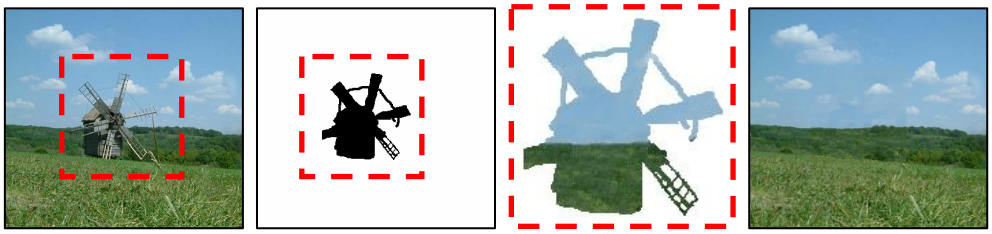
Fast Image Replacement through Texture Synthesis

Chih-Wei Fang (方志偉) and Jenn-Jier James Lien (連震杰)

Robotics Laboratory, Dept. of Computer Science and Information Engineering
National Cheng Kung University

No. 1, Ta-Hsueh Road, Tainan, Taiwan,
Tel: (06) 2757575 ext. 62540, Fax: (06) 2747076
E-mail: {nat, jjlien}@csie.ncku.edu.tw, web: http://robotics.csie.ncku.edu.tw



(a) (b) (c) (d)

Figure 1: (a) Input image. (b) Inverse matte that defines the removed (or replaced) region. (c) The result of the synthetic region. Showing the result by zooming in the replaced region. (d) Output synthesized image.

Abstract

We developed a system including two modules: the texture analysis module and the texture synthesis module. The analysis module is capable of analyzing an input image and performing training using this image data. The properties of principal component analysis (PCA) are used to reduce the dimensions of the data representation and to recombine the appearance of the features. Additionally, the vector quantization (VQ) algorithm is employed to reduce the time spent on comparison. For the synthesis module, the training data is used to rapidly synthesize a large output texture, or is employed to rapidly replace the removed regions of an image. The multi-resolution approach is applied to speed up the procedure of our algorithm: the down-sampling step is the training process and the up-sampling step is in the order of reconstructing (or synthesizing) the replaced region. That is, our system can rapidly obtain a high image quality and a promising result.

1. Introduction

Photographs sometimes include unwanted objects. Conversely, it is sometimes desirable to replace an existing object in a photograph by a new object. Although techniques, which enable the unwanted object to be removed and replaced by a new object, do exist a problem frequently arises in that the shapes and the sizes of the two objects differ, and hence a gap is apparent in the reconstructed image. Consequently, the present study develops an approach to fill in the gap left in an image when an unwanted object is removed.

Most existing algorithms are based on obtaining a minimum difference between the synthesizing patches. These represent having the minimum parallax in vision. In this approach, the continuous structures are important and the discontinuous features are conscious. However, these methods are very slow and result in a loss of definition in the reconstructed images. Clearly then, there exists a requirement to develop enhanced image replacement algorithms. Accordingly, the present study develops a method in which an approximation is used to establish suitable patches with which to fill the gap in the image. In the developed process, additional techniques are adopted in order to accelerate the process without losing its detail features.

2. Related Work

Several researchers have developed gap-filling methods based on an examination of the neighborhood pixels [9], [23], [24], [25]. However, these methods have a gradual change in two different sources. Hence, sharp-pointed edges become blurred. Some studies use patch-based approaches to synthesize image, but may make the structure of reconstructed image discontinuous [12], [18].

It has been shown that multi-resolution approach provides a reasonable technique for obtaining an initial value or for developing an approximate result, called push-pull [2], [4], [8], [27]. The approximate result can then be used as the basis to develop a superior result. Although this method can yield reasonable results, it involves the use of repeated reconstruction procedures, with the result that errors can occur. Furthermore, these errors can be compounded in the subsequent reconstruction steps.

In an alternative approach, an inverse matte is used to preserve the necessary background while removing the unnecessary region [3], [10]. Similar patches are then developed to fill in the resulting holes. In this technique, weighting information is used to blend patches that will lose the detailed features.

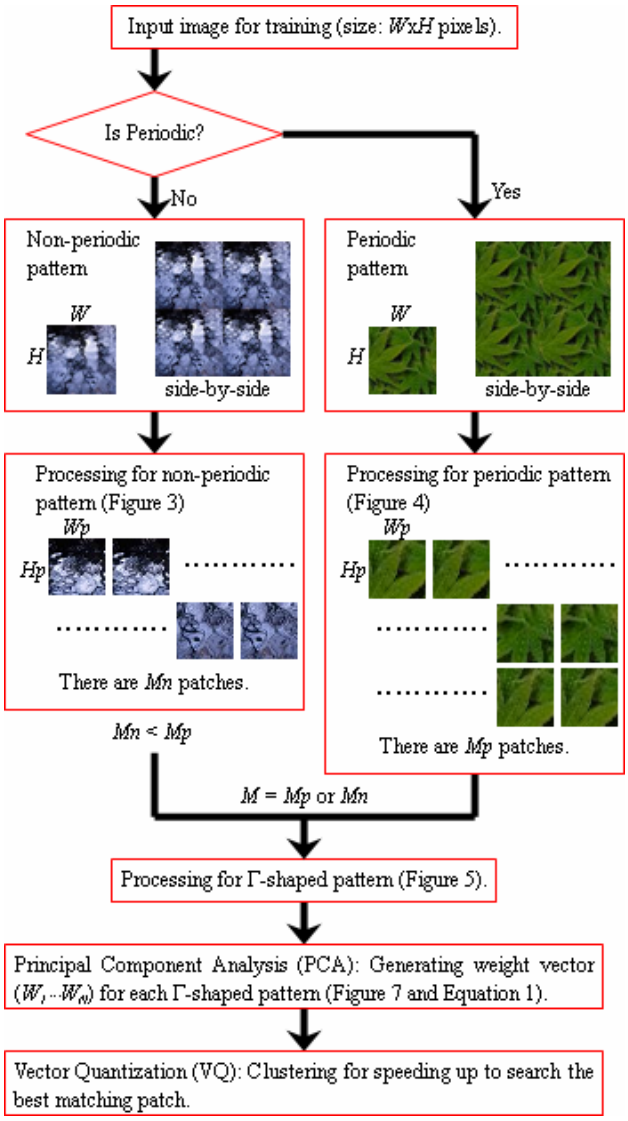


Figure 2: Framework of the texture analysis (training) module.

