Number1 Volume 13 march 2007 Journal of Engineering

TEXTURE SYNTHESIS BY GENETIC ALGORITHM

Asst. Prof. Dr. Mayada F. Abdul-Halim
Noor Adnan Ibraheem,
University of Baghdad
College of Science
Department of Computer Science

ABSTRACT

One way to synthesis texture in a fast and easy way is image quilting proposed by Efros and Freeman in 2001. This research brings the adaptive search power of genetic algorithm and combines it with the concept of image quilting to propose new texture synthesis algorithm. The proposed GA is ran on many different images from standard texture sets. Visual comparison of our proposed GA with image quilting algorithm is considered. The texture results generated by the proposed GA are roughly comparable in quality to those generated from Efros and Freeman algorithm.

الخلاصه

واحدة من طرق تكوين نسيج شئ ما بطريقة سريعة وسهلة هي طريقة خياطة الصورة التي اقترحها ايفروس و فريمان فسي 2001. هذا البحث يقوم بجمع قوة البحث التي توفرها الخوارزمية الجينية مع مفهوم طريقة خياطة الصورة لاقتراح طريقة جديدة لتكوين صورة نسيج شئ ما. الخوارزمية الجينية المقترحة نفذت على عدد مختلف من الصور الموجودة في المجاميع القياسية للصور. المقارنة المرئية للخوارزمية المقترحة مع خوارزمية خياطة الصورة اظهرت ان جودة النتائج متقاربة.

KEYWORDS

Patch-based texture synthesis, image quilting, genetic algorithms.

INTRODUCTION

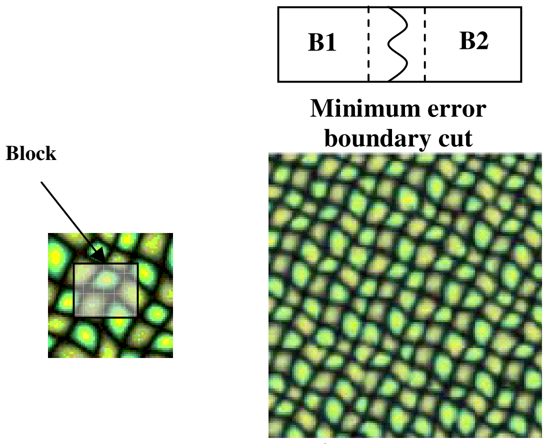
Texture is a common seen scenario in the real world and it usually characterizes certain types of surfaces of objects (e.g. walls, clouds, and piles of food cans). Therefore, reproducing the textures for these objects is usually required when rendering their synthetic images. One way to reproduce textures is from scanned photographs. However, this method is often suffered from inadequate size or visible repetition and seams if a simple tiling is directly used. Texture synthesis is an alternative way to create textures. Given a texture sample that contains adequate stochastic and structural information, the goal of texture synthesis is to grow a new texture that visually appears to be generated by the same underlying pattern as in the input texture sample. This method has a variety of applications in computer vision, graphics, and image processing. For example, textures have long been used to decorate object surfaces in computer rendered images. However, natural textures are often difficult to generate manually; therefore an algorithm to synthesize a large texture from a small scanned patch will be desirable.

Until 2001, most texture synthesis algorithms compute the value of each pixel in the synthesized texture individually. However, in 2001, Efros and Freeman published their paper “Image Quilting for Texture Synthesis and Transfer”. In this paper, they note that most pixels in a synthesized texture do not have a choice about their final pixel value. Using this observation, Efros and Freeman present a method to synthesis texture based on blocks of texture rather than individual pixels. This process –image quilting- is a simple and fast patch-based method: the block substitution is here optimized by stitching together small blocks of existing images and minimizes the error on the boundary cut where the blocks join. By using blocks, the texture synthesis process becomes easier and faster whiles still producing excellent results for both stochastic and structured textures.

In this paper, we bring the adaptive search power of the genetic algorithm and combines it with the concept of image quilting to propose a new texture synthesis algorithm. Because our method is based on the patch-based approach, particully image quilting, we provide a brief overview of this approach. An overview of genetic algorithms also is given. Then we present the proposed genetic algorithm and how we utilize it for synthesizing texture. We show the results obtained and conclude by outlining some possible extensions of this work.

