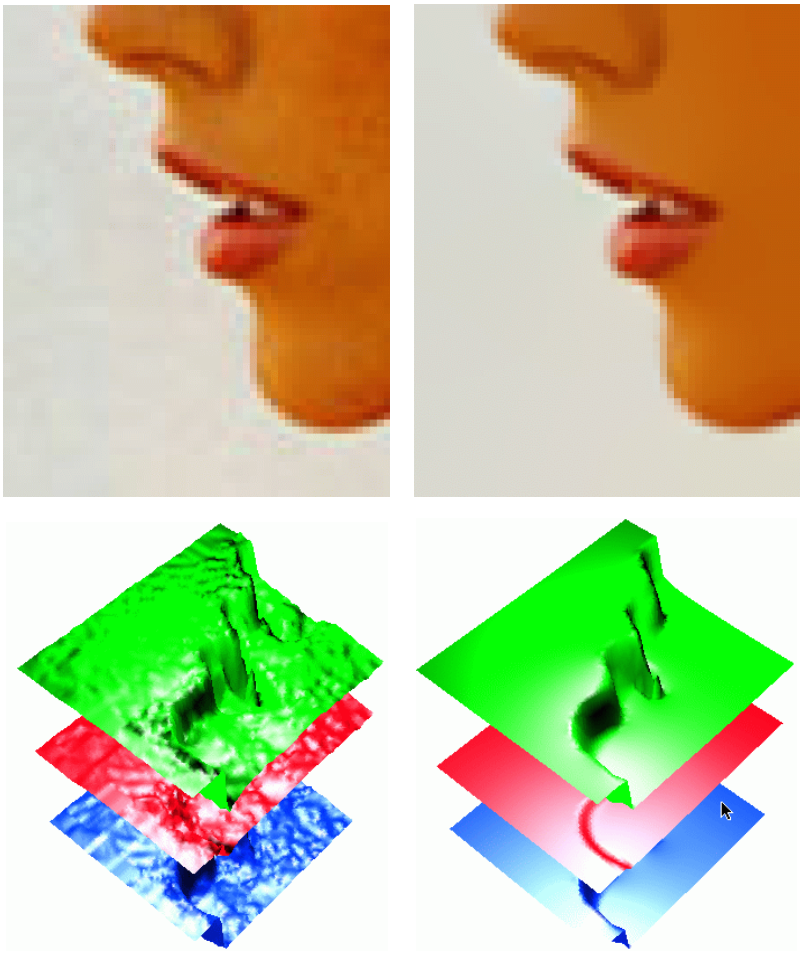

Fig. 8. The results of the Beltrami filter. The original image is on the left and the filtered one on the right

goal [24]. Once such an arc length is defined, all we need to do is to plug it into our Beltrami framework and we have a filter for texture.
Such filters were reported in [16], where the texture is represented by using the Gabor-Morlet wavelet transform $W(x, y, \theta, \sigma)$ [19], with x and y the spatial coordinates, θ the wavelet orientation parameter and σ the wavelet scale parameter. The texture image is the embedding $(x, y, \theta, \sigma) \rightarrow (x, y, \theta, \sigma, R, J)$, where $R = real(W)$ and $J = imag(W)$. Each scale is considered as a different space, resulting with the metric

$$\mathsf{G} = (g_{ij}) = \begin{pmatrix} 1 + R_x^2 + J_x^2 & R_x R_y + J_x J_y & R_x R_\theta + J_x J_\theta \\ R_x R_y + J_x J_y & 1 + R_y^2 + J_y^2 & R_y R_\theta + J_y J_\theta \\ R_x R_\theta + J_x J_\theta & R_y R_\theta + J_y J_\theta & 1 + R_\theta^2 + J_\theta^2 \end{pmatrix},$$

and the Beltrami flow

$$\begin{aligned} R_t &= \Delta_g R \\ J_t &= \Delta_g J. \end{aligned}$$

Consequently, each scale can be filtered in a different way and to a different extent. See Fig. 9 for a demonstration of texture enhancement using the Beltrami flow.

