Efficient Beltrami Flow in Patch-Space

Aaron Wetzler and Ron Kimmel

Department of Computer Science,
Technion, Israel

Abstract. The Beltrami framework treats images as two dimensional manifolds embedded in a joint features-space domain. This way, a color image is considered to be a two dimensional surface embedded in a hybrid special-spectral five dimensional $\{x, y, R, G, B\}$ space. Image selective smoothing, often referred to as a denoising filter, amounts to the process of area minimization of the image surface by mean curvature flow. One interesting variant of the Beltrami framework is treating local neighboring pixels as the feature-space. A distance is defined by the amount of deformation a local patch undergoes while traversing its support in the spatial domain. The question we try to tackle in this note is how to perform patch based denoising accurately, and efficiently. As a motivation we demonstrate the performance of the Beltrami filter in patch-space, and provide useful implementation considerations that allow for parameter tuning and efficient implementation on hand-held devices like smart phones.

Keywords: Beltrami flow, patch-space, denoising.

1 Introduction

Following the success of the Non Local Means denoising method as introduced by Buades et al. in [2] much attention has been devoted to developing various types of patch based denoising techniques. A patch, in terms of an image, is generally considered to be a square region of pixels of fixed size centered at the coordinates of an image pixel. Peyrè in [6] studies patch based manifolds while a more specific analysis of a generalized patch based denoising framework is done by Tschumperlè and Brun in [12]. They show that the NL means [2] and Bilateral [11] filters are isotropic versions of their patch based diffusion framework by choosing a specific patch size and metric. In much the same way Sochen et al. present the Beltrami framework and show in [8] how choices of different metrics can be used to produce filtering methods like the anisotropic diffusion process of Perona and Malik [5] as an example. Anisotropic diffusion was also shown by Barash in [1] to have a strong connection to the Bilateral filter through the adaptive smoothing filter and Elad in [4] demonstrated its connection to other classical filtering techniques.

In [7] Maragos and Roussos, explore a generalization of the Beltrami flow using weighted patches. We will use a similar formulation while setting the weights of each neighboring pixel to be one. In this context, the Beltrami framework provides a general and natural substrate for diffusion based image manipulation and naturally extends to higher dimensions. We will show how it can be applied to an image manifold in patch-space with better visual results as well as the overall PSNR compared to strictly local-differential techniques. We will discuss numerical considerations and demonstrate how the use of an integral image eliminates the algorithm's dependency on the patch size allowing for good performance on a modern smartphone.

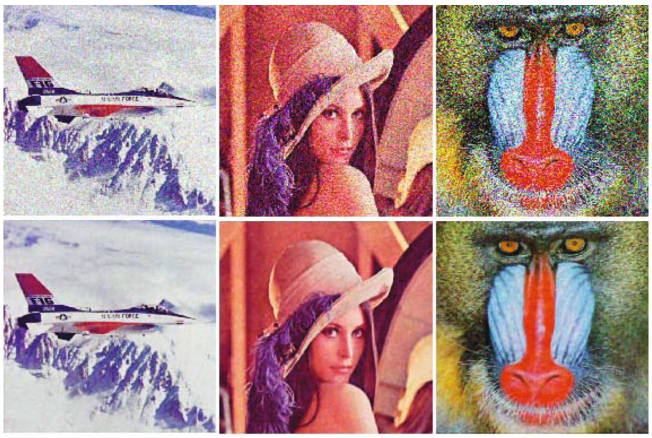

Fig. 1. Examples of Beltrami patch denoising for color images. From top to bottom, left to right a) Noisy F16, $\sigma = 20$ b) Denoised image, $PSNR = 31.51dB$ c) Noisy Lena, $\sigma = 30$ d) Denoised image, $PSNR = 29.54dB$ e) Noisy Mandrill, $\sigma = 50$ f) Denoised image, $PSNR = 21.43dB$.

