A Combined PDE and Texture Synthesis Approach to Inpainting

Harald Grossauer

Department of Computer Science
University of Innsbruck
Technikerstr. 25
A–6020 Innsbruck
AUSTRIA
harald.grossauer@uibk.ac.at
http://informatik.uibk.ac.at/infmath/

Abstract. While there is a vast amount of literature considering PDE based inpainting and inpainting by texture synthesis, only a few publications are concerned with combination of both approaches. We present a novel algorithm which combines both approaches and treats each distinct region of the image separately. Thus we are naturally lead to include a segmentation pass as a new feature. This way the correct choice of texture samples for the texture synthesis is ensured. We propose a novel concept of "local texture synthesis" which gives satisfactory results even for large domains in a complex environment.

1 Introduction

The increase in computing power and disk space over the last few decades has created new possibilities for image and movie postprocessing. Today, old photographs which are threatened by bleaching can be preserved digitally. Old celluloid movies, taking more and more damage every time they are exhibited, can be digitized and preserved. Unfortunately much material has already suffered. Typical damages are scratches or stains in photographs, peeled of coatings, or dust particles burned into celluloid. All these flaws create regions where the original image information is lost. Manual restoration of images or single movie frames is possible, but it is desirable to automate this process. Several *inpainting algorithms* have been developed to achieve this goal. In this paper we focus on single image inpainting algorithms (there exist more specialized algorithms for movie inpainting). They may roughly be divided into two categories:

1. Usually PDE based algorithms are designed to connect edges (discontinuities in boundary data) or to extend level lines in some adequate manner into the inpainting domain, see [1,2,3,4,5,6,7,8,9,10,11]. They are targeted on extrapolating geometric image features, especially edges. I.e. they create regions inside the inpainting domain. Most of them produce disturbing artifacts if the inpainting domain is surrounded by textured regions, see figure 1.

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3022, pp. 214–224, 2004.

2. Texture synthesis algorithms use a sample of the available image data and aim to fill the inpainting domain such that the color relationship statistic between neighbored pixels matches those of the sample, see [12,13,14,15,16, 17,18]. They aim for creating intra–region details. If the inpainting domain is surrounded by differently textured regions, these algorithms can produce disturbing artifacts, see figure 2.

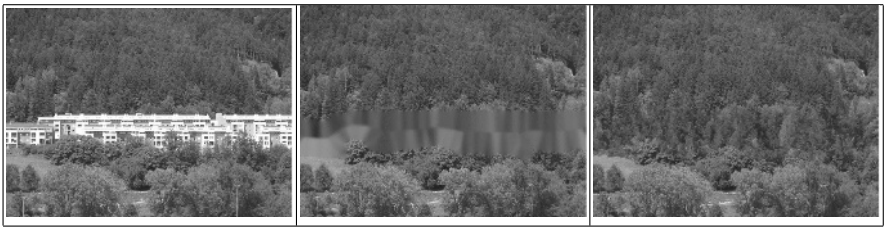

Fig. 1. An example which is not well suited for PDE inpainting. The texture synthesis algorithm achieves visually attractive results *(right picture)*, PDE based inpainting algorithms fail for large sized domains surrounded by strongly textured areas *(middle picture)*

