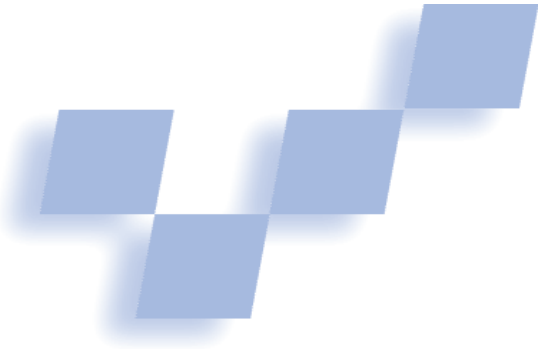


Fast Texture Transfer

Michael Ashikhmin
Stony Brook University

A fast texture transfer technique produces results similar to state-of-the-art methods. This article presents several applications of the method including artistic style transfer, image enhancement, and novel nonphotorealistic filter creation.

In many applications it's useful to have the ability to create a texture of arbitrary size given a small input sample. *Texture synthesis* techniques perform this operation. A *texture transfer* algorithm, on the other hand, takes two images—the source texture and the target image—as input. The algorithm modifies the target image, replacing some high-frequency information with the source texture. Although synthesis and transfer operations share many of the same challenges, there are significant differences. First, a clear criterion of success exists in texture synthesis: the result has to look like the input. For texture transfer, the degree of similarity with the original target image is usually adjusted based on user preferences. The case of artistic style transfer is probably the best illustration of this. The definition of artistic style is subjective; success in attaining this style is a matter of personal preference. Therefore, in a typical texture transfer algorithm application, users would take a trial and error approach and experiment with different parameter values.

This inherent human involvement imposes additional requirements on a texture transfer algorithm. The algorithm must provide a sufficiently rich space of results to explore. This is contrary to texture synthesis where an algorithm that doesn't require any human involvement is a better choice in most cases. Although current texture transfer methods often allow different degrees of similarity between the result and the target image, insufficient computational speed hampers truly exploratory use of these techniques. This is not just an inconvenience—as it would be in texture synthesis—but effectively reduces the space of results to just a few possibilities that the user has the patience to wait for.

This article presents an algorithm for texture transfer between images that is up to several orders of magnitude faster than current state-of-the-art techniques. I will demonstrate how the technique can leverage self-similarity of complex images to increase resolution of some types of images and to create novel, artistic looking images from photographs without any prior artistic source. Compared to other alternatives, methods based on texture transfer are global in the sense that the user need not deal with details such as defining and painting individual brush strokes. Texture transfer methods are also more general since they don't need to emulate any particular artistic style (line drawing, hatching, realistic oil painting, and so on). Not surprisingly, there is a price to pay for this generality—an algorithm designed for a specific artistic style will most likely produce results superior to those presented here for that particular case.

Basic technique

I use the coherent synthesis technique as the basis of my method.[1] Primarily a hands-off texture synthesis algorithm, coherent synthesis works by growing texture patches of irregular size, one pixel at a time. It proceeds in scan line order, choosing the best pixel from a short candidate list. This list is based on locations that already synthesized pixels were taken from. Each already synthesized pixel in a small (typically, L-shaped 5×2.5) neighborhood contributes its appropriately forward-shifted neighbor in the texture image to the list. I can create certain texture transfer examples by extending the notion of the neighborhood to a full square that includes corresponding parts of the target image, as shown in Figure 1.

This technique is fast but has several important limitations when applied to texture transfer. First, dozens of iterations might be necessary to obtain a sufficient similarity between the target image and the result, defeating the algorithm's speed advantages and creating flat images by destroying fine texture features. A simple increase of the target image's weight in the neighborhood norm does not fix this problem since it's mostly due to the small number of candidates examined

 Published by the IEEE Computer Society 0272-1716/03/$17.00 © 2003 IEEE

by the algorithm at each synthesis step. Usually the approach creates just one or two candidates as dictated by growing patches, and the algorithm can't terminate this growth process, potentially, until it runs into the end of the original image. This also can produce noticeable horizontal edges when the growth process finally terminates.

