BRIDGES
Mathematical Connections
in Art, Music, and Science

Combining Abstract Images using Texture Transfer

Gary R. Greenfield
Department of Mathematics & Computer Science
University of Richmond
Richmond, VA 23173, U.S.A.
ggreenfi@richmond.edu

Abstract

We present a new method for combining gray scale images to achieve an aesthetic composite. Our method treats one image as the source, the other as the destination, and invokes a fast texture transfer algorithm to replace colors in the destination with colors from the source in a manner dependent on the local structure of the two images. The stochastic nature of the algorithm allows for many variants of the composite to be examined. Our test suite consists of nine abstract images selected from images previously obtained using co-evolutionary simulated evolution for aesthetic purposes. We present and discuss several examples and indicate directions for future work.

1. Introduction

For computer generated art works in general, and evolutionary art works in particular, it is often desirable to combine characteristics from one or more images in order to achieve enhanced aesthetic content. The traditional way of doing this is by alpha blending (portions) of images to obtain a composite image. Alpha blending gives a very formal looking collage in the sense that it combines images by varying the degree of transparency. We investigate the use of texture transfer to combine the thematic content of two abstract gray scale images with the intent of producing an aesthetic composite image with identifiable characteristics of both original images but without the transparency effect. Since texture transfer is a re-targeting method, dependent on a source image and a destination image, we introduce a weighting scheme to bias in favor of one image over the other. Because our texture transfer method is fast and incorporates stochastic elements it is especially well-suited for generating a sequence of variations admitting a range of aesthetic possibilities.

This paper is organized as follows. In section two we develop our texture transfer technique. In section three we present the test suite of images from prior work where we evolved abstract gray scale images using a co-evolutionary simulation modelled after hosts and parasites. In section four we present and discuss some of the transfer results we obtained. In section five we give our conclusions and consider directions for future work.

2. Texture Transfer Methodology

Given a texture sample from say a scanned image, the texture *synthesis* problem is the problem of generating additional texture material based on the properties of the sample. The texture synthesis problem has received considerable attention in the computer graphics literature because of its importance in computer gaming, web design, and computer animation. An ambitious theoretical treatment based on "analogies" is found in Hertzmann et al [9]. More focused approaches are to be found in Ashikhmin [1], De Bonet [3], Effros [4] [5], Heeger [8], and Levoy [11]. This is not an exhaustive list of the contributions made to the texture synthesis problem. The texture *transfer* problem, on the other hand, falls within the realm of the more general image re-targeting problem which asks: Given a source image S and a destination image D how can we transfer attribute A from S to D. Besides texture, attribute A could be color, luminance, or some more ethereal aspect of style. The texture transfer technique we describe in this section closely follows the method outlined by Ashikhmin [2]. The key difference is that we assign an integer attribute index $I = I(p)$ to each pixel p of an image and use this index in our computations. For our gray scale abstract images, I is the color look-up table index of the pixel. Thus $0 \leq I \leq 255$, and a pixel with index I has RGB color vector $(I/255, I/255, I/255)$.

To transfer texture from source image S to pixel p of destination image D, consider a candidate transfer pixel t from the source. Both t and p determine 3×3 pixel neighborhoods in their respective images as follows:

s_1	s_2	s_3
s_4	t	s_6
s_7	s_8	s_9

d_1	d_2	d_3
d_4	p	d_6
d_7	d_8	d_9

The criterion for accepting t as the transfer pixel is that it minimizes the function

$$f(t) = 2(I(t) - I(p))^2 + \Sigma_i\, w_i(I(s_i) - I(d_i))^2,$$

where the weight w_i is one, if $i \leq 4$, and two otherwise. To make texture transfer effectively computable as part of a scan line algorithm, consider the *template* obtained by gluing the L-shaped halves of the two neighborhoods together to yield,

s_1	s_2	s_3
s_4	p	d_6
d_7	d_8	d_9

This template indicates that texture transfer to the 3×3 pixel neighborhood of p in D has progressed to the stage where destination pixels d_1, d_2, d_3, d_4 have already been replaced by source pixels s_1, s_2, s_3, s_4 and only now is it p's turn. Each previously transferred s_i defines a source neighborhood whose central pixel t_i is eligible as a transfer candidate to replace p. The subtlety here is the relative position of s_i within the template must be taken into consideration. For example, the source neighborhoods used for s_2 and s_3 must be defined so that the transfer candidates t_2 and t_3 from the source will appear as follows:

*	s_2	*
*	t_2	*
*	*	*

*	*	s_3
*	t_3	*
*	*	*

To make transfer a stochastic process, five percent of the time one additional randomly chosen source pixel will be eligible as a transfer candidate. To calculate $f(t)$ for such a pixel, we use the source neighborhood that has it as the central pixel. This means the $f(t)$ calculation will need to be made at most five times per destination pixel. We initialize transfer at boundary pixels of the destination image by choosing the best index match from among a random sample of five source image pixels. We ignore those source pixels that do not properly define 3×3 neighborhoods when doing the $f(t)$ calculation. We chose the weights to bias in favor of the integrity of the destination image in order to compensate for the fact that texture transfer, at least for gray scale images, means outright pixel replacement.

3. Suite of Test Images

A full description of the co-evolutionary simulation we used to generate our test suite of abstract imagery can be found in [6] and [7]. Here, it will suffice to recall that our simulation relies on the method of evolving expressions first introduced by Sims [10]. This method generates an image from an expression tree — a tree consisting of nodes that contain functions in two variables. To assign an index to each pixel, its coordinates are passed to the expression tree as inputs and the corresponding output is scaled to an integer I between 0 and 255 which serves as the gray scale index for the RGB color $(I/255, I/255, I/255)$. Aesthetic fitness of images is determined by passing digital convolution filters over fixed portions of the image and maximizing the deviation between the original image and the convolved image. This controls the way entropy is introduced into the image population. Since both filters and expressions evolve over the course of the simulation, the simulation is co-evolutionary. We selected nine images for our test suite from a pool of several hundred that we evolved over the course of a multi-year period. They were chosen to reflect the diversity within the pool. They are shown in Figure 1.

4. Transfer Results and Discussion

We use the notation $\#S \rightarrow \#D$ to indicate that we chose test suite image $\#S$ as the source and test suite image $\#D$ as the destination. Taking advantage of the stochastic nature of our texture transfer method, our transfer program generated a sequence of ten variations of the texture transfer for each source and destination selected. All transfers were done using 200×200 pixel resolutions of the test images. Texture transfer takes only a few seconds for images of this size. As was to be expected, some transfers were more successful than others. Figure 2 shows examples of the $\#3 \rightarrow \#2$ transfer, while Figure 3 shows examples of the $\#2 \rightarrow \#3$ transfer. Most transfers worked better in one direction than the other. For example, we feel the $\#7 \rightarrow \#0$ variations of Figure 4 are superior to the best transfer results from the $\#0 \rightarrow \#7$ transfer shown in Figure 5.

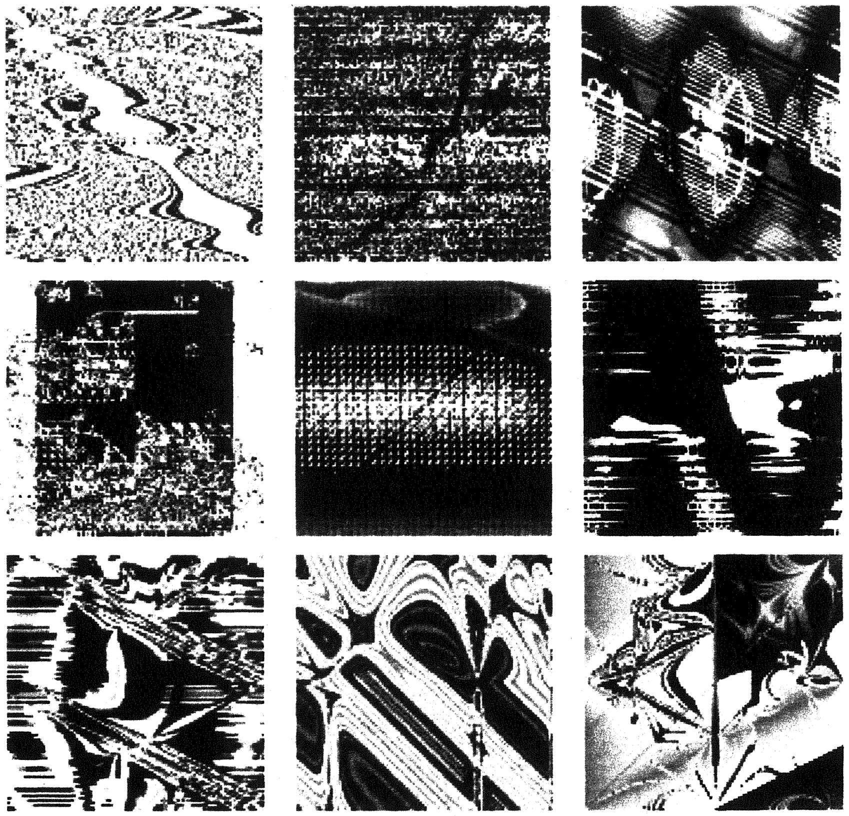

Figure 1: *Test suite of images labelled top-to-bottom left-to-right as #0, #1, ...,#8.*