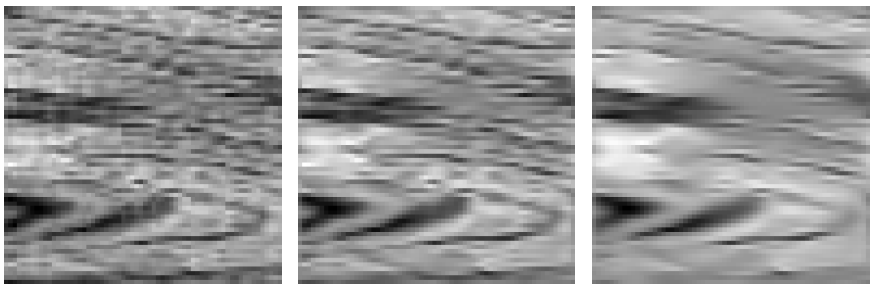

Fig. 9. Texture enhancement by using the Beltrami filter on the Gabor-Morlet wavelet transform of the texture image. The original image is on the left

The Beltrami filter for gray level video and volumetric medical data (such as CT or MRI) is accomplished by considering them as the embedding $(x, y, z) \to (x, y, z, I)$, where for video z represents time and for volumetric data the third spatial coordinate. The induced metric in this case is

$$\mathsf{G} = (g_{ij}) = \begin{pmatrix} 1 + I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & 1 + I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & 1 + I_z^2 \end{pmatrix},$$

and the Beltrami flow is

$$I_t = \frac{1}{\sqrt{g}} \mathrm{div} \left(\frac{\nabla I}{\sqrt{g}} \right),$$

where $\nabla I = (I_x, I_y, I_z)$ and $g = 1 + I_x^2 + I_y^2 + I_z^2$.

3.2 Non-Euclidean Feature Spaces

We have seen above that the image is represented as an embedding of a surface in a spatial-feature space. In the previous subsections we treated many tasks

in which the feature space is Euclidean and endowed with a Cartesian coordinate system. There are many instances where the situation is different. We shall present below two such cases: perceptual color denoising and orientation diffusion.

Color Image Denoising

The construction of the RGB color space was described in the section on color images. While the coordinates in this color space are perfectly defined from a physical point of view they are not enough in order to denoise color images aimed to be seen by human beings. The most important notion in denoising is distance. What is relevant in denoising color images is to understand how distances between colors are *perceived* by humans. In other words we treat the perceptual color space as a three-dimensional manifold whose local coordinates are given by the RGB system. What is needed in order to complete the picture is to provide the *metric* on this manifolds such that distances between colors can be measured with accordance to perception. This distance can not be deduced from physics and must be given from psychophysical experiments and considerations. Albeit its modern appearance, this paradigm is more than a century old. The first formulation of the perceptual color space as a Riemannian manifold is due to Helmholtz [11] in 1896 ! Helmholtz suggested a metric which is based on the famous log response of our senses. While it is good as a first approximation it was soon realized that his metric is inappropriate and does not describe well the experiments in various regions of the perceptual color space. The experiments are based on the notion of Just Noticeable Differences (JND). In a typical JND experiment two squares of the same color are shown to a subject. One of these squares gradually changes its color until the subject declares that the colors are different. This gives a map of infinitesimal distances in color space and can be compared directly to metrics that model this human color perception. The construction of such metrics captured the interest of prominent scientist such as Helmholtz and Schrödinger [28]. The Helmholtz model is given simply by the following line element:

$$ds^2 = (d\log R)^2 + (d\log G)^2 + (d\log B)^2$$

This equation ignores the dependence of the JND on the overall luminance. Schrödinger tried to rectify this line element and suggested the following model:

$$ds^2 = \frac{1}{R+G+B}\left(\frac{dR^2}{R} + \frac{dG^2}{G} + \frac{dB^2}{B}\right) .$$

More recent efforts to model the metric of the perceptual color space include Stiles [36] and Vos and Walraven [41].
We will demonstrate here the denoising with respect to the Helmholtz and Schrödinger metrics only. For a thorough discussion refer to [32]. Let us denote the perceptual color Riemannian manifold by M_c. The Beltrami framework

describes a color image as the embedding of a two-dimensional surface in the fiber bundle $\mathbb{R}^2 \times M_c$. The base manifold is $\mathbb{R}^2$. At each point in the base manifold the fiber M_c is attached. A color image is a *section* of this fiber bundle. The metric on the fiber bundle is simply

$$ds^2 = ds^2_{spatial} + ds^2_{color} = dx^2 + dy^2 + dI^i dI^j h_{ij}$$

where for the Helmholtz model

$$(h_{ij}) = \begin{pmatrix} \frac{1}{R^2} & 0 & 0 \\ 0 & \frac{1}{G^2} & 0 \\ 0 & 0 & \frac{1}{B^2} \end{pmatrix},$$

and for the Schrödinger model it is

$$(h_{ij}) = \frac{1}{R+G+B} \begin{pmatrix} \frac{1}{R} & 0 & 0 \\ 0 & \frac{1}{G} & 0 \\ 0 & 0 & \frac{1}{B} \end{pmatrix}.$$

