Image Restoration using Multiresolution Texture Synthesis and Image Inpainting

Hitoshi Yamauchi, Jörg Haber, and Hans-Peter Seidel
Max-Planck-Institut für Informatik, Saarbrücken, Germany
E-mail: {hitoshi,haberj,hpseidel}@mpi-sb.mpg.de

Abstract

We present a new method for the restoration of digitized photographs. Restoration in this context refers to removal of image defects such as scratches and blotches as well as to removal of disturbing objects as, for instance, subtitles, logos, wires, and microphones.

Our method combines techniques from texture synthesis and image inpainting, bridging the gap between these two approaches that have recently attracted strong research interest. Combining image inpainting and texture synthesis in a multiresolution approach gives us the best of both worlds and enables us to overcome the limitations of each of those individual approaches.

The restored images obtained with our method look plausible in general and surprisingly good in some cases. This is demonstrated for a variety of input images that exhibit different kinds of defects.

Keywords: *multiresolution texture synthesis, image inpainting, image restoration, frequency decomposition*

1 Introduction and Related Work

Today, digital photo cameras have established themselves on both consumer and professional markets. Apart from the immediate availability of photos for viewing and/or electronic transfer to an editorial office, digital cameras have the big advantage of producing electronic images that can easily be stored and copied without loss of quality for the next decades to come. Although these advantages may sound great, one has to consider that the number of analog camera sales worldwide each year is still a multiple of the number of corresponding digital camera sales: the quality and resolution of analog images is still hard to achieve even for high-end (and high-price) digital cameras.

As a result, the amount of analog images that have to be digitized in order to "live forever" is still growing. In addition, many photographs from the pre-digital era still need to be digitized to prevent them from decay. Unfortunately, this material often exhibits defects such as scratches or blotches. Equally disturbing artifacts are, for instance, subtitles, logos, and physical objects such as wires and microphones, which should be removed from the image. Obviously, it is desirable to remove defects or disturbing objects in a fully automatic way. Such automatism would include detection of the image regions to be repaired as well. Although an automatic detection should be possible for obvious defects, the detection of unwanted objects is a completely subjective process that cannot be performed without user interaction. Due to this restriction, and to avoid going beyond the scope of this paper, we do not address automatic detection here. For the special case of image sequences, detection and restoration of image defects has been addressed in [14].

In this paper we present a new method to automatically repair "damaged" areas of digitized photographs. Our method performs a frequency decomposition of the input image to combine techniques and ideas from two different areas of research: *texture synthesis* and *image inpainting*.

Recently, texture synthesis [9, 19] has become an active area of research. The common idea of all texture synthesis approaches is to create a new texture from a (typically small) initial seed texture such that the appearance (i.e. structure and color) of the synthesized texture resembles the sample texture. Early approaches employed image pyramids and histogram matching to create two- or three-dimensional textures from a 2-D sample image [11], or synthesized textures by sampling successive spatial frequency bands from an input texture, which has been decomposed using a Laplacian pyramid [7]. Texture synthesis based on statistical measurements of the seed image and its wavelet decomposition has been proposed in [17]. The approach presented in [10] is based on the original work in [9] and introduces a scheme to select the order of pixels that are synthesized. Recent publications focus on texture synthesis on surfaces [20, 18, 21], texture synthesis for "natural textures" [2], real-time texture synthesis [15, 22], or on texture transfer [8, 12].

In principle, texture synthesis techniques could be employed to repair digitized photographs, see also Figure 1. Especially if a damaged area needs to be filled with some pattern or structure, texture synthesis does a good job —

input image | texture synthesis [19] | image inpainting [16] | our method

Figure 1. Comparison between texture synthesis, image inpainting, and our method for input images with texture (top row) and with texture and additional intensity gradient (bottom row). Each row left to right: input image (damaged areas are masked out); resulting images from texture synthesis [19], from image inpainting [16], and from our new method.

given that the sample area in the same image is large enough. However, texture synthesis usually fails, if the area to be reconstructed contains an additional color or intensity gradient as it is shown in Figure 1 (bottom row).