Recent studies have presented the use of analysis and sampling techniques for the regular or near-regular patterns, and have then used tiles to synthesize the image [6], [13], [14], [17]. These approaches commence by identifying the features or structures. These are then arranged as skeletons images, and the texture is wrapped onto these skeletons [1], [15], [26], [28]. Such methods are suitable for the regular or near-regular patterns. They can solve the discontinuous condition. But these methods are inapplicable to non-structured and non-periodic images.

In nature, the noises are Gaussian signals. These noises spread everywhere and cause a photograph to be truer and more pellucid. In order to merge the fragments more smoothly and with no seam, Drori et al. [3] used a Gaussian mask to blend the various fragments. This approach successfully creates a flawless image. However, the results tend to be more blurred and some detailed feature information is lost.

To solve above existing problems, this paper proposes two modules: texture analysis (training) module and texture synthesis module. The texture analysis module is capable of analyzing input images (or textures) and training by using these data very rapidly (Section 3). The synthesis module can synthesize a large image (or texture) based on these training data (Section 4). The synthesis process performs on a real-time basis. The synthesis process is then modified slightly to fit the image replacement process (Section 5). Additionally, the multi-resolution technique is also applied to speed up image replacement process. Section 6 presents the current experimental result and evaluates time cost of the training and reconstruction (or synthesis) process. Finally, Section 7 presents some conclusions and enhancements to our developed approach.

3. Texture Analysis (Training) Module

The output big image is synthesized based on the small single input texture. The input image is divided into several patches, which are investigated in the subsequent analysis process. The important data of the input image are selected for training purposes. Principal component analysis (PCA) is employed to reduce the dimensions of the data, and vector quantization (VQ) is adopted to reduce the time required for comparison. Figure 2 illustrates the system framework extending from the input image to the output image.

3.1. Processing for non-periodic and periodic patterns

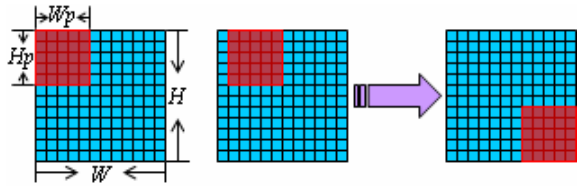


Figure 3: Processing for non-periodic pattern. A pixel-by-pixel shifting method is used to divide the input texture into M patches, where $M = (W–Wp+1)\times(H–Hp+1)$, in which W is the width of the input image (or texture), H is the height of the input image, Wp is the width of the patch, and Hp is the height of the patch.

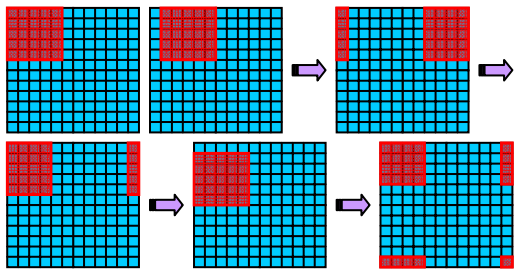


Figure 4: Processing for periodic pattern. The pattern is divided into M patches, where $M=W\times H$. When the patch reaches the boundary of the input image, in order to form a complete patch, additional pixels are supplied from the opposite border.

The training data is obtained by cropped patches (or window) of pre-defined size (WpxHp pixels) from the original input image (WxH pixels). Two different patch-dividing schemes are employed depending on whether the image has a non-periodic pattern or a periodic pattern, as shown in Figure 3 and Figure 4, respectively. Periodic patterns have continuous veins between two equivalent patterns when positioned side by side (see Figure 2). Non-periodic patterns do not have this property. When non-periodic patterns are placed side by side, there is a visible discontinuity seam between the textures (also see Figure

2). Therefore, periodic patterns can be divided into more patches than non-periodic patterns, as shown in Figure 3 and Figure 4. For example, an input image of size 64x64 pixels can be divided into 4096 patches of size 32x32 pixels if the pattern is periodic, but can only be divided into 1089 patches if the pattern is non-periodic.

But the number of patches is not the most important criterion in deciding whether the synthetic result is good or bad. Instead, if the patch size is too small, each patch will contain insufficient information to identify the correct patches. However, if the size of the patch is too large, it will result in fewer patches, with the consequence that the system will not have a sufficient number of patches to perform the best match. These problems can result in the output image having a visible seam between the patches.

3.2. Processing for Γ-shaped pattern

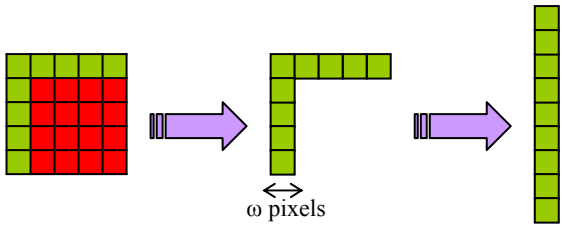
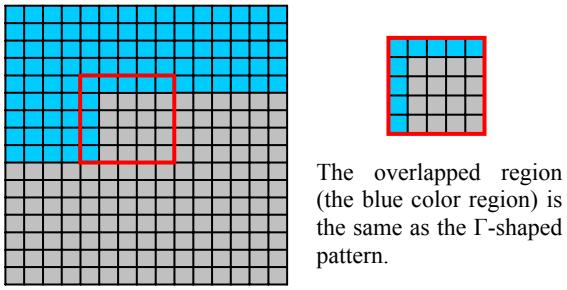


Figure 5: Using only the left border and the top border in each search patch, which has a thickness of ω pixels (ω=2 pixels in this study) and is called a Γ-shaped pattern. Each pattern contains K pixels, where $K = \omega \times (Wp + Hp - \omega)$. So the size of each patch is reduced from a WpxHp dimensional (pixel) vector to a K-dimensional (pixel) vector ($P_1 \ldots P_K$). ($K << Wp$xHp)



Output synthesized image.