The induced metric on the section is simply

$$g_{\mu\nu} = \delta_{\mu\nu} + I^i_\mu I^j_\nu h_{ij},$$

and the Levi-Civita coefficients are given by Eq. (2). The Beltrami flow then is

$$I^i_t = \Delta_g I^I + \Gamma^I_{jk} \partial_\mu X^j \partial_\nu X^k g^{\mu\nu}$$

Orientation Diffusion

Another example of a non-Euclidean feature space is the orientation [18]. In this case the feature manifold is the unit circle $\mathbf{S}^1$. We construct again the fiber bundle $\mathbb{R}^2 \times \mathbf{S}^1$ and regard the orientation vector field as a section of this fiber bundle. In order to express the metric on this fiber bundle we cover $\mathbf{S}^1$ with two coordinate patches. This can be done in various ways. We present here the hemispheric coordinates for simplicity. Embedding the orientation circle in $\mathbb{R}^2$ with Cartesian coordinates u and v we find that $\mathbf{S^1}$ is given by $u^2 + v^2 = 1$. We write the metric on the patch of $\mathbf{S}$ described by u as

$$ds^2 = du^2 + dv^2 = (1 + \frac{u^2}{1-u^2})du^2 = \frac{1}{1-u^2}du^2 = A(u)du^2$$

Having calculated the metric on the fiber we can now deduce the induced metric on the section

$$ds^2 = dx^2 + dy^2 + A(u)du^2 = (1 + A(u)u_x^2)dx^2 + 2A(u)u_x u_y dxdy + (1 + A(u)u_y^2)dy^2$$

Note that the metric on the fiber bundle is given by

$$(h_{ij}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{1-u^2} \end{pmatrix}.$$

The Levi-Civita coefficients can be calculated by Eq. (2). The Beltrami flow equation reads:

$$u_t = \Delta_g u + \Gamma^u_{jk} \partial_\mu X^j \partial_\nu X^k g^{\mu\nu}.$$

The Beltrami flow modifies the features *in the feature manifold* such that a unit length vector stays always a unit length vector along the flow.

3.3 Inverse Diffusion Across Edges

An interesting approach to extend Gabor's original idea [9] for image enhancement via

$$I_t = I_{\eta\eta} - \epsilon I_{\xi\xi},$$

is to try to manipulate the eigenvalues of the inverse metric matrix in the Beltrami operator. If these values are kept positive, the result is a diffusion that can be enhanced in a specific direction as proposed by Weickert in his 'coherence enhancement' filters [43]. More interesting, yet obviously less stable, is the concept of negative eigenvalues that mimic Gabor's inverse diffusion across the edge. This was first introduced in [16].
The concept is simple. We first extract the inverse metric matrix (g^{ij}) and compute its eigenstructure, $(g^{ij}) = U \Lambda U^T$. Next, manipulate the eigenvalues so that the smaller one gets a negative sign. This way, the inverse diffusion across the edge, due to the negative sign, enhances and sharpens the edges in the image, while the diffusion along the edges (the direction orthogonal to the maximal change direction) smooths the boundaries and adds some control to the process. See Fig. 10 for an example of this process. This is an extension to Gabor's original idea from 1965, that exploits the geometric structure of the color image, where there are no level-sets or 'isophots' due to its multi-channel nature.

4 Numerical Schemes

The PDEs describing the diffusion processes are continuous, but they are implemented on discrete digital images by computer algorithms with discrete representations. The means to bridge this gap are the numerical schemes that ensure that the discrete solution will converge to the continuous one as the grid is refined.
Many numerical schemes are used for the solution of the image diffusion PDEs. Among them the fast fourier transform (FFT), wavelet transforms, finite element techniques, neural networks, multigrid methods and many more. However, in most cases *finite difference* schemes are used. In these

Fig. 10. Edge enhancement by diffusion along the edge and inverse diffusion across it. The original image is on the left

schemes continuous derivatives are approximated by discrete differences. The parameter domain is covered by a grid with step sizes k in time and h in space and the variables are discretized, for instance $u(t,x)$ is replaced by $u_m^n \triangleq u\,(t = nk, x = mh)$, see Fig. 11.

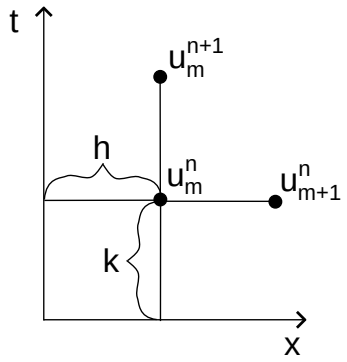

Fig. 11. The numerical grid for finite difference schemes

In most cases the design of satisfactory finite difference numerical schemes is quite straightforward. However, due to the size of the data, simplistic schemes might be inefficient and require a long run time. In the following subsections the main principles of the finite difference schemes are presented along with a few more elaborate schemes required to efficiently tackle the more challenging PDEs used for image processing.

