A Short Time Beltrami Kernel for Smoothing Images and Manifolds

Alon Spira, Ron Kimmel, *Senior Member, IEEE*, and Nir Sochen

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 solution of the heat equation by a convolution with a Gaussian kernel. The kernel is appropriate for smoothing regular (flat) 2D images, for smoothing images painted on manifolds, and for simultaneously smoothing images and the manifolds they are painted on.

The kernel combines the geometry of the image and that of the manifold into one metric tensor, thus enabling a natural unified approach for the manipulation of both. Additionally, the derivation of the kernel gives a better geometrical understanding of the Beltrami flow and shows that the bilateral filter is a Euclidean approximation of it.

On a practical level, the use of the kernel allows arbitrarily large time steps as opposed to the existing explicit numerical schemes for the Beltrami flow. In addition, the kernel works with equal ease on regular 2D images and on images painted on parametric or triangulated manifolds. We demonstrate the denoising properties of the kernel by applying it to various types of images and manifolds.

Index Terms

Beltrami, kernel, denoising, smoothing, manifold.

Contact information:

Alon Spira:

Computer Science department, Technion, Technion city, Haifa, 32000, Israel.

salon@cs.technion.ac.il

phone: 972-4-8294975

fax: 972-4-8293900

Ron Kimmel:

Computer Science department, Technion, Technion city, Haifa, 32000, Israel.

ron@cs.technion.ac.il

phone: 972-4-8294616

fax: 972-4-8293900

Nir Sochen:

Applied mathematics department, Tel-Aviv university, Tel-Aviv, 69978, Israel.

sochen@math.tau.ac.il

phone: 972-3-6409622

fax: 972-3-6409357

A Short Time Beltrami Kernel for Smoothing Images and Manifolds

I. Introduction

PDE based algorithms have proven their worth in the fields of image processing and computer vision. Non-linear advanced equations are capable of excellent smoothing of images while preserving their visually important features. The geometric and variational origin of many of these methods adds to the understanding of their qualities. Important related work include [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11] and references within.

The Beltrami framework [12], [13] enables state of the art image regularization based on geometric and variational sound grounds. It produces a spectrum of image enhancing algorithms ranging from the L_2 linear diffusion to the L_1 non-linear flows. See [14] for the connection between the Beltrami flow and previously proposed feature-preserving image smoothing algorithms. Apart from regular (flat) 2D images, the framework was used for textures, video, and volumetric data [15], as well as for orientation diffusion [16].

Sochen et. al. [17], [18], [19] extended the Beltrami flow for images painted on explicit and implicit manifolds. They have also shown the Beltrami flow to be a generalization of the harmonic maps approach for enhancing images painted on implicit manifolds [20], [21], [22]. In [23] they extend their results to images painted on triangulated manifolds. All these implementations of the Beltrami flow are done by explicit numerical schemes that require an upper bound on the time step used and might result in many iterations (semi-implicit schemes based on operator splitting, as done in [24], [25], [26], exist only for gray level images).

Many methods exist for the smoothing of meshes and graphic objects. For a relevant approach, where the bilateral filter is used for mesh denoising see [27]. See the references there for other approaches based on image regularization methods. There are also methods for the joint smoothing of manifolds and the images painted on them. For PDE based algorithms see [28] and [29], where the triangulated manifolds and the images undergo anisotropic diffusions. The numerical scheme in [28] consists of Loop's subdivision while [29] uses a finite element discretization in space. Both use semi-implicit finite difference discretizations in time. These methods for either manifold or for joint manifold and image smoothing do not enjoy the natural and simple geometrical model that the Beltrami kernel introduces. They usually do not have its smoothing quality, efficiency and simplicity of implementation.

A short time kernel has been suggested for 1D non-linear diffusion in [30] and an approximation for the 2D Beltrami operator in [31]. In [11] a short time oriented Gaussian kernel is used to implement vector-valued image regularization. Here we briefly review some of the ideas introduced in [32], [33] and construct a 2D short time

kernel for the Beltrami flow. This kernel enables the implementation of the Beltrami flow by 'convolving' the image with the kernel, similarly to the solution of the heat equation by a convolution with a Gaussian kernel. This implementation replaces the conventional method of solving the first variation of the Beltrami functional as a gradient descent PDE process by the appropriate numerical schemes.

On a theoretical level, the derivation of the kernel gives a better geometrical understanding of the Beltrami flow and shows that the bilateral filter [34], [35], [36] is actually a Euclidean approximation of it. For images painted on manifolds, the kernel combines the geometry of the image and that of the manifold into one metric tensor, thus enabling a natural unified approach for the manipulation of both. The main practical advantages of the kernel include the ability to select an arbitrary time step for the Beltrami flow and a straightforward applicability of the kernel to all common types of images (gray scale, color, painted on manifolds, etc.) and manifolds (parametric and triangulated).

In order to compute the short time kernel we need to calculate geodesic distances between pixels in the image. For regular 2D images and images painted on parametric manifolds we use fast marching [37], [38], [39] on parametric manifolds [40], [41]. For images painted on triangulated manifolds we use fast marching on triangulated manifolds [42].

This paper is organized as follows. Section 2 describes the Beltrami flow for regular 2D images and for images painted on manifolds. The derivation of the short time kernel is presented in Section 3. Section 4 reviews shortly the method 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 5 and the conclusion appears in Section 6.

II. The Beltrami Flow for Images Painted on Manifolds

In the Beltrami framework [12], [13] 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 (see [43], [44] to learn about differential geometry of manifolds). The image is represented by $\{X^1, X^2, \ldots, X^N\} = \{x^1, x^2, \ldots, x^L, I^1, I^2, \ldots, I^M\}$, with x^i the spatial coordinates, I^j the intensity components and $N = L + M$. Gray level images, for instance, have $L = 2$, $M = 1$ and $\{X^1, X^2, X^3\} = \{u^1, u^2, I(u^1, u^2)\}$, see Figure 1. The following derivation will assume the more general case of color images painted on parametric manifolds embedded in $\mathbb{R}^6$, where we have $L = 3, M = 3$ and the image is $\{x(u^1, u^2), y(u^1, u^2), z(u^1, u^2), I^1(u^1, u^2), I^2(u^1, u^2), I^3(u^1, u^2)\}$. For other values of L and M the derivation is similar.

If we choose the embedding space to be Euclidean, its metric h_{ij} is represented by the diagonal matrix H, with ones in the first L rows and β^2 in the next M. β is the relative scale between the spatial coordinates and the intensity components. See [15] on the meaning of its value. Non-Euclidean embedding spaces were addressed in [14], [45]. The metric elements g_{ij} of the image are derived from the metric elements h_{ij} and the embedding by