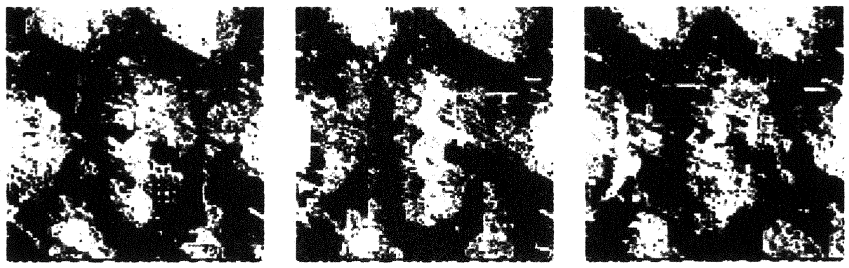

Figure 2: *Three examples of the* $\#3 \rightarrow \#2$ *transfer.*

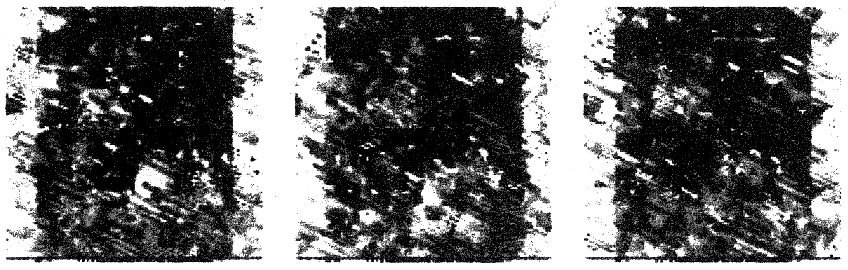

Figure 3: *Three examples of the #2 → #3 transfer.*

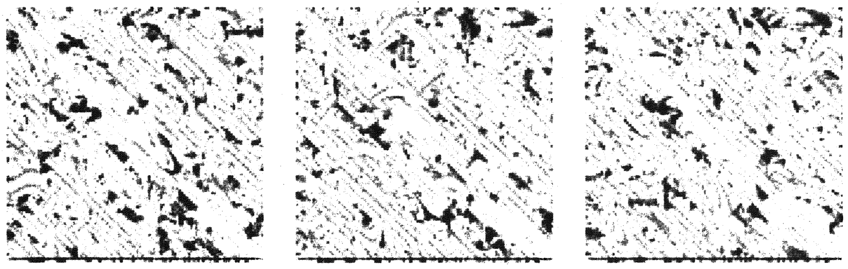

Figure 4: *Three examples of the #7 → #0 transfer.*

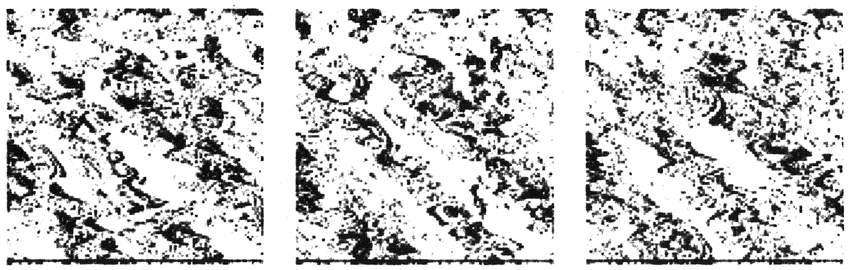

Figure 5: *Three examples of the #0 → #7 transfer.*

In Figure 6 we show the results from the #6 → #4 transfer, and in Figure 7 the results from the #4 → #2 transfer. In these two cases transfers in the opposite direction were usually unsuccessful. In Figure 8 we the show the best aesthetic result we obtained from three other transfers. The rightmost image of Figure 8 is a rare example where test image #8 was used successfully as a source. We were never able to use it successfully as a destination. Perhaps this reflects the fact that it would be better classified as a composition than a texture. Similarly, we were never successfully able to use test image #1 as a source. Perhaps this reflects the fact that the aesthetics of this image depend heavily on its randomness characteristics. It therefore came as a surprise that the #8 → #1 transfer not only gave unexpected results but some of our most stunning images. Three examples from this transfer are shown in Figure 9. From an artistic point of view, it was gratifying to find that the results