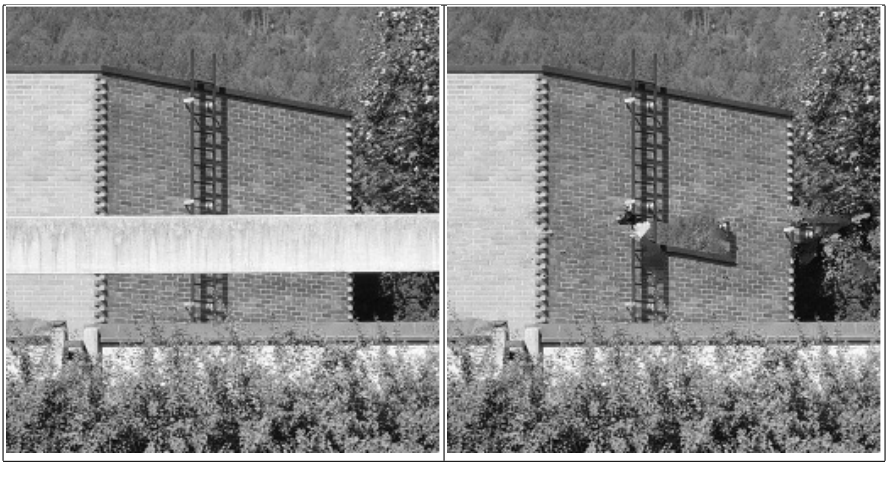

Fig. 2. Texture synthesis may run into problems, if the sampling domain is chosen inappropriately. The balustrade in the *left* picture should be removed by resynthesizing image contents, taking the rest of the picture as sample texture. The result can be seen on the *right*: the ladder initiated spurious sampling of trees and leaves into the brick wall

Until now there are only a few algorithms trying to treat geometric image features and texture simultaneously:

- An algorithm based on texture template spectrum matching has been proposed in [19]. This algorithm does not fit into either one of the two categories mentioned above.
- A special purpose algorithm was used in [20] for restoring missing blocks in wireless transmitted compressed images. Common lossy compression algorithms (like e.g. JPEG) divide an image into 8x8 pixel blocks that are independently compressed. If a corrupted block is detected it is reconstructed rather than retransmitted to decrease latency and bandwidth usage. Reconstruction occurs by classifying the contents of adjacent blocks into either structure or texture. Depending on this classification the missing block is restored by invoking either a PDE inpainting or a texture synthesis algorithm.
- Closely related to our inpainting technique are the algorithms proposed in [21,22], which are most natural to compare with in the course of this paper. They differ from our algorithm by the choice of the subalgorithms in each step. Moreover, we propose to perform a segmentation step to determine appropriate texture sample regions. This prevents artifacts arising in the texture synthesis step, as exemplified in figure 2. This problem is not adressed in [21,22].

2 The Algorithm

The proposed inpainting algorithm consists of five steps:

1. Filtering the image data and decomposing it into geometry and texture
2. PDE inpainting of the geometry part
3. Postprocessing of the geometry inpainting
4. Segmentation of the inpainted geometry image
5. Synthesizing texture for each segment

We will describe each step in detail in the following subsections. The image, denoted by a function $u : D \to \mathbb{R}$ (or $\mathbb{R}^3$ for color images), is defined on the image domain $D \subset \mathbb{R}^2$. A user specified *mask function* $m : D \to [0, 1]$ marks the inpainting domain $\Omega = \text{supp}(m)$. A value of 1 in the mask function highlights the flawed region. The mask is designated to continuously drop to zero in a small neighborhood (i.e., a few pixels) outside of the flawed region. This "drop down" zone will later be used to smoothly blend the inpainting into the original image.

2.1 Filtering and Decomposition

A nonlinear diffusion filter of Perona–Malik type [23] is applied to the image u, i.e. it is evolved according the partial differential equation

$$\frac{\partial u}{\partial \tau} = \nabla \cdot (d(\|\nabla u(x, y)\|) \cdot \nabla u) \tag{1}$$

where the diffusivity $d(s)$ is chosen to be

$$d(s) = \frac{1}{1 + \frac{s^2}{\lambda^2}} \tag{2}$$

with a suitably chosen parameter λ. The filtered image g is the solution of (1) at a specified time τ_0, characterizing the strength of filtering. The effect of the filter is that g reveals piecewise constant intensities and contains little texture and noise. Thus we call g the *geometry part*. We set $u = g + t$ and refer to t as the texture part. All along this paper we consider noise as a texture pattern.

Recently several advanced techniques for image decomposition have been proposed, see [24,25,26]. In [21] decomposition is done by using the model from [24], i.e. by jointly minimizing the BV seminorm of the function g and Meyers G-norm (see [27]) of the function t. Using this model allows one to extract texture *without* noise, which leads to an approximative decomposition $u \approx g + t$, with geometry part g and texture t. For inpainting this seems not to be optimal, since a noisefree inpainting in a noisy image may look inadequate.