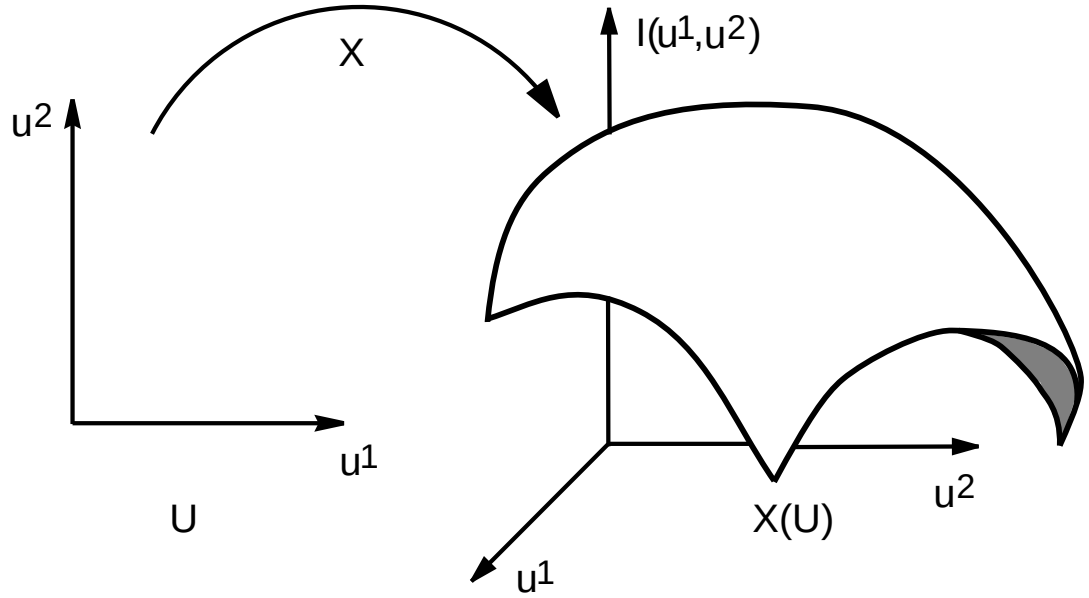

Fig. 1. A gray level image according to the Beltrami framework

the pullback procedure

$$\begin{aligned} \mathsf{G} &= (g_{ij}) = \\ &= \begin{pmatrix} \sum_i (x_1^i)^2 + \beta^2 \sum_j (I_1^j)^2 & \sum_i x_1^i x_2^i + \beta^2 \sum_j I_1^j I_2^j \\ \sum_i x_1^i x_2^i + \beta^2 \sum_j I_1^j I_2^j & \sum_i (x_2^i)^2 + \beta^2 \sum_j (I_2^j)^2 \end{pmatrix}, \end{aligned} \tag{1}$$

with x_j^i the derivative of x^i with respect to u^j.

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

$$S = \iint \sqrt{g} du_1 du_2, \tag{2}$$

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

$$\begin{aligned} X_t^a &= -g^{-\frac{1}{2}} h^{ab} \frac{\delta S}{\delta X^b} = \\ &= g^{-\frac{1}{2}} \partial_i (g^{\frac{1}{2}} g^{ij} \partial_j X^a) + \Gamma_{bc}^a \partial_i X^b \partial_j X^c g^{ij} \ , \end{aligned} \tag{3}$$

with g^{ij} and h^{ab} the components of the contravariant metrics of the image manifold G^{-1} (the inverse of the metric tensor G) and the embedding space H^{-1} respectively. Einstein's summation convention is used. The Christoffel symbols (also known as the Levi-Civita coefficients) Γ_{bc}^a are defined in terms of the metric H

$$\Gamma_{bc}^a = \frac{1}{2} h^{ad} \left(\partial_b h_{dc} + \partial_c h_{bd} - \partial_d h_{bc} \right). \tag{4}$$

Since we have chosen the embedding space H to be Euclidean, we have $\Gamma_{bc}^a = 0$. See [14] for a detailed derivation.

In matrix form Equation (3) reads

$$X_t^a = \underbrace{\frac{1}{\sqrt{g}}\text{div}\left(\sqrt{g}\mathsf{G}^{-1}\nabla X^a\right)}_{\Delta_g X^a}. \tag{5}$$

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

$$\begin{aligned} I_t &= \Delta_g I = H\langle \hat{I}, \vec{N}\rangle = \\ &= \frac{\left(1+\beta^2 I_x^2\right) I_{yy} - 2\beta^2 I_x I_y I_{xy} + \left(1+\beta^2 I_y^2\right) I_{xx}}{1+\beta^2 I_x^2 + \beta^2 I_y^2}, \end{aligned} \tag{6}$$

i.e., the image surface moves according to the intensity component of the mean curvature flow . For a color image painted on a manifold the diffusion equation for each component reads

$$X_t^i = \Delta_g X^i. \tag{7}$$

This is a joint smoothing of the manifold and the image painted on it. For smoothing only the manifold, Equation (7) should be applied only to the first L components of $\{X^1, X^2, \ldots, X^N\}$. Applying it to the last M components will result in only image smoothing.

III. A Short Time Kernel for the Beltrami Flow

It can be shown that applying the heat equation

$$I_t = \Delta I \tag{8}$$

to the 2D regular image $I(u^1, u^2, t_0)$ for the duration t is equivalent to convolving the image with a Gaussian (linear) kernel

$$\begin{aligned} &I\left(u^1, u^2, t_0 + t\right) = \\ &= \iint I(\tilde{u}^1, \tilde{u}^2, t_0) K(|u^1 - \tilde{u}^1|, |u^2 - \tilde{u}^2|; t) d\tilde{u}^1 d\tilde{u}^2 \;\; = \\ &= I(u^1, u^2, t_0) * K(u^1, u^2; t)\ , \end{aligned} \tag{9}$$

where the kernel is given by

$$K(u^1, u^2; t) = \frac{1}{4\pi t}\exp\left(-\frac{(u^1)^2 + (u^2)^2}{4t}\right)\ . \tag{10}$$

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 image 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.

The main idea behind the kernel is presented in Figure 2 for a 1D signal. For the Gaussian kernel the amplitude of the filtered signal at a specific point is the sum of the neighboring points' amplitudes weighted according to their

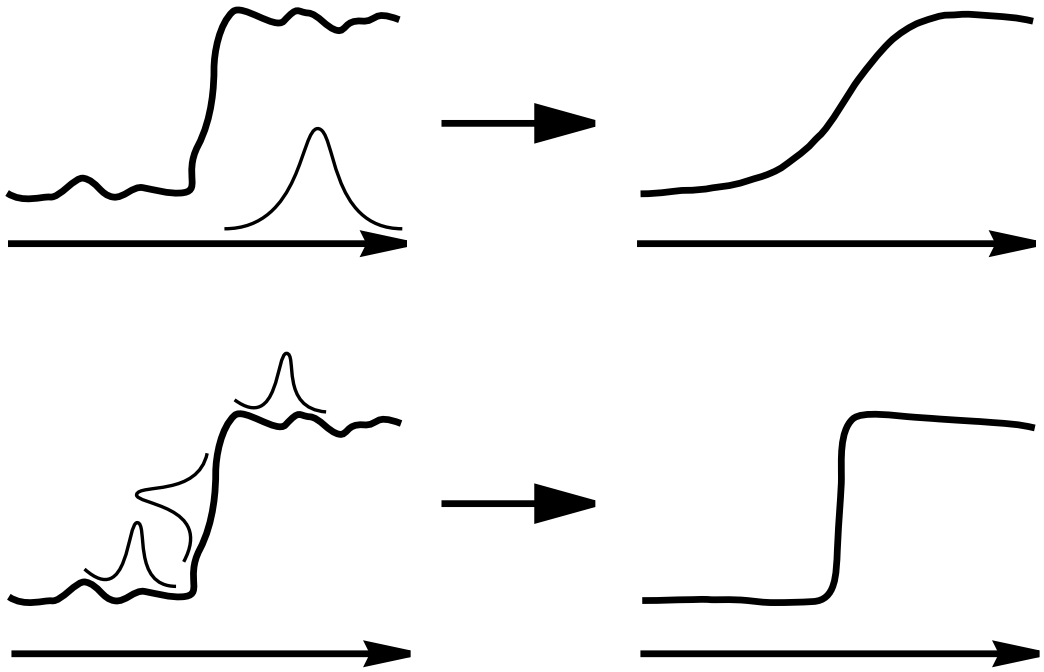

Fig. 2. Filtering a signal with a linear Gaussian kernel (top) and a nonlinear Beltrami kernel (bottom)

distance along the coordinate axis. For the nonlinear Beltrami kernel the weighting is according to the distance on the signal itself. The Beltrami kernel 'resides' on the signal, while for the linear kernel the Gaussian 'resides' on the coordinate axis.

