(1) Workshop

Workshop on Multimedia Technologies

(2) Title

Image Synthesis by Numbers of Texture Patches

(3) Abstract

In this paper we investigate two kinds of prevalent applications including the texture synthesis and the texture transfer. The texture synthesis technique takes an input sample and generates synthetic textures of unlimited size, which perceived by human being to be very similar to the given source. Initially, we adopt patch-based sampling algorithm to complete the goal, which derives from the *Markov Random Field* model and performs faster than the pixel-based counterpart. Secondly, rather than exhaustive search we explore the accelerative technique of nearest neighbors search to improve the performance in the synthesis process of best match. Thirdly, to find the minimum error path the dynamic programming technique is used for tackling the problem of consistent transition between two overlapped patches. In particular, our automatic system framework is amenable for synthesizing the tiling and constrained textures. The usage of tiling texture is very conventional in texture-mapped geometric models and webpage images, and the constrained texture synthesis is applicable to a wide category of applications such as hole filling and object removing.

In addition, we propose a non-iterative method for texture transfer algorithm, which renders the target image by transferring source texture patches very similar to the texture synthesis algorithm. The method runs more than an order of magnitude faster than multi-pass counterpart, and takes into account two match principles of target fidelity and neighbor coherence. Moreover, we succeed in extending the technique for the label controlled texture transfer, and demonstrate its employment on landscape gardening.

(4) Authors

- Jiunn-Shyan Lee

jslee@cs.nchu.edu.tw

Institute of Computer Science, National Chung Hsing University

250 Kuo-Kuan Road, 402, Taichung, Taiwan, R.O.C.

Tel: 886-4-22840497 Fax: 886-4-22853869

- Jin-Ren Chern

jrchern@cs.nchu.edu.tw

Institute of Computer Science, National Chung Hsing University

250 Kuo-Kuan Road, 402, Taichung, Taiwan, R.O.C.
Tel: 886-4-22840497 Fax: 886-4-22853869

- Chung-Ming Wang

cmwang@cs.nchu.edu.tw
Institute of Computer Science, National Chung Hsing University
250 Kuo-Kuan Road, 402, Taichung, Taiwan, R.O.C.
Tel: 886-4-22840497 Fax: 886-4-22853869

(5) Correspondence

Jiunn-Shyan Lee (jslee@cs.nchu.edu.tw)

(6) Categories

Computer Graphics, Texture/Image Synthesis, Example-Based Rendering, Non-Photorealistic Rendering

Image Synthesis by Numbers of Texture Patches

Jiunn-Shyan Lee[1], Jin-Ren Chern[1,2], Chung-Ming Wang[1]
[1]Institute of Computer Science, National Chung Hsing University
[2]Department of Information Management, Chien Kuo Institute of Technology
{jslee, jrchern, cmwang}@cs.nchu.edu.tw

Abstract

In this paper we investigate two kinds of prevalent applications including the texture synthesis and the texture transfer. The texture synthesis technique takes an input sample and generates synthetic textures of unlimited size, which perceived by human being to be very similar to the given source. Initially, we adopt patch-based sampling algorithm to complete the goal, which derives from the *Markov Random Field* model and performs faster than the pixel-based counterpart. Secondly, rather than exhaustive search we explore the accelerative technique of nearest neighbors search to improve the performance in the synthesis process of best match. Thirdly, to find the minimum error path the dynamic programming technique is used for tackling the problem of consistent transition between two overlapped patches. In particular, our automatic system framework is amenable for synthesizing the tiling and constrained textures. The usage of tiling texture is very conventional in texture-mapped geometric models and webpage images, and the constrained texture synthesis is applicable to a wide category of applications such as hole filling and object removing.

In addition, we propose a non-iterative method for texture transfer algorithm, which renders the target image by transferring source texture patches very similar to the texture synthesis algorithm. The method runs more than an order of magnitude faster than multi-pass counterpart, and takes into account two match principles of target fidelity and neighbor coherence. Moreover, we succeed in extending the technique for the label controlled texture transfer, and demonstrate its employment on landscape gardening.

Keywords: Texture Synthesis, Texture Transfer, Markov Random Field, Dynamic Programming, Nearest Neighbors Search, Tiling Texture, Object Removal

1 Introduction

In image-based rendering techniques, applying textures extracted from reality onto virtual worlds to synthesize novel views is compulsory. The captured textures lead to some common penalties such as texture seam, texture repetition, texture extrapolation, object occlusion, and highlight/shadow presence etc. As a consequence, to develop techniques tackling the problem is an urgent research subject. Recently, texture synthesis [4, 5, 9, 14] algorithm is one of prominent approach among them.

Texture mapping is ubiquitous technique for adding realism to computer-generated applications such as interactive virtual environments and first-person shooting games. Due to its diverse employments in computer graphics, it is being provided as standard rendering interface both in graphics software and hardware. However, a scanned photograph may be too small so that to be tiled on the entire object surface. Consequently, undesired artifacts such as seams and repetition are obvious. A significant solution adopts the texture synthesis approach to produce unlimited texture size or to re-synthesize a tiling texture.