IMAGE QUILTING TEXTURE SYNTHESIS

The basic idea of image quilting texture synthesis procedure is as follows (Efros and Freeman, 2001). Assume that the unit of synthesis B_i is a square block of user specified size from the set S_B of all such overlapping blocks in the input texture image is defined. To synthesize a new texture image, as a first step tile it with blocks taken random from S_B. Before placing a chosen block into the texture looking at the error in the overlap region between it and the other blocks. A minimum cost path throw that error surface is computed and declare that to be the boundary of the new block. Figure 1 shows the result of this process.



Input texture **Output texture**
Fig.(1): Quilting Texture

The minimal cost path throw the error surface is computed in the following manner. If B_1 and B_2 are two blocks that overlap along their vertical edge (Figure 1) with the regions of overlap B_1^{ov} and B_2^{ov}, respectively, then the error surface is defined as $e = (B_1^{ov} - B_2^{ov})^2$. To find the minimal vertical cut through this surface traverse e ($i = 2..N$) and compute comulative minimum error E for all paths (Efros and Freeman, 2001):

$$E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}) \qquad \text{........................} (1)$$

In the end, the minimum value of the last row in E will indicate the end of minimum vertical path through the surface and one can trace back and find the path of the best cut. Similar procedure can be applied to horizontal overlaps. When there is both a vertical and a horizontal overlap, the minimal paths meet in the middle and the overall minimum is chosen for the cut.

AN OVERVIEW OF GENETIC ALGORITHM

Genetic Algorithms (GAs) are a class of stochastic search algorithms for solving many difficult optimization problems (Goldberg, 1989). They are motivated by the computational process in natural evolution that enable organisms to adapt more to their environment over many generations. GAs operate on a set of possible *individuals*, which is called the *population*. In biological terms, the individual's bit string (i.e., *gene*) is the *genotype* and the solution represents the *phenotype* of a particular individual or *chromosome*[1]. The most basic operations used by GAs are *selection, crossover,* and *mutation*. The selection operator identifies (according to *fitness* value) the individuals of the current population, which will serve as parents for the next generation. Crossover randomly chooses pairs of individuals to combine properties of them by creating offspring. Crossover occurs with probability p_c, which is typically near one. Mutation is usually considered as a secondary operator, which makes small changes on single individuals to restore *diversity* of the population that may be lost from the repeated application of selection and crossover (Badros, 1995). Mutation occurs with some small probability p_m.

DESIGN OF GENETIC ALGORITH FOR TEXTURE SYNTHESIS

In this paper we bring the adaptive search power of GA and combines it with the concept of image quilting to propose a new texture synthesis algorithm, as we coined conventional genetic texture synthesis (CGTS).

The general steps of CGTS are:

1. Generate a random population of synthesized texture chromosomes with a pre-selected size.
2. Evaluate fitness and determine the best synthesized texture chromosome and carry it to the new population (elitist strategy). In this way there is a guarantee that the good synthesized texture chromosome is not lost.
3. Repeating the following steps until new population is complete.
   - 3.1 Select two parent chromosomes from the population according to their fitness (the better synthesized texture chromosome, the bigger chance to be selected).
   - 3.2 With a crossover probability, p_c, cross over the selected parents to form a new offspring. If no crossover was performed, offspring is an exact copy of parents.
   - 3.3 With a mutation probability, p_m, mutate new offspring at each gene.
   - 3.4 Place new offspring in a new population.
4. Use new generated population instead of old one for a further run of algorithm.
5. If the termination criterion is satisfied, stop, and return the best solution in current population.
6. Go to step 2.

The following subsections clarify the CGTS chromosome representation.. Depending on the chromosome representation, crossover and mutation are put in plain words.

Chromosome Representation and Population Initialization

The genetic algorithm used in this research process populations (old population and new population) of synthesized image textures. A synthesized image texture is a chromosome in the population. In the proposed GA, the chromosome representation is based on the idea of blocks arrangement, where the texture blocks are pasted from top to bottom, left to right. Each chromosome is represented using an array of $N \times M$ genes decimal variables inclusively. Gene variables represent block number i of the block B_i generated randomly from the set S_B of all such overlapping blocks in the input texture image I_{sample}. While $N \times M$ represent the encoding (genotype) of the output texture image I_{out}. Figure 2 gives an illustration of the chromosome representation used.