In order to develop the Beltrami kernel for images painted on manifolds we replace Equation (9) with

$$\begin{aligned} &X^i(u^1,u^2,t_0+t) = \\ &= \iint X^i(\tilde{u}^1,\tilde{u}^2,t_0)K(u^1,u^2,\tilde{u}^1,\tilde{u}^2;t)d\tilde{u}^1 d\tilde{u}^2 \ , \end{aligned} \tag{11}$$

which we denote by

$$X^i(u^1,u^2,t_0+t) = X^i(u^1,u^2,t_0) *_g K(u^1,u^2;t). \tag{12}$$

This is not a convolution in the strict sense, because K does not depend just 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

$$K(u^1,u^2;t) = \frac{H(u^1,u^2;t)}{t}\exp\left(-\frac{\psi^2(u^1,u^2)}{t}\right), \tag{13}$$

where we take, without loss 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 (7). Therefore,

$$K_t = \Delta_g K. \tag{14}$$

The left hand side of the equation is

$$\begin{aligned} K_t &= \partial_t \left(\frac{H}{t} \exp\left(-\frac{\psi^2}{t}\right)\right) = \\ &= \left(\frac{H_1}{t} - \frac{H_0}{t^2} - \frac{H_1}{t} + \frac{H_0\psi^2}{t^3} + \frac{H_1\psi^2}{t^2} + O(\frac{1}{t}) \right) \cdot \\ &\cdot \exp\left(-\frac{\psi^2}{t}\right) = \\ &= \left(\frac{H_0\psi^2}{t^3} + \frac{H_1\psi^2 - H_0}{t^2} + O(\frac{1}{t}) \right) \exp\left(-\frac{\psi^2}{t}\right) = \\ &= \frac{\psi^2}{t^2} K + O(\frac{1}{t^2}). \end{aligned} \tag{15}$$

For the right hand side of Equation (14) we calculate

$$\begin{aligned} K_i &= \frac{\partial K}{\partial u^i} = \\ &= \left(\frac{H_{0i}}{t} - \frac{2H_0\psi\psi_i}{t^2} - \frac{2H_1\psi\psi_i}{t} + O(1) \right) \exp\left(-\frac{\psi^2}{t}\right) \ . \end{aligned} \tag{16}$$

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

$$\frac{4H_0\psi^2\psi_i\psi_j}{t^3} - \frac{2(H_{0i} - 2H_1\psi\psi_i)\psi\psi_j + \partial_j(2H_0\psi\psi_i)}{t^2} \ . \tag{17}$$

Putting everything together, we get for the leading order

$$\Delta_g K = \frac{4\psi^2}{t^2} g^{ij}\psi_i\psi_j K + O(\frac{1}{t^2}). \tag{18}$$

Equating the leading terms in Equations (15) and (18), yields

$$g^{ij}\psi_i\psi_j = \|\nabla_g \psi\|^2 = \frac{1}{4}, \tag{19}$$

with ∇_g the gradient according to the metric G. This is the eikonal equation on the 2D image manifold, and its viscosity solution is a geodesic distance map ψ on this manifold. The method for efficiently solving the eikonal equation for image manifolds is reviewed 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 [30] for an example of such a computation).

The resulting short time kernel is thereby

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

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

where ds is an arc-length element on the image manifold, and $d_g(p_1,p_2)$ is the geodesic distance between two points, p_1 and p_2, on the image 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

$$\begin{aligned} &X^i(u^1,u^2) *_g K(u^1,u^2;t) = \\ &= \iint X^i(\tilde{u}^1,\tilde{u}^2) K\left(d_g\left((u^1,u^2),(\tilde{u}^1,\tilde{u}^2)\right);t\right) d\tilde{u}^1 d\tilde{u}^2. \end{aligned} \tag{21}$$

The update step for jointly smoothing the manifold and the image painted on it is

$$\begin{aligned} &X^i(u^1,u^2,t_0+t) = \\ &= \frac{H_0}{t} \iint_{(\tilde{u}^1,\tilde{u}^2)\in N(u^1,u^2)} X^i(\tilde{u}^1,\tilde{u}^2,t_0) \exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)} ds\right)^2}{4t} \right) d\tilde{u}^1 d\tilde{u}^2. \end{aligned} \tag{22}$$

As is the case with Equation (7), the smoothing process can be restricted to either the manifold or the image simply by applying Equation (22) to the appropriate components of $\{X^1, X^2, \ldots, X^N\}$. $N(u^1,u^2)$ in the last equation is 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 (20) is very similar to the bilateral filter kernel [34], [35], [36], which for the Gaussian case and Euclidean metric is of the form

$$\begin{aligned} &I^i(u^1,u^2,t_0+t) = \\ &= C \iint_{(\tilde{u}^1,\tilde{u}^2)} I^i(\tilde{u}^1,\tilde{u}^2,t_0) \exp\left(-\frac{1}{2}\left(\frac{|(u^1,u^2)-(\tilde{u}^1,\tilde{u}^2)|}{\sigma_d} \right)^2 \right) \cdot \\ &\cdot \exp\left(-\frac{1}{2}\left(\frac{|I^i(u^1,u^2,t_0) - I^i(\tilde{u}^1,\tilde{u}^2,t_0)|}{\sigma_I} \right)^2 \right) d\tilde{u}^1 d\tilde{u}^2, \end{aligned} \tag{23}$$

with C, σ_d and σ_I constants.

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 [46].

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 pixels and thus ignores connectivity. A pixel can have a relatively high kernel value, although it belongs to a different object than that of the filtered pixel. The Beltrami kernel takes this effect into account and penalizes a pixel 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. Furthermore, the value of the bilateral kernel for pixels close to the smoothed pixel is not necessarily larger than that of further away ones. This means that its numerical support is not as effectively bounded as that of the Beltrami kernel.

IV. Solving the Eikonal Equation on Images

As shown in the previous section, the construction of the kernel for a pixel requires the calculation of the geodesic distances between the pixel and its neighbors. We place the origin of the coordinate system of the image ($u^1 = u^2 = 0$) at the pixel. Next, in order to solve the eikonal equation on the image we use the fast marching method.

Regular 2D images are parametric manifolds, where the metric G is given for every pixel. Therefore, calculating the geodesic distances needed for implementing the kernel to these images is done by an extension of the fast marching method [37], [38], [39] to parametric manifolds [40], [41]. The same method is used for images painted on parametric manifolds. For images painted on triangulated manifolds we use fast marching on triangulated manifolds [42]. In [40], [41] the calculations are done on the 2D parameterization plane. In [42] the calculations are restricted to the 2D image manifold (that can be embedded in $\mathbb{R}^N$ of any N or any other manifold). Since for color images painted on manifolds and embedded in $\mathbb{R}^3$ we already have $N = 6$, calculating the distances explicitly on the 2D image is advantageous. This is contrary to methods such as [47] where the calculations are done in $\mathbb{R}^N$ inside a band around the manifold.

The original fast marching method 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. For images, where $g_{12} \neq 0$, we get a non-orthogonal coordinate system on the image, see Figure 3. The numerical support should include non-neighboring grid points (pixels). For parametric manifolds the method uses the metric of the image at each pixel in order to find the pixels used for the numerical scheme. In the case of triangulated

manifolds, the triangulation is given in advance and it determines the numerical support for each vertex.