The texture synthesis algorithm should be able to take a texture sample as input and generate a synthetic texture of arbitrary size. While the resulting image is not exactly like the original, it will be perceived by human to be very similar to the given texture. In literal there are many algorithms pursuing this goal. One class of the algorithm is based on the *Markov Random Field* model to avoid explicit probability function construction, but uses a deterministic searching method on established samples from the source. In other words, the resulting pixel is predictable from a small set of neighboring pixels. According to the pixel size at each synthesis time these algorithms can be divided into two categories: pixel-based sampling [2, 4, 6, 14] and patch-based sampling [5, 9]. Patch-based sampling algorithm has many advantages over pixel-based counterpart. Firstly, it runs more than an order of magnitude faster than pixel-based algorithm. Secondly, it produces high quality resultant textures and works well for a wide category of textures ranging from stochastic to regular. Thirdly, pixel-based approach is a special case of patch-based algorithm, while the patch size is properly configured.

In this study our methods of texture synthesis and transfer are based on Efros and Freeman's image quilting algorithm [5], which employs overlapped texture patches to synthesize large textures and finds minimum error cut in boundary transition. However, we accomplish several improvements in this work, and generate plausible visual effect and synthesize faster than their work. Firstly, we explore the accelerative technique of nearest neighbors search, whereas they do not take speed into account. Secondly, we address the problem of constrained texture synthesis, which is applicable to a variety of applications such as object removal. A strikingly result is illustrated in Figure 1. Thirdly, we propose a non-iterative procedure for texture transfer, which runs more than an order of magnitude

faster than the multi-pass method. We also experiment the algorithm to label controlled texture transfer, as the demonstration in Figure 2.

The remainder of the paper is organized as follows. Section 2 outlines the related background of pixel-based and patch-based methods for texture synthesis. In section 3 we introduce the automatic framework of patch-based texture synthesis, especially for the applications of constrained texture synthesis. Then, we describe non-iterative texture transfer scheme and augments this technique on labeled image for landscape gardening in section 4. Section 5 demonstrates some results and performance evaluations. Finally, section 6 concludes our investigations and highlights some future avenues.

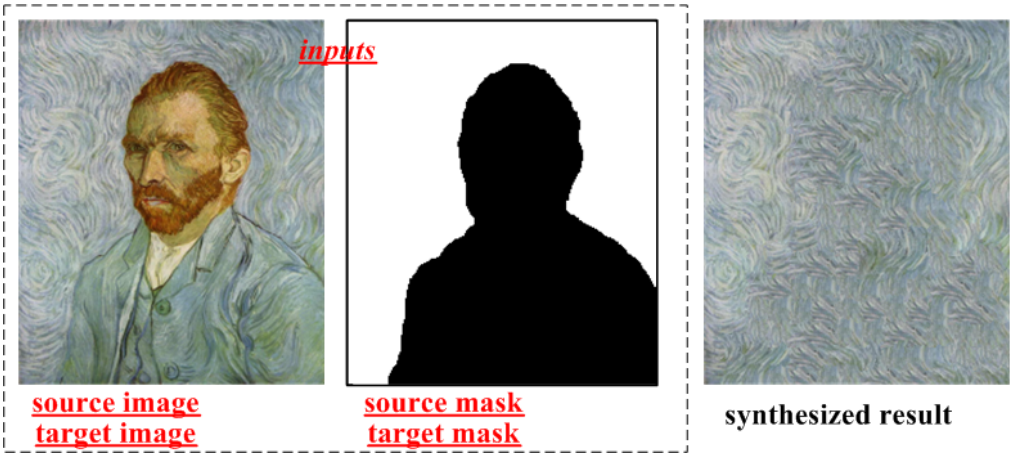


Figure 1: Example of constrained texture synthesis for foreground object removing on Van Gogh's painting. With four provided ingredients (source, target, and masks), our system can fill the desired holes with the neighboring texture patches without any user intervention in a short time.

Figure 2: Example of labeled texture transfer for landscape gardening. With three supplied ingredients (source, source label, and target label), our system automatically transfers the source patches to the labeled target in a few seconds.

2 Related Work

Procedural texture synthesis is well-established technique in computer graphics community comprising solid texture [10], reaction-diffusion [11], and clonal mosaic [13] etc. Perlin [10] introduces the noise function that returns a scalar value at a given point to create patterns such as wood and marble. The reaction-diffusion mechanism [11] governs chemical concentrations to a stable state for spotted and striped pattern formation occurring in coats of mammals. Another biological mechanism proposed by Walter [13] simulates cell division and interactions to produce spotted and striped patterns usually seen on animals. The methods of procedural texture synthesis have limitations in producing a small range of textures and problems in designing a proper algorithm to generate the desired textures.