The overlapped region (the blue color region) is the same as the Γ-shaped pattern.

Figure 6: For the output synthetic image and the search window (or patch), the blue region has been already completed; the gray region has a random value originally, and is not synthesized; and the red window indicates the region undergoing synthesis.

Taking the whole pieces of the patch as training data may lead to an overestimation of the underlying structure of the patch and will certainly increase the length of the training time. In addition, searching the matching patches by considering their whole contents generally produce unsatisfactory results since the whole contents tend to be quite different from the initial random values (which will be mentioned in Section 4 and Figure 8) and it may cause the rim effect to become distinct. According to our experience, more suitable approach is to choose just the left border and the top border of the training patch as the training data, as shown in Figure 5. An additional reason for selecting just the border part is that the image scanning convention adopted in this study is from top-left to bottom-right, as shown in Figure 6.

3.3. Principal Component Analysis (PCA)

$$\begin{bmatrix} P_{11} & P_{12} & \cdots & \cdots & \cdots & P_{1M} \\ P_{21} & P_{22} & \ddots & & & P_{2M} \\ \vdots & & & \ddots & & \vdots \\ P_{K1} & P_{K2} & \cdots & \cdots & \cdots & P_{KM} \end{bmatrix} \xrightarrow{\text{PCA}} \Psi = \begin{bmatrix} E_{11} & E_{12} & \cdots & E_{1K} \\ E_{21} & E_{22} & \cdots & E_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ E_{N1} & E_{N2} & \cdots & E_{NK} \end{bmatrix}$$

Figure 7: M patches can be obtained from Section 3.1. Each patch has K elements (see Section 3.2.) During training, PCA is used to transform the original $K \times M$ matrix to an $N \times K$ matrix, where $N < K << M$. And there are N eigenvectors.

PCA is applied to the training data to obtain their eigenspace, Ψ, as shown in Figure 7. Two important properties of PCA are employed to obtain the best performance: (1) reducing the dimensions of the data representation from K dimensions to N dimensions, where $N<K$, as shown in Figure 7. This can reduce most of the time complexity operations, and hence dramatically increases the performance; (2) recombining the appearance of the features while maintaining the coherence of the characteristic content. After the PCA process and a sort based on the eigenvalues with corresponding eigenvectors, it is found that the first several eigenvectors control the global geometrical structure, while the middle eigenvectors control the local features. Meanwhile, some noises are controlled by the last few eigenvectors. These noises cause the photograph to appear truer, but have no influence on the geometric structure. Therefore a good matching structure need only compare the first several eigenvectors. Consequently, this study uses only the first N eigenvectors, whose corresponding eigenvalues occupy 98% of total eigenvalues, for comparison purposes to identify the patch which results in the best matching of the geometrical structure. This approach makes the result more fitted visually.

Because there are M patches for any single input image, and K elements in each patch P (i.e. $P=[P_1 \ldots P_k]^T$) projected to N eigenvectors ($E_{1i} \ldots E_{Ni}$, where $i=1 \sim K$), as shown in Equation (1).

$$\begin{bmatrix} E_{11} & E_{12} & \cdots & \cdots & \cdots & E_{1K} \\ E_{21} & E_{22} & \ddots & & & E_{2K} \\ \vdots & \vdots & & \ddots & & \vdots \\ E_{N1} & E_{N2} & \cdots & \cdots & \cdots & E_{NK} \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ \vdots \\ \vdots \\ P_K \end{bmatrix} = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_N \end{bmatrix} \quad (1)$$

So the corresponding N-dimensional weight vector can be obtained. This weight vector represents the information relating to the origin of each patch. Therefore, there are a total of M N-dimensional weight vectors.

3.4. Vector Quantization (VQ)

Searching for the best matching pattern from the eigenspace Ψ during the synthesis process is a computationally expensive task. In order to speeding the synthesis process, the training data is initially projected onto the eigenspace Ψ in order to retrieve the weight vectors. The data is then separated into C clusters by means of VQ. Using this approach, the computational time can be reduced from $O(M)$ to $O(\sqrt{M})$.