of texture transfer were not entirely predictable. On the other hand, the #7 → #5 texture transfer examples shown in Figure 10 are almost precisely what one would expect. From a technical point of view, it was gratifying that for the most part texture transfer results seemed "reasonable."

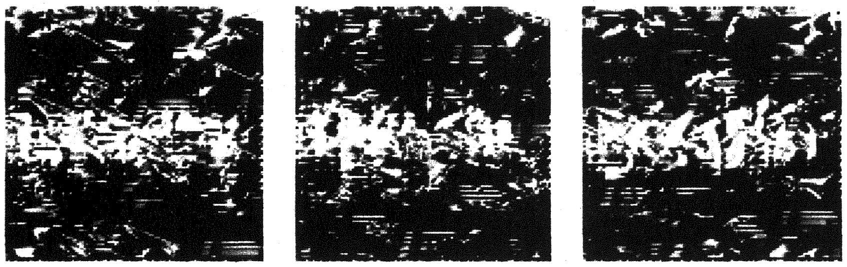

Figure 6: *Three examples of the #6 → #4 transfer.*

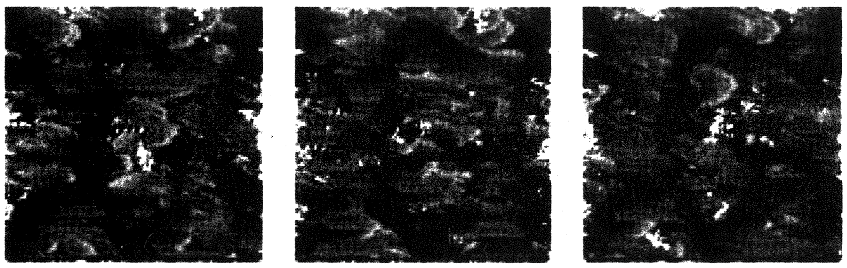

Figure 7: *Three examples of the #4 → #2 transfer.*

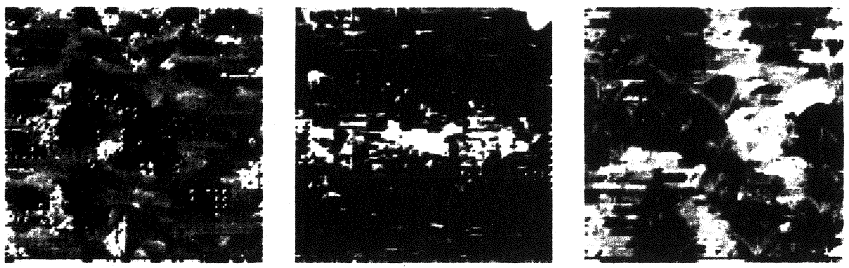

Figure 8: *The best result from the #4 → #6, #5 → #4, and #8 → #5 transfers.*

For completeness, in Figure 11 we show an example of a texture transfer from a test suite image to itself (here #5 → #5) as an aid to understanding to what extent the stochastic nature of the transfer algorithm plays a role. In Figure 12 we show examples from a #3 → #2 texture transfer where the source weights and destination weights were exchanged. By comparing Figure 12 with Figure 2, one can see how weights were used to bias in favor of the integrity of the destination image.

Figure 9: *Three examples of the #8 → #1 transfer. This transfer produced the most unexpected results.*

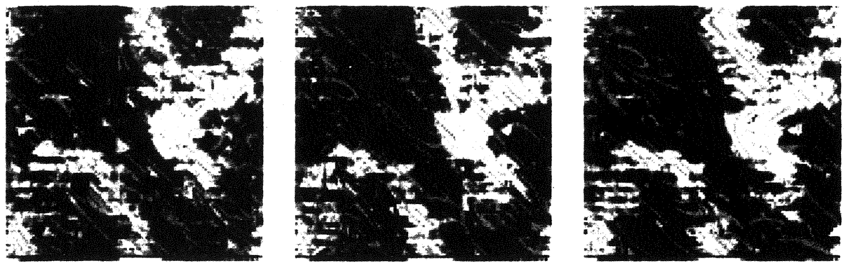

Figure 10: *Three examples of the #7 → #5 transfer. This transfer is arguably the one for which it easiest to infer what the source and destination images are.*