1	5	12	11	4	5	8	10
9	10	2	9	2	5	7	11
1	11	7	5	1	6	10	12
12	3	11	3	10	2	6	8
6	4	5	2	8	4	7	2
2	6	4	12	5	6	5	7
7	4	7	8	9	3	1	2
11	8	3	5	6	1	8	3

Block number 2 in input texture image, I_{sample}

Fig.(2). GA Texture Chromosome Representation, where maximum number of overlapping blocks , here, is 12.

Consider that an output texture image of size $O_1 * O_2$ is to be synthesized from an input texture image of size $I_1 * I_2$ with block size $w_B * w_B$ and overlap region size w_e, of course $I_1 > w_B$ and $I_2 > w_B$. Then, the dimension $N * M$ of an array of genes which define the GA chromosome is calculated as follows.

$$N = \lceil O_1 / (w_B - w_e) \rceil \quad (2)$$

$$M = \lceil O_2 / (w_B - w_e) \rceil \quad (3)$$

Each gene can hold a block number ranges from 1 to the maximum number of such overlapping blocks in the input texture image.

For research purposes, random initializing of population is the best. Moving from a randomly created population to a well-adopted population is a good test of the algorithm, since resulted synthesized texture will have been produced by the search of algorithm rather than initialization procedures. Therefore, random initialization for population is used here for CGTS. It is a simple matter to create new offspring from the members of old population using genetic operators, place those new texture chromosomes in new population. The choice of population size p_{size} ranges from 30 to 200 in conventional GA (Grefenstette 1986) .

Objective and Fitness Functions

The *objective function* is used to provide a measure of how chromosomes have performed in the problem domain. As the objective of texture synthesis problem is to minimize the error on the boundary where the patches join, GA deals with image texture synthesis problem as a minimization problem, i.e., the fit chromosomes will have the lowest numerical value of the associated objective function. This raw measure of fitness is usually only used as an intermediate stage in determining the relative performance of chromosomes in a GA. Another function, *fitness function*, is normally used to transform the objective function value into measure of relative fitness, thus:

$$fitness = g(objective) \qquad \text{.........................} \ (4)$$

Where g transforms the value of the objective function *objective* to a non-negative number, and *fitness* is the resulting relative fitness. This mapping is always necessary when the objective function is to be minimized as the lower objective function values correspond to fitter chromosomes.

The objective function of any synthesized texture chromosome can be calculated by summing all vertical and horizontal Euclidean distance measured in RGB space at overlapping location.

$$objective = \sum_{i=1}^{N*M} (ve_i + he_i) \qquad \text{.........................} \ (5)$$

Where ve_i is the sum-of-squared difference-SSD of the vertical overlap region between block i and its left block, and he_i is the SSD of the horizontal overlap region between block i and its upper block in I_{out}. Then the fitness function can be computed as follows:

$$fitness = \frac{1}{objective} \qquad \text{.........................} \ (6)$$

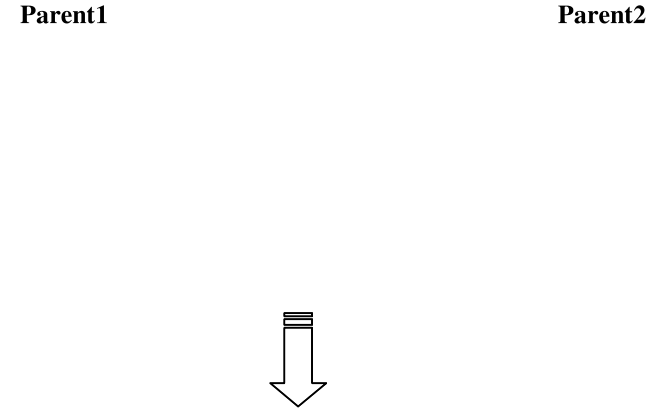
For every synthesized texture chromosome, the fitness value is calculated. Better synthesized texture (i.e., chromosomes with larger fitness values) will get higher score. Evolution function directs population towards progress because good fitness will be selected during selection process and poor one will be rejected.