2 The Beltrami Framework

We consider an image to be a $2D$ Riemannian manifold embedded in $D = d + 2$ dimensional space where $d = 1$ for grayscale images and $d = 3$ for color images. We can thus write the map $X : \Sigma \to M$ where X is the mapping of the image manifold into the embedding space feature manifold M. For a grayscale mapping we can write

$$X(\sigma_1, \sigma_2) = (x(\sigma_1, \sigma_2), y(\sigma_1, \sigma_2), z(\sigma_1, \sigma_2)). \quad (1)$$

If we further specify that $\sigma_1 = x$, $\sigma_2 = y$ and I is the image intensity map, then from (1) we have the graph of I given by

$$X(x, y) = (x, y, I(x, y)). \quad (2)$$

Both Σ and M are Riemannian manifolds and hence are equipped with metrics G and H respectively which enable measurement of lengths over each manifold.

We require the lengths as measured on each manifold to be the same. Thus we can write that

$$ds^2 = (dx\,dy\,dI)\,H \begin{pmatrix} dx \\ dy \\ dI \end{pmatrix} = (dx\,dy)\,G \begin{pmatrix} dx \\ dy \end{pmatrix}. \tag{3}$$

We can equate these and write the result compactly using Einstein notation where repeated upper and lower indices are summed over

$$g_{uv} = h_{ij}\partial_u X^i \partial_v X^j \quad u,v = 1..2 \quad i,j = 1..3 \tag{4}$$

Here the meaning of $\partial_{u,v}$ is just the partial derivative with respect to x or y. For the simple case in (4) where $H = (h_{ij})$ is the identity matrix we use the chain rule $dI = I_x dx + I_y dy$ and determine that for (2) the induced metric tensor $G = (g_{uv})$ is

$$G = \begin{pmatrix} 1+I_x^2 & I_x I_y \\ I_x I_y & 1+I_y^2 \end{pmatrix}. \tag{5}$$

Having a metric enables us to define a measure on the manifold which, for a Euclidean embedding in M, turns out to be the area of the surface as measured by the local coordinates in Σ

$$S[X,G] = \iint \sqrt{g}dxdy = A = \iint \sqrt{1+I_x^2+I_y^2}dxdy. \tag{6}$$

Here $g = \operatorname{div}(G)$. There is a more general version of the above measure called the Polyakov action which can be useful for non-Euclidean embeddings and details of its application to the Beltrami framework can be found in [8]. We now minimize the functional in (6) using the methods of variational calculus with the resulting Euler-Lagrange relation given by

$$-\frac{d}{dx}\left(\frac{I_x}{\sqrt{g}}\right) - \frac{d}{dy}\left(\frac{I_y}{\sqrt{g}}\right) = -\operatorname{div}\left(\sqrt{g}G^{-1}\nabla I\right) = 0. \tag{7}$$

We excersize freedom of paramaterization and multiply by $g^{-1/2}$ which allows (7) to be compactly written as $\Delta_g I = 0$ where Δ_g is the second order differential operator of Beltrami. We now formulate a geometric flow of the manifold

$$I_t = \Delta_g I, \tag{8}$$

which creates a scale space via the generalization of the Laplace operator onto Riemannian manifolds. The discretized version of (8) allows us to perform iterative traversal through this scale space on a computer and produces a very effective technique for denoising grayscale images when using the metric in (5).

3 Operating in Patch-Space

A patch is a window centered at a given pixel. We therefore define the mapping $P : \Sigma \to \mathbb{R}^{nw^2+2}$ in the form

$$P(x,y) = \left(x, y, \left\{I^k(x+iw, y+jw)\right\}\right) \quad i,j = -w,..,w, \quad k = 1,..,n. \tag{9}$$