In [22] a lowpass/highpass filter is applied to the image to attain g resp. t. The decomposition is thus not into geometry and texture but rather into high and low frequencies.

2.2 PDE Inpainting

For the inpainting of the geometry part we use the Ginzburg–Landau algorithm proposed in [11]. Here we give a short overview:

We calculate a complex valued function $\tilde{g}$, whose real part is the geometry image g scaled to a range of values between -1 and 1. Further we demand that $\|\tilde{g}(x,y)\|_{\max} = 1$ for all (x,y), and the imaginary part be nonnegative, $\Im\tilde{g} \geq 0$. The function $\tilde{g}$ is evolved using the *complex* Ginzburg–Landau equation

$$\frac{\partial \tilde{g}}{\partial \tau} = \Delta\tilde{g} + \frac{1}{\varepsilon^2}\left(1 - \|\tilde{g}(x,y)\|^2_{\max}\right)\tilde{g} \tag{3}$$

inside Ω, where the available data $\tilde{g}|_{\partial\Omega}$ is specified as Dirichlet boundary condition. Here $\varepsilon \in \mathbb{R}$ is a length parameter specifying the width of edges in the inpainting. $\|\cdot\|_{\max}$ denotes the maximum norm of the components of $\tilde{g}(x,y)$, which is just the absolute value $|\tilde{g}(x,y)|$ if g is a grayscale image, and $\max\{|\tilde{g}^{red}|, |\tilde{g}^{green}|, |\tilde{g}^{blue}|\}$ for RGB images. The real part $\Re\tilde{g}$ of the evolved image at some time τ_0 (i.e., if $\tilde{g}$ is "close enough" to steady state) is rescaled to the intensity range of g and constitutes the inpainting.

For a more detailed description we refer to our presentation in [11]. Theoretical results about the Ginzburg–Landau equation and similar reaction–diffusion equations can be found in [28,29].

In [21] the PDE inpainting method from [4] is used. In comparison with (3) this algorithm (judging from the examples given in [4]) creates smoother and better aligned edges, but the Ginzburg–Landau algorithm reveals higher contrast and less color smearing.

In [22] the inpainting technique from [10] is utilized. This algorithm is not designed to create edges in the inpainting domain. For the particular application to inpaint a low pass filtered image this is no problem.

2.3 Postprocessing

The Ginzburg–Landau algorithm sometimes produces kinks and corners in edges. A detailed discussion of this phenomenon has been given in [11]. This is the price for high contrast edges in the inpainting domain. To straighten the kinks we apply a coherence enhancing anisotropic diffusion filter to the image g on the inpainting domain Ω. This filter is described in [30]. We implemented this diffusion filter using the multi–grid algorithm outlined in [31]. Since diffusion happens mostly along edges and not across, the contrast does not suffer significantly.

2.4 Segmentation

As a preparation for the texture synthesizing step the inpainted and postprocessed geometry image (which for the sake of simplicity is again denoted by g) is segmented. We employ a gradient controlled region growing algorithm, inspired by the *scalar* Ginzburg–Landau equation:

Let $(\mathcal{S}_i \subseteq D)_{i=0..N}$ be the (a–priori unknown) segmentation of D, i.e. $\bigcap_i \mathcal{S}_i = \emptyset$ and $\bigcup_i \mathcal{S}_i \supseteq \Omega$. We assume that every pixel in Ω belongs to exactly one segment $\mathcal{S}_i$. We do not need to segment the whole image domain D since segments which have no intersection with Ω do not affect the final result. We derive the sets $\mathcal{S}_i$ from a set of auxiliary functions $(S_i : D \to \mathbb{R})_{i=0..N}$:

1. Set $i = 0$.
2. Choose an arbitrary pixel $(j,k) \in \Omega \setminus \bigcup_{0 \le n < i} \mathcal{S}_n$. Set $S_i = -1$, except for $S_i(j,k) = +1$.
3. Evolve S_i according to the equation

$$\frac{\partial S_i}{\partial t} = \Delta S_i - P'(S_i) - \alpha \|\nabla g(x,y)\| \tag{4}$$

   where $P'(x)$ is the derivative of the polynomial potential

$$P(x) = \frac{9}{4}x^4 + \frac{19}{8}x^3 - \frac{9}{2}x^2 - \frac{57}{8}x \tag{5}$$

   until a steady state is reached.
4. Set $\mathcal{S}_i = \operatorname{supp}\left(\max\left(0, S_i^f\right)\right)$, where S_i^f is the steady state solution from the previous step.
5. If $\bigcup_{0 \le n \le i} \mathcal{S}_n \supseteq \Omega$ terminate the algorithm, else set $i \leftarrow i + 1$ and continue with step 2.