Another prominent direction devotes to analyze a texture and extract its parameters in order to synthesize a new texture with similar characteristics [3, 7]. Recently, efficient and deterministic algorithms derived from *Markov Random Field* (MRF) are proposed, which avoid constructing explicit probability distribution. Efros and Leung's [4] method grows a new image outward from seed pixel that copied from the texture sample in a spiral order. To synthesize one pixel at a time, the algorithm looks for the best few matches of surrounding pixels in the square patch compared to the source patches. They randomly select one matched center, and copy the center color for the synthesized pixel. Wei and Levoy [14] develop a similar method to visit the pixel in a raster order and consider an L-shaped neighborhood of current pixel. In particular, they perform the match process based on two adjacent levels in multi-resolution pyramid to improve the quality of synthetic image. They also use tree-structured vector quantization (TSVQ) to speed up the match searching process. These techniques produce good results on the class of stochastic textures such as grass. Besides, their generality frees the user from any intervention. However, they fail in the class of repetitive pattern and structured textures such as bricks. A problem indicated by [4] tends to wander into the wrong part of search space and grow the garbage.

To address the problem of structured texture, patch-based sampling techniques of texture synthesis are emerged. In recent years, the technique has moved from the domain of pixel sampling to that patch sampling. In the literal, two studies of patch-based texture synthesis are developed concurrently, which are proposed by Efros [5] and Liang [9]. The core difference between them is the adopted technique to tackle the problem of overlapped boundary transition. Efros applies dynamic programming to find a minimum error boundary cut, while Liang uses feathering technique to blend two overlapped patches. Liang [9] reports that the feathering approach can produce more smooth color change than minimum error cut, but it also produces a smeary effect in some case. In the paper we adopt the patch-based method to accomplish the goal of constrained texture synthesis. In particular, we extend the technique to render images by the given texture, denoted as texture

transfer.

3 The Framework of Texture Synthesis

The texture synthesis algorithm takes an input sample and synthesizes output textures to any desired resolution, which perceived by human to be very similar to the input sample. The algorithm is based on the *Markov Random Field* model to avoid explicit probability function construction, but uses a deterministic searching method on input sample. Our texture synthesis algorithm adopts patch-based sampling method, since it performs more than an order of magnitude faster than the pixel-based algorithm [2, 4, 6, 14] and is applicable to a wide category of stochastic and regular textures.

3.1 Patch-based Texture Synthesis

Our texture synthesis framework is based on Efros and Freeman's image quilting algorithm [5], which employs overlapped texture patches to synthesize the larger texture and finds minimum error cut in boundary transition. However, in their work they don't concentrate on the time of best match search and on the case of constrained texture synthesis. In the paper we propose an automatic framework to accomplish these synthesis goals. First, we explore the accelerative technique of nearest neighbors search [1] to minify the search time in the synthesis process. Second, we address the problem of constrained texture synthesis, which succeeds in synthesizing tiling textures and filling the image holes without user intervention.

First, the source texture can provide a set of candidate patches B_s. Our algorithm proceeds in a raster scan order. For each of to be synthesized target patch $B_t(i)$, its overlapped region (usually has been synthesized) is compared against all possible candidate's correspondent region from B_s. Then, a best match patch $B_s(j)$ with minimum distance is selected and pasted onto the patch $B_t(i)$'s location. We use *L2* norm (sum of squared difference) to measure the similarity between two compared patches. To measure a distance between two overlapped boundary regions, the *L2* norm is illustrated in equation 1:

$$D(S,T) = \sum_{k \in O} [\,(r_S(k) - r_T(k))^2 + (g_S(k) - g_T(k))^2 + (b_S(k) - b_T(k))^2\,] \qquad (1)$$

The distance measure of two overlapped regions *S* and *T* is computed according to *L2* norm distance, which is a sum of squared difference of all pixels in the regions. Where *rgb* are pixel colors at position *k* in red, green, and blue channel respectively. *O* represents the set of pixels in the overlapped boundary region.

The key operation of algorithm is a deterministic search on patch's boundary match. We have to search the patch with the minimum distance among all source candidates. We can formulate this search as nearest neighbors search problem in the high-dimensional space. There are efficient algorithms available [1], since the approximate nearest neighbors is a well-studied problem. Therefore, instead of brute force method we construct a kd-tree comprised of candidate patches to serve as a basis for the remaining query operation. In order to avoid building numerous kd-trees for each of boundary configurations, we choose using four boundaries as compared feature vector. As the illustration in Figure 3, the patch configuration with four non-overlapped boundaries is our adopted solution. In particular, besides saving the time and space of kd-tree construction we find that four boundaries configuration is of worth, while performing constrained texture synthesis or synthesizing tiling textures.

