Enhancing Images Painted on Manifolds

A. Spira and R. Kimmel

Department of Computer Science, Technion, Israel
{salon,ron}@cs.technion.ac.il

Abstract. The fields of image processing, computer vision and computer graphics have concentrated traditionally on regular 2D images. Recently, images painted on 2D manifolds are becoming more popular and are used in face recognition, volumetric medical image processing, 3D computer graphics, and many other applications. The need has risen to regularize this type of images.
Various manifold representations are the input for these applications. Among the main representations are triangulated manifolds and parametric manifolds. We extend the short time image enhancing Beltrami kernel from 2D images to these manifold representations. This approach suits also other manifold representations that can be easily converted to triangulated manifolds, such as implicit manifolds and point clouds.
The arbitrary time step enabled by the use of the kernel filtering approach offers a tradeoff between the accuracy of the flow and its execution time. The numerical scheme used to construct the kernel makes the method applicable to all types of manifolds, including open manifolds and self intersecting manifolds. The calculations are done on the 2D manifold itself and are not affected by the complexity of the manifold or the dimension of the space in which it is embedded. The method is demonstrated on images painted on synthetic manifolds and is used to selectively smooth face images. Incorporating the geometrical information of the face manifolds in the regularization process yields improved results.

1 Introduction

The Beltrami framework [5, 17] enables state of the art image regularization. It produces a spectrum of image enhancing algorithms ranging from the L_2 linear diffusion to the L_1 non-linear flows. Apart from regular 2D images, the framework was used for textures, video, and volumetric data [6], non-Euclidean color spaces [18], and orientation diffusion [8]. A detailed review can be found in [23].

The recent increase in applications using images painted on 2D manifolds, requires the development of computational tools for regularizing such images. An approach based on harmonic maps was developed to enhance images painted on implicit manifolds [2, 3, 9]. In this approach the manifold is the zero set of a level set function [10] defined in the space in which the manifold is embedded. As noted before [19], this approach has three main drawbacks: the need to extend the

manifold to the embedding space, performing the calculations there (which might be computationally prohibitive for spaces with more than three dimensions) and the method's applicability only to manifolds represented by a level set and thus excluding more general manifolds, such as open manifolds and self intersecting ones.

Sochen et. al. [15, 14] 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 approach discussed in the previous paragraph. Still, the explicit numerical schemes used to implement the Beltrami flow require an upper bound on the time step used and might result in many iterations. Furthermore, the method was not extended to triangulated manifolds, which are common in many applications.

Recently, Bajaj et. al. [1] and Clarenz et. al. [4] presented combined regularizations of triangulated manifolds and the images painted on them. Both the manifolds and the images undergo anisotropic diffusions. The numerical scheme in [1] consists of Loop's subdivision while [4] uses a finite element discretization in space. Both use semi-implicit finite difference discretizations in time.

A short time kernel for the Beltrami flow on regular 2D images was presented in [22]. It followed the introduction of a short time kernel for 1D non-linear diffusion [16] and an approximation for the 2D Beltrami operator [13]. These kernels implement the flows by 'convolving' the signals with the kernels, similar to the implementation of the heat equation by a convolution with a gaussian kernel. The numerical implementation of the kernels enables an arbitrary time step that gives a tradeoff between the accuracy of the flow and its execution time.

We present here an extension of the short time Beltrami kernel to images painted on manifolds. This kernel enjoys several important advantages,

- Efficiency achieved by performing the calculations on the 2D manifold itself.
- Flexibility through the tradeoff enabled by the selection of an arbitrary time step.
- Robustness by the applicability of the kernel to all possible 2D manifolds, including open manifolds and self intersecting ones.
- Simplicity due to the applicability of the method to all popular manifold representations including triangulated manifolds, parametric manifolds, implicit manifolds and point clouds. The difference in the implementation of the method to the various manifold representations lies only in the preprocessing stage.