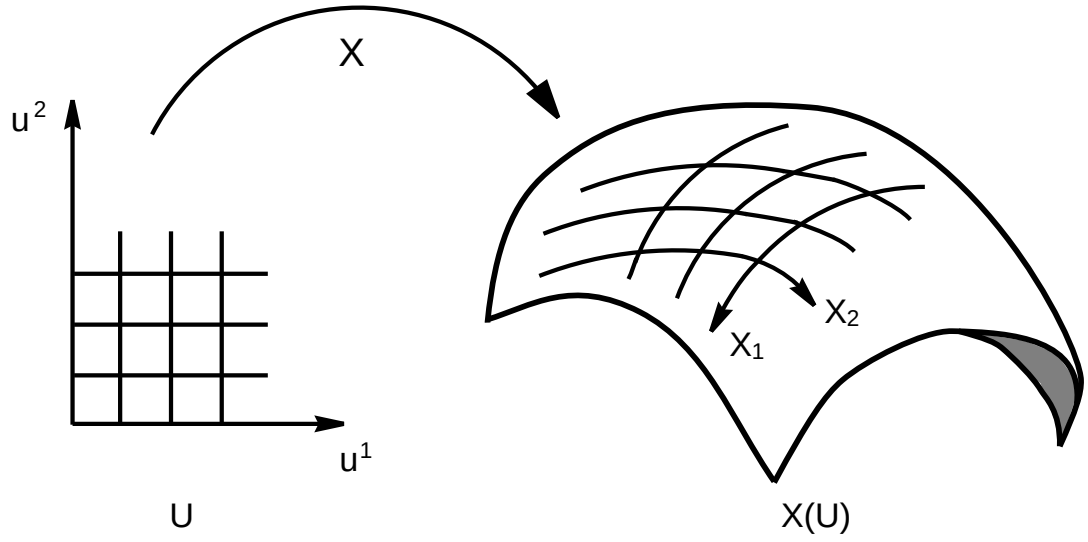

Fig. 3. The orthogonal grid on the parameterization plane is transformed into a non-orthogonal one on the image manifold.

The updated pixel together with the two other pixels in its numerical support constitute the vertices of a triangle. This triangle is the numerical stencil for updating the pixel. If the triangle is obtuse, it should be split and replaced by two acute triangles. For parametric manifolds the splitting is done according to the metric at the updated pixel, see [40], [41]. For triangulated manifolds an "unfolding" scheme is used, see [42].

After this pre-processing stage, all the triangles in the numerical grid are acute, as in Figure 4. The figure shows the method by which the vertex (pixel) C is updated according to the vertices A and B. The objective is to find t such that $\frac{t-u}{h} = 1$ and use it to calculate $\psi(C)$ based on $\psi(A)$ and $\psi(B)$.

The numerical scheme according to [42] is

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

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

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

The numerical scheme described in the previous paragraph enables the update of a pixel according to two of its neighbors. In order to use this scheme to generate the entire distance map the following algorithm [38] is used.

Initialization:

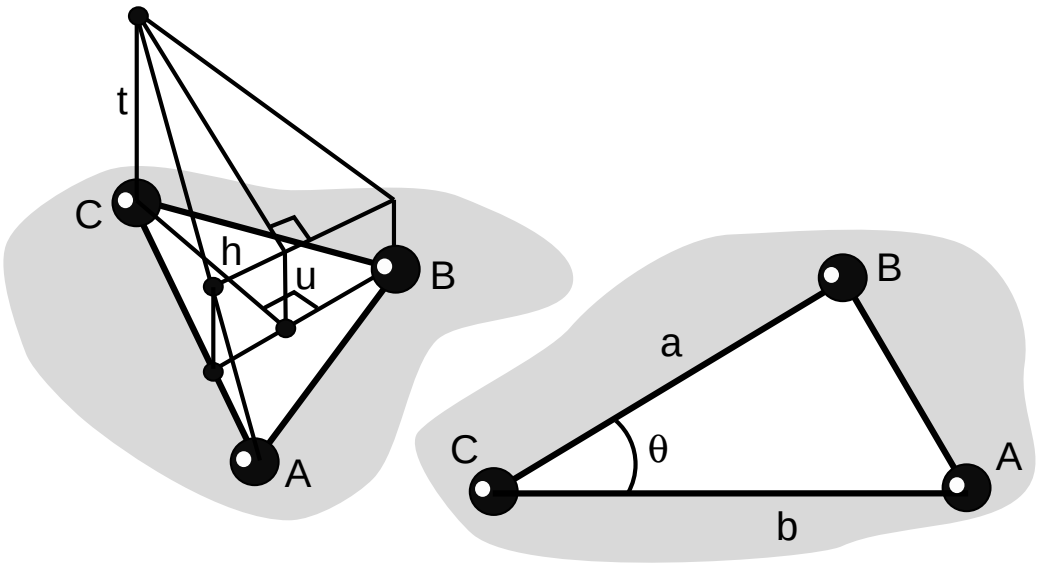

Fig. 4. The numerical stencil used to update $\psi(C)$ according to $\psi(A)$ and $\psi(B)$.

1) The pixel at the origin (for which the kernel is constructed) is defined as *Accepted* and given an initial value of zero.
2) All the other pixels are defined as *Far* and given the value infinity.

Iterations:

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

We use the term 'neighbors' to describe pixels that belong to the same numerical stencil. These pixels are not necessarily neighboring pixels in the image. We find these 'neighbors' during the pre-processing stage described previously.

The complexity of the algorithm for triangulated manifolds is upper bounded by

$$B_{triangulated} = O(n \log n), \tag{24}$$

where n is the number of pixels in the image. The $\log n$ results from using a min-heap data structure for sorting the *Close* pixels [38]. The complexity of the algorithm for parametric manifolds is upper bounded by

$$B_{parametric} = O\left(n \cdot \max\left(\log n, \log\left(\left\lceil \frac{|g_{12}|g_{11}}{g} \right\rceil\right)\right)\right), \tag{25}$$

where the components of G are calculated at each pixel. The term $\log\left(\left\lceil \frac{|g_{12}|g_{11}}{g} \right\rceil\right)$ originates from the splitting procedure, see [40], [41]. Since there is no need to use all the pixels in the image in order to update one pixel (the

value of the kernel for most of these pixels is negligible), we can bound in advance the neighborhood in which the eikonal equation is solved. Thus, we decrease substantially the practical complexity of the algorithm.

The accuracy of the numerical scheme is $O(h)$, where h is the length of the longest triangle edge in the numerical grid. It is determined for triangulated manifolds according to the given triangulation and for parametric manifolds according to the local metric at the image pixels.

V. Simulations and Results

Figure 5 shows the denoising of a regular 2D color image corrupted by zero mean Gaussian noise using the short time Beltrami kernel. In this case $\beta = 3$, the time step taken was $t = 0.1$, and only grid points with a kernel value above 0.01 were used for the filtering. The dynamic range of the color components was between 0 and 1.

The use of pixels with a weight larger than 0.01 resulted in an average of 21 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 filtered pixel. Across edges the significant change in intensity is translated into a long geodesic distance, which results in negligible 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.

Figure 7 shows the implementation of the Beltrami kernel to images painted on manifolds. The figure demonstrates the difference between smoothing a noisy color image as a regular 2D image and smoothing it as an image painted on its original manifold. The noise added is a zero mean Gaussian noise and for the kernel $\beta = 1$, the time step taken was $t = 0.1$, and only grid points with a kernel value above 0.01 were used for the filtering. An average of 5 pixels were used in the kernel as a result of these values. Both methods remove the noise, but smoothing the image on the manifold better preserves the edges of the image since these edges coincide with those of the manifold. This can be seen also in Figure 8 which contains a plot of the signal to noise ratios (SNR) of the images smoothed according to the two approaches. A better SNR is achieved by smoothing the image on the manifold, and this SNR deteriorates much slower if the image is over smoothed.

Figure 9 shows the smoothing of a triangulated face manifold with Gaussian noise added to it. After one iteration of the Beltrami kernel (t=2.0) the added noise is removed. After the second iteration the scanning errors in the original manifold are also smoothed away. Since every vertex in the triangulation is moved independently of its

neighbors, the uniformity of the triangulation might be degraded. This can be corrected by re-triangulation of the manifold [48], again using the fast marching method on manifolds.

Figure 10 demonstrates the joint smoothing of a manifold and the image painted on it. To the scanned face manifold and face image were added Gaussian noises which are independent from one another. It is evident from the figure that most of both noises is removed. The amplitude of the remaining noise signals is too large and the boundaries they create prevent the flow from smoothing them. This is more noticeable in the face manifold than in the face image.