4.1 Linear Diffusion

For 1-dimensional linear diffusion, the first derivative in time of the function $u(t,x)$ can be approximated by the first order accurate forward difference

$$D_t^+ u_m^n \triangleq \frac{u_m^{n+1} - u_m^n}{k},$$

and the second derivative in space can be approximated by the second order central difference

$$D_{xx}^0 u_m^n \triangleq \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2}.$$

The resulting numerical scheme for the linear diffusion is

$$\frac{u_m^{n+1} - u_m^n}{k} = \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2},$$

and if we define

$$r \triangleq \frac{k}{h^2},$$

we get

$$u_m^{n+1} = (1-2r)\, u_m^n + r\left(u_{m+1}^n + u_{m-1}^n\right).$$

All we need is to add the initial condition

$$u_m^0 = f_m,$$

and to define the boundary conditions.
This is an *explicit* numerical scheme, because the value of u at iteration $n+1$ is given explicitly by the value of u at previous times, see Fig. 12. The update step consists of merely additions and multiplications. The problem with explicit schemes is that their time step is limited by reasons of stability. For linear diffusion we require $r \leq 1/2$. Taking a bigger time step may result in an unstable process, whose outcome does not depend on the initial data but on the computation errors. In many equations the allowed time step is rather small and necessitates many iterations till the required output is reached. One solution is *implicit* numerical schemes, where the desired value u_m^{n+1} depends on the value of u at the same time $n+1$ and at other spatial locations, see Fig. 12. One example is the Crank-Nicolson second order accurate scheme in time and space

$$\frac{u_m^{n+1} - u_m^n}{k} = \frac{1}{2}\left[\frac{u_{m+1}^{n+1} - 2u_m^{n+1} + u_{m-1}^{n+1}}{h^2} + \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2}\right].$$

In this case we need to solve a tridiagonal system of equations in every update step. This can be done efficiently by the Thomas algorithm. A large time step would affect the accuracy of the solution, but it would not generate any instabilities.

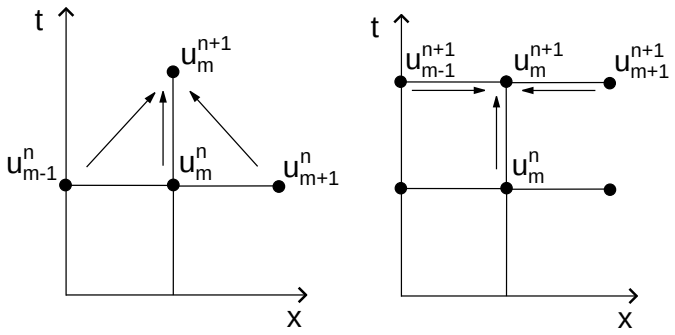

Fig. 12. The time and space dependencies of the explicit (left) and implicit (right) schemes for linear diffusion

For images where the equations have more than one dimension in space, explicit schemes are usually impractical due to the decrease of the bound on the time step. For linear diffusion we have $r \leq 1/(2D)$, with D the spatial dimension of the equation. On the other hand, implicit schemes result in a system of equations that is not tridiagonal and usually cannot be solved efficiently. More elaborate implicit schemes are required.
One such numerical scheme is the Alternating Direction Implicit (ADI) scheme. Peaceman and Rachford's [22] version is

$$\begin{aligned}\left(\mathbb{I} - \frac{k}{2}A_1\right)\tilde{u}^{n+\frac{1}{2}} &= \left(\mathbb{I} + \frac{k}{2}A_2\right)u^n \\ \left(\mathbb{I} - \frac{k}{2}A_2\right)u^{n+1} &= \left(\mathbb{I} + \frac{k}{2}A_1\right)\tilde{u}^{n+\frac{1}{2}},\end{aligned} \tag{4}$$

with $\mathbb{I}$ the identity matrix and the operators $A_1 u = u_{xx}$ and $A_2 u = u_{yy}$ replaced by their second order approximations. It can be seen from Eq. (4) that each iteration includes two steps where first the x direction is solved implicitly and the y direction explicitly, and then the opposite. Both steps consist of solving a tridiagonal system of equation, which can be done efficiently by the Thomas algorithm.

