B. A. Attea and L. M. Rashid: A genetic algorithm for texture synthesis and transfer.
In *Texture 2005: Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, pp. 59–64, 2005.

A Genetic Algorithm for Texture Synthesis and Transfer

Dr. Bara'a Ali Attea Laylan Mohammad Rashid
baraaali@yahoo.com Laylan_m_r@yahoo.com
Computer Science Department \ College of Science \ University of Baghdad

Abstract

We present a patch based method for texture synthesis and transfer. The method combines the non-iterative version of the quilting technique of Efros and Freeman as a fast, general, and simple texture synthesizer with the evolutionary model of genetic algorithm. Within a single generation, our algorithm adaptively searches the candidate patches of the available solutions so as to inherent as more as possible vital patches. The mechanism of the traditional uniform crossover used in genetic algorithm is altered here so as to accommodate texture synthesis and transfer problem. We call this crossover Semi Uniform crossover (SUX). Moreover, the mating process of the genetic algorithm is turned from its usual bi-sexual recombination scheme into a global recombination fashion where SUX is applied among all individuals to form a new single offspring. All other offspring are then obtained by mutation only. Examples and visual comparisons with image quilting, coherent synthesis, and image analogies demonstrate that the proposed algorithm can give acceptable results.

Keywords: texture synthesis, texture transfer, genetic algorithm, SUX operator, image quilting.

1. Introduction

Texture is a ubiquitous experience. It can describe a variety of natural phenomena with repetition, such as sound, motion, visual appearance, and human activities. As the importance of texture to computer vision, graphics and image processing, so dose the importance of texture synthesis and transfer. *Texture synthesis* is a method that starts with sample image and attempts to produce a texture with a visual appearance similar to that sample. Unfortunately, creating a general texture synthesis algorithm has proved difficult [1; 2; 3]. 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 [3]. In other words, texture transfer means basically rendering an object with a texture taken from a different object. In this paper we combine the technique of image quilting texture synthesis of Efros and Freeman [4] with a new genetic algorithm to address texture synthesis and transfer problems. In the proposed approach, we would like to satisfy the following:

1. Quality; it *should* produce pleasing results.
2. Generality; it should work remarkably well for a wide range of texture.
3. User-friendly; the number of tunable input parameters should be minimal.
4. Simplicity; the algorithm implementation could be simple.
5. Efficiency; computational cost should be (if possible) comparable to times required by techniques provided by other authors.

2. Previous Work

Several approaches have been recently proposed for texture synthesis. They mainly divided into either pixel-based or patch-based approaches. In general, pixel-based algorithms require up to several hours of computations to produce impressive results. The researches accelerated their algorithms (at the expense of implementation complexity) using for example, multiresolution and Tree-Vector Quantization (TVQ) techniques. Among these works are those proposed by Hertzmann et al. [5] and Wei and Levoy [1]. For natural textures M. Ashikhmin [2; 3] proposed a simple, intelligent and fast texture synthesis algorithm. On the other hand, in patch-based algorithms, the process of texture synthesis is akin to putting together patches, quilting them, making sure they all fit together. Examples of patch-based texture synthesis algorithms include the simple, general image quilting algorithm [4], efficient graph-cut algorithm [6], and the recent smart-copy algorithm [7]. Although of the large body of texture synthesis work, only a few such algorithms extended to texture transfer problem. These are image analogies [5; 8], coherent synthesis [2; 3], and image quilting [4].

Here, we mention only the patch-based texture synthesis and transfer work directly related to this research and most widely algorithm used in comparison [2; 3; 8]. Current work of Efros and Freeman (*image quilting*) [4] introduce a method of synthesizing a new image by stitching together small patches of existing images. It also, transfers a texture onto another image. Let B_i be a square block of user-specified size from the set S_B of all such overlapping blocks in the input texture image. To synthesize a new texture image, search S_B for such a block that agrees with its neighbors along the region of overlap. Then, the boundary is the minimum cost path from one end of the overlap region to the other, where the cost of being at a pixel is defined as the L_2 norm on the overlapping Red, Green, and Blue pixel values see figure 1 [9].

