

國立交通大學

多媒體工程研究所

碩士論文

利用權重彩色與灰階共現機率之三維紋理合成



3D Texture Synthesis Using Weighted Color and Grey Level

Co-occurrence Probabilities

研究生：張家文

指導教授：施仁忠 教授

張勤振 教授

中華民國 九十九 年 九月

利用權重彩色與灰階共現機率之三維紋理合成

3D Texture Synthesis Using Weighted Color and Grey Level Co-occurrence  
Probabilities

研究生：張家文

Student : Chia-Wen Chang

指導教授：施仁忠 教授

Advisor : Zen-Chun Shih

張勤振 教授

Chin-Chen Chang

國立交通大學  
多媒體工程研究所  
碩士論文



Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年九月

# 利用權重彩色與灰階共現機率之三維紋理合成

## 3D Texture Synthesis Using Weighted Color and Grey Level Co-occurrence Probabilities

研究生：張家文

指導教授：施仁忠 教授

張勤振 教授

國立交通大學多媒體工程研究所



本論文提出了一個利用權重彩色和灰階共現機率來做三維紋理合成(3D texture synthesis)的方法。因為由輸入的貼圖中用於鄰近點比對時所擷取的特徵會影響合成結果的品質，我們首先利用彩色和灰階共現機率這兩種方法來擷取輸入貼圖中的特徵。其中彩色的部分使用表面向量值(appearance vector)取代傳統只用色彩值(RGB color value)作為特徵向量，而灰階共現機率可隨輸入貼圖的特性來選擇不同的統計量作為特徵向量。接著透過輸入貼圖的特性改變兩種方法之間的權重值。再預先計算輸入圖片中的特徵所組成的相似集(similarity set)和三維候選點(3D-candidates)來做三維紋理合成。接著利用三維金字塔合成方法(pyramid synthesis method)來完成三維紋理的合成。

# **3D Texture Synthesis Using Weighted Color and Grey Level Co-occurrence Probabilities**

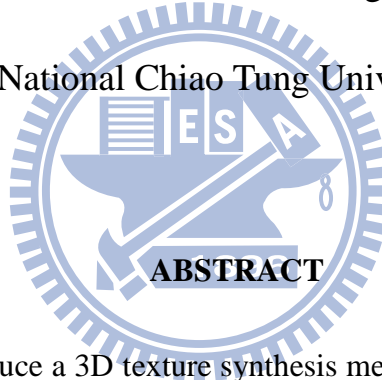
Student : Chia-Wen Chang

Advisor : Dr. Zen-Chung Shih

Dr. Chin-Chen Chang

Institute of Multimedia Engineering

National Chiao Tung University



In this thesis, we introduce a 3D texture synthesis method using weighted color and grey level co-occurrence probabilities. Because the extracted features from an input 2D texture can affect the quality of the result, we use color and grey level co-occurrence probabilities to extract the features. Appearance vectors are used to replace RGB color values. And various statistics are used according to the characteristic of the input 2D texture. We achieve 3D texture synthesis by using a pre-computed similarity set and 3D-candidate set. Then we can synthesize desired 3D textures by applying the 3D pyramid synthesis method.

# Contents

摘 要 .....	I
ABSTRACT .....	II
Contents .....	III
List of Figures .....	V
Chapter 1 Introduction .....	1
1.1 Motivation .....	1
1.2 System Overview .....	3
1.3 Thesis Organization .....	4
Chapter 2 Related Works .....	6
2.1 3D Texture Synthesis Based on Neighborhood Matching .....	6
2.2 Grey Level Co-occurrence Matrix .....	9
Chapter 3 3D Texture Synthesis .....	13
3.1 Feature Vector Generation .....	13
3.1.1 Color Features .....	14
3.1.2 Grey Level Co-occurrence Probability Features .....	15
3.1.3 Weighted Color and GLCP Features .....	18
3.2 Similarity Set and 3D-Candidate Construction .....	19

3.3	3D Pyramid Synthesis .....	19
3.3.1	Pyramid Upsampling .....	19
3.3.2	Jitter .....	21
3.3.3	Voxel Correction .....	21
Chapter 4	Implementation and Results .....	25
4.1	Implementation .....	25
4.2	Results .....	26
Chapter 5	Conclusions and Future Works .....	38
References	.....	39