Image inpainting techniques are, in a way, complementary to texture synthesis. Pioneered by Bertalmio *et al.* [3], approaches have been presented that propagate information from the surroundings of masked areas into their interior. Unlike texture synthesis, image inpainting handles color/intensity gradients correctly, but fails to reconstruct areas that should contain textures with fine detail. The approach presented in [3] is based on PDEs. Information from the environment of masked areas is propagated along isophotes, which are computed to be perpendicular to the discretized gradient vector for each pixel along the inpainting contour. A 2-D Laplacian is used to estimate color/intensity variation and propagates this variation along the isophote direction. Anisotropic diffusion iterations are run from time to time to smooth the inpainted region. To speed up the computation time of this approach, an isotropic diffusion model with user-defined diffusion barriers has been proposed in [16]. The mathematical background of image inpainting has been thoroughly investigated in [6, 5].

The only approach we are aware of that can handle both texture and intensity variation for image restoration has been presented by Hirani and Totsuka [13]. Their algorithm is based on projections onto convex sets and employs a Fourier transform together with a clipping procedure, which are carried out alternatively for a user-defined number of iterations. In addition to the damaged area, the user must interactively select a sample region, which is needed to repair the damaged area. The sample region is restricted to be a translated version of the defective region w.r.t. its texture. Different intensities of sample and defective region are taken care of automatically.

2 Overview of our Method

As it is mentioned in the introduction, we do not address automatic detection of damaged image regions in this paper for several reasons. Thus, we require the user to provide a binary mask M that identifies the image regions to be reconstructed. We assume this mask to be of equal size as the input image I, i.e. for each pixel in I we have a corresponding binary value in M. Yet, we do not require that the mask is given in pixel precision: damaged areas in I may be covered generously in M. If, however, the masked area in M is much larger than the actual damaged area in I, unnecessary degradation of the resulting image quality may occur.

Our algorithm proceeds as follows (cf. also Figure 2):

1. The input image I is decomposed into a high-frequency part H and a low-frequency part L using a discrete cosine transformation (DCT) (cf. Section 3.1).

2. The fast image inpainting algorithm proposed in [16] is applied to the interior of the masked areas of the low-frequency image L to obtain the inpainted image L^*. During this step, information from the whole input image may be used by the image inpainting algorithm. However, only the pixels inside the masked areas will be modified.

3. The high-frequency image H is decomposed into a Gaussian pyramid with $n+1$ levels H_i $(i = 0, \ldots, n)$. Section 3.2 provides some more details about this step.

4. Starting from the highest level H_n, we apply multiresolution texture synthesis [19] inside the masked areas in H_i $(i = n, \ldots, 0)$:

   4.1. First, a kD-tree for fast nearest neighbor look-up is built [1]. However, the search space for texture synthesis in level H_i does not only contain the non-masked areas of H_i, but additionally includes the corresponding areas of the already synthesized higher levels H_k^* $(k = i+1, \ldots, n)$. To obtain the seed image for the highest level H_n, we simply apply the complementary mask $\overline{M_n}$ to H_n.

   4.2. Texture synthesis is applied inside the masked area of H_i. The *neighborhood vector* (cf. [19]), which is used to perform a look-up in the kD-tree, is composed of the pixel information from the texture synthesis kernel in level H_i **and** of all corresponding kernels from the higher levels H_k^* $(k = i+1, \ldots, n)$. This ensures high coherency among all texture synthesis levels.

   More details about this texture synthesis step are given in Section 3.3.

5. The synthesized high-frequency image H_0^* and the inpainted low-frequency image L^* are summed up to yield I^*, which represents the restored version of the input image I.

Details of our implementation are given in the following Section 3. Figure 3 shows some of the intermediate levels of our algorithm for a sample input image.