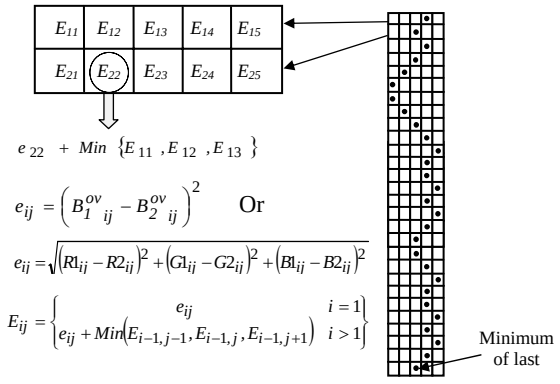

Figure 1: Minimum error boundary cut algorithm. B_1 and B_2 are two blocks that overlap along their vertical with the region of overlap B_1^{ov} and B_2^{ov} respectively, and e_{ij} is the error surface at the overlapped position (i, j).

Efros and Freeman augment their algorithm for texture transfer by requiring that each block satisfy a desired *correspondence map*, as well as satisfy the overlap requirements. The correspondence map is a spatial corresponding quantity map (e.g., image intensity, low frequency image components, image orientation, or even three dimension surface slope, over both the texture source image and a controlling target image. Thus, the error term of the image quilting algorithm is the weighted sum, α times the block overlap matching error plus (1- α) times the squared error between the correspondence map pixels within the source texture block and those at the current target image position. The parameter α determines the tradeoff between the texture synthesis and the fidelity to the target image correspondence map. The algorithm iterated over the synthesized image several times, reducing the block size with each iteration and the blocks are matched not just with their neighbor blocks on the overlap regions, but also with whatever was synthesized at this block in the previous iteration [4].

3. The Proposed Algorithm

This section presents how to combine the non-iterative version of image quilting, in a simple way, with a new genetic algorithm to tackle texture synthesis and transfer problems

3.1 Genetic algorithm: an overview

Genetic algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. Genetic algorithms have been developed by John Holland, his colleagues, and his students at the University of Michigan. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance [10].

3.2 Search Space Size and Problem Complexity

First it is important to consider the complexity of the problem by evaluating search space size. The total number of square blocks B_i in the input source texture determines the search space size. Consider a source texture of size 128×128 (see figure 2 (a)), then the set S_B of all *non-overlapping* 32×32 blocks in this texture will contain 16 distinct blocks, i.e., $|S_B|$=16 (see figure 2 (b)). Then for an output image of size 256×256 (see figure 2 (c)), the algorithm should *paste* the best blocks among the 16 source blocks onto each of the 64 target blocks. This yields an exponential search space size equal to 16^{64} = 1.1579e+077.

Although when $|S_B|$=16 yields a huge search space, it may not be sufficient enough to produce pleasing results as S_B dose not contain enough variability of the input texture. Increasing number of blocks in S_B by letting also overlapping blocks gives a huge search space size.

(a) (b) (c)

Figure 2: (a) 128×128 source texture. (b) The source texture has 16 non-overlapping square blocks each of size 32×32. (c) 256×256 target image, requires 64 texture blocks for pasting..

The extreme case is extracted when S_B contains all overlapping blocks drawn after every pixel of the input texture starting from top left corner. For the mentioned example, the extreme case gives a total of 9409 valid blocks which gives 9409^{64} potential solutions to the problem

3.3 Individual Representation: Genotype Encoding

At the beginning of the computation a number of individuals are randomly initialized. An individual, here, is represented as a two dimensional array of m × n genes where m and n are:

$$m = \frac{h_t}{w_B} \;;\; n = \frac{w_t}{w_B} \tag{1}$$

Where

h_t :height of the output image.

w_t: :width of the output image.

w_B :width of the source texture patch.

Each gene in the chromosome can hold an integer value *i* ranges from 1 to $|S_B|$ which refers to block B_i in S_B. Figure 3 depicts an individual representation with a random initialization for a 128×128 texture, 32×32 patch, a 256×256 output image, and | S_B| =16.