Here $w \in \mathbb{N}$ is known as the window size or patch size, and n is the number of channels in the image. For example, a single channel image where $n = 1$ and $w = 5$ produces patches of size 11×11 centered about each pixel in the image I. We can see that the above definition reduces to the grayscale embedding (2) for $w = 0$ and $n = 1$ as described in the previous section. From here on we will denote $\left\{I^k\left(x + iw, y + jw\right)\right\}$ $i, j = -w, .., w \quad k = 1, .., n$, as $I^k_{i,j}$. Note that I^k is simply the kth color channel. We wish to derive the induced metric tensor G for this new embedding. For that goal we first consider the arclength measurement in the embedding space which we assume to be Euclidean and therefore

$$ds^2 = \langle \boldsymbol{dP}, \boldsymbol{dP} \rangle_H = dx^2 + dy^2 + \sum_{i,j,k} \left(dI^k_{i,j}\right)^2 . \tag{10}$$

In reality, the coordinates x and y do not possess the same physical measure as the intensity values of the image so we need to introduce a scaling factor into the patch-space metric given by

$$h_{ij} = \begin{cases} \delta_{ij} & i, j \leqslant 2 \\ \beta^2 \delta_{ij} & \text{otherwise} \end{cases} , \tag{11}$$

where δ_{ij} is the Kronecker delta. Following the same procedure as before and using the chain rule $dI^k_{i,j} = I^k_{i,j\,x} dx + I^k_{i,j\,y} dy$ we pullback the metric from the embedding to determine that the new induced metric tensor for the 2D image manifold embedded into patch-space is given by

$$G = \begin{pmatrix} 1 + \beta^2 \sum_{i,j,k} I^{k\,2}_{i,j\,x} & \beta^2 \sum_{i,j,k} I^k_{i,j\,x} I^k_{i,j\,y} \\ \beta^2 \sum_{i,j,k} I^k_{i,j\,x} I^k_{i,j\,y} & 1 + \beta^2 \sum_{i,j,k} I^{k\,2}_{i,j\,y} \end{pmatrix} . \tag{12}$$

This metric combined with (8) gives the Beltrami flow in patch-space as

$$I_t = \Delta_g I = \frac{1}{\sqrt{g}} \operatorname{div}\left(\sqrt{g} G^{-1} \nabla I\right) . \tag{13}$$

4 Implementation and Results

We use the flow given by (13) to progress through the scale space on image manifolds embedded into patch-space for both grayscale and color images. The algorithm was tested on a desktop PC and the color version was efficiently implemented on an iPhone 4 smartphone. To measure the success we visually inspected the results as well as measured the standard Peak Signal to Noise Ratio for images: $PSNR = 10\log_{10}\left(255^2 / E\left[\left(I_{est} - I\right)^2\right]\right)$ where I_{est} is the estimation of the denoised version of I.

4.1 Parameter Optimization

Given an image with additive Gaussian white noise and standard deviation σ we need to find a set of parameters that produces the best $PSNR$ value. The normal approach is to fix β and change the number of iterations which allows traversal of the scale space. The obvious disadvantage is that more iterations mean longer execution times. One efficient alternative which has been used here is to fix the number of iterations and vary β. This has the effect of artificially moving through the scale space by causing a change of the distances on the embedded image manifold. The output therefore depends on the window size, the number of iterations of the update, and the parameter β. The time complexity of the algorithm is $O(KN^2W^2)$ where K is the number of iterations, N is the width of an image (assuming it is square) and $W = 2w + 1$ for a patch size w. We

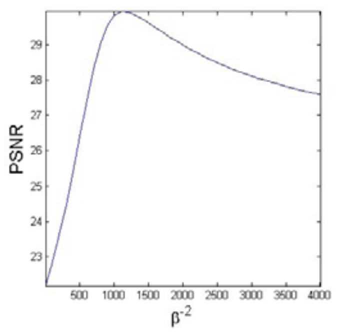

Fig. 2. Example of $PSNR$ as a function of β^{-2} with a typical global maximum

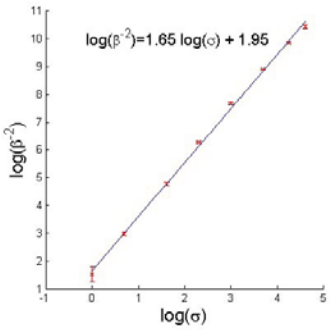

Fig. 3. Loglinear relationship between σ and β^{-2}. Error bars indicate one standard deviation from the mean over a set of different images.