VI. Conclusion

We have presented a short time kernel for the Beltrami flow for regular 2D images, images painted on manifolds, and the smoothing of the manifolds themselves. 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. It was also shown that for images painted on manifolds, incorporating the metric of the manifold in the flow produces better results.

From a practical stand point, the numerical implementation of the kernel handles every possible manifold represented by all common manifold representation. It enjoys low computational complexity further enhanced by an arbitrary time step for a Beltrami-like adaptive smoothing, which is impossible for the explicit numerical schemes currently existing for color images. All these attributes make the Beltrami kernel a highly practical tool for real life image processing and graphics applications.

Acknowledgment

We thank T. Treibitz and S. Rozenfeld for their assistance in the implementation of triangulated manifold smoothing. We also thank the anonymous reviewers for their helpful remarks.

References

[1] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” *IEEE-PAMI*, vol. 12, pp. 629–639, 1990.

[2] L. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D*, vol. 60, pp. 259–268, 1992.

[3] L. Alvarez, F. Guichard, P. L. Lions, and J. M. Morel, “Axioms and fundamental equations of image processing,” *Arch. Rational Mechanics*, vol. 123, pp. 199–257, 1993.

[4] M. Lindenbaum, M. Fischer, and A. M. Bruckstein, “On Gabor’s contribution to image enhancement,” *Pattern Recognition*, vol. 27, no. 1, pp. 1–8, 1994.

[5] L. Alvarez and L. Mazorra, “Signal and image restoration using shock filters and anisotropic diffusion,” *SIAM J. Numer. Anal*, vol. 31, pp. 590–605, 1994.

[6] G. Sapiro and D. L. Ringach, “Anisotropic diffusion of multivalued images with applications to color filtering,” *IEEE Trans. Image Proc.*, vol. 5, pp. 1582–1586, 1996.

[7] J. Weickert, *Anisotropic Diffusion in Image Processing*. Teubner Stuttgart, 1998, iSBN 3-519-02606-6.

[8] M. Desbrun, M. Meyer, P. Schroder, and A. H. Barr, "Anisotropic feature-preserving denoising of height fields and bivariate data," *Graphics Interface*, vol. 11, no. 10, pp. 145–152, 2000.

[9] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*. Springer, 2002.

[10] R. Kimmel, *Numerical Geometry of Images: Theory, Algorithms, and Applications*. Springer, November 2003.

[11] D. Tschumperle and R. Deriche, "Vector-valued image regularization with pde's : A common framework for different applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 4, pp. 506–517, 2005.

[12] 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.

[13] N. Sochen, R. Kimmel, and R. Malladi, "A general framework for low level vision," *IEEE Trans. on Image Processing*, vol. 7, no. 3, pp. 310–318, 1998.

[14] ——, "From high energy physics to low level vision," UC Berkeley, LBNL report LBNL-39243, August 1996.

[15] 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*, vol. 39, no. 2, pp. 111–129, 2000.

[16] 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*, vol. 13, pp. 238–248, 2002.

[17] N. Sochen, R. Deriche, and L. Lopez-Perez, "The Beltrami flow over manifolds," INRIA Sophia-Antipolis, Sophia Antipolis, France, Technical Report TR-4897, 2003.

[18] ——, "Variational Beltrami flows over manifolds," in *International Conference on Image Processing*, 2003.

[19] ——, "The Beltrami flow over implicit manifolds," in *Proc. of 9th International Conference on Computer Vision*, Nice, October 2003.

[20] M. Bertalmio, L. T. Cheng, S. Osher, and G. Sapiro, "Variational problems and partial differential equations on implicit surfaces," *Journal of Computational Physics*, vol. 174, pp. 759–780, October 2001.

[21] M. Bertalmio, F. Mémoli, L. T. Cheng, G. Sapiro, and S. Osher, "Variational problems and partial differential equations on implicit surfaces: Bye bye triangulated surfaces?" *Geometric Level Set Methods in Imaging, Vision, and Graphics, S. Osher, N. Paragios, eds., Springer, New York*, pp. 381–398, 2003.

[22] F. Mémoli, G. Sapiro, and S. Osher, "Solving variational problems and partial differential equations mapping into general target manifolds," *Journal of Computational Physics*, vol. 195, no. 1, pp. 263–292, 2004.

[23] L. Lopez-Perez, R. Deriche, and N. Sochen, "The Beltrami flow over triangulated manifolds," in *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis: ECCV 2004 Workshops CVAMIA and MMBIA, Lecture Notes in Computer Science (vol. 3117)*, Prague, Czech Republic, May 2004, pp. 135–144.

[24] T. Lu, P. Neittaanmaki, and X. C. Tai, "A parallel splitting up method and its application to Navier-Stokes equations," *Applied Mathematics Letters*, vol. 4, no. 2, pp. 25–29, 1991.

[25] ——, "A parallel splitting up method for partial differential equations and its application to Navier-Stokes equations," *RAIRO Math. Model. and Numer. Anal.*, vol. 26, no. 6, pp. 673–708, 1992.

[26] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Transactions on Image Processing*, vol. 7, pp. 398–410, 1998.

[27] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 950–953, 2003.

[28] C. L. Bajaj and G. Xu, "Anisotropic diffusion of surfaces and functions on surfaces," *ACM Transactions on Graphics*, vol. 22, pp. 4–32, January 2003.

[29] U. Clarenz, U. Diewald, and M. Rumpf, "Processing textured surfaces via anisotropic geometric diffusion," *IEEE Transactions on Image Processing*, vol. 13, no. 2, pp. 248–261, 2004.

[30] N. Sochen, R. Kimmel, and A. M. Bruckstein, "Diffusions and confusions in signal and image processing," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 3, pp. 195–209, 2001.

[31] N. Sochen, "Stochastic processes in vision: From Langevin to Beltrami," in *Proc. of International Conference on Computer Vision*, Vancouver, Canada, July 2001.

[32] A. Spira, R. Kimmel, and N. Sochen, "Efficient Beltrami flow using a short time kernel," in *Proc. of Scale Space 2003, Lecture Notes in Computer Science (vol. 2695)*, Isle of Skye, Scotland, UK, June 2003, pp. 511–522.

[33] A. Spira and R. Kimmel, "Enhancing images painted on manifolds," in *Proc. of Scale Spcace 2005, Lecture Notes in Computer Science (vol. 3459)*, Hofgeismar, Germany, April 2005, pp. 492–502.

[34] S. M. Smith and J. M. Brady, "SUSAN - a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[35] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision*, Bombay, India, January 1998.

[36] M. Elad, "On the bilateral filter and ways to improve it," *IEEE Transactions on Image Processing*, vol. 11, no. 10, pp. 1141–1151, October 2002.

[37] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[38] ——, *Level Set Methods and Fast Marching Methods*. Cambridge university press, 1996.

[39] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Trans. on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, 1995.

[40] 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.

[41] ——, "An efficient solution to the eikonal equation on parametric manifolds," *Interfaces and Free Boundaries*, vol. 6, no. 3, pp. 315–327, September 2004.

[42] R. Kimmel and J. A. Sethian, "Computing geodesic paths on manifolds," *Proceedings of National Academy of Sciences*, vol. 95, no. 15, pp. 8431–8435, July 1998.

[43] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces*. New Jersey: Prentice-Hall Inc., 1976.

[44] E. Kreyszig, *Differential Geometry*. New York: Dover Publications, Inc., 1991.

[45] N. Sochen and Y. Y. Zeevi, "Representation of colored images by manifolds embedded in higher dimensional non-Euclidean space," in *Proc. of ICIP98*, Chicago, IL, January 1998, pp. 166–170.

[46] D. Barash, "Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 844–847, June 2002.

[47] F. Mémoli and G. Sapiro, "Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces," *Journal of Computational Physics*, vol. 173, no. 2, pp. 730–764, 2001.