In order to compute the short time kernel we need to calculate geodesic distances between pixels in the image. For images painted on parametric manifolds we use fast marching [11, 12, 24] on parametric manifolds [20, 21]. For images painted on triangulated manifolds we use fast marching on triangulated manifolds [7].

This paper is organized as follows. The first section describes the Beltrami flow for images painted on manifolds. In Section 2 it is implemented by the short time kernel. Section 3 describes the calculation of geodesic distance maps on

images. In Section 4 the kernel is demonstrated on images painted on synthetic manifolds and is used to regularize face images. The conclusions are in Section 5.

2 The Beltrami Flow

According to the Beltrami framework [5, 17] the image is represented by $\{X^1, X^2, \ldots, X^M, I^1, I^2, \ldots, I^N\}$, with X^i the spatial coordinates and I^j the intensity components. The following derivation will assume color images painted on parametric manifolds embedded in $\mathbb{R}^3$, where we have $M = 3, N = 3$ 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 M and N the derivation is virtually identical.

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 M rows and β^2 in the next N. β is the relative scale between the spatial coordinates and the intensity components. The metric elements g_{ij} of the image are derived from the metric elements h_{ij} and the embedding by the pullback procedure

$$G = (g_{ij}) = \begin{pmatrix} \sum_i (X_1^i)^2 + \beta^2 \sum_j (I_1^j)^2 & \sum_i X_1^i X_2^i + \beta^2 \sum_j I_1^j I_2^j \\ \sum_i X_1^i X_2^i + \beta^2 \sum_j I_1^j I_2^j & \sum_i (X_2^i)^2 + \beta^2 \sum_j (I_2^j)^2 \end{pmatrix}, \quad (1)$$

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

$$S = \iint \sqrt{g} du_1 du_2, \quad (2)$$

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

$$I_t^i = \Delta_g I^i, \quad (3)$$

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

For gray level images we have

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

i.e., the image surface moves according to the intensity component of the mean curvature flow, see Fig. 1.

3 A Short Time Kernel for the Beltrami Flow

As shown in [22], a kernel exists for the Beltrami flow on 2D regular images. This is similar to the simpler case of linear diffusion, where applying the PDE

$$I_t = \Delta I \quad (4)$$

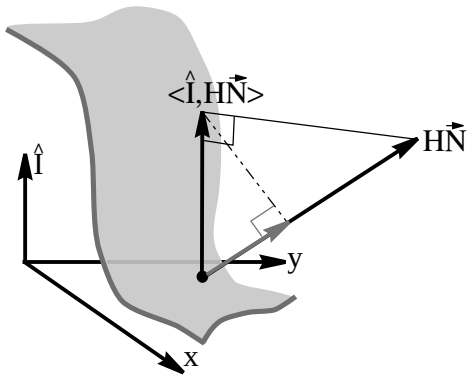

Fig. 1. In the Beltrami flow for 2D regular gray level images the image surface moves according to the intensity component of the mean curvature flow. Geometrically, only the projection of this movement on the normal to the surface matters.

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 kernel

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

where the kernel is given by

$$K(u^1, u^2; t) = \frac{1}{4\pi t} \exp\left(-\frac{(u^1)^2 + (u^2)^2}{4t}\right) \ . \tag{6}$$

Because of the non-linearity of the Beltrami flow (the Beltrami operator depends on the data I), the Beltrami kernel is 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 Fig 2. For the Gaussian kernel the amplitude of the filtered image at a specific pixel is the sum of the neighboring pixels' amplitudes weighted according to their distance along the coordinate axis. For the nonlinear Beltrami kernel the weighting is according to the geodesic distance on the image itself. The Beltrami kernel 'resides' on the image while for the linear kernel the Gaussian 'resides' on the coordinate axis. This is the reason why linear diffusion blurs the image while the Beltrami flow removes the noise but keeps the edges intact.