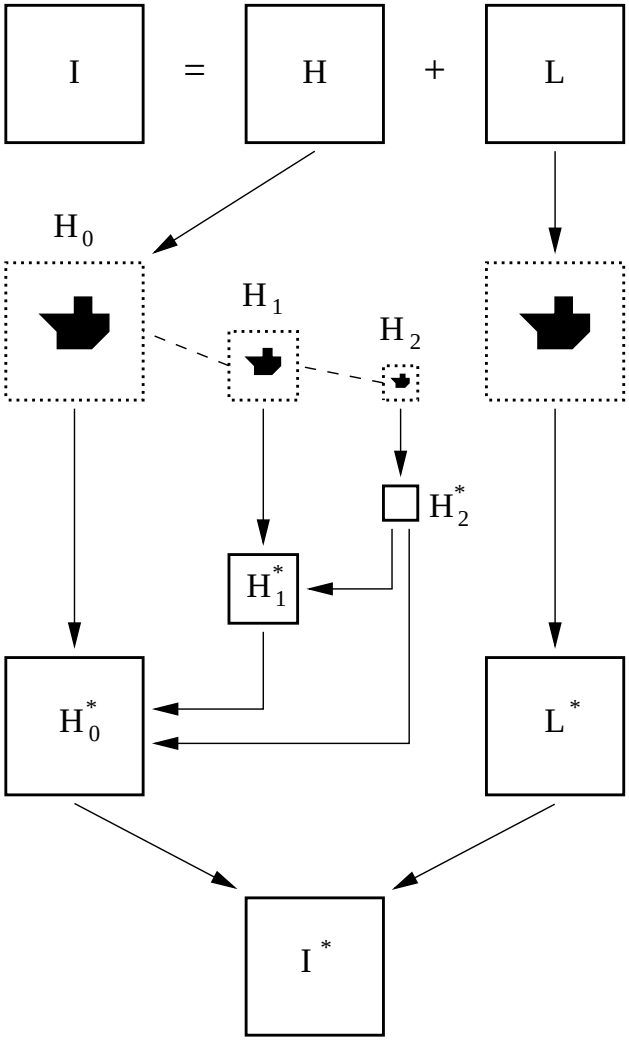

Figure 2. Overview of our method. Top to bottom: the input image I is decomposed into a high-frequency image H and a low-frequency image L using a DCT. Image inpainting is applied to the low-frequency part L to obtain the inpainted image L^*. The high-frequency part H is decomposed into a Gaussian pyramid (shown up to level 2 in this example). Starting from the highest level (H_2), multiresolution texture synthesis is applied to the masked areas of the levels H_i. For each level, the neighborhood vector for the texture synthesis (cf. [19]) is composed of the kernels of that level and of all higher levels. In this way, coherency is maintained throughout all texture synthesis levels. Finally, the resulting high-frequency image H_0^* and the low-frequency image L^* are summed up to yield the restored image I^*.

3 Implementation Details

3.1 Frequency Decomposition

In the first step of our algorithm, the input image I is decomposed into a set of spectral sub-bands using a discrete cosine transform (DCT). Next, we select the first κ sub-bands and compute the inverse DCT of this subset. The resulting image is used as the low-frequency image L_κ.

I L^* H_0, H_1 H_0^*, H_1^*

Figure 3. Multiresolution texture synthesis. Left to right: input image I; inpainted low-frequency image L^*; two levels (H_0, H_1) of the Gaussian decomposition of the high-frequency image H; the same two levels after texture synthesis (H_0^*, H_1^*). The detail images H_i and H_i^* $(i = 0, 1)$ are shown with gamma correction to emphasize the high-frequency detail. The restored image I^* is shown in Figure 1 bottom right.

The corresponding high-frequency image H_κ is obtained by subtraction: $H_\kappa := I - L_\kappa$. Obviously, the parameter κ determines an upper bound for the (low) frequencies that are contained in L_κ. Our goal is, to have as much detail (= high frequencies) as possible in H_κ, while making sure that low-frequency gradients are completely contained in L_κ. To this end, we compute the autocorrelation matrix[1] A_κ of H_κ: $A_\kappa := \text{DCT}^{-1}(\text{DCT}(H_\kappa) \cdot \text{DCT}(H_\kappa))$. For a non-square input image I, we pad H_κ with zeros to obtain a square matrix H'_κ and clip the zeroed border of the resulting $A'_\kappa := H'_\kappa \cdot H'_\kappa$. Next, we compute the standard deviation of the autocorrelation matrix A_κ. In Figure 4, the resulting standard deviations of the autocorrelation matrices of the input images shown in Figure 5 are plotted over the range of $\kappa = 1, \ldots, 16$. We found that choosing the lowest κ value that yields a standard deviation of less than 0.001 gives good results in general.