fixed the variables depending on whether an image was grayscale or color. To optimize for β we ran a non-linear optimization program with the $PSNR$ as the target function for a particular image. With the variables held constant except for β, the $PSNR$ function was found to always have a global maximum over the search region. An example function is shown in Fig. 2. The analytical relationship between β and σ is non-trivial however we determined experimentally that the optimal value of β is approximately related to σ by a linear model in log space for values of σ up to 100[1] by the simple relation

$$\log(\beta^{-2}) = a\log(\sigma) + b. \tag{14}$$

Fig. 3 shows this relationship graphically. The graph was obtained by running the optimization program for a set of different images and then fitting the model in (14). It was found that the value of β that globally maximized the $PSNR$ of any representative image produced $PSNR$ values very close to maximum in other images corrupted by Gaussian noise with the same σ. The error bars in Fig. 3 show that there is almost negligible deviation from the mean for the optimal

[1] Pixel intensity values range from 0 to 255.

values of β for a given σ for different images. This fact is critical and illustrates that a and b obtained from the log-linear model only need to be calculated once for a predetermined window size, color type and iteration count. They can then be used to generate a β for any given σ for any image. Alternatively, a densely populated look-up table can be generated to relate the two for even greater accuracy.

4.2 Reducing Time Complexity

The weights of nearby pixels are unitary in our method. We take advantage of this property and eliminate the W^2 component by using an integral image to calculate the sums in (12) for each color channel yielding running time complexity of $O(KN^2)$. This allows for patch size independence in performance which is especially important for a practical implementation on a mobile device.
For low values of K and images of size 256×256 the iPhone implementation performs denoising in real time. A patch size of 5×5 ($w = 2$) produced the best results for grayscale images with negligible PSNR differences for the various iterations as shown in Table 1. The same behavior occurs for color images except

Table 1. *PSNR* results from denoising of the Cameraman image corrupted with AGWN for $\sigma = 20$ using optimized β. Values are in dB.

	$w = 0$	$w = 1$	$w = 2$	$w = 3$	$w = 4$
10 iterations	28.04	29.11	**29.21**	29.09	28.96
50 iterations	27.58	29.04	**29.37**	29.27	29.15
100 iterations	27.35	28.94	**29.36**	29.28	29.16
150 iterations	27.22	28.88	**29.35**	29.28	29.16

that the optimal patch size appears to be 7×7 ($w = 3$). The *PSNR* alone is not enough as can be seen in Fig. 4. The denoising properties of the Beltrami flow are reasonable for a small number of iterations, however higher quality visual results require more iterations. It was found that grayscale images are best denoised by $K = 150$ iterations and $w = 2$, whereas color images require only $K = 10$ iterations at a window size of $w = 3$.

Table 2. Run times in seconds for Patch Beltrami color denoising on an iPhone 4

	$N = 256$	$N = 512$
$K = 1$	0.14	0.54
$K = 5$	0.65	2.60
$K = 10$	1.27	5.13
$K = 20$	2.55	10.37
$K = 50$	6.42	25.45

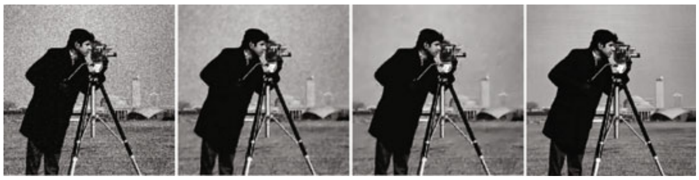

Fig. 4. From left to right a) Noisy image at $\sigma = 20$ b) Denoised with 10 iterations, $PSNR = 29.21$ c) Denoised with 150 iterations $PSNR = 29.35$ d) Original image

Running times for different iteration counts of the iPhone implementation for color images are shown in Table 2. Depending on an application's speed requirements, K can be further reduced down to $K = 1$ with a gradual decrease in output quality as shown in Fig. 5, where it is seen that after $K = 10$ there is virtually no improvement. For each iteration the update of a pixel is independent of the update of any other pixel, so the process is highly parallelizable, however this characteristic has not been exploited in the current implementation.