In each iteration of the Beltrami kernel we use

$$I^i(u^1, u^2, t_0 + t) = \iint I^i(\tilde{u}^1, \tilde{u}^2, t_0) K(u^1, u^2, \tilde{u}^1, \tilde{u}^2; t) d\tilde{u}^1 d\tilde{u}^2 \ , \tag{7}$$

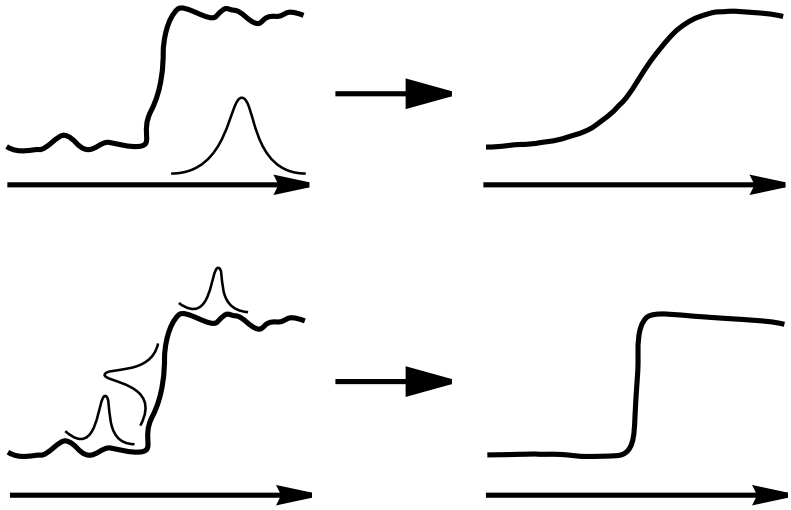

Fig. 2. Filtering an image with a linear Gaussian kernel (top) and a nonlinear Beltrami kernel (bottom).

with the kernel

$$K(u^1,u^2,\tilde{u}^1,\tilde{u}^2;t) = \frac{H_0}{t}\exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)} ds\right)^2}{4t}\right) = \frac{H_0}{t}\exp\left(-\frac{d_g^2\left((u^1,u^2),(\tilde{u}^1,\tilde{u}^2)\right)}{4t}\right), \tag{8}$$

where ds is an arc-length element on the image, and $d_g(p_1,p_2)$ is the geodesic distance between two pixels p_1 and p_2. For the full derivation of the kernel see [22]. The derivation of the short time kernel for the Beltrami flow on images painted on manifolds is the same. The difference lies in the calculation of geodesic distances on this kind of images, which will be detailed in the next section.

The resulting update step for the Beltrami kernel is

$$I^i(u^1,u^2,t_0+t) = \frac{H_0}{t}\iint_{(\tilde{u}^1,\tilde{u}^2)\in N(u^1,u^2)} I^i(\tilde{u}^1,\tilde{u}^2,t_0)\exp\left(-\frac{\left(\int_{(u^1,u^2)}^{(\tilde{u}^1,\tilde{u}^2)} ds\right)^2}{4t}\right) d\tilde{u}^1 d\tilde{u}^2, \tag{9}$$

with $N(u^1,u^2)$ the neighborhood of the pixel (u^1,u^2), where the value of the kernel is above a certain threshold. Due to the monotone nature of the fast marching algorithm used in the next section for the solution of the eikonal equation, once a pixel 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 pixels is negligible. Therefore, the eikonal equation is solved only in a small neighborhood of each pixel. H_0 is taken such that integration over the kernel in the neighborhood $N(u^1, u^2)$ of the pixel equals one.

4 Solving the Eikonal Equation on Images Painted on Manifolds

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. The viscosity solution $\phi(u^1, u^2)$ of the eikonal equation

$$\|\nabla_g \phi\| = 1, \tag{10}$$

is the required geodesic distance map from the pixel to its neighbors, where $\nabla_g \phi$ is the gradient of ϕ on the image. To solve the eikonal equation on the image we use the fast marching method.