Explanation of equation (4): like in the scalar Ginzburg–Landau equation $P'(x)$ is the derivative of a bistable polynomial potential $P(x)$, forcing $S_i(x)$ to take on values close to $+1$ or -1. Here $P(x)$ is chosen to be nonsymmetric, having a shallow minimum at $x = -1$ and a deep minimum at $x = +1$. Assume $\alpha = 0$.

$P(x)$ is constructed such that the diffusion caused by the Laplacian is strong enough to move S_i in the surrounding of the seeding pixel from the negative to the positive minimum. Thus under continued evolution the $+1$ domain will spread out all over D. If $\alpha > 0$ the term depending on ∇g is a forcing term acting against diffusion and thus eventually stops propagation at edges of g. Eventually further terms could be added, e.g. penalizing curvature of S_i to prevent the segments from crossing small narrow gaps. This turned out not to be required by our application.

Large changes in S_i occur only in a small region between the $+1$ and the -1 domain, so this algorithm can be efficiently implemented using a front tracking method. Therefore only a small fraction of all pixels has to be processed at each iteration.

2.5 Texture Synthesis

For the texture synthesis step we employed the algorithm from [13]. In [22] the same algorithm with a different implementation has been used. In [21] the texture synthesis algorithm from [18] was used, which should give similar synthesis results as [13]. In both [21,22] all of the available image data $t|_{D\setminus\Omega}$ is taken as sample to synthesize texture in all of $t|_\Omega$. Since the texture synthesis proceeds from the border on inwards into the inpainting domain it is evident that texture can be continued without artifacts. A few "perturbing pixels" might suffice though to make the algorithm use texture sample data from unsuitable image locations, see figure 2.

To circumvent the shortcomings of this "global sampling texture synthesis" we introduce "texture synthesis by local sampling": for every segment $\mathcal{S}_i$ we take $\Omega_i^{sample} = \mathcal{S}_i \setminus \Omega$ and $\Omega_i^{synth} = \mathcal{S}_i \cap \Omega$ to be the texture sample region, resp. the texture synthesizing region. Then the texture synthesis algorithm from [13] is applied to each pair $(\Omega_i^{sample}, \Omega_i^{synth})$ individually. Here we tacitly assume that differently textured regions belong to different segments. Two neighboring regions with different textures belong to the same segment if they have similar intensities, due to the initial diffusion filtering. However, our experiments have shown that the impact of a few wrong texture samples is not significant.

The texture synthesis is the most time consuming part of our inpainting algorithm: for every $(j,k) \in \Omega^{synth}$ set $t_{j,k} \leftarrow t_{m,n}$, where $(m,n) \in \Omega^{sample}$ is chosen such that t in a neighborhood of (m,n) most closely resembles t in a neighborhood of (j,k). Finding the most similar neighborhood for every pixel in Ω_{synth} leads to a considerable amount of nearest neighbor searches in a high dimensional vector space. In most of our examples the number of *test vectors* (i.e. the number of pixels in Ω_{sample}) was too small – resp. the dimension of the vector space was too high – for a binary search tree to be effective. The runtime of the texture synthesis using a search tree could not be improved compared to an exhaustive search. See [32] for a efficiency discussion of nearest neighbor search algorithms and the presentation of the algorithm that we used.

2.6 Assembling

In a last step the synthesized textures are added to the geometry inpainting. The final inpainting is blended onto the initial (flawed) image, i.e. $u_{final} = m \cdot (g + t) + (1 - m) \cdot u_{initial}$ where m is the mask function. This is to soften the impact of discontinuities which could arise from texture synthesis.