Furthermore, to make a consistent transition between two overlapped regions where two textures match best, we adopt *dynamic programming* technique to find the path with the lowest error. B1 and B2 are two overlapped regions with the same pixel size m×n. The error weight is defined as $e_{i,j} = (B1_{i,j} - B2_{i,j})^2 = (r1_{i,j} - r2_{i,j})^2 + (g1_{i,j} - g2_{i,j})^2 + (b1_{i,j} - b2_{i,j})^2$. We want to find a minimum cost path from the source (row 0) to the destination (row *m-1*). $E_{i,j}$ represents the found minimum cost form pixel location *(i, j)* to the destination (row *m-1*). Therefore, using equation 2 to traverse from *i=m-1* to *i=0*, we can find out the minimum path.

$$\begin{aligned} E_{i,j} &= e_{i,j} \ for \ \mathrm{i} = \mathrm{m}-1 \\ E_{i,j} &= e_{i,j} + \min(E_{i+1,j-1}, E_{i+1,j}, E_{i+1,j+1}) \ for \ \mathrm{i} = 0..\mathrm{m}-2 \end{aligned} \tag{2}$$

Finally, we summarize our texture synthesis framework in the following pseudo code. Besides, Figure 4 and 5 shows the synthesis process and some synthesized results.

Input: source texture with mask image, target texture with mask image
Output: synthesized resultant image, which resembles the source

Procedure TextureSynthesis()

construct noise texture for the result from source pixels

copy the non-synthesized pixels of the target to the result

check suitable patches of the source and build a kd-tree based on overlapped boundary

check to be synthesized patches of the result and insert into a linklist

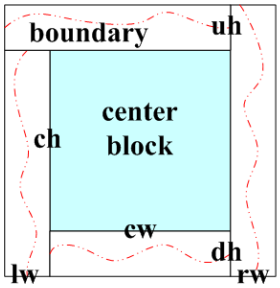
for each node in the linklist of the result

acquire the overlapped boundary from the result based on the node's location

search the best match of the boundary feature from the source kd-tree

adopt dynamic programming for minimum cut path in the overlapped boundary

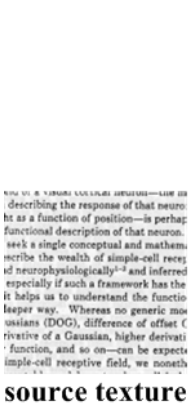
paste the selected source patch onto the result based on minimum cut path



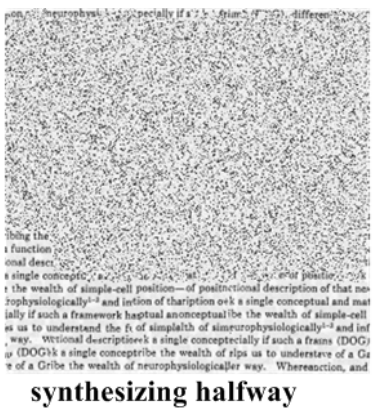
dotted line:
cut path of
minimum cost

sw×sh: source resolution
tw×th: target resolution
pw = lw+cw+rw
ph = dh+ch+uh
#candidate = (sw-pw+1)(sh-ph+1)
for non-tiling source
#candidate = sw*sh
for tiling source
#task = $\lceil tw/cw \rceil * \lceil th/ch \rceil$
boundary_size = pw*ph-cw*ch
feature_vector_size = boundary_size*3
for RGB color

Figure 3: The configuration of adopted patch in our system framework. It is feasible to accomplish a variety of synthesis applications such as tiling texture and constrained synthesis.



source texture



synthesizing halfway

synthesized result

Figure 4: We can synthesize a tiling texture to any desired resolution resembling input source. We use the accelerative techniques of patch-based sampling and kd-tree search, and dynamic programming technique to find minimum cut path between two overlapped regions to stitch the overlapped boundaries in a consistent transition.

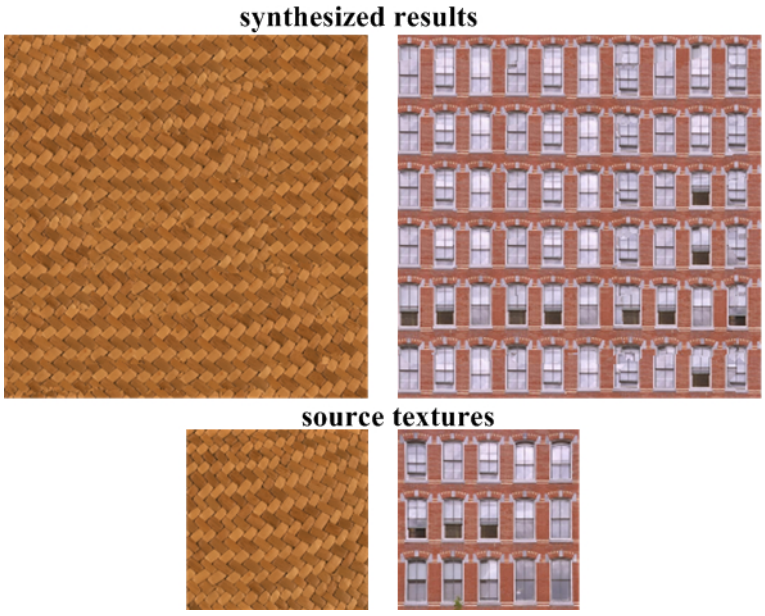


Figure 5: More texture synthesis examples. Our patch-based algorithm works well for a wide category of textures ranging from stochastic to regular.