Regular 2D images are parametric manifolds, where the metric g_{ij} is given for every point. Therefore, calculating the geodesic distances needed for implementing the kernel to these images [22] was done by an extension of the fast marching method [11, 12, 24] to parametric manifolds [20, 21]. The same method is used here for images painted on parametric manifolds. For images painted on triangulated manifolds we use fast marching on triangulated manifolds [7]. Since the embedding space in our case has at least four dimensions (gray scale images painted on manifolds), calculating the distances explicitly on the 2D image is advantageous.

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. 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 pixel.

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 [20, 21]. For triangulated manifolds an "unfolding" scheme is used, see [7].

After this pre-processing stage, all the triangles in the numerical grid are acute, as in Fig. 3. 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 $\phi(C)$ based on $\phi(A)$ and $\phi(B)$.

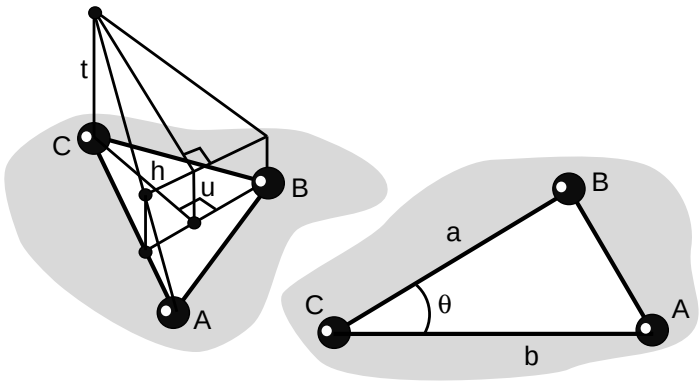

Fig. 3. The numerical stencil used to update $\phi(C)$ according to $\phi(A)$ and $\phi(B)$.

The numerical scheme according to [7] is

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

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

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

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 [12] is used.

Initialization:

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 is upper bounded by $O(n \log n)$, 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 [12]. 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 size of the heap used for the fast marching and enhance its efficiency.

5 Simulations and Results

We first demonstrate the effect of the manifold on the resulting enhanced image. In Fig. 4 a texture image is painted on a flat plane (a regular 2D image) as well as on the manifold $\{x, y, \sin(2\pi x)\sin(2\pi y)\}$. Both images are enhanced using the short time Beltrami kernel. While the texture in the regular 2D image is smoothed evenly, the degree of smoothing in the image painted on the manifold differs according to the geometry of the manifold. In planar areas of the manifold (its peaks and troughs) the smoothing is the same as in the regular image, but on the slopes the spatial extent of the kernel is smaller and there is less smoothing, as can be expected from Fig. 2.

Figure 5 shows the difference between enhancing a color face image as a regular 2D image and enhancing it as an image painted on the face manifold. In both cases one iteration of the kernel with a time step of $t = 0.5$ was applied, only grid points with a kernel value above 0.01 were used for the filtering and the fast marching was restricted to a neighborhood of 7×7 around each updated pixel. An average of 20 pixels were used in the kernel as a result of these parameters. A comparison between the output images shows that a kernel that takes into account the geometry of the manifold smoothes more in flat regions such as the forehead and smoothes less in edges of the manifold such as the lips. The overall effect is a more selective smoothing and a better looking output image.

6 Conclusions

We have presented a short time kernel for the Beltrami flow for images painted on manifolds. Incorporating the metric of the manifold in the flow produces better results in applications such as face image regularization. The numerical implementation of the kernel handles every possible manifold represented by virtually every common manifold representation. It enjoys low computational complexity further enhanced by an arbitrary time step that enables trading accuracy for a shorter execution time. All these attributes make the Beltrami kernel a highly practical tool for real life graphics and 3D image processing applications.