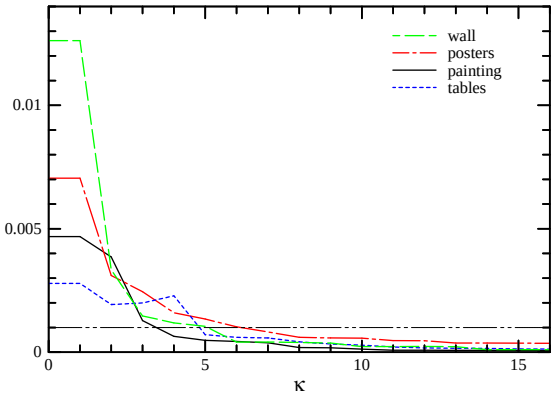

Figure 4. Standard deviations of the autocorrelation matrices A_κ of the input images shown in Figure 5 plotted over the range of $\kappa = 1, \ldots, 16$.

3.2 Gaussian Decomposition

The decomposition of the high-frequency image H into a Gaussian pyramid is based on the approach proposed by Burt and Adelson [4]. In particular, we employ the 5x5 Gaussian kernel ω proposed in [4] with the recommended parameter value $a = 0.4$:

$$\omega(u, v) = \widehat{\omega}(u)\,\widehat{\omega}(v)$$
$$\widehat{\omega}(0) = 0.4,\ \widehat{\omega}(\pm 1) = 0.25,\ \widehat{\omega}(\pm 2) = 0.05$$

The pyramid decomposition proposed in [4] requires that the input image has a resolution of $(p\,2^N + 1) \times (q\,2^N + 1)$ pixels ($p, q, N \in \mathbb{N}$) to ensure that a Gaussian pyramid of $N+1$ levels may be constructed. In our case, we may safely terminate the pyramid decomposition at a level $n \ll N+1$. This can be explained as follows: during the texture synthesis in level i, we include the pixel information from the kernels of all higher levels $i+1, \ldots, n$ into the nearest neighbor search. To be successful, however, the search needs to have enough candidates (contiguous groups of pixels). Thus, the size of the smallest H_n must not be too small. In practice, we obtained good results for pyramid decompositions up to level three. As a consequence, the resolution of the input image I is practically unrestricted for our method.

3.3 Texture Synthesis

For the texture synthesis (Step 4 in Section 2), we implemented and tested the approaches presented in [9], [19],

[1] Actually, the autocorrelation matrix A of a matrix H is defined as $A := \text{DCT}^{-1}(\text{DCT}(H) \cdot \overline{\text{DCT}(H)})$, where $\overline{H}$ denotes the conjugate of H. In our case, the matrices H_κ (and thus also $\text{DCT}(H_\kappa)$) contain real numbers only.

and [2]. Apart from the advantage of speed-up of the algorithm in [19], we found the results of all approaches to be pretty similar. We finally implemented the approach proposed in [12], which basically switches between the texture synthesis algorithms from [19] and [2] from level to level, depending on a local distance criterion. During the texture synthesis in level H_i, we typically used the 5x5 causal kernel from [19] within H_i, a standard 3x3 kernel for level H_{i+1}, and a 1x1 kernel for the higher levels H_k $(k = i+2, \ldots, n)$.