6	5	3	9	13	5	2	16
10	6	6	6	13	7	8	2
6	1	3	10	10	1	12	7
10	15	5	6	10	14	13	10
3	6	6	6	6	3	6	6
3	15	7	4	4	10	6	6
6	15	7	15	8	7	3	3
3	3	3	3	6	7	3	3

Figure 3: An individual representation example.

3.4 Objective Function

The objective function is used to provide a measure of how individuals have performed in the problem domain. In the case of a minimization problem, the most fitted individuals will have the lowest numerical value of the associated objective function. Here, the objective function is calculated as the error term, *E*. In texture synthesis, *E* is the sum of block overlap matching error E_{ov} overall individual genes. Each gene in a GA individual has an associated with it E_{ov} (calculated from the block referred to by this gene, and its left and top preceding blocks in the output image). E_{ov} between two blocks *B*1 and *B*2 is computed as a weighted difference in YIQ decorrelated color space as:

$$E_{ov} = \sum_{p_{in}^{ov}} \frac{10}{16}|Y_1(p) - Y_2(p)| + \frac{3}{16}|I_1(p) - I_2(p)| + \frac{3}{16}|Q_1(p) - Q_2(p)| \tag{2}$$

Where subscripts refer to the overlapped blocks.

On the other hand, for texture transfer, *E* is calculated as a weighted sum of E_{ov} and a corresponding map error E_{cm} summed over all individual genes. In addition to E_{ov} each gene in a GA individual has an E_{cm} (pixel wise luminance

or blurred luminance differences calculated from this block and the block of the target image at the current target image position). Formally speaking, E_{cm} is:

$$E_{cm} = \sum_{p_{in}^{cm}} |Y_1(p) - Y_2(p)| \quad (3)$$

3.5 Genetic Operators

After population initialization, the proposed GA *discards selection* operator and lets all parents to recombine to generate a single offspring which would be better than all of these parents. The remaining offspring are created by perturbing the generated offspring via mutation operator. The proposed crossover operator called Semi-Uniform-Crossover; SUX modifies the mechanism of the traditional uniform crossover into: *spreading vital genes at the expense of lethal genes between the matted parents rather than exchanging their genes* (*with equal probability*) figure 5 depicts the process of SUX between two parents for a texture transfer example.

A gene here can be classified as vital or lethal according to associated error. The two competing genes of the matted parents are compared in their errors. The gene with less error is then considered as the vital one, while the latter is defined to be lethal gene.

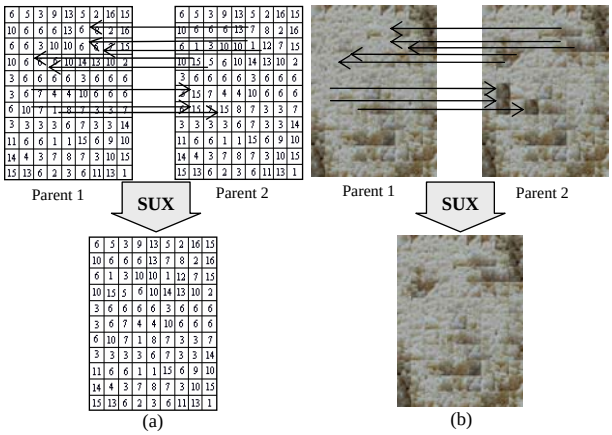

Figure 4: The proposed SUX. (a) In the genotype representation. (b) The phenotype result.

The degree of sexuality is extended from its usual fashion bisexuality to the global meaning of multi-sexuality by which it means: letting more than two parents to recombine at one time to produce a single offspring. As SUX focuses on vital genes at the expense of lethal ones, surviving vital genes among more than two matted parents can then be taken into account and replacing lethal ones to generate a single offspring.

The second perturbation operator is mutation that alters a gene value *i* by another value *j* taken randomly from [1, $|S_B|$] as shown in figure 5.