3 Results and Discussion

The results presented in this section are chosen to highlight various situations an inpainting algorithm has to tackle:

1. In figure 3 the inpainting domain consists of thin and long structures, as occurring in scratch and text removal. This is easier to inpaint than equally sized compact shaped areas, since edges have to be established over short distances only. Additionally, if the size of typical texture features is comparable to the "width" of the inpainting domain, then even inappropriately synthesized texture does not necessarily produce an artifact.
2. In figure 4 inpainting is easy because Ω is surrounded by a single homogeneous region. The PDE inpainting only needs to adapt the appropriate color. An appropriate texture sampling region is easily found.
3. Inpainting is difficult if the object to be removed covers a variety of different regions containing complex textures, which happens mainly in airbrushing applications, see figure 5.

One difficult example is shown in figure 5: the balustrade to be removed covers three adjacent regions with two different textures. The brick wall is considered as two distinct regions, because of the noticeable difference in brightness. Compared to the plain texture synthesis result from figure 2 no improper texture is synthesized into the wall, due to the local sampling. Unfortunately the corners of the building are found to be another segment and the brick pattern on the corners is not synthesized satisfactorily (neither is it in figure 2). More examples can be found in [33].

3.1 Choice of Parameters

Our proposed algorithm contains several numeric parameters that may be tuned: the edge sensitivity λ in the pre–filtering, the edge width ε in the PDE inpainting, an edge sensitivity and a regularization parameter for the post–processing (which have not been explicitly mentioned) and the strength α of the forcing term in the segmentation. Further, for the diffusion equations in the pre– and post–filtering stopping times (resp. stopping cireteria) have to be specified (not for the inpainting and the segmentation, which are evolved to steady state). Moreover, the size and the shapes of the neighbourhood regions in the texture synthesis phase could also be adjusted.

The examples in this paper have been created with a fixed parameter setting that has been tuned on an appropriate training set. We found that the quality of

Fig. 3. This example is easy since the inpainting domain consists of long thin structures only

Fig. 4. This example is easy because the object to be removed is surrounded by a single weakly textured region

the inpainting was increased only marginally if the parameters were fine–tuned for each image separately. Automatic content based parameter determination resulted in better quality only in a few cases but produced serious artifacts more often. Besides, not all parameters may be chosen independently, i.e., if ε is increased then so should be α.

3.2 Future Work

As already pointed out in [21] each separate step of the inpainting algorithm could be performed with several different subalgorithms. Since it is unlikely that one combination performs optimally, it would be desirable to have a criterion for automatically choosing the appropriate algorithms. This criterion would have to include the form of the inpainting domain, image contents, amount of texture and probably more.