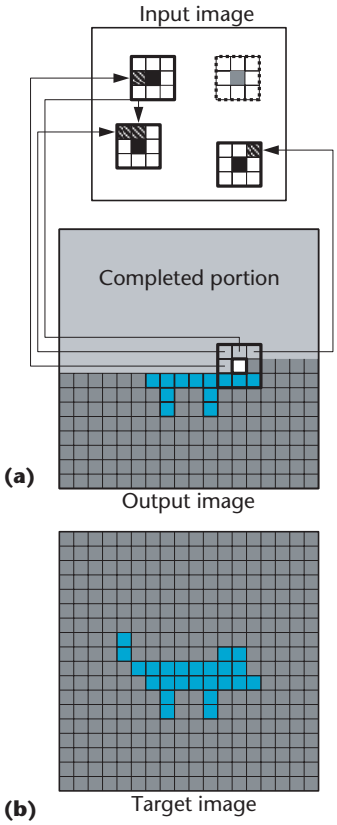
My solution is to increase the algorithm's search space. You must carry this out carefully since the low number of candidates is the primary source of the original method's efficiency and the computational cost can become prohibitive if you need interactive feedback. You must also make sure that visual results are not degraded. An unrestricted search over an entire texture image gives the results found in Wei and Levoy,[2] which are not always superior to coherent synthesis.[1] However, a slight increase in search space dramatically improves the convergence rate of coherent synthesis without compromising visual results.

In particular, after creating a candidate list for a pixel as I've described elsewhere,[1] with some user controlled probability p, I add a single candidate taken from a random location in the texture image. Surprisingly small values around $p = 0.05$—which corresponds to considering (but not necessarily accepting), on average, one extra candidate per 20 synthesized pixels—can make big difference in terms of the quality of the synthesized image. This procedure also reduces the amount of perceived boundary discontinuity since the smaller patches produced generally fit together better. This obtains artistic filtering results similar to image analogies[3] but at a fraction of the cost while using only the simpler-to-implement coherent synthesis strategy. It also solves the problem of horizontal edges mentioned previously, since it becomes unlikely that the algorithm does not terminate patch growth before running into the bottom image boundary. Of course, multiple iterations can still occur[1,4] but a single iteration is sufficient in most cases. Better results could potentially occur by adding a specially chosen extra candidate instead of a purely random one, as suggested by Zelinka and Garland.[5] Because this would require costly preprocessing of the source image, I use the simpler and faster random choice. The proposed extension will not, in most cases, benefit texture synthesis since smaller patches will fail to capture large-scale features of the underlying texture.

A second useful modification is problem-specific image similarity metrics. The original neighborhood difference measure D for the case of texture transfer is the pixelwise L_2 difference between a neighborhood in the source texture and a combined neighborhood made from two L-shaped parts put together[1] (see Figure 1):