4.2 Nonlinear Diffusion

The original Perona-Malik filter [23] suffered from instabilities. The regularization presented by Catté, Lions, Morel and Coll [4] consists of replacing $f(|\nabla I|)$ with $f(|\nabla I_\sigma|)$ where I_σ is the convolution of I with a gaussian kernel with a standard deviation of σ. This smoothing of I eliminates some of the small scale noise and makes the filter well-posed.
Weickert, ter Haar Romeny and Viergever [45] introduced the first order accurate Additive Operator Splitting (AOS) scheme to numerically implement this filter. The update step is

$$I^{n+1} = \frac{1}{m} \sum_{i=1}^{m} \left(\mathbb{I} - mkA_i(I^n)\right)^{-1} I^n,$$

with m the dimension of the image and the elements of the matrix A_i are given by

$$(A_i)_{pq} = \begin{cases} \frac{f_p + f_q}{2h^2} & q \in N(p) \\ -\sum_{l \in N(p)} \frac{f_p + f_l}{2h^2} & p = q \\ 0 & \text{otherwise} \end{cases}$$

with $N(p)$ the neighbors of the grid point p in the i-th direction, and f_p the value of $f(|\nabla I_\sigma^n|)$ at grid point p.
The AOS scheme is semi-implicit and the size of the time step does not affect its stability. The scheme is efficient because it only requires the solution of tridiagonal systems of equations. It creates a discrete scale space [43] and its additivity gives equal importance to all coordinate axes as opposed to the multiplicative Locally One Dimensional (LOD) scheme which uses the update step

$$I^{n+1} = \prod_{i=1}^{m} \left(\mathbb{I} - kA_i(I^n)\right)^{-1} I^n.$$

The AOS may be used also for some anisotropic nonlinear filters applied to gray level images. For color images and filters like the Beltrami flow, where each color component depends on the value of the others, the splitting is impossible. To date, there is no PDE based implicit scheme for the color Beltrami. This is one of the main motivations for the construction of numerical kernels, described in the next section.

5 Kernels

It was shown in the previous section that the bound on the time step of some of the explicit numerical schemes can be alleviated by the use of implicit schemes. This enables a tradeoff between the efficiency of the scheme and its accuracy. Unfortunately, this is not the case in some of the important geometric filters, such as the Beltrami filter. Another approach, namely the use of kernels, is the answer in some of these cases. Moreover, the kernels add a new perspective to these filters and present connections to other existing image enhancing procedures.

5.1 The Gaussian Kernel for the Heat Equation

It can be shown that linear diffusion of an image can be accomplished by convolving it with a Gaussian kernel. Applying the heat equation to the 2-dimensional data $I(u^1, u^2, t_0)$ for the duration t is equivalent to the convolution

$$I(u^1, u^2, t_0 + t) = \iint I(\tilde{u}^1, \tilde{u}^2, t_0) K(|u^1 - \tilde{u}^1|, |u^2 - \tilde{u}^2|; t) d\tilde{u}^1 d\tilde{u}^2$$
$$= I(u^1, u^2, t_0) * K(u^1, u^2; t) \ , \tag{5}$$

where the kernel is given by

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

The use of the kernel enables to replace the iterative application of the numerical scheme for the PDE with a one step filter.

5.2 1-Dimensional Kernel for Nonlinear Diffusion

A kernel for the nonlinear diffusion of 1-dimensional signals was presented in [30]. The nonlinear kernel adapts itself to the local amplitude of the signal. Adaptive filtering has been done before, mainly by using robust estimation techniques. However, the nonlinear kernel relates to the signal as a curve and its adaptivity originates from the geometry of this curve.
The main idea behind the nonlinear kernel is presented in Fig 13. For the linear kernel the amplitude of the filtered signal at a specific point is the sum of the neighboring points' amplitudes weighted according to their distance along the coordinate axis. For the nonlinear kernel the weighting is according to the distance on the signal itself. The nonlinear kernel 'resides' on the signal while for the linear kernel the Gaussian 'resides' on the coordinate axis.