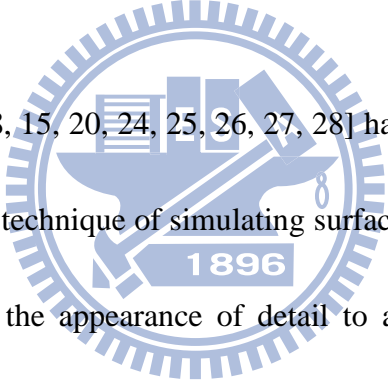
# List of Figures

Figure 1.1 System flowchart.....	5
Figure 2.1 [7] 3D neighborhood matching synthesis schemes.....	7
Figure 2.2 The process for computing grey level co-occurrence probabilities.....	10
Figure 3.1 Overview of color feature extraction.....	15
Figure 3.2 (a) grey level image generated from the input 2D texture (b) 15×15 window for each pixel (c) GLCP features computed from the window.....	16
Figure 4.1 The comparison between the previous method and our method.....	30
Figure 4.2 The comparison between the previous method and our method.....	31
Figure 4.3 The comparison between the previous method and our method.....	32
Figure 4.4 The comparison between the previous method and our method.....	33
Figure 4.5 The comparison between the previous method and our method.....	34
Figure 4.6 The comparison between the previous method and our method.....	35
Figure 4.7 The comparison between the previous method and our method.....	36
Figure 4.8 The comparison between the previous method and our method.....	37

# Chapter 1

## Introduction

### 1.1 Motivation



Texturing mapping [8, 15, 20, 24, 25, 26, 27, 28] has been applied widely in computer graphics. It is an effective technique of simulating surface detail at relatively low cost. This technique can easily add the appearance of detail to a large variety of object surfaces. Without increasing 3D model detail complexity, it enhances the visual realism of 3D models by adding fine texture details. But it suffers some well-know problems such as distortion and discontinuity. Moreover, the size and resolution of textures can not be changed dynamically.

3D textures [19, 20] can be used to solve the above problems. A 3D texture is defined as colored points in 3D to represent a real-world material. Users do not need to find a parameterization for the surface of the object to be textured. Furthermore, 3D textures provide texture information inside the entire volume. Procedural approaches and



image-based approaches both can generate 3D textures, but they may be suffered from some problems. For procedural approaches, it is difficult to express procedurally a desired texture for users. It only can be defined for a limited set of materials. For image-based approaches, it cannot do well for large structural textures. Furthermore, it is non-reusable, that is, textures generated for one 3D texture cannot be used for other 3D textures.

Recently, a lot of 3D textures have been synthesized from 2D textures by 3D texture synthesis approaches [3, 4, 6, 7, 11, 14, 19, 21, 22, 23]. Several methods [6, 7, 14, 22] used three orthogonal slices for neighborhood matching to synthesize 3D textures. They are applicable to a wide variety of textures, including anisotropic textures, textures with large coherent structures, and multi-channel textures. However, the quality problems still exist for a lot of textures. Further improvements are often required to extract more reliable texture features. Haralick et al. [9] introduced a method using the conditional joint probabilities of all pair wise combinations of grey levels in the spatial window to extract texture features for image classification. Recently, several methods [2, 5, 13, 18] extended the method to some applications, such as face detection, and image segmentation.

In this thesis, we extract color and grey level co-occurrence probability (GLCP) features from a 2D texture for 3D texture synthesis. With the characteristic of an input 2D texture, we adjust weighting values between color and GLCP features. We use weighted color and GLCP features for neighborhood matching. Our approach can provide better

synthesized results for a wide range of textures.

The major contributions of this thesis are as follows: We present an approach for synthesizing 3D textures from a 2D texture using weighted color and GLCP features. With the characteristic of an input 2D texture, we can adjust the weighting values between color and GLCP features for 3D texture synthesis.

## 1.2 System Overview

The flowchart of our system is shown in Figure 1.1. First, input a 2D texture and repeat the texture thrice as three directional exemplars.

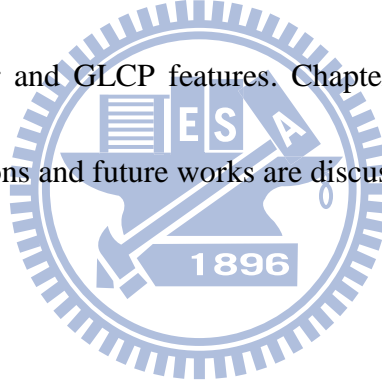
For feature vector generation, it divides into two parts: one part is to extract color features; the other part is to extract GLCP features. For the color feature extraction, it captures  $5 \times 5$  window information and uses principal component analysis (PCA) to decrease the dimensions. For the GLCP feature extraction, it converts the input 2D texture to a grey level texture, and then it captures  $15 \times 15$  window information and assigns the number of quantized grey levels. It selects the displacement value ( $\delta$ ) and the orientation value ( $\theta$ ) to create a grey level co-occurrence matrix and selects the acceptable statistical parameters. According to the characteristic of the input 2D texture, we assign weighted values between color and GLCP features.