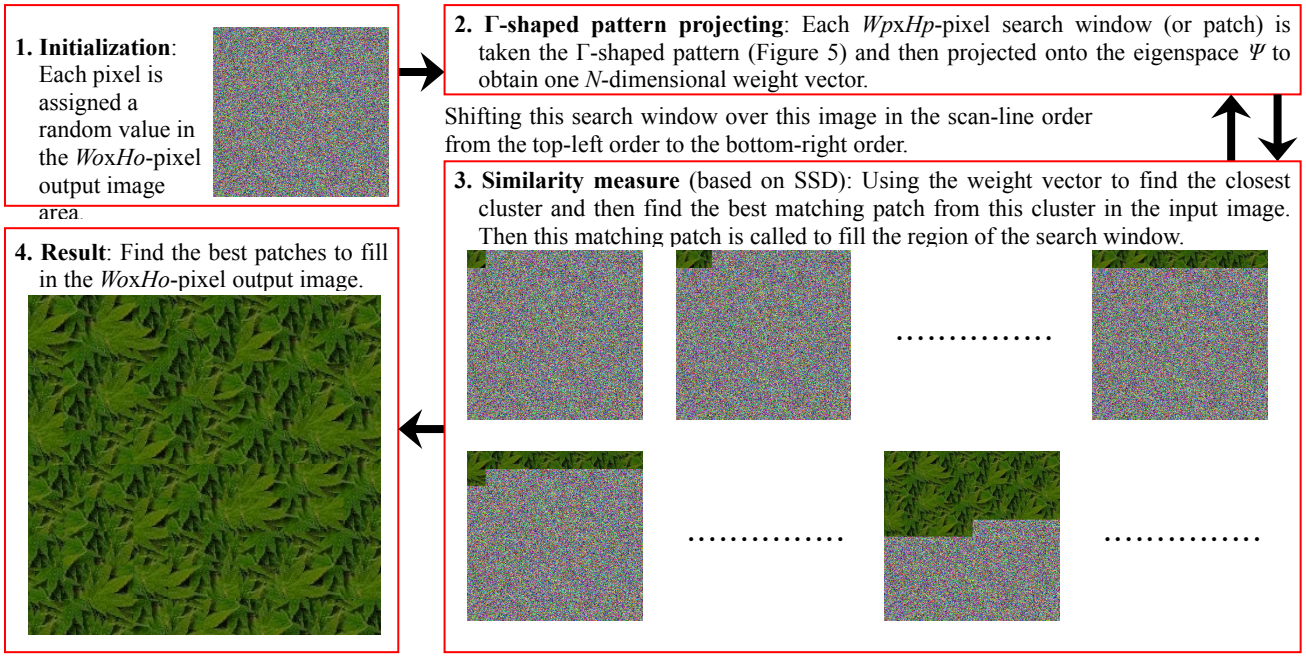


Figure 8: Framework of the texture synthesis module. All images here have the same size of *Wox*H*o* pixels.

The use of Tree-structured Vector Quantization (TSVQ) has the potential to reduce computation costs [22]. However, there is a risk in applying this technique in the present case. For example, any misclassification in the parent node will affect the final classification result. Therefore, in our work, the closest cluster is directly identified according to its mean vector, and this cluster is then searched exhaustively to locate the best matching vector.

4. Texture Synthesis Module

The texture synthesis module comprises four steps, as shown in Figure 8. Given a predefined size (*Wox*H*o* pixels) of the RGB image, the RGB values of each pixel in the output image are initialized randomly from 0~255, individually. The whole content of the output image is retrieved by shifting a search window (or patch) of size *Wp* x *Hp* over this image buffer in the scan-line order mentioned previously and acquiring the Γ-shaped pattern from the region of the search window.

Traditionally, synthesis processes using PCA reconstructed the desired patches via the linear combination of the first *N* eigenvectors. This study attempted to project the whole patch having random value for each pixel onto the training patch's eigenspace, and then to reconstruct the whole patch in one step for the output synthetic image. However, in implementation, it was found that this approach produced a blurred result and that much of the original detail was lost.

Accordingly, an indirect approach was developed in which the Γ-shaped pattern of each search window, which contains random value for each pixel, was first projected onto the eigenspace Ψ to have the corresponding weight vector. Second, based on the similarity measure of using the sum of squared difference (SSD) between the weight vectors of this search window and *M* patches in the original input image, this weight vector was classified to the closest cluster and then this cluster was searched to find the best matching weight vector. Third, the matching patch, which corresponds to the best matching weight vector, in the original input image was taken to fill the region of this search window.

The proposed system generates the output image in scan-line order. As discussed above, synthesis performed by considering the whole content of patch may lead to a poor result. Hence, two schemes are introduced to overcome this problem: (1) reducing the shift step size of the search window (or patch), which also increases the overlap region, then applying weighting by distance [24], or (2) using dynamic programming (DP) [5], [11] to find the minimum error for cutting. However, these methods blur the image details and the boundaries. Furthermore, their use increases the size of the overlap region and reduces the shift distance of the pixels. Consequently, the time required for the synthesis process increases.

So according to our experimented results; the best overlap border (region) is a 2-pixel width, which is also the thickness ω of the Γ-shaped pattern. A single pixel width (ω=1) does not contain sufficient information to identify an appropriate patch. Although taking a width of greater than 2 pixels will yield the same results as an overlap border with 2-pixel width, this approach will increase the time of operation significantly. In addition, considering the process of the overlap border, averaging process will cause slightly blurred contents. Hence, in this study, only the part of the search window having random values is replaced by the corresponding region of the matching patch. The content of the overlap border maintains the same without replacement. This process can make the result more fitted visually without losing its geometrical structure. The reason is as we discussed in Section 3.3. Especially, using the overlap border as the Γ-shaped pattern can make the geometrical structures between patches look real and continuous.

5. Image Replacement through Texture Synthesis

Both texture analysis module and texture synthesis module can be applied to the image replacement of the particular region for a given image, as shown in Figure 1. Here an original input image is selected and is annotated as I_0 (Figure 1a). And the regions containing the region containing the replaced pixels are manually acquired and are called inverse matte, β_0 (Figure 1b).