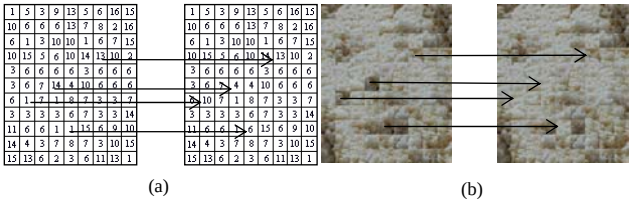

Figure 5: The mutation operator: (a) In genotype. (b) In phenotype.

3.6 Phenotype Decoding

The most common stopping condition used in genetic algorithm literature is to allow the algorithm to run to a given number of generations.

Finally, the phenotype decoding scheme consists of the following steps:

- Go through the output image and the best GA individual evolved in raster scan order (from left to right and from top to bottom) in steps of one patch for the output image and in steps of one GA gene for the GA individual.
- Pick from the source texture block B_i referred to by the current GA gene.
- Paste block B_i on the output image such that:
  - Place the left overlap region of B_i at the right overlap region of the already placed block preceding it and the top overlap region of B_i at the bottom overlap region of the block above it.
  - Find the minimum cost path along the error surface at the overlap region.
  - Make the average of the overlapping pixel luminance values at the minimum cost path the boundary of B_i.
  - Past B_i on the output image.
- Repeat.

4. Results

To facilitate results comparison with texture synthesis and transfer work, e.g., image quilting, image analogies, and coherent synthesis we use same examples. Original textures and images with their size are available at:

(www.cs.berkely.edu/~efros/research/quilting.html),

(http://www.mrl.nyu.edu/projects/image-analogies/), and

(http://www.cs.sun/sb.edu/~ash/ftt/).

All results shown here are depicted with small sizes than those experimented with so as to be within the paper size limit. The texture synthesis results of the proposed algorithm against those of Efros and Freeman are shown in figure 7. Figure 8 and figure 9 depict our results of texture transfer against those of Efros and Freeman, Hertzmann, and Ashikhmin. In the experiments, the population size was varied from 5 to 30. Maximum number of generations was varied from 3 to 10. Block width used for texture synthesis is in the range 32-50, while for texture transfer is in the range 13-20.

Overall results are comparable to those of Efros and Freeman, Hertzmann and Ashikhmin. Like texture synthesis results of image quilting, the proposed GA completely success stochastic texture, while have limitations (e.g., mismatched boundaries) on structured textures. For texture transfer, the GA algorithm successfully captures the random aspects of the stochastic textures, and capable of rendering target images with tileable results.