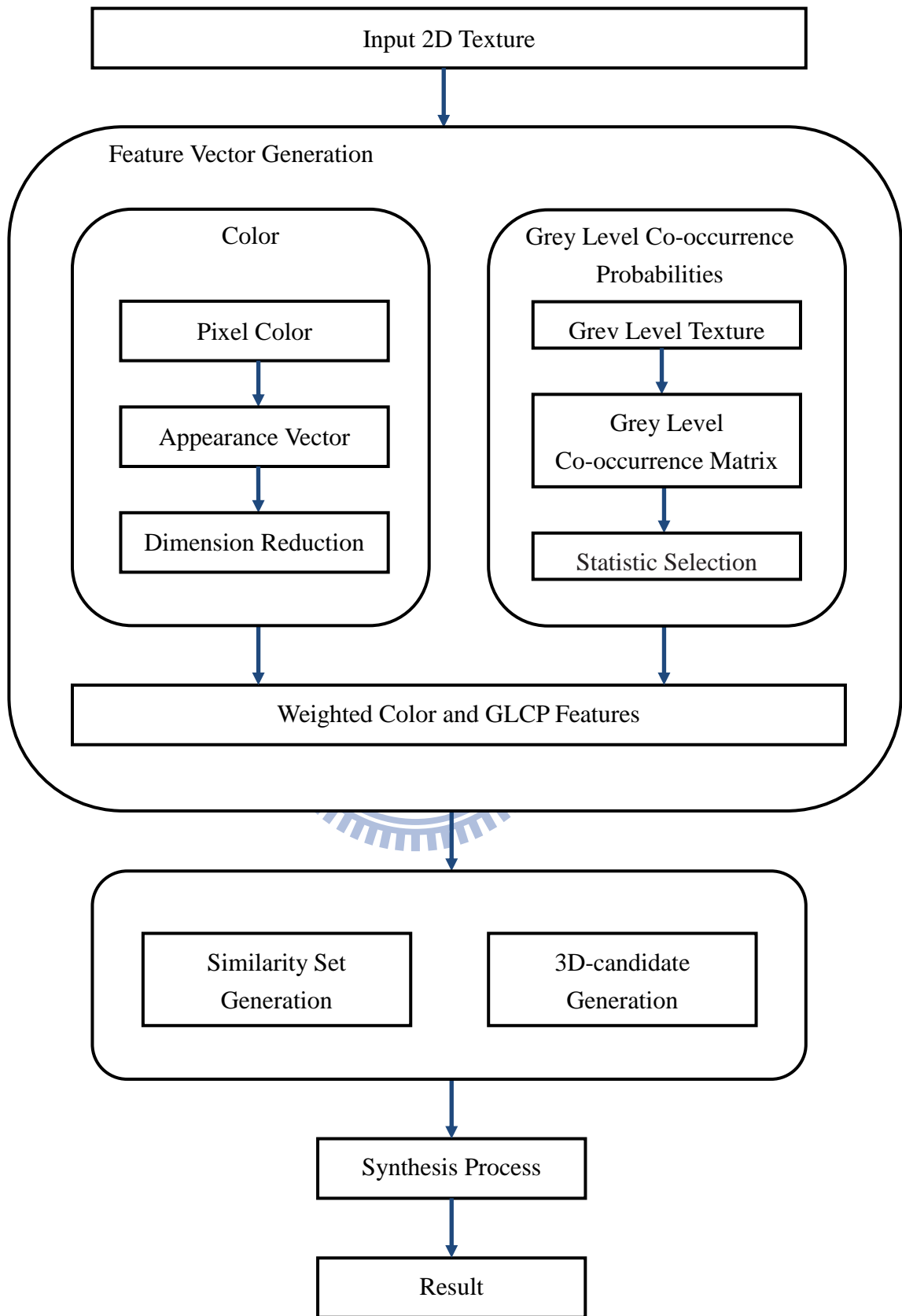
For similarity set generation and 3D-candidates generation, it finds the three pixels

most similar to each pixel. And it computes a small candidate set from the other two exemplars for each pixel. For synthesis process, we apply the pyramid synthesis method [7, 10] to our system. The pyramid synthesis method consists of the upsampling step, jitter step and correction step. They are used at each synthesis level to obtain the results.