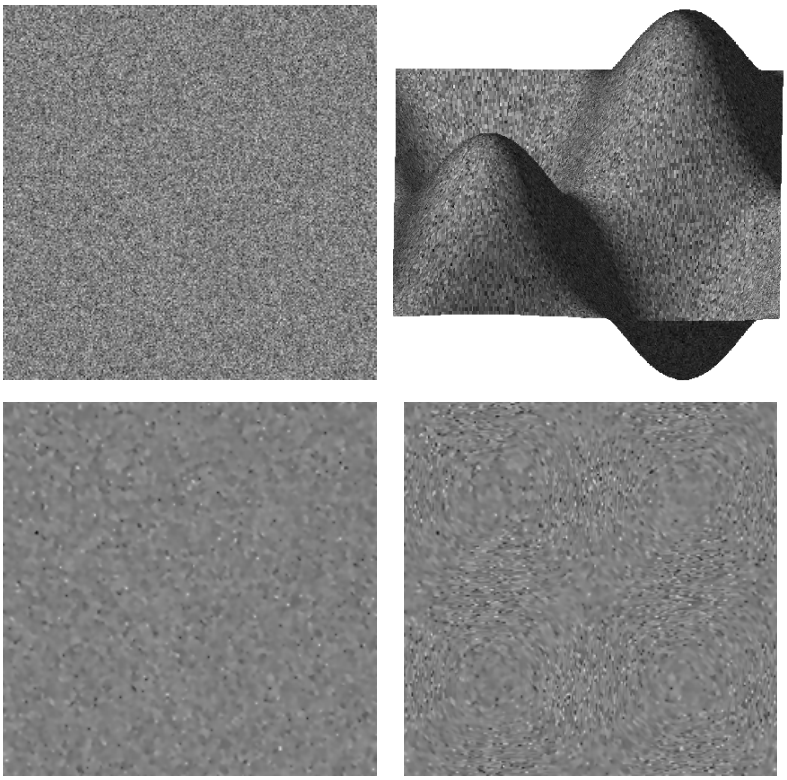

Fig. 4. The effect of the manifold on the enhanced images. The regular 2D image is on the left and the texture painted on the manifold is on the right. The original textures are on the top and the enhanced textures on the bottom.

References

1. C. Bajaj and G. Xu. Anisotropic diffusion of surfaces and functions on surfaces. *ACM Transactions on Graphics*, 22:4–32, January 2003.
2. M. Bertalmio, L. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174:759–780, October 2001.
3. M. Bertalmio, F. Memoli, L. 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*, pages 381–398, 2003.
4. U. Clarenz, U. Diewald, and M. Rumpf. Processing textured surfaces via anisotropic geometric diffusion. *IEEE Transactions on Image Processing*, 13(2):248–261, 2004.
5. 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.