3.2 Tiling Texture Synthesis

To handle the image boundaries is inevitable, especially for tiling texture synthesis. The usage of tiling texture is ubiquitous in webpage image and image-based rendering environment. We treat the image toroidally so that the synthetic image is tiled seamlessly. In other words, we treat $P(x, y) \equiv P(x \bmod W, y \bmod H)$ while x is out of legal range of image width (*0..W-1*) or y is out of image height (*0..H-1*). Figure 6 demonstrates the result of tiling texture synthesis thanks to the mechanism of toroid and our four boundaries configuration. Notice that the source tiling leads to the joint edges quite striking.

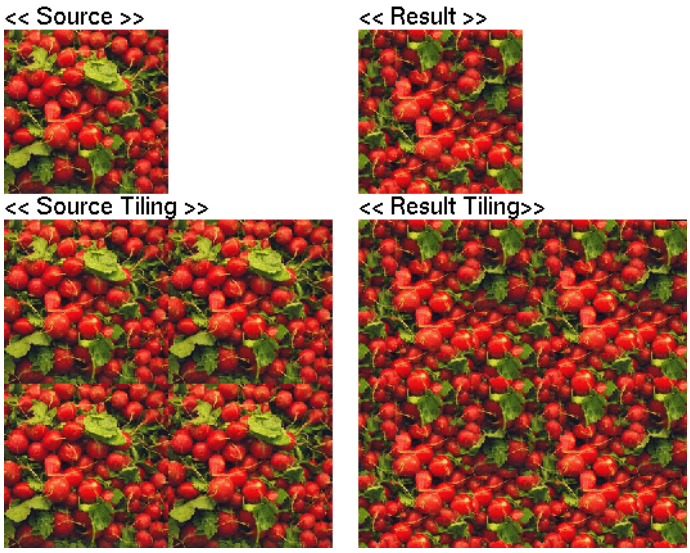


Figure 6: We tile the source texture horizontally and vertically leading to the joint edges quite noticeable. To reduce the undesired artifact of seam, our system adopts patches with four boundaries and treats the result image as toroid.

3.3 Constrained Texture Synthesis

Texture synthesis can also be used as a tool for solving several practical problems. For instance, photographs or scanned images may reveal some scratches, and our algorithms can clean and recover the undesired flaws. With four supplied ingredients (source, source mask, target, and target mask) as input, our system can fill the indicated holes with neighboring texture patches very quickly without any user intervention. The white pixels in the source mask indicate feasible texture patches. Moreover, the black pixels in the target mask define the patches to be synthesized. Figure 7 demonstrates the example of constrained texture synthesis. With the specified masks our framework is simple enough to automatically fill the holes by neighboring patches, which are match-selected.

Figure 7: Example of constrained texture synthesis for hole filling, object removing, and image inpainting etc. In each step system synthesizes one patch in scan-line order, and thus it demands the patches with four matched boundaries to synthesize the tasks in the last

column and the last row.

4 The Framework of Texture Transfer

We propose a non-iterative procedure for patch-based texture transfer so that it performs more than an order of magnitude faster than the counterpart of Efros's multi-pass method [5]. Furthermore, the algorithm is applicable to a wide category of applications such as texture-by-numbers [8], in which a realistic scene is composed of a variety of textures. In this study, we exploit the technique for label controlled texture transfer and demonstrate its employment on landscape gardening. In addition, we further develop an interactive interface for the user painting and controlling the synthesis process.

4.1 Non-iterative Texture Transfer

With the maturation of the patch-based sampling technique, we can easily extend it to the application of texture transfer, which renders the target image with numbers of texture patches from the source sample. In principle, we intend that the bright patches of target image correspond to the bright patches of source texture, and the dark patches of target image have a low match error with respect to the dark patches of the source. In a texture transfer algorithm two aspects of feature match are need to be concerned, which are "*Fidelity Match*" and "*Coherence Match*". The "*Fidelity Match*" (M_f) attempts to efficiently select a source patch, which matches the synthesizing patch of the target according to user specified features. Meanwhile, the "*Coherence Match*" (M_c) attempts to have a consistent transition and to preserve coherence with the neighboring synthesized patches. Efros [5] accomplishes the goal by issuing a composite query of both concerns, which is a weighted sum of these two matches, as following equation $M = \alpha M_c + (1-\alpha) M_f$. Basically, the parameter α determines the tradeoff between the texture synthesis and the fidelity to target. As a consequence, one synthesis pass is difficultly choosing a good parameter α to produce appealing result. Therefore, an iterative method is emerged from him [5] to compensate the drawback.