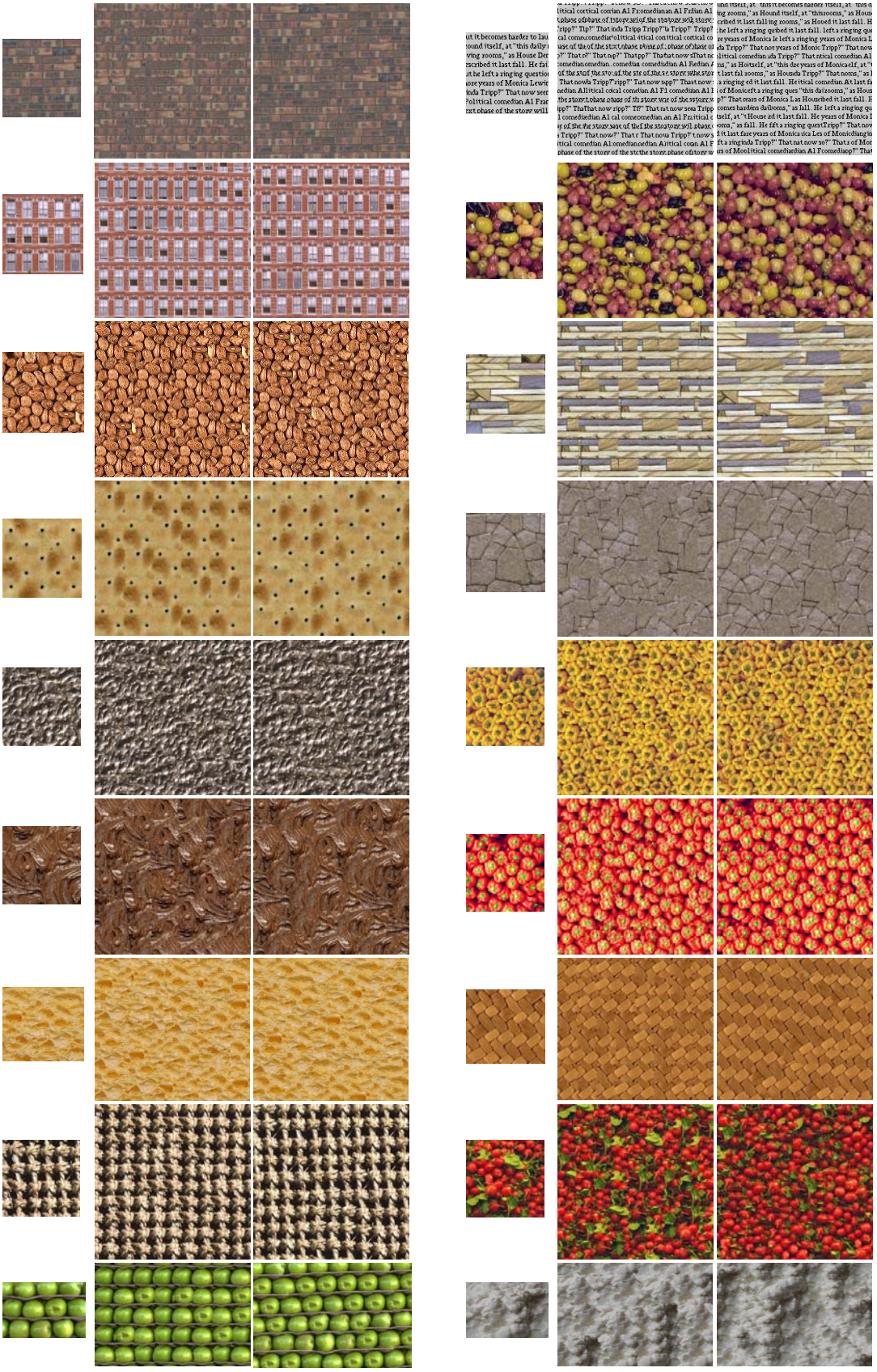

Figure 6: Texture Synthesis Results 1st column is the source texture, 2nd column synthesis results using GA with global SUX, and 3rd column synthesis results of image quilting [4].