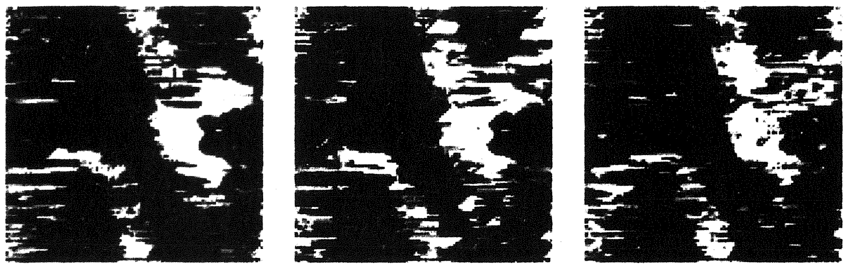

Figure 11: *Three examples of texture transfer from test image #5 to itself.*

5. Conclusions and Future Work

We have described a texture transfer technique used to provide a new way to "composite" gray scale images. Our examples were selected from a test suite of evolved abstract gray scale images. Future work should proceed in two directions. First, as noted by Ashikhmin [2], the weights used in the selection function $f(t)$ deserve further investigation. Second, gray scale texture transfers (appropriate here because of the display requirements) are only the first step toward the more ambitious goal of developing a similar texture transfer recombination technique for color images. The key idea is to use indices from a look-up table derived from a carefully chosen space-filling curve through HSV color space to replace the gray scale indices used here.

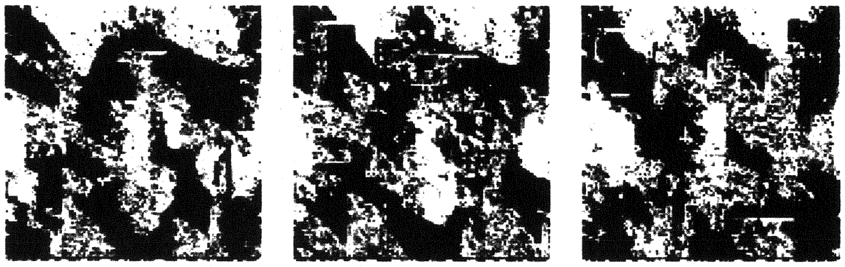

Figure 12: *Three examples of the #3 → #2 transfer transfer with source and destination weights reversed.*

References

[1] Ashikhmin, M., Synthesizing natural textures, *Proceedings ACM Symposium on 3D Graphics*, ACM Press, 2001, 217–226.

[2] Ashikhmin, M., Fast texture transfer, *IEEE Computer Graphics and Applications*, July/August 2003, 38–43.

[3] De Bonet, J., Multiresolution sampling procedure for anlaysis and synthesis of texture images, *Siggraph 1997 Conference Proceedings*, ACM press, 1997, 361–368.

[4] Effros, A., Freeman, W., Image quilting for texture synthesis, *Siggraph 2001 Conference Proceedings*, ACM Press, 2001, 341–346.

[5] Effros., A., Leung, T., Texture synthesis by nonparametric sampling, *IEEE International Conference on Computer Vision*, IEEE Press, 1999, 1033-1038.

[6] Greenfield, G., Mathematical building blocks for evolving expresssions, *2000 Bridges Conference Proceedings* (ed. R. Sarhangi), 2000, 61–70.

[7] Greenfield, G., Art and artificial life — a coevolutionary approach, *Artificial Life VII Conference Proceedings* (ed. M. Bedau et al), MIT Press, Cambridge, MA, 2000, 529–536.

[8] Heeger, D., Bergen, J., Pyramid-based texture analysis/synthesis, *Siggraph 1995 Conference Proceddings*, ACM Press, 1995, 229–238.

[9] Hertzmann, A., et al, Image analogies, *Siggraph 2001 Conference Proceedings*, ACM Press, 2001, 327–340.

[10] Sims, K., Artificial evolution for computer graphics, *Computer Graphics*, **25** (1991) 319–328.

[11] Wei, L-Y., Levoy, M., Fast texture synthesis using tree-structured vector quantization, *Siggraph 2000 Conference Proceedings*, ACM Press, 2000, 379-488.