In this paper, our patch-based texture transfer algorithm adopts a non-iterative framework, which performs more than an order of magnitude faster than Efros's multi-pass procedure. Besides, our method takes into consideration both match principles of target fidelity and neighbor coherence. The crucial scheme we adopted is that we firstly determine the k feasible candidates of "*Fidelity Match*" from source patches, and then search a best candidate of "*Coherence Match*" among them for the target's resultant patch. Obviously, the parameter k determines the tradeoff between the target fidelity and the texture synthesis. A small k leads to more resemblance to the target image, and a large k more neighbors coherence can be. However, a large k also causes a longer time for patch selection. Figure 8 illustrates a resulting example of proposed texture transfer algorithm, and more results are demonstrated in Figure 9. Finally, we summarize this non-iterative algorithm for texture transfer in the following pseudo code:

```
Input: a source texture and a to be transferred target image
Output: resultant image of texture transferred

Procedure TextureTransfer()
    copy all pixels of the target onto the result
    check suitable patches of the source and build a kd-tree based on center feature vector
    check to be synthesized patches of the result and insert into a linklist
    for each node in the linklist of the result
        acquire the center feature vector from the result based on the node's location
        determine k best matches of the center feature vector from the source kd-tree
        select a best match of the overlapped boundary among k center matches
        adopt dynamic programming for minimum cut path in the overlapped boundary
        paste the selected source patch onto the result based on minimum cut path
```

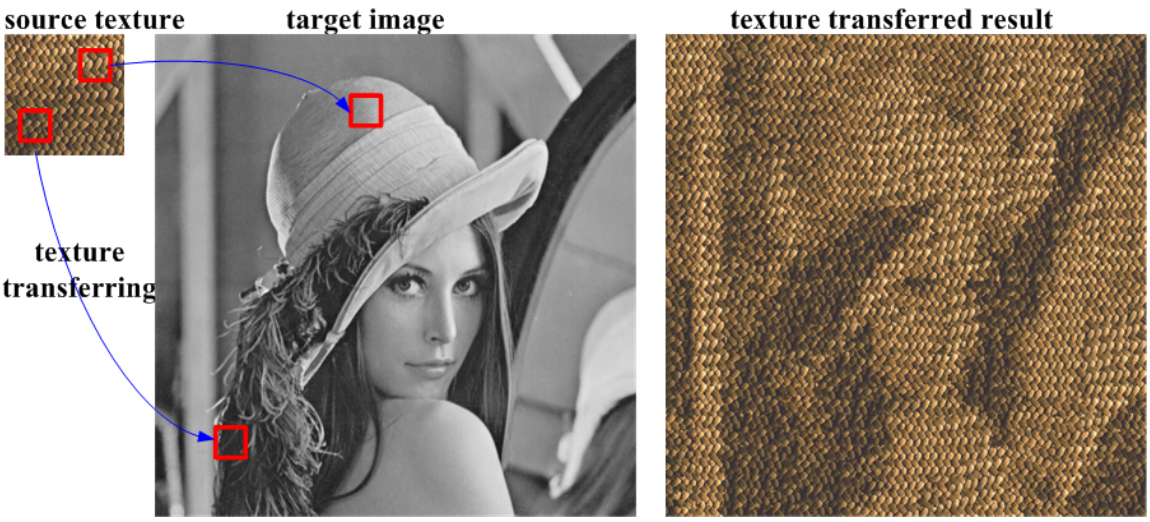


Figure 8: To synthesize a transferred patch, we determine k nearest matches of the center feature vector from the kd-tree composed of source patches. Following, we select the best match of the overlapped boundary among these k center matched candidates.