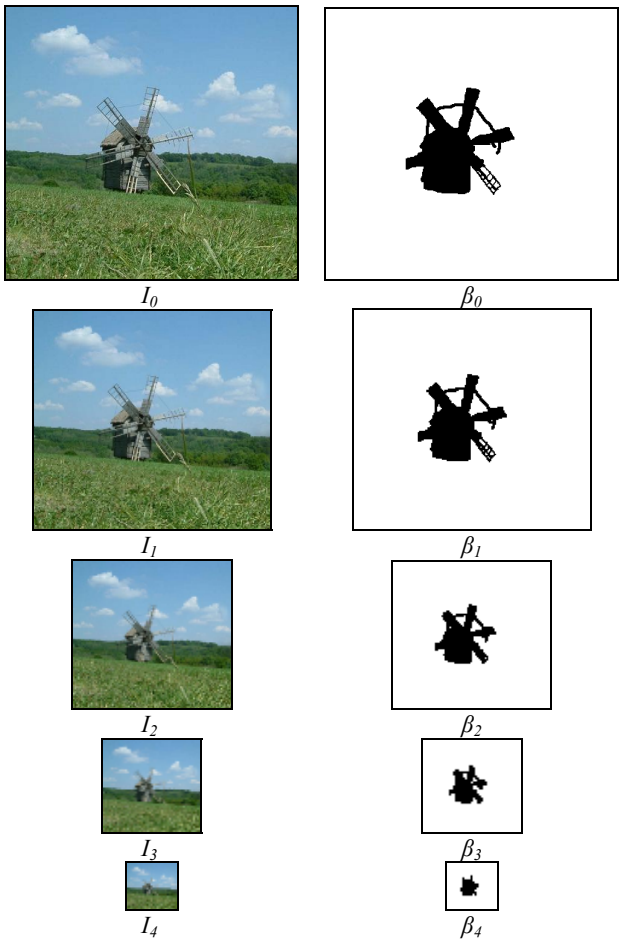


Figure 9: We have input image I_0 and inverse matte β_0, and then do l times down-sampling ↓ to get each level of input image I_i and inverse matte β_i, i=1~l.

This matte is binary, such that the white regions which are to be retained are set to 1, and are the known regions, while the black regions which are to be replaced regions are set to 0, and are the replaced region. In addition, the replaced regions can comprise many sub-regions, must contain the removable objects, can exceed the boundaries of the removable objects, and can be of any shapes. But too many or too large replaced will lead to a poor quality result. The known region serves as the source of the replaced regions. The analysis module developed in Section 3 is used for training, while the synthesis module presented in Section 4 is employed for filling the replaced region. But some revisions are performed to be suitable for the process of image replacement.

5.1. Preprocessing Using the Multi-resolution Approach

Traditionally, others' approaches for the process of image replacement must assign an initial value to each pixel in the replaced region, which here is defined as the foreground region. These initial values are used to search the background region in order to find suitable patches with which to fill the replaced region. Hence, we apply the multi-resolution approach to our system, so we do not need the above-mentioned method. The major purpose is to avoid the condition, which is unable to converge and getting the wrong result of reconstructed image. The multi-resolution approach makes the input image I_0 and the inverse matte β_0 to do l times down-sampling ↓, then to get each level of input image I_i and inverse matte β_i, i=1~l, as shown in Figure 9. The value of the level i can serve as the initial values of the level i-1, and the size of the replaced region at the level l of image I_l is less than that of one patch, so we need not assign an initial value to each pixel in the replaced region. In addition, B_i denotes the background region, which must be preserved for training data, as in Equation (2),

$$B_i = I_i \beta_i \quad , \quad i = 0 \sim l \tag{2}$$

and the foreground region F_i, as in Equation (3),

$$F_i = I_i \overline{\beta}_i \quad , \quad i = 0 \sim l \tag{3}$$

is then utilized to search the background B_i to locate the best patch with which to fill in the replaced region.

5.2. Training Process Based on Background Region

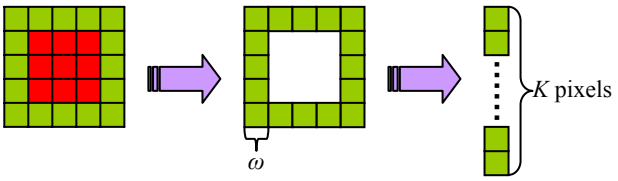


Figure 10: Acquire the four borders with thickness ω (ω=2) pixels for each search patch. There are K pixels in each patch, where $K = 2\times\omega\times(Wp+Hp-2)$.

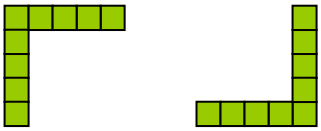
For the training process, we get the whole piece of the patches from the background regions, which are preserved for training data from the level 1 image I_1 to the level l image I_l. We drop the patches at the level 0 image I_0 for the following reasons: (1) The total amount of patches are too many (usually 100~300 thousands of patches) and cost much time in many unnecessary operators. (2) Using plenty of patches is unnecessary to fill in the removed (or replaced) region. VQ need to separate such many patches into more clusters for speeding up searching the matching patches. Furthermore clusters raise the probability of matching the wrong patches, and the time of VQ training increases exponentially following the increasing cluster. (3) The level 0 image I_0, has more noise influence on the process of PCA training. That results easily in getting the wrong information. (4) While reconstructing the lower level images (ex. I_2~I_l), the system need stronger structural patches to fill the replaced region. Because of applying multi-resolution to input image I_0, the patches at the lower level of image structurally are more powerful.

The training process of image replacement is the same as that of the texture analysis in Section 3. However, if the same Γ-shaped pattern of the search window (or patch) is applied (Figure 5), the fragment between two different textures will be mapped by only one texture rather than being mapped by the mixing of two different textures. Hence, the search window is modified, as shown in Figure 10, and the performance looks promising as shown in Figure 11 , 13, and 15.

Here, we analyze the problem when applying the Γ-shaped pattern to the process of image replacement. For example, in Figure 11(a), the top half of the image is sky, the bottom half of the image is grass, and the middle part shows some trees. If our previous proposed approach is used to remove the tree region and to fill in the removed (or replaced) region by

(a) (b)

Figure 11: (a) Input image with the removed object between two different appearances (textures) and including light effect. (b) Output image reconstructed using the modified patch shown in Figure 10. The generated patches merge well with the original background in terms of their geometrical structure. However, the color does not match well because of the light effect.



(a) (b)

Figure 12: (a) The top-left Γ-shaped pattern. (b) The bottom-right Γ-shaped pattern.

using the Γ-shaped pattern to reconstruct the image. It can be seen that the resulting image is filled well near the sky in the original location of the trees, but the region of the image towards the grass is filled in with sky data and hence does not merge satisfactorily with the grass region.