$$D^2(N_r, N_s) = D^2(N_{Lr}, N_{Ls}) + D^2(N_{Lt}, N_{Ls}) \quad (1)$$

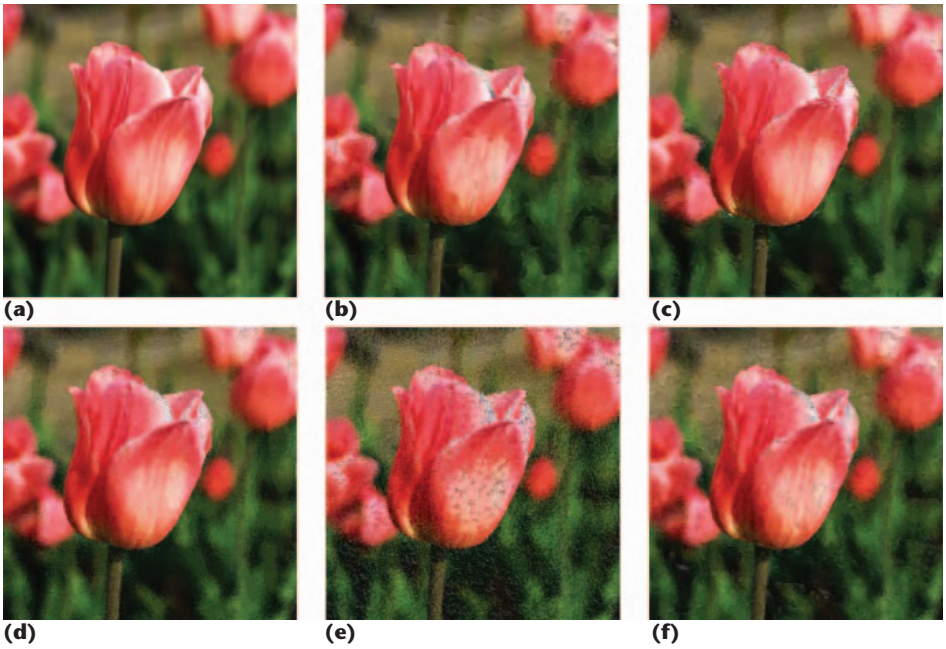
where indexes refer to source texture (s), target (t), and result (r) images, and subscript L in N_L indicates an L-shaped neighborhood. The two parts of Equation 1 can be arbitrarily weighted with respect to each other. In particular, it's beneficial to use only the second part of Equation 1—that is, the difference with the target—as the measure of neighborhood similarity for some of the applications I discuss in later sections. Visual patch coherence is sufficiently maintained in this case by the appropriate choice of candidates, which favors patch growing by itself, so additional coherence enforcement by the difference measure is not necessary. This modification can significantly improve the results of the basic coherent synthesis technique for these applications. Since neighborhood comparison is a part of the inner loop of the algorithm, I've also effectively reduced computational load by a factor of two. For artistic style transfer I will completely replace Equation 1 by a different measure. Along with the search space extension, these changes enable effective use of the algorithm in several applications.



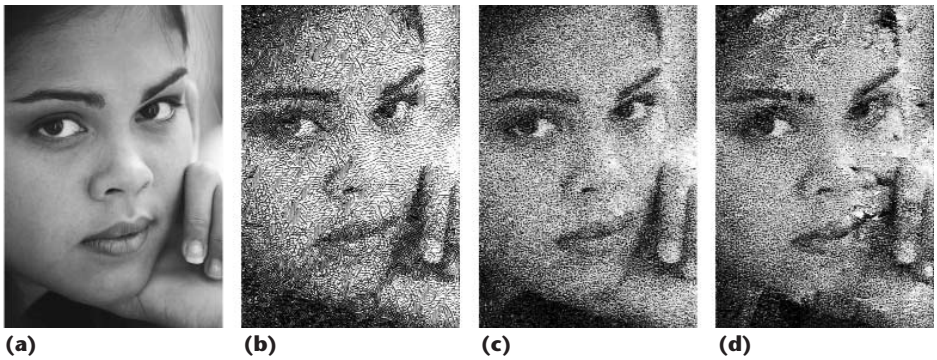
1 (a) Each pixel in this L-shaped neighborhood generates a shifted candidate pixel (black) according to its original position in the input image (hatched). A single random candidate (light blue with dashed lines) is added with probability *p*. The candidate whose neighborhood best matches the one in the output image according to an application-specific similarity metric is chosen as the next pixel value. In the original algorithm, the complete neighborhood for matching is composed from two L-shaped halves, with top half coming from the already synthesized part of the output image and (b) the bottom half from the target image.