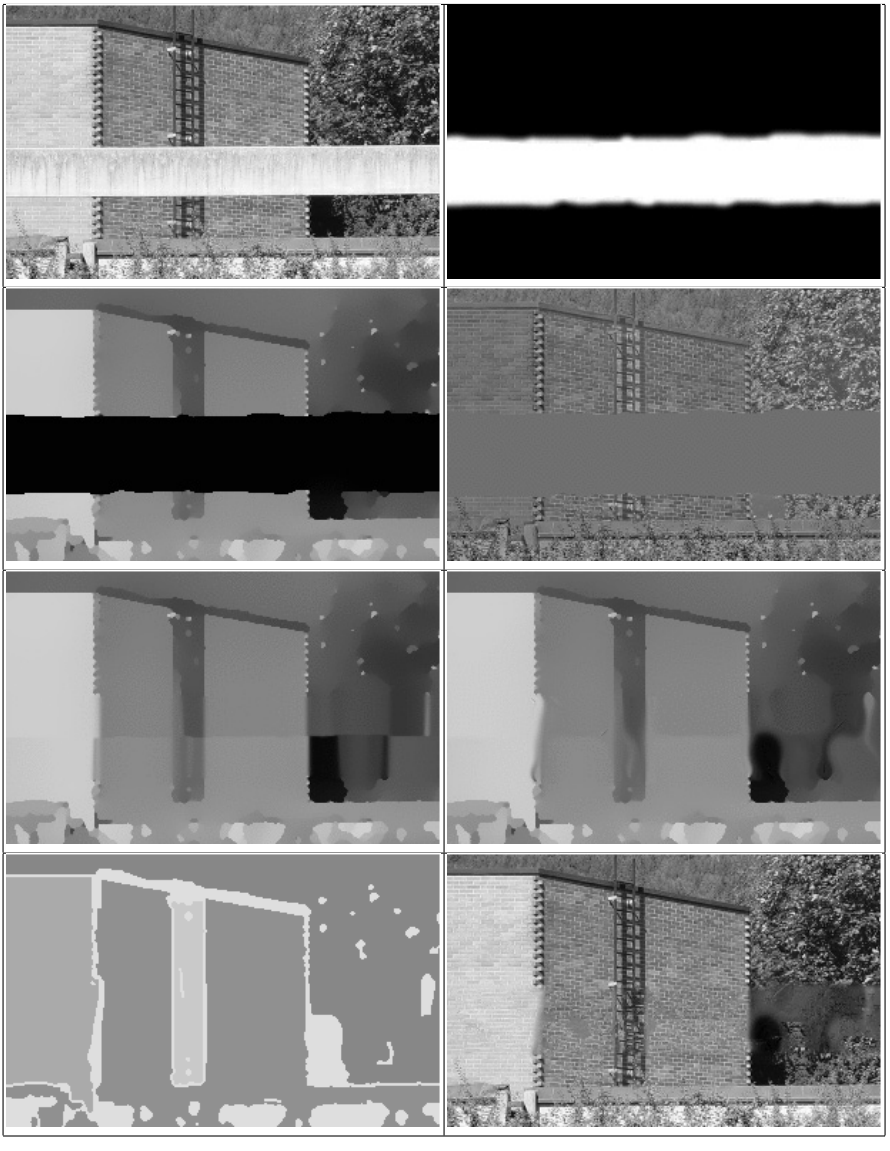

Fig. 5. The airbrushed image during each step of the algorithm. **First row**: the original image and the mask, with the inpainting domain being white. **Second row**: the filtered image and the difference image (i.e. the texture part). **Third row**: the inpainted geometry part before (*left*) and after (*right*) postprocessing. **Fourth row**: result of the segmentation and final result of the complete inpainting algorithm. Note that compared to figure 2 textures are synthesized using only appropriate information from the surrounding region. Unfortunately the salient brick pattern on the edge of the building was not correctly recognized by the segmentation

Acknowledgment. This work is supported by the Austrian Science Fund (FWF), grants Y–123 INF and P15617-N04.