Table 3. PSNR comparison between Regular Beltrami, Patch Beltrami and NL means for some standard grayscale images. All values are in dB.

	CMan	Lena	Barbara	House	
Regular Beltrami	36.97	36.90	35.85	36.92	
Patch Beltrami	**37.70**	**38.11**	**37.01**	**38.16**	$\sigma = 5$
NL means	33.91	37.55	36.06	38.05	
Regular Beltrami	27.22	29.03	26.20	28.98	
Patch Beltrami	29.35	**32.05**	28.71	**32.13**	$\sigma = 20$
NL means	**29.37**	31.56	**29.86**	31.97	
Regular Beltrami	20.55	23.00	20.95	22.63	
Patch Beltrami	23.83	**27.13**	23.52	**26.88**	$\sigma = 50$
NL means	**23.93**	26.46	**24.26**	26.09	

Using the optimally chosen parameters, the denoising process can now be used automatically with the only input parameter being σ as is the norm for denoising images. The experiments in Table 3 reveal that apart from causing a significant improvement over the original application of the Beltrami flow, the new patch-based metric in fact produces results comparable to or even better than the Non-Local means method [2]. Furthermore, the results are within about $2dB$ of the state of the art, such as the block matching algorithms of Dabov et al. [3].

4.3 Residual Noise

Another way to compare the effectiveness of a denoising process is by evaluating the residual as noise as introduced by Baudes et al. in [2]. Here, we look at

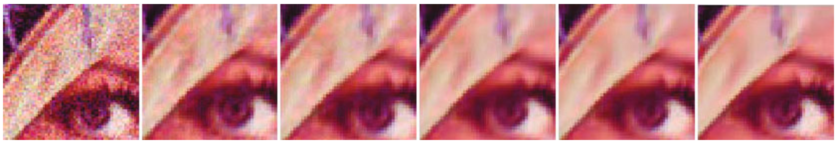

Fig. 5. Optimized denoising for different values of K a) Noisy image, $\sigma = 20$, $PSNR = 22.25$ b) $PSNR = 28.93$, $K = 1$ c) $PSNR = 29.91$, $K = 2$ d) $PSNR = 31.15$,$K = 5$ e) $PSNR = 31.45$, $K = 10$ f) $PSNR = 31.55$, $K = 40$

the differences between the estimated output image and the noisy input image. Ideally, the resulting difference image should also appear as Gaussian white noise. Fig. 6 shows that patch based Beltrami flow produces significantly less structure in the difference image compared to the original pixel based version.

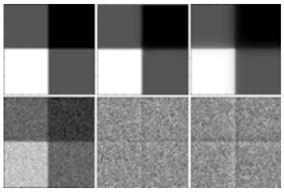

Fig. 6. Method noise. From top to bottom, left to right a) Original image b) Noisy image with $\sigma = 20$ c) Beltrami patch denoising. 150 iterations, $w = 2$ d) Method noise for Beltrami patch denoising e) Regular Beltrami denoising f) Method noise for regular Beltrami denoising.

4.4 Non-gaussian Denoising

In addition to filtering Gaussian noise, the Beltrami flow has other desirable properties. The process tends to align colors along boundaries which lends itself to solving the problem of antialiasing images with jagged, unmatched edges. Another fundamental characteristic of the method is the traversal of a scale space which flattens out smooth, weakly textured objects. An example of the effect of applying the Beltrami patch filter in both types of examples is shown in Fig. 7. It is interesting to note that a state of the art denoising method, BM3D [3], copes very poorly with these two situations because it is optimized for Gaussian noise removal alone.