Applications

All examples presented here use 5×5 neighborhoods. Similarly to Hertzmann et al.,[3] I use luminance values only during synthesis and keep color components of the target image unchanged, although no distribution equalization of target and texture images is necessary. I measured runtimes on a 1.2-GHz Pentium IV PC; they are all less than a few seconds. This time is sufficient to provide interactive feedback for the user and allow fast exploration of a rich result space by adjusting algorithm parameters such as extra random candidate probability p and an extra location-specific candidate probability. A single iteration of the algorithm obtained all results. Original images are available at http://www.cs.sunysb.edu/~ash/ftt/. The



2 Varying the algorithm's parameters can obtain different artistic styles: (a) original image, $m = 1$, $p = 0.1$, $p_l = 0$; (b) $m = 2$, $p = 0$, $p_l = 0.03$; (c) $m = 4$, $p = 1.0$, $p_l = 0.1$; (d) $m = 8$, $p = 0.5$, $p_l = 0.01$; (e) $m = 2$, $p = 0.2$, $p_l = 0$. Image size is 430×430 pixels. No additional artistic samples were used.



3 (a) Style transfer results for a photograph using (b) hatching and (c) charcoal drawing. Extra candidate probability p values: 0.2, 0.5. Total times: 0.6 and 0.9 second. (d) Charcoal example after six iterations of coherent synthesis. Total time: 3.5 seconds. All images are 366×554 pixels.

reader is encouraged to examine them directly since differences among some images shown are quite subtle. This is especially true for Figure 2.

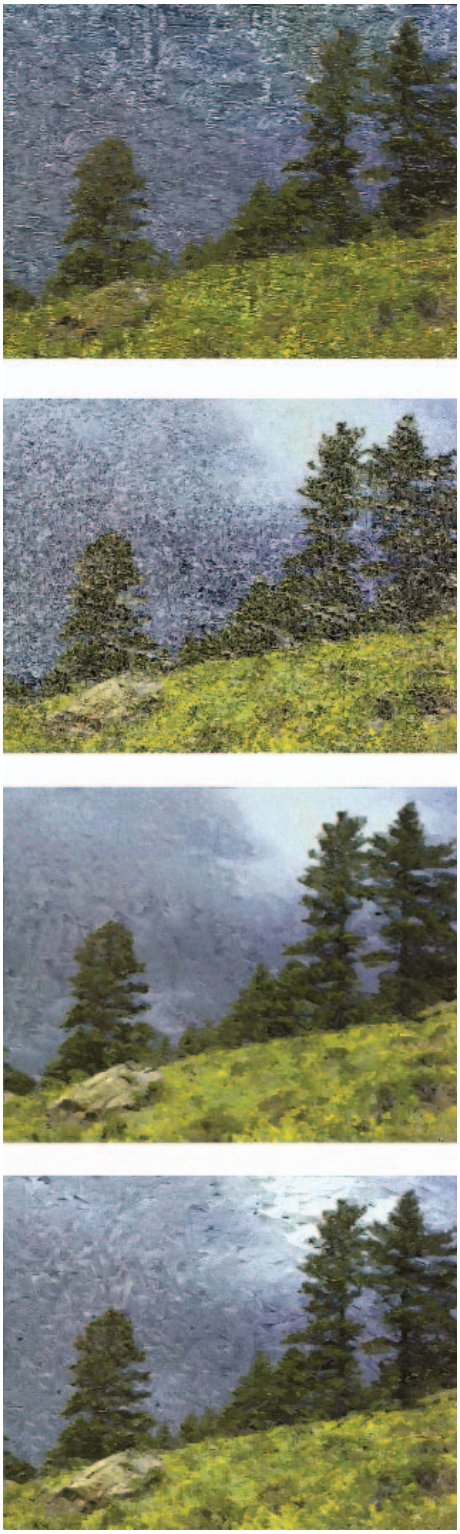
Artistic style transfer

Texture transfer applications dealing with artistic styles will probably benefit the most from a fast algorithm. These applications are inherently user-oriented and interactive feedback is therefore highly desirable. In artistic style transfer I set an artistic drawing or painting as the source texture and a photograph as the target image. (I don't use extra correspondence maps.) I modified the neighborhood difference metric to find the best candidate from the now extended list based on developments presented in Hertzmann et al. and in Efros and Freeman.[3,4] Both of these papers found blurred versions of images useful for artistic style transfer. This corresponds to the general idea of preserving large features of the photograph but changing high-frequency details to match those of the artistic image. I therefore define the measure of neighborhood difference as the sum of two parts: the difference of neighborhood averages between the source and the target, and the L_2 pixelwise difference of only high-frequency components in the L-shaped neighborhoods of the result and the source texture image:

$$D^2\left(N_r, N_s\right) = w\left(\overline{N_s} - \overline{N_t}\right)^2 + \left(1/n\right) D^2\left(N_{Lr} - \overline{N_{Lr}}, N_{Ls} - \overline{N_{Ls}}\right)$$

Here $\overline{N}$ denotes corresponding neighborhood averages and n is the number of pixels used to compute the L_2 norm. To make this definition work properly, I initialize the result by copying the target image into it. The user can adjust the weighting w of the first part and extra candidate probability p based on a subjective notion of style; for all examples shown in this article, $w = 1$.

Figures 3 and 4 present the results of this process for several images and reflect my subjective judgment choices. Figure 5 shows the original photograph. To facilitate comparison with previous results, I use the same examples used in Hertzmann et al.,[3] even though some present rather difficult cases for this algorithm (such as the hatching example in Figure 3b for which using directional information would have been beneficial). Results obtained using an image analogies technique are available at http://www.mrl.nyu.edu/projects/image-analogies/. Figure 3d shows an image obtained with the original algorithm, which still is not completely converged even after six iterations.[1]



4 (a) Artistic style transfer results for Van Gogh (extra candidate probability $p = 0.04$), (b) pointillist (0.15), (c) watercolor (0.15), (d) Manet (0.08). Running times were 0.9, 0.95, 1.0, and 0.95 seconds respectively. All images are 640 × 500 pixels.

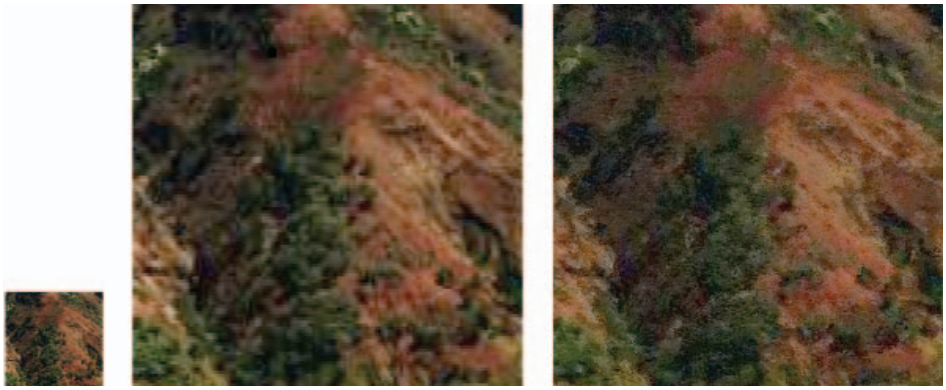
Artistic image examples can be found at http://www.cs.sunysb.edu/~ash/ftt. A pointillist painting example demonstrates limitations of using just a luminance channel for synthesis. Overall results are comparable to those presented in Hertzmann et al.[3] and Efros and Freeman,[4] with runtimes around 1 or 2 seconds compared to dozens of minutes for the image analogies technique, and about a minute for the image quilting technique (numbers for alternative techniques are rough estimates obtained by scaling those provided by the authors). Both of these techniques are significantly more complex to implement than the algorithm discussed here. In the case of line art drawings, my results don't have long, well-separated lines, but they are still visually pleasing. Sticking to the choice of simplicity and computational efficiency, I did not include oriented filter response in the neighborhood difference measure as suggested by Hertzmann et al.[3] This feature can be easily added if necessary. It's also easy to restrict the area of the artistic image used in the transfer process to avoid effects similar to somewhat strange looking vertically striped areas in the hatched image, which are due to transfer from the corresponding area in the original artistic sample. I did not perform this extra intervention for any of presented examples.