[48] A. Liav, "Fast computation of geodesic distances: Graphic applications," M.Sc. Thesis, Technion - Israel Institute of Technology, 2001.

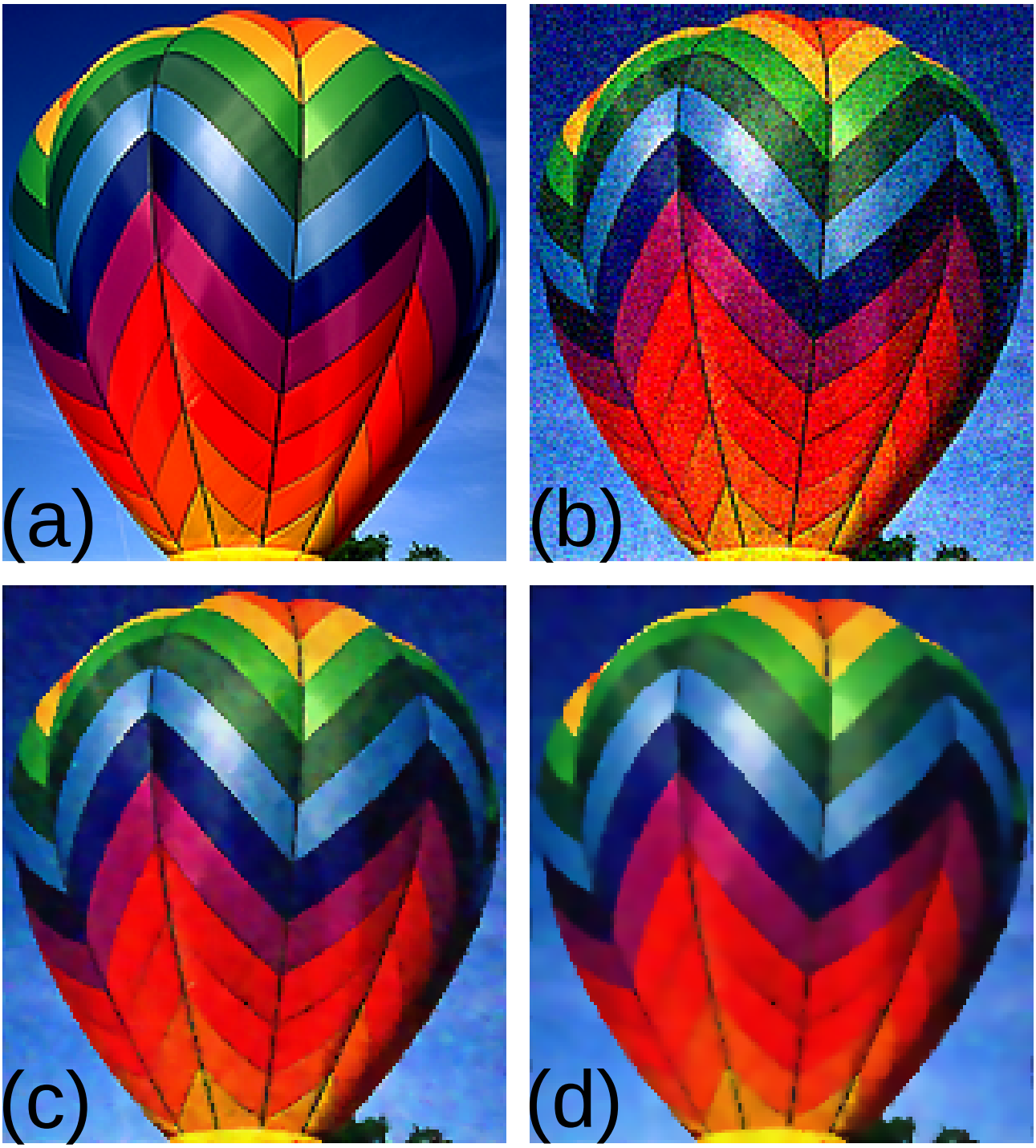

Fig. 5. Denoising a color image by the short time Beltrami kernel. a) The original image, b) the noisy image, c) after one iteration of the kernel, d) after two iterations.

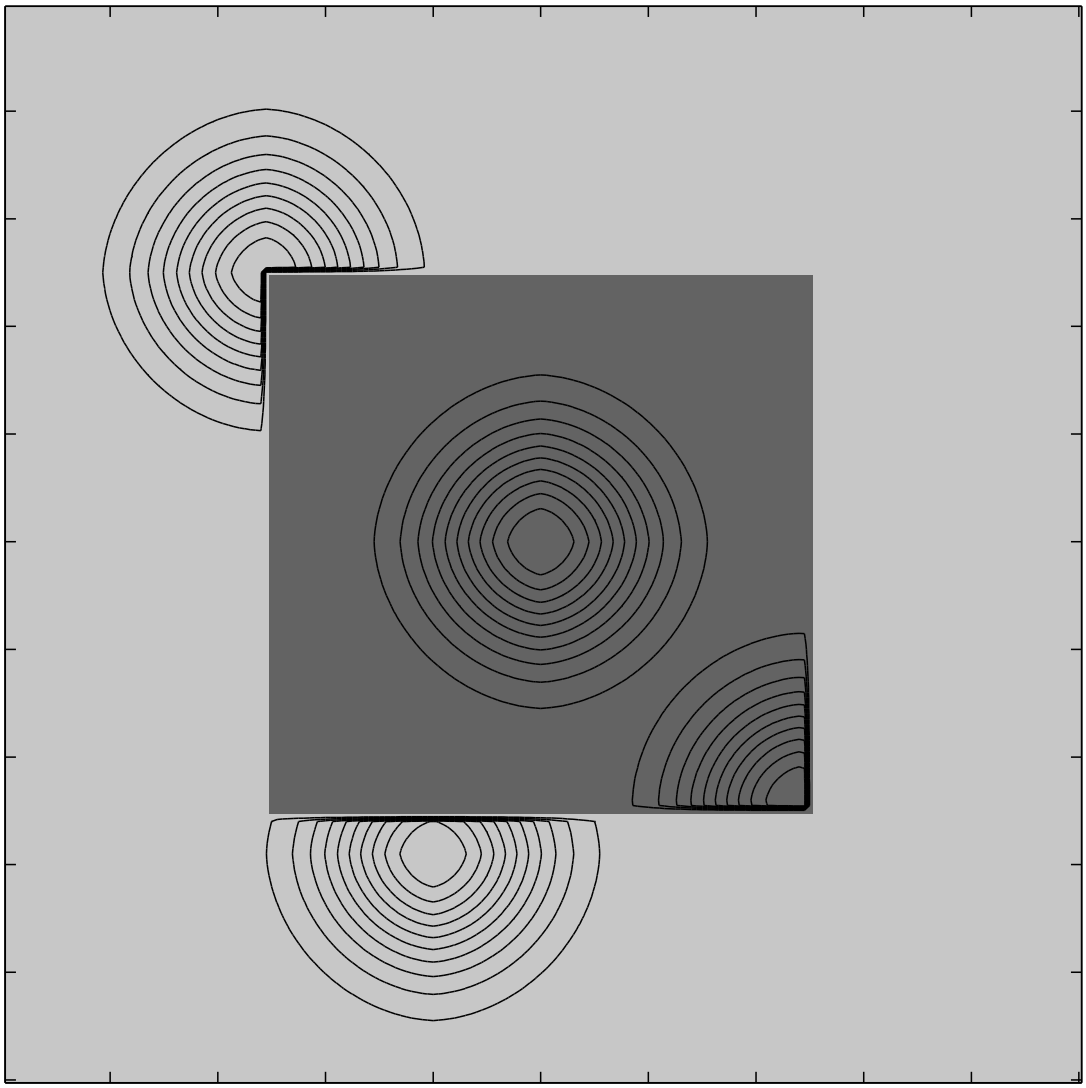

Fig. 6. Level curves of the Beltrami kernel at various locations in a synthetic image.