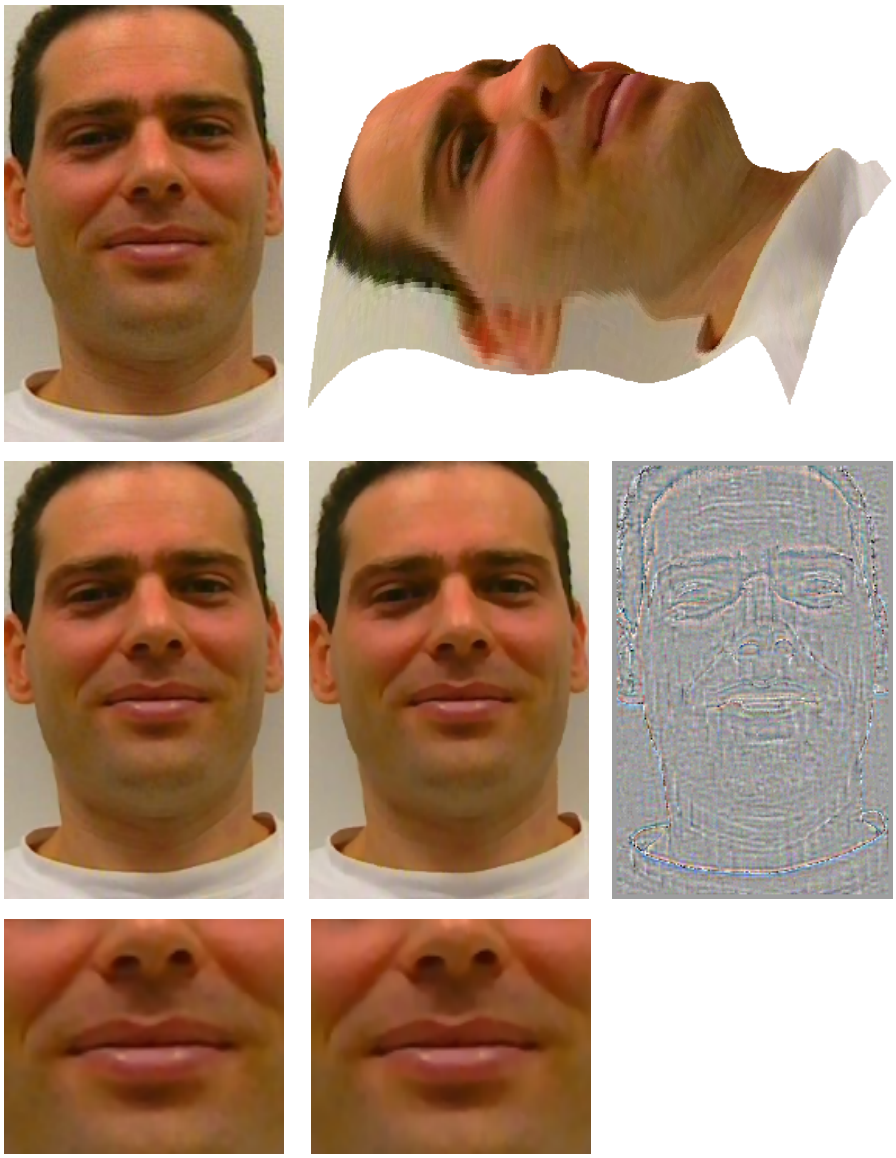

Fig. 5. Enhancing a color face image with the Beltrami kernel. On the left the image is treated as a regular 2D image and on the right as an image painted on the face manifold. On the top are the original images and in the middle and the bottom are the enhanced images. The image in the middle right is the difference between the images on its left.

6. 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.
7. R. Kimmel and J. Sethian. Computing geodesic paths on manifolds. *Proceedings of National Academy of Sciences*, 95(15):8431–8435, July 1998.
8. R. Kimmel and 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.
9. F. Memoli, G. Sapiro, and S. Osher. Solving variational problems and partial differential equations mapping into general target manifolds. *Journal of Computational Physics*, 195(1):263–292, 2004.
10. S. Osher and J. Sethian. Fronts propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.
11. J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of National Academy of Sciences*, 93(4):1591–1595, 1996.
12. J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge university press, 1996.
13. N. Sochen. Stochastic processes in vision: From langevin to beltrami. In *Proc. of International Conference on Computer Vision*, Vancouver, Canada, July 2001.
14. N. Sochen, R. Deriche, and L. Lopez-Perez. The beltrami flow over implicit manifolds. In *Proc. of 9th International Conference on Computer Vision*, Nice, October 2003.
15. N. Sochen, R. Deriche, and L. Lopez-Perez. The beltrami flow over manifolds. Technical Report TR-4897, INRIA Sophia-Antipolis, Sophia Antipolis, France, 2003.
16. 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.
17. N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Trans. on Image Processing*, 7(3):310–318, 1998.
18. 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.
19. 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.
20. 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.
21. A. Spira and R. Kimmel. An efficient solution to the eikonal equation on parametric manifolds. *Interfaces and Free Boundaries*, 6(3):315–327, September 2004.
22. 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.
23. A. Spira, N. Sochen, and R. Kimmel. Geometric filters, diffusion flows, and kernels in image processing. *Handbook of Geometric Computing- Applications in Pattern Recogntion, Computer Vision, Neuralcomputing, and Robotics, Bayro Corrochano Eduardo (Ed.), Springer Verlag, Heidelberg*, pages 203–230, February 2005.
24. J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. on Automatic Control*, 40(9):1528–1538, 1995.