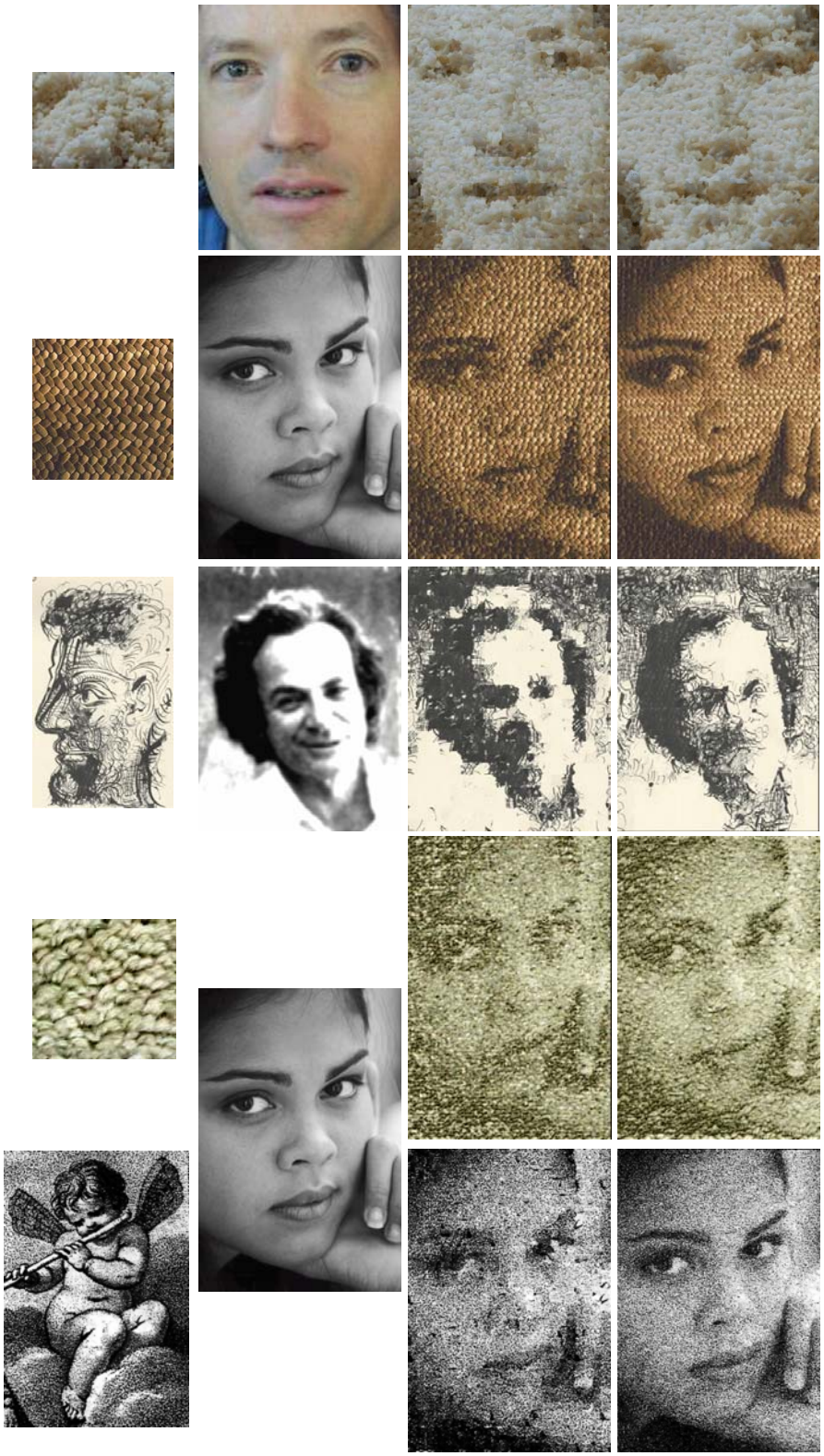

Figure 7: Texture Transfer Results: 1st column is source texture, 2nd column is the controlling target image, 3rd column are the results of proposed GA, and 4th column is the results of Efros and Freeman [4] or Hertzmann[5]., or Ashikhmin[3].