Multiresolution texture synthesis is applied inside the masked areas of each level H_i $(i = n, \ldots, 0)$. The complementary part of H_i (i.e. the part of H_i which is outside the masked areas) is used as the seed image. Since the H_i differ in size from level to level, the mask has to be adapted. Let $M_0 := M$ denote the user-defined binary mask in the size of the input image. We decompose this mask into a Gaussian pyramid up to level n using the same approach and kernel as for the image data (see Section 3.2). This operation is carried out in floating point arithmetic with 1.0 and 0.0 representing true and false for the initial level M_0, respectively. Thus, the higher levels M_i $(i = 1, \ldots, n)$ contain blurry images of the initial mask M. Next, we quantize every M_i back into a binary mask such that 0.0 maps to false and any other value in $]0, 1]$ is mapped to true. Given that the number of levels of the pyramid is typically three or four in our application, the clear distinction between 0.0 and any value larger than zero is not an issue in single precision floating point.

4 Results

Our algorithm is controlled by two different parameters: the number of DCT sub-bands (κ), from which the low-frequency image L_κ is computed (cf. Section 3.1), and the number of levels ($n + 1$) in the Gaussian decomposition of the high-frequency image (cf. Section 2). In practice, we obtained very good results when choosing the lowest κ value that yields a standard deviation of less than 0.001 as described in Section 3.1. Thus, the choice of κ is fully automated in our approach. The optimal number of levels in the Gaussian decomposition is somewhat hard to predict, though. In general, we obtained good results when using three or four levels, i.e. setting $n = 2$ or $n = 3$.

Figure 5 shows some results obtained with our method. Each input image is shown with its mask applied. For the purpose of illustration, the color of each mask has been chosen to differ significantly from the content of the input image. We found that the restored images look plausible in general. In some cases we obtained results that looked surprisingly good. One example is the *table* image (Figure 5, right column), where the flare of the highlight that is reflected from the marble floor is restored very well after the masked tables have been removed. We have not performed numerical comparisons of the results of different image restoration techniques, though. We believe that a simple RMS comparison is useless in the context of image restoration, since it does not take into account relevant perception issues.

In our approach, we apply image inpainting to handle intensity gradients in the input images. During our simulations, we found that multiresolution texture synthesis alone can solve the intensity variation problem to a certain extent. However, the larger and the more irregularly shaped the masked areas are, the more favorably it is to use image inpainting in addition to multiresolution texture synthesis.

Currently, our implementation is rather experimental: no optimizations have been performed, and the timings include gathering of quite a lot of statistical data. All timings were collected on a 1.7 GHz Pentium4 PC and are given for an input image size of 600×450 pixels. The time to restore an image depends heavily on the percentage of the masked pixels. In our simulations, we typically used masks that covered 4–6 % of the input image. For these masks, our algorithm took about 5–10 min to complete (including I/O). The initial fast image inpainting took 4–20 sec, depending on the convergence of the (iterative) inpainting algorithm.

5 Conclusion and Future Work

We have presented a new method that combines texture synthesis and image inpainting in a multiresolution framework for the restoration of digitized photographs. The combination of these two powerful approaches enables us to simultaneously recover texture and color/intensity gradient information. We have demonstrated the quality of the results of our method for a variety of defective input images.

One of the ideas we'd like to put into practice is the applicability of a "fuzzy mask", i.e. a mask where each pixel has a probability between 0.0 and 1.0 instead of a binary true/false. Such masks would be useful to create smooth transitions between the content of the image that is kept and the modified pixels.

In addition, we are planning to investigate a tight coupling of the initial image inpainting and the results from a texture synthesis step. Image inpainting propagates information from the environment of a masked area to the interior. However, the structure inside the masked area is not known. Texture synthesis does not know about the structure either, but it can give some reasonable results if the unknown structure is assumed to be similar to what is still visible in other parts of the input image. Therefore it seems promising to "guide" the initial image inpainting using the result of a previous texture synthesis. In this way, diffusion could proceed along important feature lines, thereby improving the quality of the inpainting step significantly.

painting ($n=2, \kappa=4$) posters ($n=2, \kappa=6$) wall ($n=2, \kappa=5$) tables ($n=3, \kappa=5$)

Figure 5. Top row: input images with masked areas. Bottom row: restored images (see also Section 4). The parameter κ (= number of DCT sub-bands used to compute the low-frequency image L_κ) has been chosen automatically according to our autocorrelation metric (cf. Section 3.1).