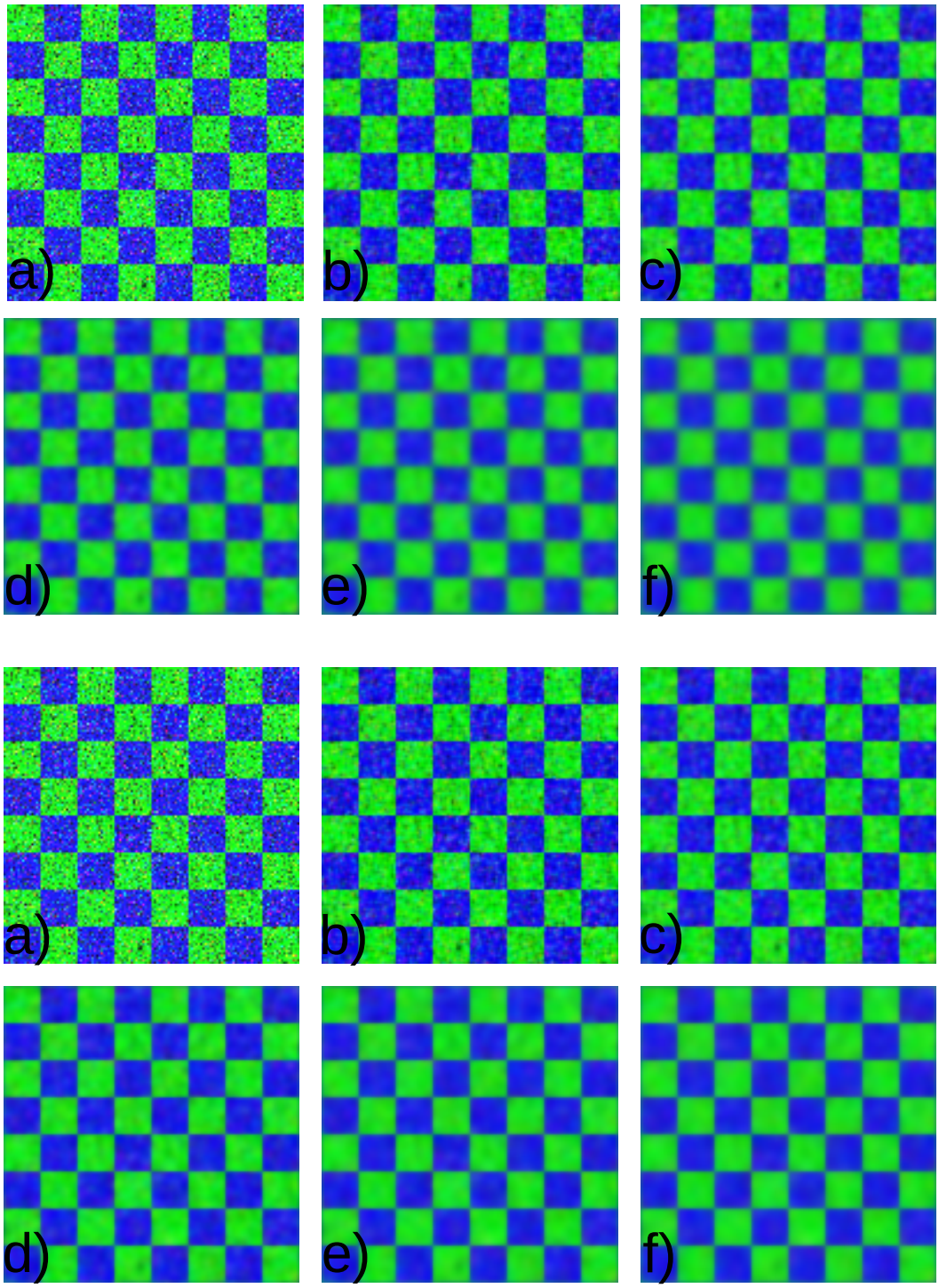

Fig. 7. The difference between smoothing an image as a regular 2D image and smoothing it as an image painted on its original manifold. On the top two rows the image is smoothed as a regular 2D image. On the bottom two rows the image is smoothed on the original manifold, where $z = 1$ at the green squares and $z = 0$ at the blue squares. The respective times of the images are: a) T=0.0, b) T=0.5, c) T=1.0, d) T=2.0, e) T=3.0, and f) T=4.0.

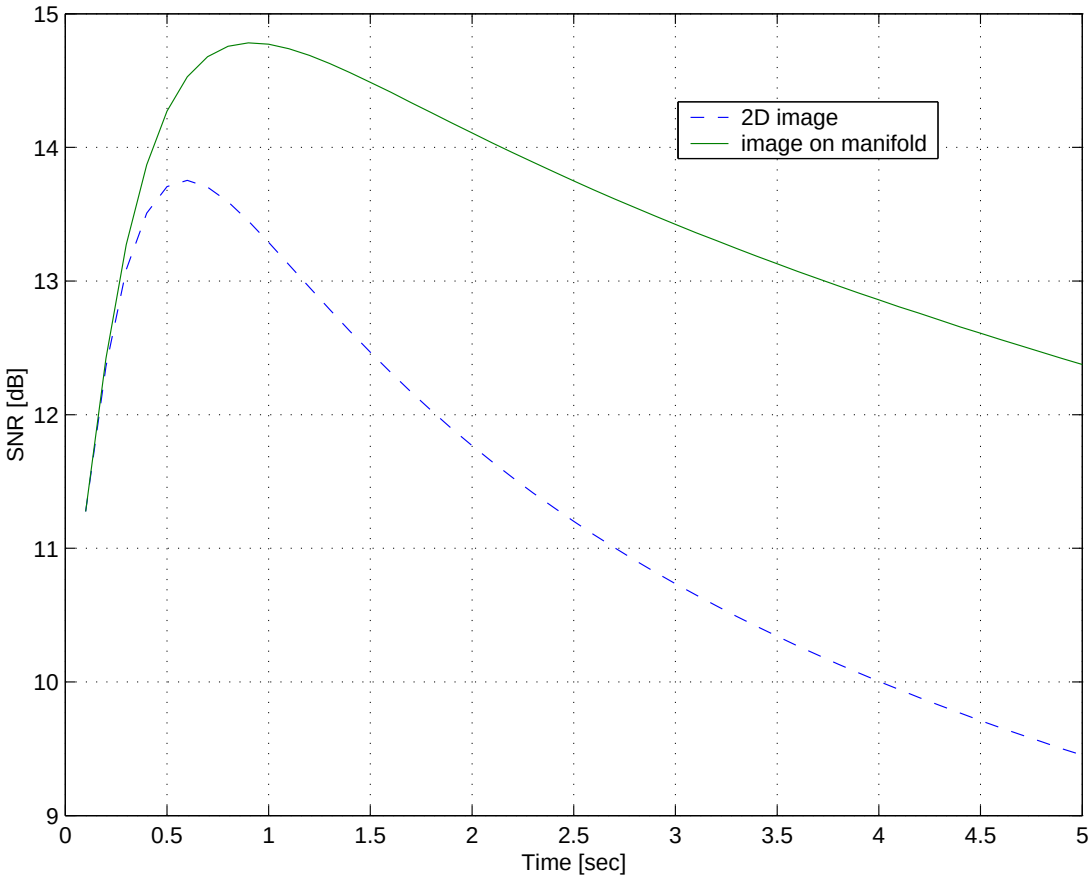

Fig. 8. The SNR of the smoothed images as a function of time.