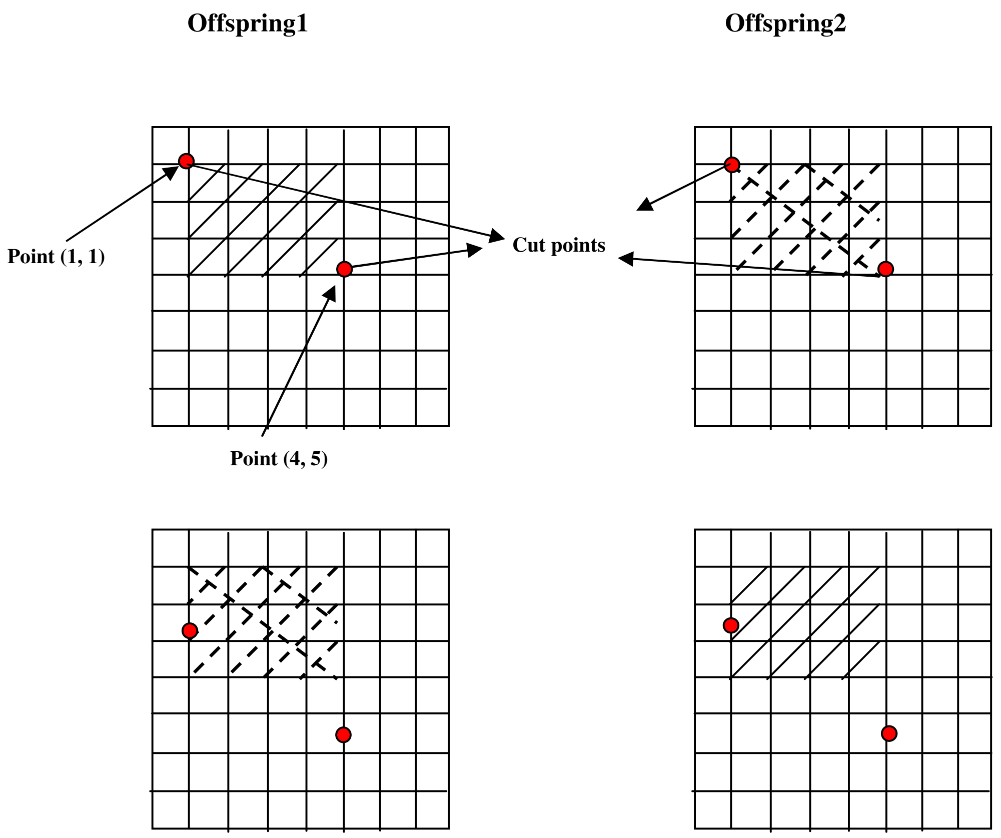
Selection

In this paper tournament selection is used, where two individuals are taken at random from initial generation, and the better individual is selected from them. The winner of the tournament is the individual with higher fitness of the tournament competitors, and the winner is inserted into mating pool, the mating pool, being filled with tournament winners. In addition, the *elitist* strategy is used so that the best individual will be automatically survived to the next generation. Elitism makes the GAs retain the best individual at each generation. The best individual can be lost if it is not selected to produce or if it destroyed by recombination or mutation (Mitchell 1998).

Crossover

There are numerous ways to implement crossover. Some forms of crossover are more appropriate for certain problems than others are (Spears 1997). Two point crossover is used for the proposed GAs. After choosing two parents from the mating pool, two crossover points are selected randomly. Each cut point represents a row and column coordinates of the parent chromosome array. Then the blocks of textures between these twopoints from the selected individuals are swapped to form two new offspring. Crossover occurs with probability 0.6 for CGTS. Figure 3 clarifies the crossover operation.





Fig(3). Crossover operation where the cut points (1, 1) and (4, 5) are chosen randomly.

Mutation

In CGTS, mutation is applied to each offspring chromosome after crossover. Each gene in the offspring is changed to include any source block number taken randomly from S_B. Mutation is performed with low probability, $p_m = 0.1$. Figre 4 clarifies this operation.

Offspring

Gene block number is to be perturbed by mutation

1	**5**	12	11	4	5	8	10
9	10	2	9	2	5	**7**	11
1	11	7	**5**	1	6	10	12
12	3	11	3	10	2	6	8
6	**4**	5	2	8	4	7	2
2	6	4	12	5	6	5	7
7	4	7	8	9	**3**	1	2
11	8	**3**	5	6	1	8	3

Before mutation

new Offspring

1	2	12	11	4	5	8	10
9	10	2	9	2	5	9	11
1	11	7	3	1	6	10	12
12	3	11	3	10	2	6	8
6	12	5	2	8	4	7	2
2	6	4	12	5	6	5	7
7	4	7	8	9	11	1	2
11	8	7	5	6	1	8	3

After mutation

Fig(4). Mutation operation

Termination Criteria

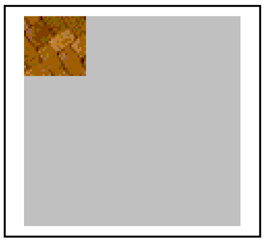
A common practice to terminate the GA is after a pre-specified number of generations. Then the quality of the best members of the population against the problem definition is tested. If no acceptable solution is found, the GA may be restarted or a fresh search is initiated (pohleheim 1998). Henceforth, the application of this termination criterion is used.