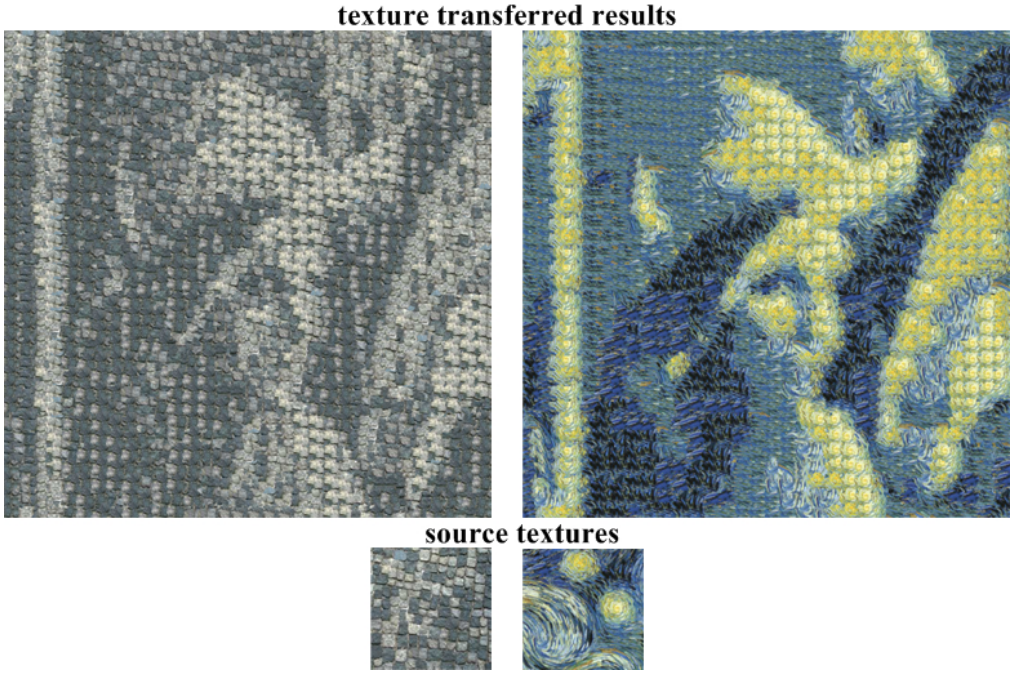


Figure 9: More texture transferred examples. The synthesized result on right shows inconsistent boundary transition because the input source belongs to an image not a texture (doesn't satisfy the stationary property [14]).

4.2 Texture Transfer by Label

Motivated by Ashikhmin [2] and Hertzmann [8]'s working system, we develop a practical system for labeled texture transfer as well. It is straightforward to extend the texture transfer algorithm to apply a labeled example to another labeled image. In practice,

instead of using RGB or grayscale color as feature vector, we may utilize user specified label as feature vector to dominate the process of best match search. In principle, our algorithm allows any kind of information or composite channels of information being used to control the synthesis process. Moreover, we not only develop an automatic framework for labeled texture transfer based on the supply of source and target label maps, but also provide a user interface for the interested user with intuitive control over the synthesis process. Benefiting from one patch sampling a time rather than numerous pixel samplings, the system can provide immediate update by using an accelerative search to the established source kd-tree.

Figure 10: We develop an interactive system for the interested user painting the target label map to control the synthesis. Benefiting from one patch sampling at a time, we can provide the immediate update by issuing a query to the established kd-tree.

5 Results

As the illustration in Figure 3, we adopt four boundaries as to be compared feature vector for statistical constrain. The size of the patch center ($cw \times ch$) is one of parameters specified by the user and it depends on the properties of the given textures. It must be big enough to capture the relevant structure in the texture. In practice, we usually assume that the size of the texture element is known to serve as patch's center size. A smaller center size implies a weaker statistic constrain, but a bigger one costs the more computation time and memory space for constructing the kd-tree. It also lengthens the later search operations. In particular, while the patch center and boundary sizes are properly configured, we simulate the work of pixel-based approach straightforward.

Besides, the size of patch boundaries (*lw-dh-rw-uh*) should be sufficient large to avoid

mismatching boundary feature. A wider boundary zone implies a strong statistical constrain, but it also costs more time and space for the kd-tree construction. In order to improve the system performance we adopt a simple method, which reduces the candidate number of source by successive candidates. In other words, we only add one patch into the kd-tree for every R successive patches. The selection of R is a tradeoff between the synthesis speed and the synthesis fidelity. A large R significant decreases the synthesis time, but it also decreases the synthesis perfection. In our experiment, a small R (<5) produces no visible difference from R=1 (accepts all source patches). Besides, the statistical amounts for candidate number, task number, feature vector size, and total kd-tree size are listed in equation 3:

$$\begin{aligned}
&\#candidate = (sw - pw + 1)(sh - ph + 1) \quad for\ non-tiling\ source \\
&\#candidate = sw * sh \quad for\ tiling\ source \\
&\#task = \lceil tw / cw \rceil * \lceil th / ch \rceil \\
&feature_vector_size = boundary_size * 3 \quad for\ RGB\ color \\
&kd - tree_size = feature_vector_size * (\#candidate)
\end{aligned} \tag{3}$$

Table 1 summarizes the adopted parameters for the demonstrated examples, and Table 2 shows their respective synthesis time. All time are measured on a Pentium 4 - 1.5GHz PC with 128M RAM. Our experiments show that our algorithms are fast enough to enable real-time texture synthesis and generate plausible visual effects.

Table 1: Quantitative descriptions of test examples.

	Texture synthesis (Figure 4)	Texture synthesis (Figure 6)	Object removal (Figure 7)	Texture transfer (Figure 8) (k=30)	Texture transfer (Figure 10) (k=30)
Source texture ($sw \times sh$)	128×128	128×128	200×150	128×128	128×128
Target image ($tw \times th$)	256×256	128×128	200×150	512×512	512×256
Patch center ($cw \times ch$)	32×32	32×32	16×16	16×16	20×20
Patch boundary (lw-dh-rw-uh)	8-8-8-8	8-8-8-8	6-6-6-6	6-6-6-6	6-6-6-6
Candidate patches ($\#C$)	6,561	6,561	5,320	10,201	9,409
Reduced candidates ($\#C/R$) (R=4)	1,641	1,641	1,330	2,551	2,353
Task patches ($\#N$)	64	16	65	1024	338

Table 2: Timing statistics of test examples.