5 Original photograph used in artistic style transfer examples shown in Figure 4. (Courtesy of John Shaw)

Super resolution

Texture transfer methods can artificially increase resolution of some types of images. It's well known that visually complex images contain a significant amount of details on many scales. Moreover, some images have self-similar appearance, which for my purposes means that a version of an image scaled to some different resolution looks similar to the original. (Some authors[6] use the term *self-similarity* to describe similarity of texture appearance across the image—a property more commonly referred to as *stationarity*.)

Typical examples of such self-similar behavior include various images of natural fractal objects such as clouds, waves, woods, or entire terrains. You can leverage this property to increase an image's resolution by applying my texture transfer algorithm. First, prepare a high-resolution version of the image by magnifying the original using standard methods—for example, by applying a cubic or a box filter. Then set this image as the target and use the original (or its down-sampled version) as the source texture. Since the desired result should look similar to the initial image, we can infuse high-frequency details by taking them from the original. Simple magnification, on the other hand, will produce a noticeably



6 Super-resolution example. (a) Original image from VisTex (http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html) texture database, (b) magnification by a factor of $m = 4$ using a box filter, and (c) using texture transfer where $p = 0$; $p_l = 0.03$—image in (b) acted as the target. In this case, using only location-specific candidates was sufficient to obtain good enough results.

Related Work

Extensive review of research on either nonphotorealistic rendering or textures and texture synthesis is beyond the scope of this article. More detail is available elsewhere.[1,2] Here I mention only the work most directly related to my research. Recently, significant advances have occurred in texture synthesis[3-9] and the ability of several such algorithms to perform texture transfer has been demonstrated,[2,7,8,10] including examples of artistic style transfer. However, with the exception of image analogies,[2] the primary goal of these methods is texture synthesis rather than transfer, and they usually do not address specific requirements mentioned in the main article text. Combining two texture synthesis algorithms, the image analogies method produces impressive results but requires up to several hours of computations. Even with potential fivefold speed increase with optimized implementation suggested by Hertzmann et al.,[2] this is not sufficient for fast exploration of the result space. Another restriction of the technique is that the user must provide additional unfiltered maps restricted to having exact pointwise correspondence with the original image.

Coherent synthesis provides interactive feedback to the user but converges to the result too slowly, requiring many dozens of iterations for examples similar to those presented in this article.[7] Nevertheless, this algorithm is currently the simplest and fastest among those with demonstrated texture transfer abilities and I use it as the basis for my method. Even faster and simpler algorithms based on image mosaics are available[6] but texture transfer has yet to be demonstrated using these methods. The image quilting algorithm[4] probably gives the best synthesis results overall, but when applied to texture transfer, is slower and more complex than coherent synthesis and requires multiple iterations (although generally a smaller number than with coherent synthesis). Similarly to image analogies, this algorithm uses special correspondence maps during the transfer process.

References

1. B. Gooch and A. Gooch, *Nonphotorealistic Rendering*, A.K. Peters, 2001.
2. A. Hertzmann et al., "Image Analogies," *Proc. Siggraph,* ACM Press, 2001, pp. 327-340, http://mrl.nyu.edu/publications/image-analogies/.
3. A. Efros and T.K. Leung, "Texture Synthesis by Nonparametric Sampling," *IEEE Int'l Conf. Computer Vision*, IEEE CS Press, 1999, pp. 1033-1038.
4. L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. Siggraph*, ACM Press, 2000, pp. 379-488.
5. J. Portilla and A. Simoncelli, "A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients," *Int'l J. Computer Vision*, vol. 40, no. 1, 2000, pp. 49-71.
6. L. Liang, et al., *Real-Time Texture Synthesis by Patch-Based Sampling*, tech. rep. MSR-TR-2001-40, Microsoft, Mar. 2001.
7. M. Ashikhmin, "Synthesizing Natural Textures," *Proc. ACM Symp. Interactive 3D Graphics*, ACM Press, 2001, pp. 217-226.
8. A. Efros and W. Freeman, "Image Quilting for Texture Synthesis and Transfer, *Proc. Siggraph*, ACM Press, 2001, pp. 341-346.
9. S. Zelinka and M. Garland, "Towards Real-Time Texture Synthesis with the Jump Map," *Proc. 13th Eurographics Workshop on Rendering*, Eurographics Association, 2002, pp. 101-107.
10. P. Harrison, "A Non-Hierarchical Procedure for Resynthesis of Complex Textures," *WSCG Winter School of Computer Graphics Conf. Proc. (WSCG)*, Univ. of West Bohemia, 2001, pp. 190-197.