References

1. S. Masnou, *Disocclusion: A Variational Approach Using Level Lines*, IEEE Transactions on Signal Processing, 11(2), February 2002, p.68–76
2. S. Masnou, J.-M. Morel, *Level Lines based Disocclusion*, Proceedings of the 1998 IEEE International Conference on Image Processing, p.259–263
3. C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, J. Verdera, *Filling–In by Joint Interpolation of Vector Fields and Gray Levels*, IEEE Transactions on Signal Processing, 10(8), August 2001, p.1200–1211
4. M. Bertalmio, G. Sapiro, C. Ballester, V. Caselles, *Image inpainting*, Computer Graphics, SIGGRAPH 2000, July 2000
5. M. Bertalmio, A. Bertozzi, G. Sapiro, *Navier–Stokes, Fluid Dynamics, and Image and Video Inpainting*, IEEE CVPR 2001, Hawaii, USA, December 2001
6. T. Chan, J. Shen, *Mathematical Models for Local Nontexture Inpaintings*, SIAM Journal of Applied Mathematics, 62(3), 2002, p.1019–1043
7. T. Chan, J. Shen, *Non–Texture Inpainting by Curvature–Driven Diffusions (CDD)*, Journal of Visual Communication and Image Representation , 12(4), 2001, p.436–449
8. T. Chan, S. Kang, J. Shen, *Euler's Elastica and Curvature Based Inpainting*, SIAM Journal of Applied Mathematics, 63(2), pp. 564–592, 2002
9. S. Esedoglu, J. Shen, *Digital Inpainting Based on the Mumford–Shah–Euler Image Model*, European Journal of Applied Mathematics, 13, pp. 353–370, 2002
10. M. Oliveira, B. Bowen, R. McKenna, Y. Chang, *Fast Digital Inpainting*, Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain, pp. 261–266
11. H. Grossauer, O. Scherzer, *Using the Complex Ginzburg–Landau Equation for Digital Inpainting in 2D and 3D*, Scale Space Methods in Computer Vision, Lecture Notes in Computer Science 2695, Springer
12. H. Igehy and L. Pereira, *Image replacement through texture synthesis*, Proceedings of the 1997 IEEE International Conf. on Image Processing, 1997
13. Li-Yi Wei, M. Levoy, *Fast Texture Synthesis using Tree-structured Vector Quantization*, Proceedings of SIGGRAPH 2000
14. Li-Yi Wei, M. Levoy, *Order-Independent Texture Synthesis*, Technical Report TR-2002-01, Computer Science Department, Stanford University, April, 2002
15. P. Harrison, *A non–hiearachical procedure for re–synthesis of complex textures*, WSCG'2001, pages 190–197, Plzen, Czech Republic, 2001, University of West Bohemia
16. M. Ashikhmin, *Synthesizing Natural Textures*, Proceedings of 2001 ACM Symposium on Interactive 3D Graphics, Research Triangle Park, NorthCarolina March 19-21, pp. 217-226
17. R. Paget, D. Longstaff, *Texture Synthesis via a Nonparametric Markov Random Field*, Proceedings of DICTA-95, Digital Image Computing: Techniques and Applications, 1, pp. 547–552, 6-8th December 1995
18. A. Efros, T. Leung, *Texture Synthesis by Non-parametric Sampling*, IEEE International Conference on Computer Vision (ICCV'99), Corfu, Greece, September 1999

19. A. Hirani, T. Totsuka, *Combining frequency and spatial domain information for fast interactive image noise removal*, ACM SIGGRAPH, pp 269–276, 1996
20. S. Rane, M. Bertalmio, G. Sapiro, *Structure and Texture Filling-in of missing image blocks for Wireless Transmission and Compression*, IEEE Transactions on Image Processing, 12(3), March 2003
21. M. Bertalmio, L. Vese, G. Sapiro, S. Osher, *Simultaneous texture and structure image inpainting*, IEEE Transactions on Image Processing, 12(8), August 2003
22. H. Yamauchi, J. Haber, H.-P. Seidel, *Image Restoration using Multiresolution Texture Synthesis and Image Inpainting*, Proc. Computer Graphics International (CGI) 2003, pp.120-125, 9-11 July, Tokyo, Japan
23. P. Perona, J. Malik, *Scale–Space and Edge Detection Using Anisotropic Diffusion*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.12, No. 7, July 1990
24. L. Vese, S. Osher, *Modeling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing*, UCLA CAM Report 02-19, May 2002, to appear in Journal of Scientific Computing
25. S. Osher, A. Sole, L. Vese, *Image decomposition and restoration using total variation minimization and the H^{-1} norm*, UCLA Computational and Applied Mathematics Reports, October 2002
26. J.-F. Aujol et.al., *Image Decomposition Application to SAR Images*, Scale Space Methods in Computer Vision, Lecture Notes in Computer Science 2695, Springer
27. Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, AMS University Lecture Series 22, 2002
28. F. Bethuel, H. Brezis, F. Helein, *Ginzburg–Landau Vortices*, in Progress in Nonlinear Partial Differential Equations, Vol. 13, Birkhäuser, 1994
29. L. Ambrosio, N. Dancer, *Calculus of Variations and Partial Differential Equations*, Springer, 2000
30. J. Weickert, *Anisotropic Diffusion in Image Processing*, B.G.Teubner, Stuttgart, 1998
31. S. Acton, *Multigrid Anisotropic Diffusion*, IEEE Transactions on Image Processing, 7(3), March 1998
32. S.A. Nene, S.K. Nayar, *A Simple Algorithm for Nearest Neighbor Search in High Dimensions*, IEEE Trans. Pattern Anal. Machine Intell., vol. 19, 989-1003, 1997
33. http://informatik.uibk.ac.at/infmath/