In an alternative approach, two patterns are used, i.e. the top-left Γ-shaped pattern and the bottom-right Γ-shaped pattern, as shown in Figure 12. Initially, the top-left Γ-shaped pattern is used to reconstruct the image in a top-down order. The bottom-right Γ-shaped pattern is then employed to reconstruct the image in a bottom-up order. The two steps are then repeated. In the case of Figure 11, the replaced region is initially filled completely with sky data and then filled in completely with grass data. As the two-step process is repeated, these two actions are repeated continuously and convergence cannot be achieved.

5.3. Completion Process

In order to practice for convenient, one minimum rectangular boundary covering the entire replaced region is identified. Then the same texture synthesis process as in Section 4 is applied to this rectangular region. We only change the compared part to four borders of the patch, whose thickness is ω pixels, as shown in Figure 10. In our algorithm, the reconstructed order is from the level l image I_l to the level 0 image I_0. First, the system reconstructs the level i (i=l) image I_i. The replaced part is reconstructed (or synthesized), called I_i'. Only the reconstructed part of I_i' is preserved at position F_i. The other parts are replaced by the original background B_i. This gives the first step of the reconstructed image C_i, as in Equation (4).

$$C_i = I_i \beta_i + I_i' \overline{\beta}_i \tag{4}$$

The system applies the up-sampling ↑ technique from the reconstructed image C_i at level i of and proceeds to only replace the level i-1 image I_{i-1} at position F_{i-1}, as in Equation (5).

$$I_{i-1} = I_{i-1} \beta_{i-1} + (C_i \uparrow) \overline{\beta}_{i-1} \tag{5}$$

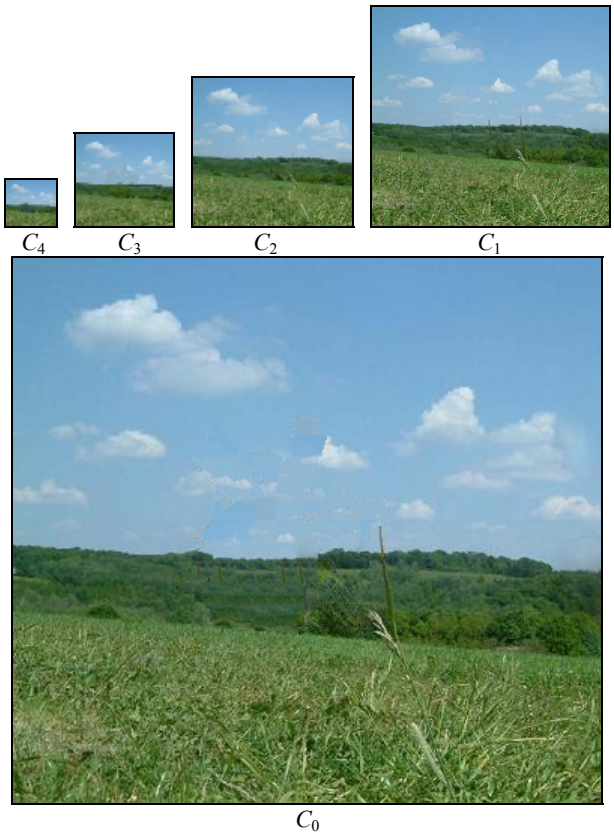


Figure 13: Show reconstructed image C_i from lowest level 4 to the origin level 0, i=0~4. C_i does up-sampling ↑, and then serves I_{i-1} as initial value for searching the matching patch to fill the removed (or replaced) region of C_{i-1}.

Then the system repeats above process of Equations (4) and (5) from the level l-1 image I_{l-1} (i=l-1) to the level 0 image I_0 (i=0). But the process of Equation (5) does not require at the last time (i=0). According to our experiments, when the system reconstructs directly the level 0 image I_0, the output image is often converge to the worse result or need to repeat many times (usually 70~100 times) of reconstructing processes. But using the multi-resolution approach, our system can reconstruct image not only fast but also promising, as shown in Figure 13.

6. Experimental Results

In acquiring the experimental statistics presented below, the process was performed ten times and the average time calculated. The experiments were performed on a personal computer with a AMD K8 3200+ (2.0 GHz) processor, 512MB DDR SDRAM, and the Windows XP SP2 operating system. The current system was written in Visual C++.

Here, Figure 14 shows various results by using our developed modules in Section 3 and Section 4. The width of the patch, Wp, is 32 pixels, the height of the patch, Hp, is 32 pixels, and the thickness of the patch, ω, is 2 pixels. As shown in Table 1, the training process can be completed rapidly, and the time complexity is proportioned to the size of the input image. Furthermore, the proposed method can synthesize a seamless texture rapidly, as shown in Table 2. Table 3 and Table 4 show the statistics of the Figure 9 and Figure 13. The required time is also proportioned to the size of the output image and the value of N for the dimension of eigenspace. Figure 15 shows various results of Section 5. Hence, the size of the patches is modified

such that *Wp* is 16 pixels and *Hp* is 16 pixels. However, the same thickness as prescribed in Section 5 is retained. The results have a higher quality and are also finished rapidly.

Table 1: Average training time in Section 3 for input images of various sizes. Units of time: milliseconds (ms).

Size of input image (pixels)	64x64	96x96	128x128
PCA	250	437	875
Projecting to eigenspace	359	703	1656
VQ	203	391	953
Total time (ms)	**812**	**1531**	**3484**

Table 2: Average synthesizing time in Section 4 for output images of various sizes.

Size of output image (pixels)	200x200	300x300	400x400	600x600
Synthesis time (ms)	11	37	59	140

Table 3: The synthesized information of each level in Figure 9 and Figure 13 for multi-resolution approach. "~0" means close to 0 ms.