Genotype Decoding (Phenotype)

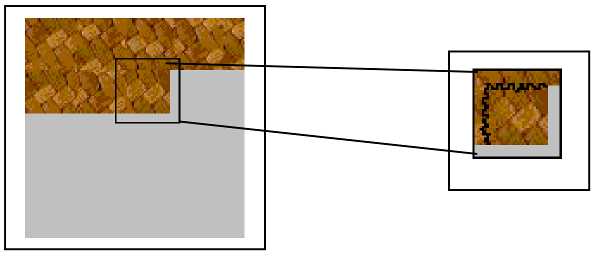
The genotype of the best resulted synthesized texture chromosome from the last generation must be decoded into the corresponding phenotype, i.e., output texture image I_{out}. This operation is accomplished as follows. The best synthesized texture chromosome from the last generation is an array of blocks with minimum overlap error. The first block number at coordinate 1,1 is replaced by its corresponding pixel values in the upper left corner of the I_{out}. The remaining blocks are replaced with its corresponding pixel values in raster scan order, i.e., from top to bottom and from left to right. The overlapping process is done before pasting the block onto the output image. The minimum cost path in the upper and left overlap regions of block is calculated. Finally the path as the boundary between the adjacent blocks is marked. One purpose of finding the minimum cost path is to level off the abrupt intensity transitions in the overlap regions. Figure 5 depicts an example of the genotype decoding process.

1	2	3	1
2	3	4	2
1	2	1	3
1	2	2	1

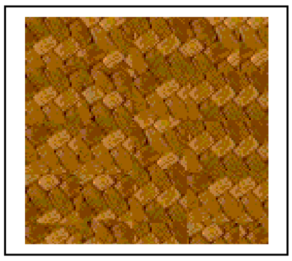
(a)



(b)



(c)



(d)

Figure 5: Genotype decoding process. The gray area is already synthesized. (a) The best synthesized texture chromosome, (b) the output texture image I_{out} after replacing the first block. (c) The minimum cost path in the upper and left overlap regions of block (d) the chromosome phenotype.

EXPERIMENTAL RESULTS

The parameter settings used in all experiments are shown in table 1.

Table1: parameters setting

Parameter	**Value**
Population size	50
Termination criteria	After 50 generation
Crossover rate	0.6
Mutation rate	0.1
Block size	32*32
w_e	5-pixel
I_{sample}	64*64
I_{out}	128*128

A visual comparison of our approach with image quilting algorithm is shown in figure 6. As depicted from the figures, the texture results generated by CGTS is roughly comparable in quality to those generated from Efros and Freeman algorithm (2001). More results from applying CGTS on different texture samples shown in figure 7. Although our algorithm is able to synthesize a wide variety of textures including stochastic and semi-structured textures, they still have several limitations as shown in figure 8. Because we use fixed shape and size of blocks, our algorithm cannot reproduce perfect results for structured textures and with perceptivity, lighting and shadow features.