5 Conclusions

We have shown that the extension of the original Beltrami filter with a more general metric produces significantly better results than the original Beltrami filter. The number of iterations required for denoising color images is $K \backsimeq 10$ resulting in a relatively fast algorithm of time complexity $O(KN^2)$ permitting an

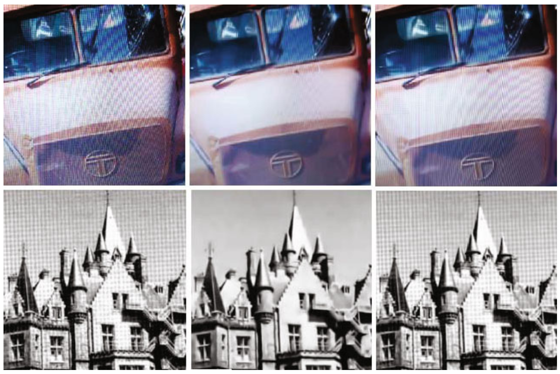

Fig. 7. Removal of aliasing and block textures. From left to right, top to bottom a) Photograph of truck with aliasing. b) Beltrami patch denoising, $\sigma = 20$ c) CBM3D denoising, $\sigma = 20$ d) Photograph of castle with weak block textures e) Beltrami patch denoising, $\sigma = 20$ f) BM3D denoising, $\sigma = 20$.

efficient implementation on a modern smartphone. We have also experimentally determined the relationship between the image intensities and their coordinates as posed in [8]. The proposed method produces $PSNR$ values close to state of the art techniques such as BM3D [3]. In addition to Gaussian denoising, the process accurately removes weak textures and aliasing while preserving the fine structure of the edges in images that other methods are not capable of dealing with.

6 Future Work

Modern smartphones have powerful graphics processing units as well as accelerated vector engine hardware. Neither of these features were utilized for the current application and further work is required to enable the method to work at optimal speed. Although this note has focused mainly on the denoising property of the Beltrami operator it would seem reasonable to further study other applications of the operator in patch-space such as inverse diffusion and other processes which control the eigen-values of the local diffusion operator. Many of these techniques have already been developed for the original Beltrami flow such as the FAB diffusion method as described by Gilboa et al. [9] and therefore it would be prudent to extend their application to patch-space. The same can be said for the short time Beltrami kernel as described by Spira et al. in [10] where it is approximated by finding local geodesic distances on the manifold via the fast marching method and the local metric tensor. The analysis and implementation of the same procedure would be a fruitful direction for future research in patch-space based flows.

Acknowledgments. This research was supported by European Community's FP7- ERC program, grant agreement no. 267414.

References

1. Barash, D.: A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(6), 844–847 (2002)
2. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: CVPR, pp. 60–65 (2005)
3. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-d transform-domain collaborative filtering. IEEE Transactions on Image Processing, 2080–2095 (2007)
4. Elad, M.: On the origin of the bilateral filter and ways to improve it. IEEE Transactions on Image Processing 11(10), 1141–1151 (2002)
5. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell., 629–639 (1990)
6. Peyre, G.: Manifold models for signals and images. Computer Vision and Image Understanding 113, 249–260 (2009)
7. Roussos, A., Maragos, P.: Tensor-based image diffusions derived from generalizations of the total variation and Beltrami functionals. In: ICIP (September 2010)
8. Sochen, N., Kimmel, R., Malladi, R.: A general framework for low level vision. IEEE Trans. on Image Processing, 310–318 (1998)
9. Sochen, N.A., Gilboa, G., Zeevi, Y.Y.: Color image enhancement by a forward-and-backward adaptive Beltrami flow. In: Sommer, G., Zeevi, Y.Y. (eds.) AFPAC 2000. LNCS, vol. 1888, pp. 319–328. Springer, Heidelberg (2000)
10. Spira, A., Kimmel, R., Sochen, N.A.: Efficient Beltrami flow using a short time kernel. In: Griffin, L.D., Lillholm, M. (eds.) Scale-Space 2003. LNCS, vol. 2695, pp. 511–522. Springer, Heidelberg (2003)
11. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proc. IEEE ICCV, pp. 836–846 (1998)
12. Tschumperlé, D., Brun, L.: Non-local image smoothing by applying anisotropic diffusion pde's in the space of patches. In: ICIP, pp. 2957–2960 (2009)