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, 46(6):891–923, 1998. Source code available from `http://www.cs.umd.edu/~mount/ANN/`.

[2] M. Ashikhmin. Synthesizing Natural Textures. *2001 ACM Symp. Interactive 3D Graphics*, pages 217–226, 2001.

[3] M. Bertalmio, S. Guillermo, V. Caselles, and C. Ballester. Image Inpainting. In *Computer Graphics (SIGGRAPH '00 Conf. Proc.)*, pages 417–424, July 2000.

[4] P. J. Burt and E. H. Adelson. A Multiresolution Spline with Application to Image Mosaics. *ACM Transactions on Graphics*, 2(4):217–236, Oct. 1983.

[5] T. F. Chan and J. Shen. Non-Texture Inpainting by Curvature-Driven Diffusions (CDD). Technical Report TR 00-35, UCLA CAM, Sept. 2000.

[6] T. F. Chan and J. Shen. Mathematical Models for Local Non-Texture Inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, June 2002.

[7] J. De Bonet. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images. In *Computer Graphics (SIGGRAPH '97 Conf. Proc.)*, pages 361–368, Aug. 1997.

[8] A. A. Efros and W. T. Freeman. Image Quilting for Texture Synthesis and Transfer. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 341–346, Aug. 2001.

[9] A. A. Efros and T. K. Leung. Texture Synthesis by Non-parametric Sampling. In *IEEE International Conf. on Computer Vision*, volume 2, pages 1033–1038, Sept. 1999.

[10] P. Harrison. A Non-Hierarchical Procedure for Re-Synthesis of Complex Textures. In *Proc. of WSCG 2001*, pages 190–197, Feb. 2001.

[11] D. J. Heeger and J. R. Bergen. Pyramid-Based Texture Analysis/Synthesis. In *Computer Graphics (SIGGRAPH '95 Conf. Proc.)*, pages 229–238, Aug. 1995.

[12] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image Analogies. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 327–340, Aug. 2001.

[13] A. N. Hirani and T. Totsuka. Combining Frequency and Spatial Domain Information for Fast Interactive Image Noise Removal. In *Computer Graphics (SIGGRAPH '96 Conf. Proc.)*, pages 269–276, Aug. 1996.

[14] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. W. Rayner. Detection of Missing Data in Image Sequences. *IEEE Trans. Image Processing*, 4(11):1496–1508, Nov. 1995.

[15] L. Liang, C. Liu, Y. Xu, B. Guo, and H.-Y. Shum. Real-Time Texture Synthesis Using Patch-Based Sampling. *ACM Transactions on Graphics*, 20(3):127–150, July 2001.

[16] M. M. Oliveira, B. Bowen, R. McKenna, and Y.-S. Chang. Fast Digital Image Inpainting. In *Proc. of the International Conf. on Visualization, Imaging and Image Processing (VIIP 2001)*, pages 261–266, Sept. 2001.

[17] E. P. Simoncelli and J. Portilla. Texture Characterization via Joint Statistics of Wavelet Coefficient Magnitudes. In *Proc. of Fifth International Conf. on Image Processing*, volume I, pages 62–66. IEEE Computer Society, Oct. 1998.

[18] G. Turk. Texture Synthesis on Surfaces. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 347–354, 2001.

[19] L.-Y. Wei and M. Levoy. Fast Texture Synthesis Using Tree-Structured Vector Quantization. In *Computer Graphics (SIGGRAPH '00 Conf. Proc.)*, pages 479–488, July 2000.

[20] L.-Y. Wei and M. Levoy. Texture Synthesis over Arbitrary Manifold Surfaces. In *Computer Graphics (SIGGRAPH '01 Conf. Proc.)*, pages 355–360, Aug. 2001.

[21] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin. Texture and Shape Synthesis on Surfaces. In *Rendering Techniques 2001*, pages 301–312, 2001.

[22] S. Zelinka and M. Garland. Towards Real-Time Texture Synthesis with the Jump Map. In *Rendering Techniques 2002*, pages 101–107, 2002.