blurry version for these types of images, but is necessary to guide the positioning of large-scale features in the result. Contrary to the super-resolution procedure where high-resolution cutouts were used,[3] I use only a single image of a given initial resolution.

To help the algorithm converge faster and to increase its result space, you can also add extra candidates not only from random positions in the source texture as before but also from positions given by dividing current (x, y) offset in the result image by the magnification factor m. This is again performed with some user controlled probability p_l. Figure 6 shows results for a typical fractal aerial photograph.

Novel artistic styles

For general rather than fractal images, the strategy presented in the previous section would not give a good super resolution appearance due to the limited, if any, self-similarity in this case. However, I still found the strategy useful to create artistic looking versions of an image.

Depending on the probabilities of the extra candidates (both random p and location-specific p_l), resolution ratio m of target images, and the source texture, you can obtain a variety of interesting styles. Figure 2 shows several examples, each created in less than half a second (runtime, not user time spent experimenting with parameters) from an image with an original resolution of 430×430 pixels. I did not use a special artistic source to create any of these results.

Conclusion

My method carefully extends the search space of the coherent synthesis in a way that doesn't carry a great computational penalty. The technique is compatible with the general spirit of developing simple methods well suited to a particular problem at hand, rather than attempting to create a single technique (that is, with a fixed image metric) suitable for most texture transfer tasks.

Recently developed simple and fast texture synthesis and transfer techniques are being used in a number of computer graphics applications. The texture transfer method presented here can provide significant value for those applications requiring greater flexibility and speed rather than hands-off operation. It would be also interesting to investigate theoretical foundations of neighborhood-based texture synthesis and transfer algorithms as well as broader applications of such methods in other areas of computer graphics. Possibly transferring them into a general modeling and rendering technique for complex systems. ■

References

1. M. Ashikhmin, "Synthesizing Natural Textures," *Proc. ACM Symp. Interactive 3D Graphics*, ACM Press, 2001, pp. 217-226.
2. L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. Siggraph*, ACM Press, 2000, pp. 379-488.
3. A. Hertzmann et al., "Image Analogies," *Proc. Siggraph,* ACM Press, 2001, pp. 327-340, http://mrl.nyu.edu/publications/image-analogies/.
4. A. Efros and W. Freeman, "Image Quilting for Texture Synthesis and Transfer, *Proc. Siggraph*, ACM Press, 2001, pp. 341-346.
5. S. Zelinka and M. Garland, "Towards Real-Time Texture Synthesis with the Jump Map," *Proc. 13th Eurographics Workshop on Rendering*, Eurographics Association, 2002, pp. 101-107.
6. S. Brooks and N. Dodgson, "Self-Similarity Based Texture Editing," *Proc. Siggraph*, ACM Press, 2002, pp. 653-656.

Michael Ashikhmin *is an assistant professor at the Center for Visual Computing, Computer Science Department at Stony Brook University. His research interests include realistic and nonphotorealistic computer graphics, animation, and visual perception. Ashikhmin received an MS in physics from the Moscow Institute of Physics and Technology, an MS in chemistry from the University of California, Berkeley, and a PhD in computer science from the University of Utah.*

Readers may contact Michael Ashikhmin at the Computer Science Dept., Stony Brook Univ., Stony Brook, NY 11794; ash@cs.sunysb.edu.

For further information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.