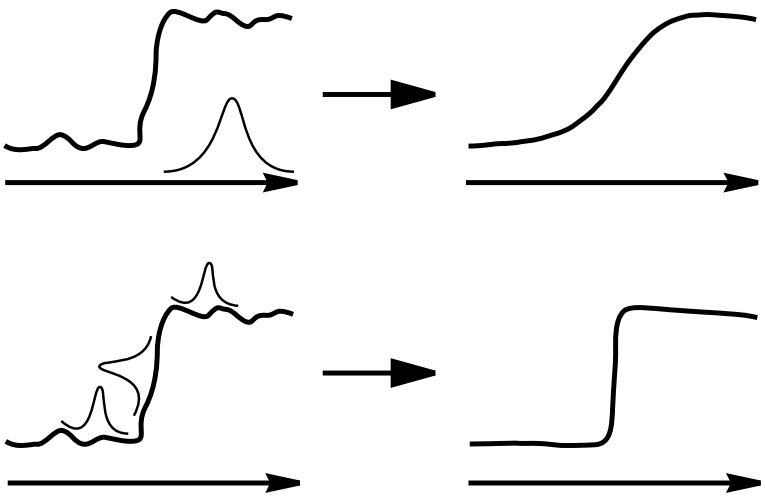

Fig. 13. Filtering a signal with a linear Gaussian kernel (top) and a nonlinear kernel (bottom)

The distance along the signal is calculated using the metric of the curve which is the signal. Various metrics are possible and they yield different filtering results. The Euclidean metric, for instance, using the curve representation $C(p) = (x(p), y(p)) = (x, y(x))$ is $g(x) = 1 + y_x^2$. The kernel is constructed for the 1-dimensional Beltrami flow

$$C_t = \Delta_g C.$$

The kernel cannot be global in time due to its non-linearity (the kernel depends on the signal's local amplitudes which change in each iteration of the kernel). Therefore the PDE cannot be replaced with a one step filter like in linear diffusion. Only a short time kernel which is applied iteratively is possible. After each iteration the signal is

$$C(p, t_0 + t) = \int C(\tilde{p}, t_0) K(p, \tilde{p}; t) d\tilde{p},$$

with the kernel

$$K(p, \tilde{p}; t) = \frac{H(p, \tilde{p}; t)}{\sqrt{t}} \exp\left(-\frac{\psi(p, \tilde{p})}{t}\right).$$

$H(p, \tilde{p}; t)$ can be taken to be a constant [30] and we get

$$\psi(p) = \frac{1}{4}\left(\int_p^{\tilde{p}} ds\right)^2,$$

where ds is an arc length element given by $ds = \sqrt{g(p)}dp$. Since $\int_p^{\tilde{p}} ds$ is the distance on the signal from point p to point $\tilde{p}$, the resulting kernel is indeed a Gaussian 'residing' on the signal, see Fig. 13.

5.3 The Short Time Kernel for the Beltrami Flow

A short time kernel for the 2-dimensional Beltrami flow was introduced in [35]. If used iteratively, it has an equivalent effect to that of the Beltrami flow. We replace Eq. (5) with

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

which we denote by

$$I^i(u^1, u^2, t_0 + t) = I^i(u^1, u^2, t_0) *_g K(u^1, u^2; t).$$

This is not a convolution in the strict sense, because K does not depend on the differences $u^i - \tilde{u}^i$. It will be shown later that $*_g$ is the geometric equivalent for manifolds of convolution. The general form of K is

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

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. In order to find K, we use the fact that it should satisfy Eq. (3) and after a few mathematical manipulations we get

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

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. The resulting short time kernel is

$$\begin{aligned}K(u^1,u^2,\tilde{u}^1,\tilde{u}^2;t)&=\frac{H_0}{t}\exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)}ds\right)^2}{4t}\right)\\&=\frac{H_0}{t}\exp\left(-\frac{d_g^2\left((u^1,u^2),(\tilde{u}^1,\tilde{u}^2)\right)}{4t}\right),\end{aligned}\tag{6}$$

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 a 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

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

The resulting update step for the image is

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

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 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 Eq. (6) is very similar to the Bilateral filter kernel [38, 6]. The difference between them is that the Beltrami kernel

uses geodesic distances on the image manifold, while the Bilateral kernel uses Euclidean distances. As can be seen from the derivation of the Beltrami kernel, the Bilateral filter originates from image manifold area minimization. The Bilateral filter can actually be viewed as an Euclidean approximation of the Beltrami flow.

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.

The short time kernel for the Beltrami flow requires the solution of the Eikonal equation on the image manifold. The image manifold is a parametric manifold, where the metric G is given for every point. The solution to the Eikonal equation on parametric manifolds [34] is based on the solution of the same problem on triangulated manifolds [17] which in turn is an extension of Sethian's fast marching method [29]. Another Eikonal solver on flat domains with regular grids was proposed by Tsitsiklis [40].

The original fast marching algorithm [29] solves the Eikonal equation in an orthogonal coordinate system. This is not the case for image manifolds. There $g_{12} \neq 0$ and we get a non-orthogonal coordinate system on the manifold. The solution for that is similar to that of [17], where a pre-processing stage is used to construct a suitable numerical stencil for each grid point. In this case there is no need to perform the unfolding step of [17] because the structure of the non-orthogonal grid on the manifold is given by its metric G. Figure 14 demonstrates the solution of the Eikonal equation for the parametric manifold $z = 0.5 \sin(4\pi x) \sin(4\pi y)$.