Level	Width (pixels)	Height (pixels)	Number of whole patches	Synthesis time (ms)
0	392	364	114034	63
1	196	182	24272	16
2	96	91	4024	~0
3	49	46	110	~0
4	25	23	0	~0

Table 4: The time in Figure 9 and Figure 13 for various processes. The total time of the training and synthesis processes include the processes of the multi-resolution approach, PCA, projection (the patches are projected onto eigenspace Ψ to obtain the weight vectors), VQ, and synthesis.

Method		Time (ms)
Training process	Multi-resolution	32
	PCA	937
	Projecting to eigenspace	1828
	VQ	437
Synthesis process	Synthesis	79
Total time		**3313**

7. Conclusions and Future Work

We developed a system including two modules: the texture analysis and synthesis modules. This approach is able to be applied to the two different purposes: the synthesis of a large image, and the replaced of local removed region. The analysis module can reduce dimension of the training data and cluster these data. When processing the synthesis step, the analysis module can reduce operation time significantly. The synthesis module can synthesize a large output image very fast and keep geometrical structures and veins continuous. The same process can be used to replace the removed regions. Additionally, the multi-resolution technique is applied to image replacement. The down-sample step is used for analysis process as training data, and the up-sampling step is used for reconstructing process as initial value. So it makes the system to obtain more realistic image very fast.

In general, the sum of squared difference (SSD) is used as the similarity measure. This study uses this measure to find the best matching weight vector. However, lacking the property of direction can lead to mismatching errors. Therefore, it is advisable to replace the SSD method by probability-based techniques [20] or fuzzy feature contrast (FFC) approaches [19].

Considering the lighting effect in Section 5, if the image is influenced by lighting effects, the synthesis results may violate the physical property of lighting. Hence, the input image must be processed first, i.e. the lighting information should initially be extracted. After synthesizing the image, the lighting information can then be added back to the output image [21], [16]. Alternatively, the normal direction of the lighting in the background region can be calculated. The information of these normal directions can then be used to interpolate the replaced region in order to have better synthesis result [7].

8. References

[1] Ashikhmin, M., "Synthesizing Natural Textures," In *ACM Symposium on Interactive 3D Graphics*, 217–226, 2001.

[2] Brooks, S., and Dodgson, N. A., "Self-Similarity Based Texture Editing," *ACM Transactions on Graphics, 21*, 3, 653–656, 2002.

[3] Drori, I., Cohen-Or, D., and Yeshurun, H., "Fragment-Based Image Completion," *ACM Transactions on Graphics, 22*, 3, 303-312, 2003.

[4] De Bonet, J. S., "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," In *Proceedings of ACM SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 361–368, 1997.

[5] Efros, A. A., and Freeman, W. T., "Image Quilting for Texture Synthesis and Transfer," In *Proceedings of ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 341–346, 2001.

[6] Efros, A. A., and Leung, T. K., "Texture Synthesis by Non-parametric Sampling," In *International Conference on Computer Vision*, 1033–1038, 1999.

[7] Fang, H., and Hart, J. C. 2004. "Textureshop: Texture Synthesis as a Photograph Editing Tool," *ACM Transactions on Graphics, 23*, 3, 354-359.

[8] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F., "The Lumigraph," In *Proceedings of ACM SIGGRAPH 96*, ACM Press, 43–54, 1996.

[9] Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H. "Image analogies," In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, 327–340, 2001.

[10] Igehy, H., and Pereira, L. "Image Replacement through Texture Synthesis," In *IEEE International conference on Image Processing*, vol. 3, 186–189, 1997.

[11] Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A., "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Transactions on Graphics, 22*, 3, 277–286, 2003.

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

[13] Lin, W.-C., Hays, J. H., Wu, C., Kwatra, V., and Liu, Y., "A Comparison Study of Four Texture Synthesis Algorithms on Regular and Near-regular Textures," Tech. Rep. CMU-RI-TR-04-01, The Robotics Institute, Carnegie Mellon University, 2004.

[14] Liu, Y., Collins, R., and Tsin, Y., "A Computational Model for Periodic Pattern Perception Based on Frieze and Wallpaper Groups," *IEEE Transaction on Pattern Analysis and Machine Intelligence, 26*, 3, 354–371, 2004.

[15] Liu, Y., and Lin, W.-C., "Deformable Texture: the Irregular-Regular-Irregular Cycle," In *Texture 2003, The 3rd International Workshop on Texture Analysis and Synthesis*, 65–70, 2003.

[16] Liu, Y., Lin, W.-C., and Hays, J., "Near-Regular Texture Analysis and Manipulation," *ACM Transactions on Graphics, 23*, 3, 368–376, 2004.

[17] Liu, Y., Tsin, Y., and Lin, W.-C., "The Promise and The Perils of Near-regular Texture," *International Journal of Computer Vision (In Press)*, 2004.

[18] Nealen, A., and Alexa, M., "Fast and High Quality Overlap

Repair for Patch-Based Texture Synthesis," In *Proceedings of Computer Graphics International 2004*, 582-585, 2004.

[19] Santini, S., and Jain, R. "Similarity Measures," In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*, 9, 871-883, 1999.

[20] Schneiderman, H., and Kanade, T., "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition," In *IEEE Computer Vision and Pattern Recognition Conference*, 45–51, 1998.

[21] Tsin, Y., Liu, Y., and Ramesh, V., "Texture Replacement in Real Images," In *IEEE Computer Vision and Pattern Recognition Conference*, 539–544, 2001.

[22] Wei, L.-Y., "Deterministic Texture Analysis and Synthesis using Tree Structure Vector Quantization," In *Proceedings of the XII Brazilian Symposium on Computer Graphics and Image Processing*, 207-214, 1998.

[23] Wei, L.-Y., *Texture Synthesis by Fixed Neighborhood Searching*, PhD thesis, Stanford University, 2002.