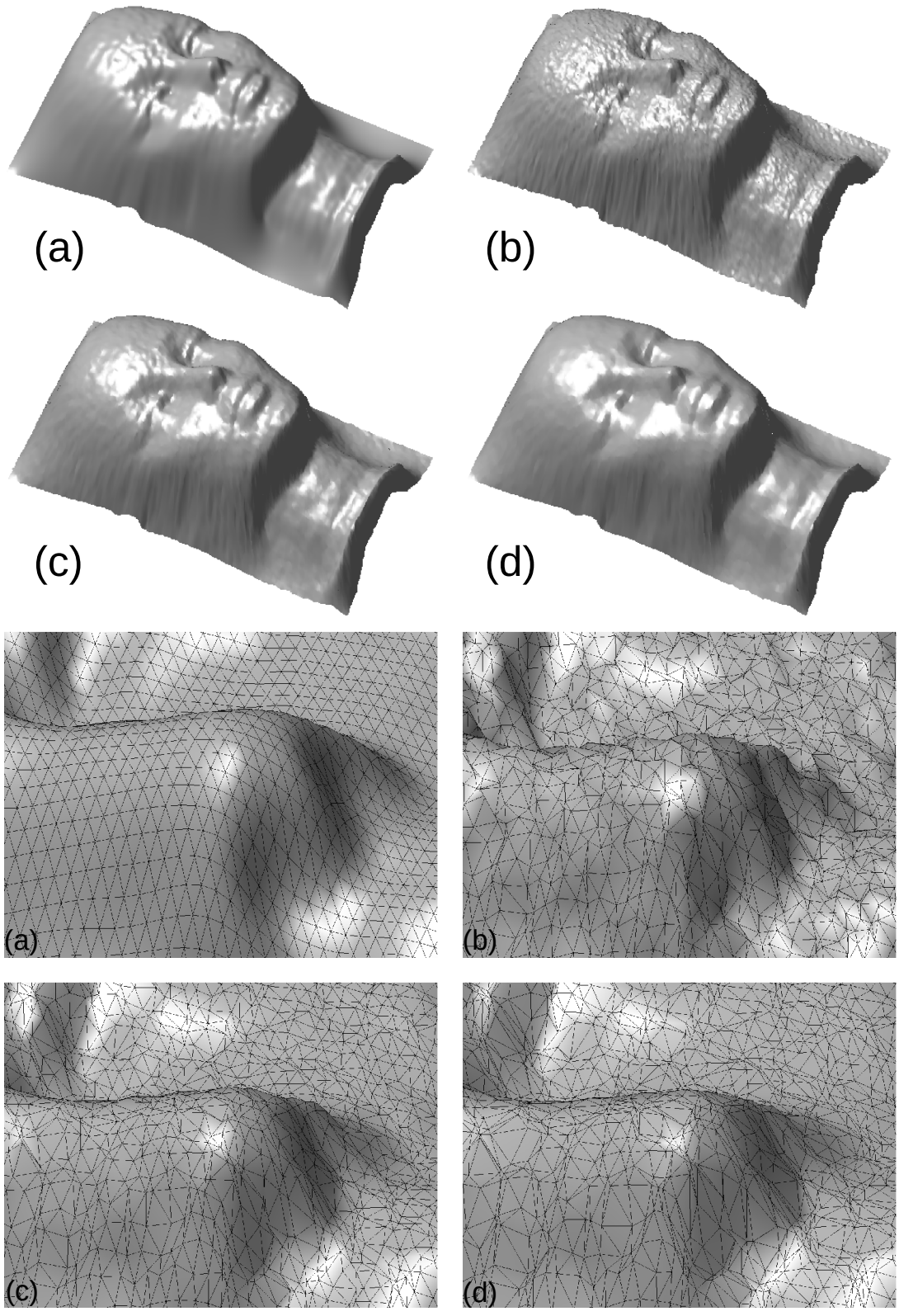

Fig. 9. Smoothing a triangulated manifold. a) the original manifold, b) noise added to the manifold, c) after one iteration of the Beltrami kernel, d) after two iterations of the kernel.

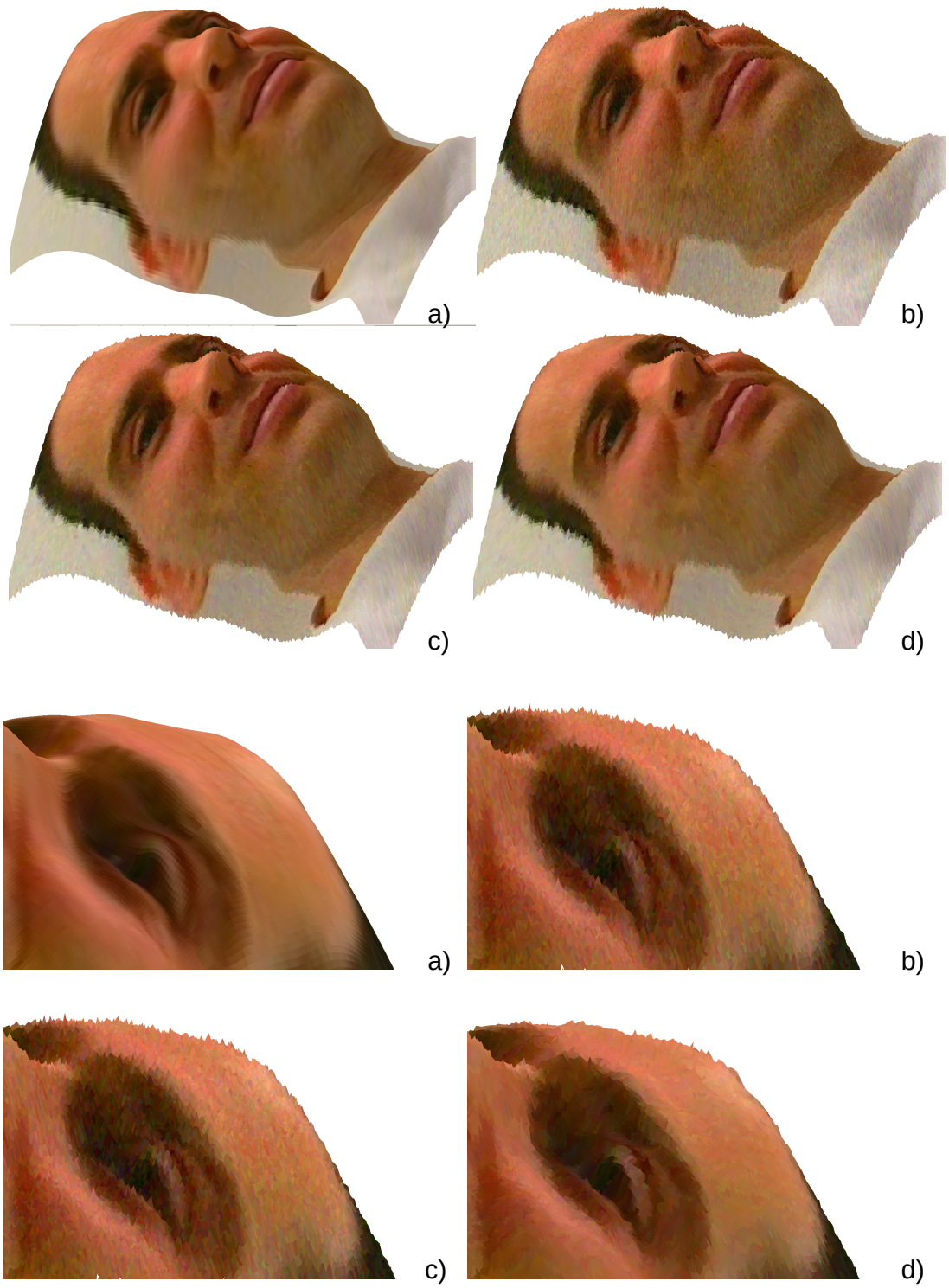

Fig. 10. Jointly smoothing a face manifold and the face image painted on it. a) the original face, b) noises added to the face, c) after one iteration of the Beltrami kernel, d) after two iterations of the kernel.