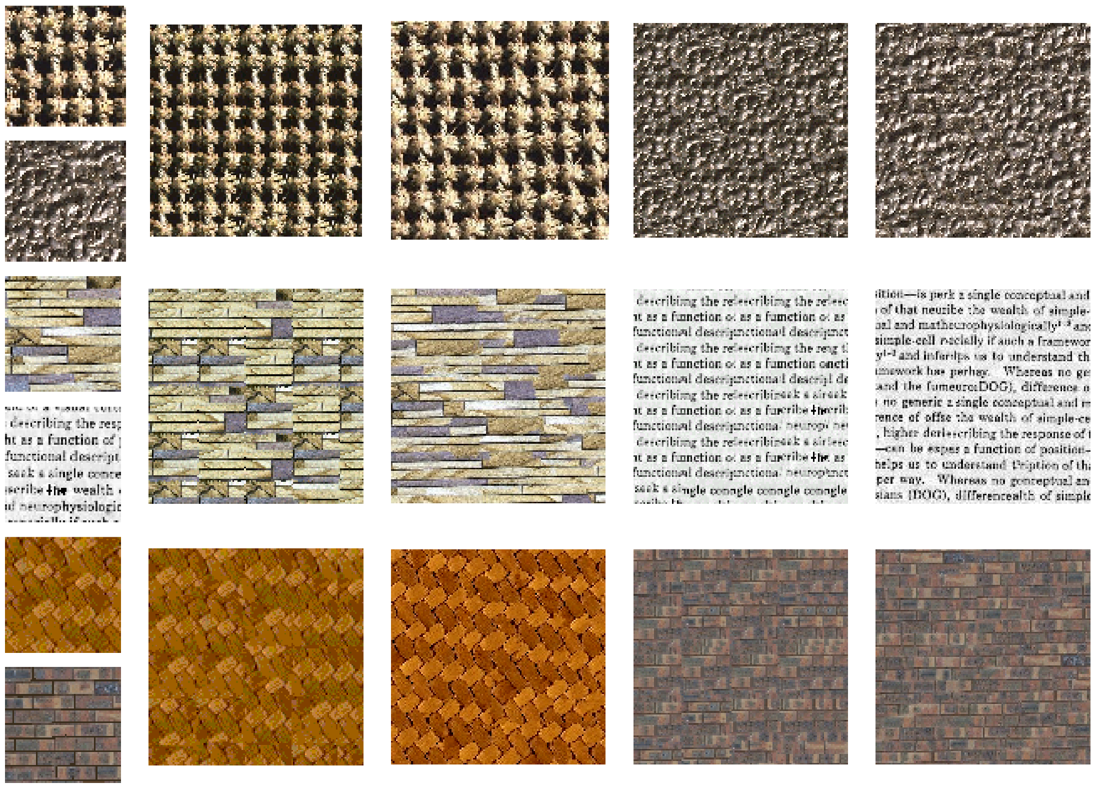


Fig.(6). Texture synthesis results. 1st column: sample texture. 2nd and 4th columns: synthesis result generated by CGTS. 3rd and 5th columns: synthesis result generated by image quilting algorithm (Efros and Freeman, 2001).