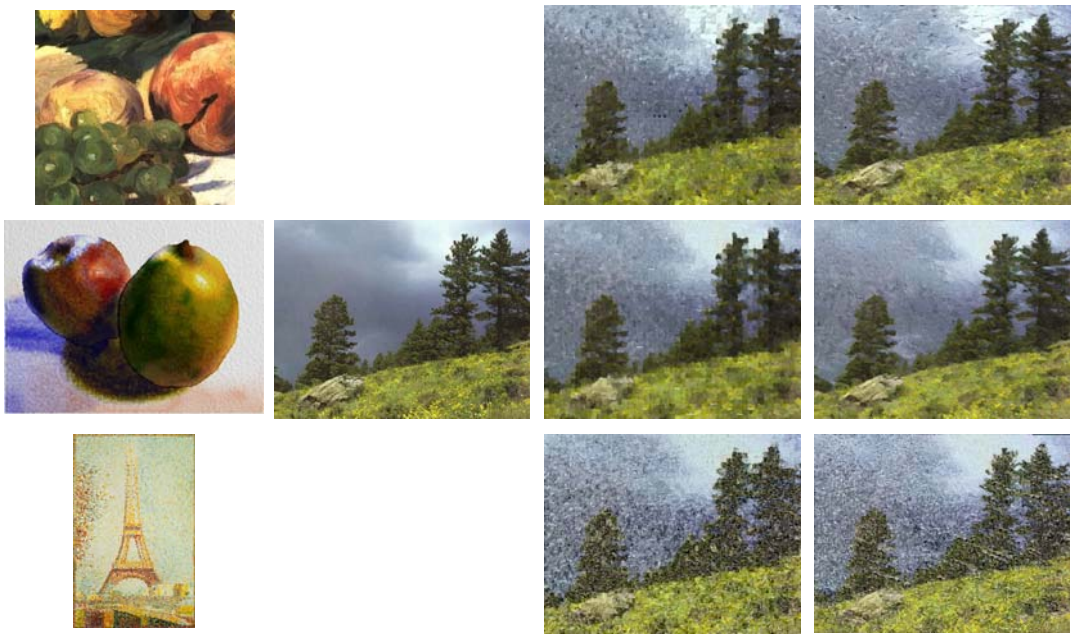

Figure 8: More Texture Transfer Results: 1st column is source texture, 2nd column is the controlling target image, 3rd column are the results of proposed GA, and 4th column is the results of Hertzmann [5], or Ashikhmin [3].

Results also demonstrate that although the algorithm does not fully capture the texture of structure ones as result of Efros and Freeman, and Hertzmann but they still provide visual results without sacrificing texture synthesis and target image rendering qualities too much. For example, in the case of Picasso, although our result does not have long, well-separated lines but they are still visually pleasing. In this pen and ink style illustration, the texture hatches delineate the Richard Feynman face tones, shades, and pattern.

The proposed algorithm ran on a single Pentium III PC computer. For texture synthesis, it takes between 17 and 61 seconds, while for texture transfer between 7 and 20 minutes per image depending on the size of input and output, block size, population size, and maximum number of GA generations. Considering the parallel behavior of the genetic algorithm, if the proposed genetic algorithm runs on multiprocessors such that each individual is processed on a single processor, we get time between 3 and 12 seconds for texture synthesis and time between 28 and 60 seconds for texture transfer. Despite technique of Hertzmann, (which requires dozens of minutes), the algorithms of Efros and Freeman, and Ashikhmin require times between seconds to several minutes for single iteration of their algorithms. Their sufficient results require from multiple of to many dozens of iterations.

5. Conclusions

This paper combines the non-iterative version of image quilting with genetic algorithm for implementing a fast, general, and simple texture synthesis and transfer algorithm. it also introduces anew crossover, SUX which alters the mechanism of the uniform crossover to be compatible with this problem. The proposed method produces results that are visually pleasing and comparable to those of Efros and Freeman, Hertzmann, and Ashikhmin.

Acknowledgments

The authors would like to thank A.A. Efros, W.T. Freeman, A. Hertzmann, and M. Ashikhmin for making their texture synthesis and transfer results imagery available online.

References

[1] L.Y. Wei, "*Texture Synthesis by Fixed Neighborhood Searching*", Ph.D. Dissertation, Stanford University, 2001.

[2] M. Ashikhmin, *"Synthesizing Natural Textures"*, University of Utah, pages 1-3, 2001.

[3] M. Ashikhmin, *"Fast Texture Transfer"* IEEE Computer Graphics and Applications, pages 1-4, 2003.

[4] A.A. Efros, and W.T. Freeman, *"Image Quilting for Texture Synthesis and transfer"*, SIGGRAPH, page 1-6, 2001.

[5] A. Hertzmann, *"Algorithms for Rendering in Artistic Styles"*, PhD thesis, New York University, 2001.

[6] V. Kwatra, I. Essa, A. Schödl, G. Turk, A. Bobick, *"Graphcut Textures: Image and Video Synthesis Using Graph Cuts"*, SIGGRAPH 2003.

[7] A. Neubeck, A. Zalesny, and L. Van Gool, *"Cut-primed smart copying"*, Texture 2003 Workshop in conjunction with ICCV 2003.

[8] J. Huttunen *"Image Analogies"*, Helsinki University of technology, pages 2-6, 2004.

[9] J. Wu, *"Patch-Based Texture Synthesis in 2D and 3D"*, University of California, San Diego, 2003.

[10] D.E. Goldberg, *"Genetic Algorithm in Search, optimization, and Machine learning"*, Addison Wesley, 1989.