	Texture synthesis (Figure 4)	Texture synthesis (Figure 6)	Object removal (Figure 7)	Texture transfer (Figure 8) (k=30)	Texture transfer (Figure 10) (k=30)
Training time for kd-tree (Tt)	2.77 sec	4.03 sec	0.55 sec	0.63 sec	0.95 sec
Synthesizing time (Ts=(T1+T2)N)	3.56 sec	0.88 sec	1.16 sec	15.90 sec	9.63 sec
Searching a best match patch (T1)	0.052 sec	0.052 sec	0.015 sec	0.013 sec	0.026 sec
Finding minimum cost cut (T2)	0.003 sec	0.003 sec	0.002 sec	0.002 sec	0.002 sec
Total executing time (T=Tt+Ts)	6.33 sec	4.91 sec	1.71 sec	16.53 sec	10.58 sec

6 Conclusions and Future Work

In the paper we present two categories of applications comprising the texture synthesis and the texture transfer. We adopt patch-based sampling method for these two accomplishments, since it runs more than an order of magnitude faster than the pixel-based counterpart and is applicable to a wide range of stochastic and regular textures.

The texture synthesis algorithm takes an input sample and generates synthetic texture of arbitrary size. While the resulting image is not exactly like the original, but perceived by human being to be very similar to the given sample. The dynamic programming technique is used for tackling the problem of consistent transition between two overlapped patches. Besides, an accelerative technique of nearest neighbors search is explored instead of exhaustive search to minify the searching time of best match. Furthermore, we propose the automatic framework to accomplish two particular categories of synthetic textures, the tiling texture and the constrained synthesis. The usage of tiling texture is very conventional in texture-mapped geometric models and webpage images, and the constrained synthesis is applicable to a wide range of applications such as hole filling and object removing.

As to the application of texture transfer, we propose a non-iterative algorithm transferring matched patches from source texture to the target image. We efficiently and significantly diminish the synthesis time compared to the multi-pass counterpart. Our method also takes into account two match principles of target fidelity and neighbor coherence. Furthermore, we succeed in extending the technique for label controlled texture transfer, and demonstrate its employment on landscape gardening. An interactive and controllable interface is provided for the interested user real-time painting, benefiting from one patch sampling a time using an accelerative kd-tree search.

While the current results are encouraging, we present some directions that could be further exploited. In our current implementation, we adopt simple *L2* norm to compute distance metric. To develop a better measure function of distance is an important open area of research. In principle, the image can be of arbitrary dimensions and contents. The feature selection for the matching constrain is also another large open problem. Besides RGB color, grayscale, and label, some other information such as depth, orientation, and another color space, may be used to represent the feature vector in order to improve the synthesis result. Moreover, our patch-based algorithm can be applied to motion synthesis such as ocean waves and clouds [14]. Another interesting direction allows direct synthesis of textures covering three-dimensional objects by proposed patch-based algorithm rather than pixel-based counterparts [12, 15, 16].

Reference

[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*, 45(6):891-923, 1998.

[2] M. Ashikhmin, "Synthesizing Natural Textures," *Proceedings of ACM Symposium on Interactive 3D Graphics*, pp. 217-226, 2001.

[3] J. S. De Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," *Proceedings of ACM SIGGRAPH '97*, pp. 361-368, 1997.

[4] A. Efros and T. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proceedings of International Conference on Computer Vision*, pp. 1033-1038, 1999.

[5] A. Efros and W. T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proceedings of ACM SIGGRAPH 2001*, pp. 341-346, 2001.

[6] P. Harrison, "A Non-hierarchical Procedure for Re-synthesis of Complex Textures," *Proceedings of Central Europe on Computer Graphics, Visualization and Computer Vision*, pp. 190-197, 2001.

[7] D. J. Heeger and J. R. Bergen, "Pyramid-based Texture Analysis/Synthesis," *Proceedings of ACM SIGGRAPH '95*, pp. 229-238, 1995.

[8] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image Analogies," *Proceedings of ACM SIGGRAPH 2001*, pp. 327-340, 2001.

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

[10] K. Perlin, "An Image Synthesizer," *Proceedings of ACM SIGGRAPH '85*, pp. 287-296, 1985.

[11] G. Turk, "Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion,"

Proceedings of ACM SIGGRAPH '91, pp. 289-298, 1991.

[12] G. Turk, "Texture Synthesis on Surfaces," *Proceedings of ACM SIGGRAPH 2001*, pp. 347-354, 2001.

[13] M. Walter, A. Fournier, and M. Reimers, "Clonal Mosaic Model for the Synthesis of Mammalian Coat Patterns," *Proceedings of Graphics Interface*, pp. 82-91, 1998.

[14] L. Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-structured Vector Quantization," *Proceedings of ACM SIGGRAPH 2000*, pp. 479-488, 2000.

[15] L. Y. Wei and M. Levoy, "Texture Synthesis over Arbitrary Manifold Surfaces," *Proceedings of ACM SIGGRAPH 2001*, pp. 355-360, 2001.

[16] L. Ying, A. Hertzmann, H. Biermann, and D. Zorin, "Texture and Shape Synthesis on Surfaces," *Proceedings of 12th Eurographics Workshop on Rendering*, pp. 301-312, 2001.