In order to demonstrate the spatial structure of the kernel, we tested it on the synthetic image in Fig. 15. At isotropic areas of the image, the kernel is isotropic and its weights are determined solely by the spatial distance from the 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 Conclusion

This chapter described image enhancement using PDE based geometric diffusion flows. On the theoretical side, starting with variational principles explains the origin of the flows, and the geometric approach results in some nice invariance properties. On the practical side, using carefully selected numerical schemes and developing kernels for the flows, enables an efficient and

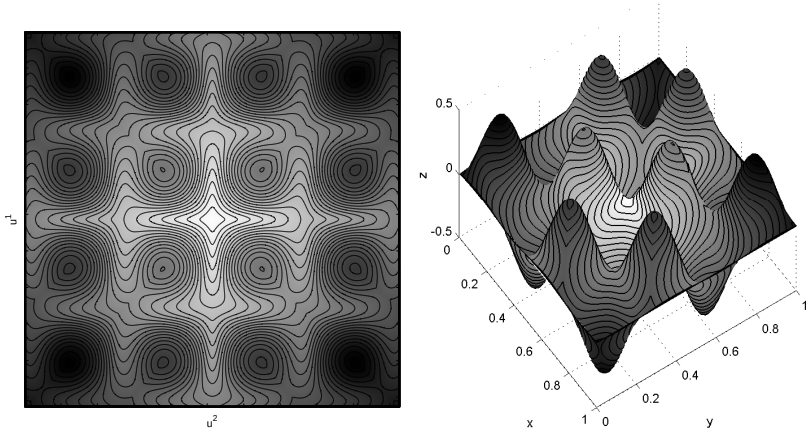

Fig. 14. Fast marching on the manifold $z = 0.5\sin(4\pi x)\sin(4\pi y)$. Left: implemented on the parameterization plane. Right: projected on the manifold. Lower values are assigned brighter colors. The black curves are the level curves

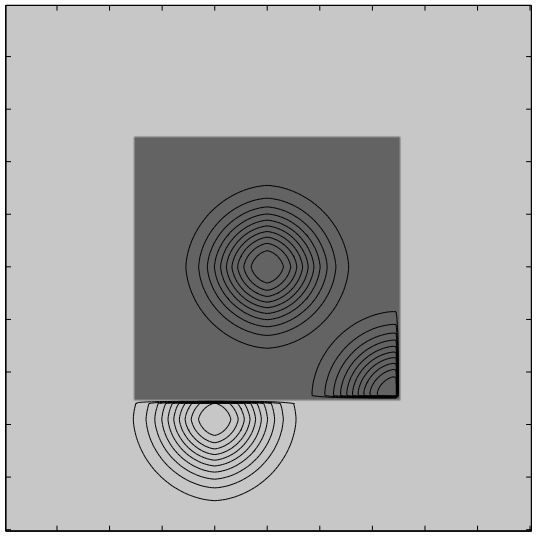

Fig. 15. Level curves of the kernel at various locations in a synthetic image

robust implementation. Combined together we get a fascinating area of research yielding state of the art algorithms.