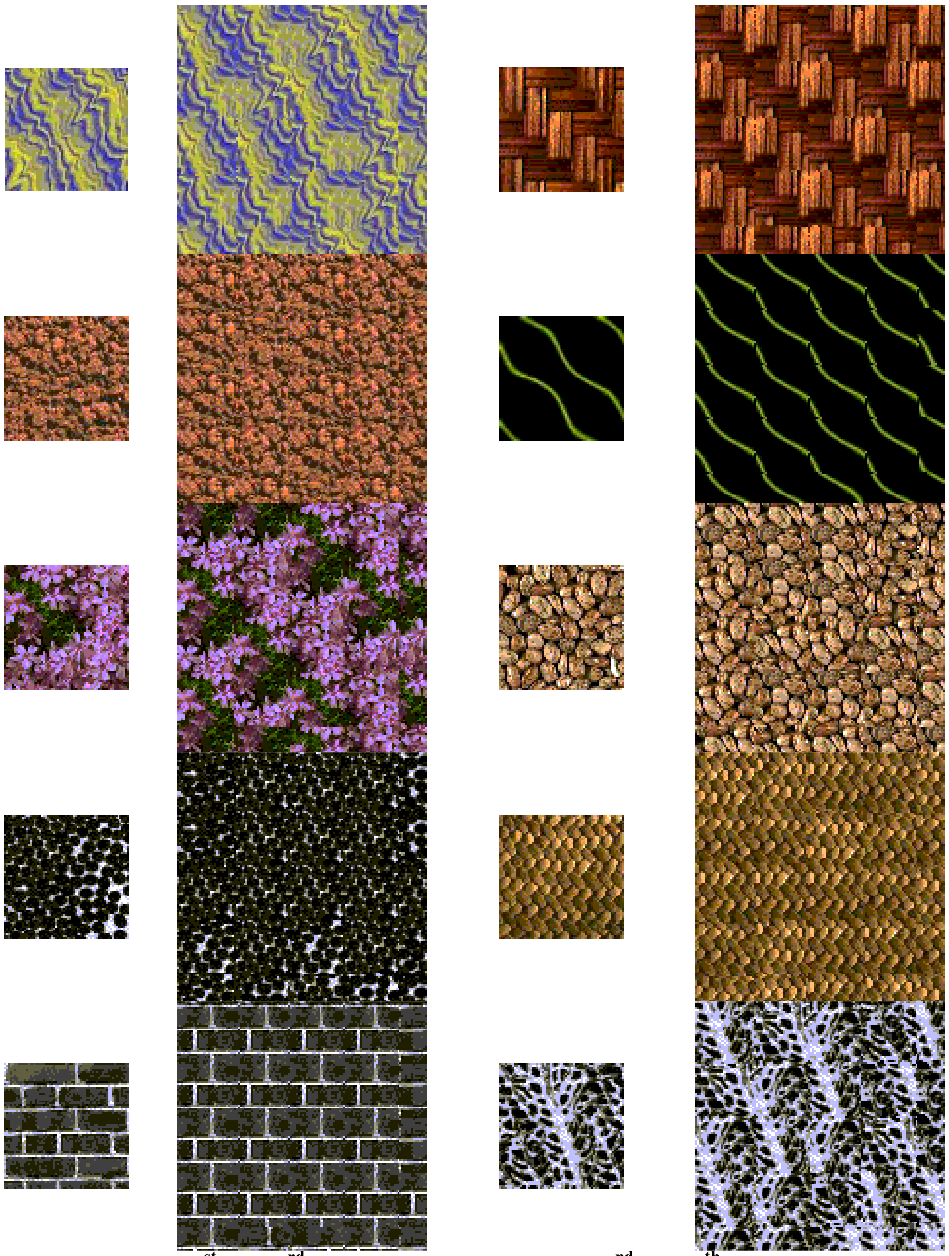


Fig.(7). Texture synthesis results. 1st and 3rd columns: sample texture. 2nd and 4th columns: synthesis result generated by CGTS.

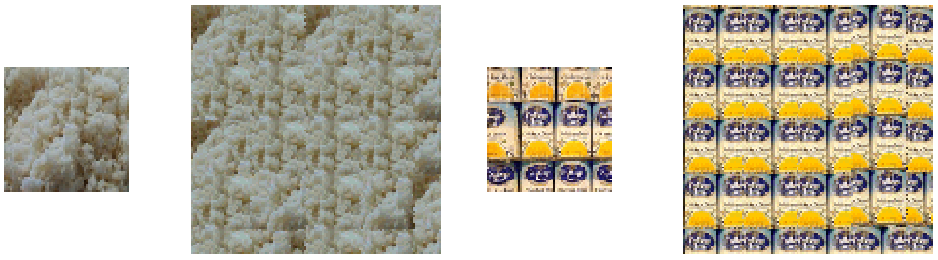


Fig.(8). Limitations of our texture synthesis algorithm. The smaller patches are the input textures, and to the right is the synthesized results by CGTS.

CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a new texture synthesis algorithm- as we coined conventional genetic texture synthesis algorithm. The proposed genetic texture synthesis algorithm combines the idea of image quilting and the power of the conventional genetic algorithm for texture synthesis problem. The results demonstrate that the texture results generated by CGTS are roughly comparable in quality to those generated from image quilting algorithm.

There are several possible directions for future work. Although our algorithm is fast enough for software applications, it needs further improvements. For example, to reduce computational efforts and provide a smooth transition between adjacent texture blocks a simple blending method called feathering can be used. this method may reduce the limitations of our method. Further, one can test the GA with smaller population size, and/or with different GA operators to see whether they produce progress in the GA results.

REFERENCES

Efros A. A. and Freeman. W. T. (2001): *Image quilting for texture synthesis and transfer*. In Proceedings of ACM SIGGRAPH 2001, Computer Graphics Proceedings, pages 341–346.
Brodatz, P. (1966): *Textures*: *A Photographic Album for Artists and Designers*. Dover, New work
Goldberg, D. E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley.
Grefenstette, J. J. (1986): *Optimization of control parameters for genetic Algorithms*, IEEE Transaction on Systems, Man, and Cybernetics, vol. SMC-16, no. 1.
Mitchell, M. (1998): *An Introduction to genetic Algorithms*, A Bradford Book, The MIT press.
Pohleheim, H. (1998): *GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB*, available at http://www.geatbx.com/index.html.
Spears, W. M. (1997): *Recombination Parameters*, Handbook of Evolutionary Computation, eds., T. Back and Z. Michalewies, IOP publishing and Oxford University Prees.