[24] Wei, L.-Y. "Texture Synthesis from Multiple Sources," *ACM Transactions on Graphics*, 1-1, 2003.

[25] Wei, L.-Y., and Levoy, M., "Fast Texture Synthesis using Tree-structured Vector Quantization," In *Proceedings of ACM SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 479–488, 2000.

[26] Wu, Q., and Yu, Y., "Feature Matching and Deformation for Texture Synthesis," *ACM Transactions on Graphics, 23*, 3, 362-365, 2004.

[27] Yamauchi, H., Haber, J., and Seidel, H.-P., "Image Restoration using Multiresolution Texture Synthesis and Image Inpainting," In *Proceedings of Computer Graphics International*, 120-125, 2003.

[28] Zelinka, S., and Garland, M., "Towards Real-Time Texture Synthesis with the Jump Map," Eurographics Organization, 99-104, 2002.

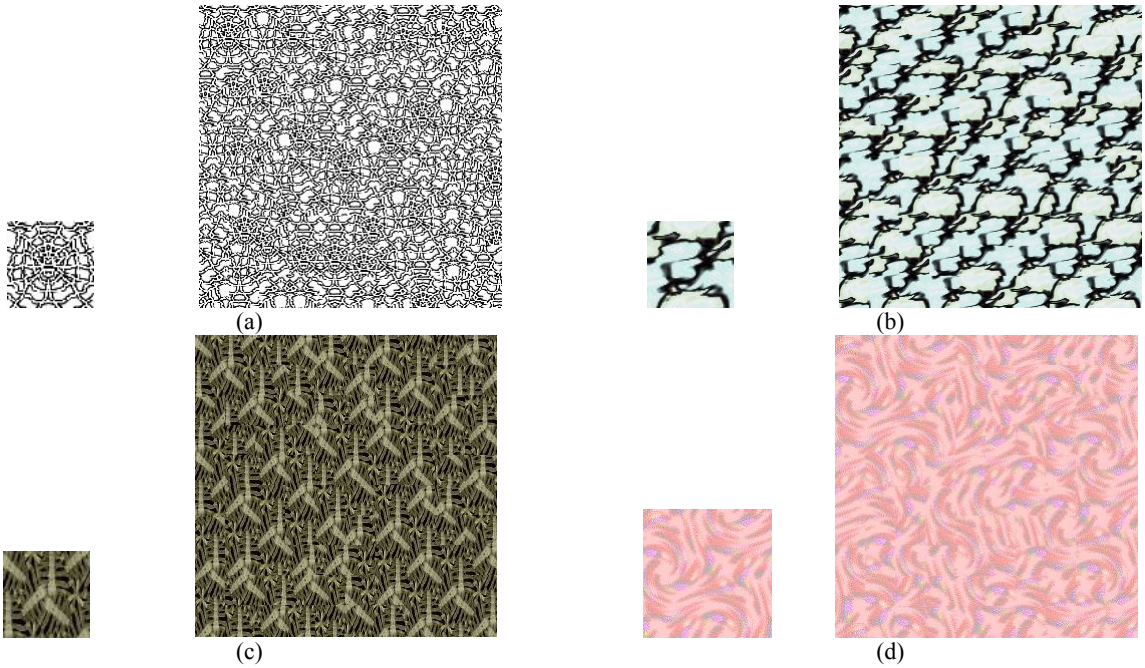


Figure 14: Left column: Input images (a)~(c) (size: 64x64 pixels) and (d) (size: 128x128 pixels). Right column: Output images using the proposed approach described in Sections 3 and 4 (size: 300x300 pixels).

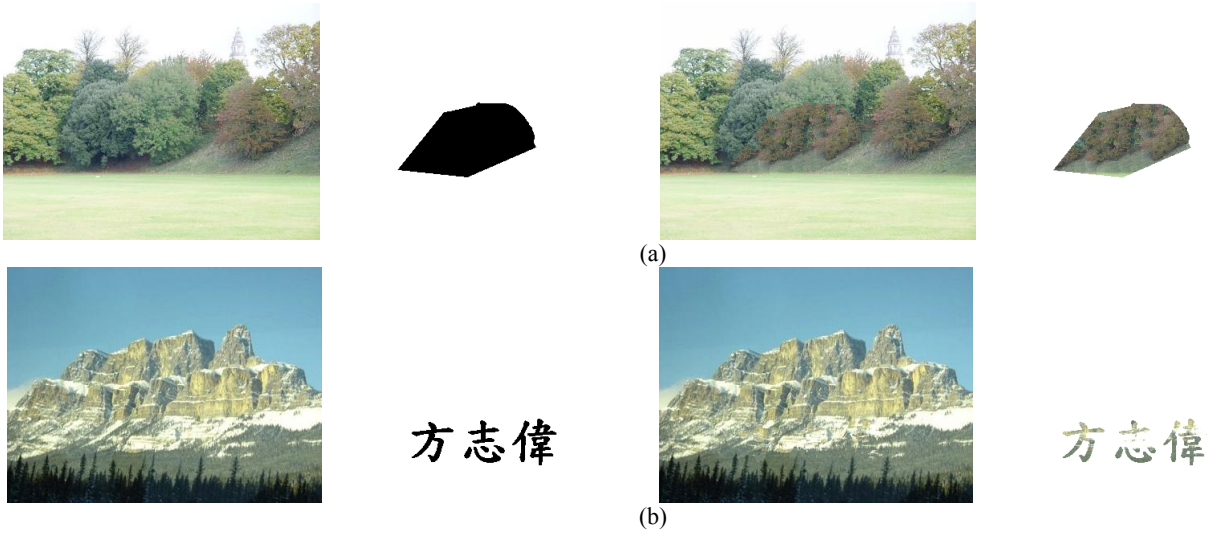


Figure 15: From left to right columns: input images, inverse mattes, output images, and the synthesis results used to replace the replaced regions.