References

1. L. Alvarez, F. Guichard, P. L. Lions, and J. M. Morel. Axioms and fundamental equations of image processing. *Arch. Rational Mechanics*, 123:199–257, 1993.
2. L. Alvarez and L. Mazora. Signal and image restoration using shock filters and anisotropic diffusion. *SIAM J. Numer. Anal*, 31:590–605, 1994.
3. C. Ballester, V. Caselles, and P. Monasse. The tree of shapes of an image. In *Preprint, C.M.L.A, No. 2001-02, Ecole Normale Supérieure de Cachan*, 2001.
4. F. Catté, P. Lions, J. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal of Numerical Analysis*, 29:182–193, 1992.
5. S. Di Zenzo. A note on the gradient of a multi image. *Computer Vision, Graphics, and Image Processing*, 33:116–125, 1986.
6. M. Elad. On the bilateral filter and ways to improve it. *IEEE Transactions on Image Processing*, 11(10):1141–1151, October 2002.
7. L. C. Evans and J. Spruck. Motion of level sets by mean curvature, I. *J. Diff. Geom.*, 33, 1991.
8. O. Faugeras and R. Keriven. Scale-space and affine curvature. In *Proceedings Europe-China Workshp on Geometrical modelling and Invariants for Computer Vision*, pages 17–24, 1995.
9. D. Gabor. Information theory in electron microscopy. *Laboratory Investigation*, 14(6):801–807, 1965.
10. M. A. Grayson. The heat equation shrinks embedded plane curves to round points. *J. Diff. Geom.*, 26, 1987.
11. H. Helmholtz. *Handbuch der Psychologischen Optik*. Voss, Hamburg, 1896.
12. R. Kimmel. Affine differential signatures for gray level images of planar shapes. In *Proc. of ICPR'96*, Vienna, Austria, August 1996.
13. R. Kimmel. Intrinsic scale space for images on surfaces: The geodesic curvature flow. *Graphics Modeling and Image Processing*, 59(5):365–372, September 1997.
14. R. Kimmel. *Numerical geometry of images*. Springer, Boston, NY, to appear.
15. 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.
16. R. Kimmel, R. Malladi, and N. Sochen. Images as embedding maps and minimal surfaces: Movies, color, texture, and volumetric medical images. *International Journal of Computer Vision*, 39(2):111–129, 2000.
17. R. Kimmel and J. Sethian. Computing geodesic paths on manifolds. *Proceedings of National Academy of Sciences*, 95(15):8431–8435, July 1998.
18. R. Kimmel and N. Sochen. Orientation diffusion or how to comb a porcupine? *special issue on PDEs in Image Processing, Computer Vision, and Computer Graphics, Journal of Visual Communication and Image Representation*, 13:238–248, 2002.
19. T. S. Lee. Image representation using 2D Gabor-wavelets. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

20. M. Lindenbaum, M. Fischer, and A. M. Bruckstein. On Gabor's contribution to image enhancement. *Pattern Recognition*, 27(1):1–8, 1994.
21. S. Osher and J. Sethian. Fronts propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.
22. D. Peaceman and H. Rachford. The numerical solution of parabolic and eliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3:28–41, 1955.
23. P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.
24. Y. Rubner and C. Tomasi. *Perceptual Metrics for Image Database Navigation.* Kluwer Academic Publishers, Boston, NY, December 2000.
25. L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.
26. G. Sapiro. *Geometric Partial Differential Equations and Image Processing.* Cambridge University Press, January 2001.
27. G. Sapiro and D. L. Ringach. Anisotropic diffusion of multivalued images with applications to color filtering. *IEEE Trans. Image Proc.*, 5:1582–1586, 1996.
28. E. Schrödinger. Grundlinien einer theorie der farbenmetrik in tagessehen. *Ann. Physik*, 63:481, 1920.
29. J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of National Academy of Sciences*, 93(4):1591–1595, 1996.
30. 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.
31. N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Trans. on Image Processing*, 7(3):310–318, 1998.
32. 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.
33. A. Spira and R. Kimmel. Geodesic curvature flow on parametric surfaces. In *Curve and Surface Design: Saint-Malo 2002*, pages 365–373, Saint-Malo, France, June 2002.
34. 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.
35. A. Spira, R. Kimmel, and N. Sochen. Efficient beltrami flow using a short time kernel. In *Proc. of Scale Spcace 2003, Lecture Notes in Computer Science (vol. 2695)*, pages 511–522, Isle of Skye, Scotland, UK, June 2003.
36. W. S. Stiles. A modified Helmholtz line element in brightness-colour space. *Proc. Phys. Soc. (London)*, 58:41, 1946.
37. B. ter Haar Romeny. *Geometry-Driven Diffusion in Computer Vision.* Kluwer Academic Publishers, Dordrecht, Netherlands, 1994.
38. C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision*, Bombay, India, January 1998.
39. D. Tschumperlé. *PDE's based regularization of multivalued images and applications.* Ph.D. thesis, December 2002.

40. J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*, 40(9):1528–1538, 1995.
41. J. J. Vos and P. L. Walraven. An analytical desription of the line element in the zone-fluctuation model of colour vision II. The derivative of the line element. *Vision Research*, 12:1345–1365, 1972.
42. J. Weickert. Theoretical foundation of anisotropic diffusion in image processing. *Computing, Suppl.*, 11:221–236, 1996.
43. J. Weickert. *Anisotropic Diffusion in Image Processing.* Teubner Verlag, Stuttgart, 1998. ISBN 3-519-02606-6.
44. J. Weickert, S. Ishikawa, and A. Imiya. Linear scale-space has first been proposed in japan. *Journal of Mathematical Imaging and Vision*, 10:237–252, 1999.
45. J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever. Efficient and reliable scheme for nonlinear diffusion filtering. *IEEE Trans. on Image Processing*, 7(3):398–410, 1998.