### **1.3 Thesis Organization**

The rest of this thesis is organized as follows: In Chapter 2, we review related works. In Chapter 3, we present our approach for synthesizing 3D textures from a 2D texture based on weighted color and GLCP features. Chapter 4 shows the implementation and results. Finally, conclusions and future works are discussed in Chapter 5.



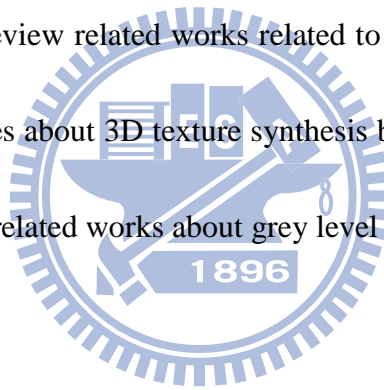


**Figure 1.1** System flowchart

# Chapter 2

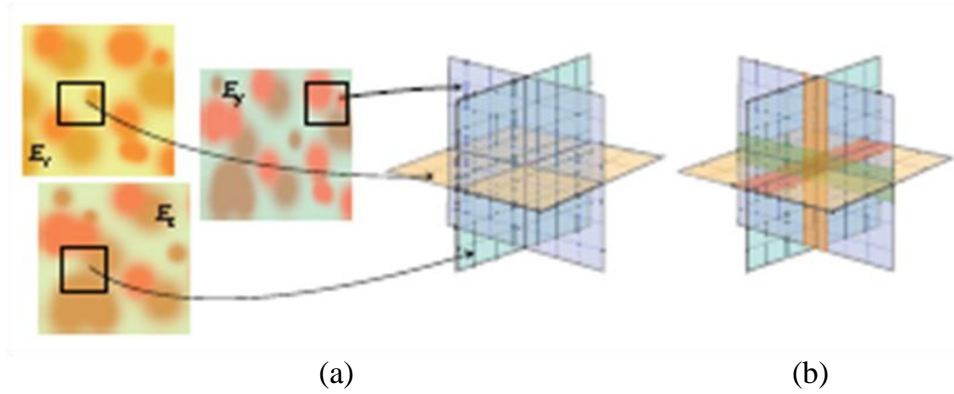
## Related Works

In this chapter, we review related works related to our approach. In Section 2.1, we review previous approaches about 3D texture synthesis based on neighborhood matching. In Section 2.2, we review related works about grey level co-occurrence matrices.



### 2.1 3D Texture Synthesis Based on Neighborhood Matching

Wei [26] first adapted 2D neighborhood matching synthesis schemes to 3D textures. The key idea is to consider three 2D exemplars for each direction from an input 2D texture. In each voxel of the output 3D texture, three interleaved 2D neighborhoods are extracted (see Fig. 2.1 [7]). The best matches are found independently in each of the three 2D exemplars. And then they attempted to converge toward the best color for all three directions.



**Figure 2.1** [7] 3D neighborhood matching synthesis schemes

(a) The three exemplars  $E_x$ ,  $E_y$ ,  $E_z$  and a corresponding 3-neighborhood.

(b) the crossbar defined by the 3-neighborhood.

Qin and Yang [22] presented a method for generating 3D textures from input examples. They replaced traditional gray-level histograms with basic gray-level aura matrices for neighborhood matching. Their method characterized each input example as a set of aura matrices and generated a 3D texture from multiple view directions. For every voxel in the volume result, they only considered the pixels on the three orthogonal slices for neighborhood matching. Furthermore, their approach can generate reliable results for both stochastic and structural textures. However, it needs large storages for large matrices.

Kopf et al. [14] introduced a 3D texture synthesis method from 2D exemplars. They extended 2D texture optimization techniques to synthesize 3D textures, and integrated

with histogram matching which forces the global statistics of the synthesized 3D texture. They also introduced a reweighting scheme for histogram matching. For each voxel, they only considered the neighborhood coherence in three orthogonal slices, and iteratively increased the similarity between the 3D textures and the exemplar. Their approach could generate good results for wide range of textures. However, they synthesized the texture with the information on the slices.

Dong et al. [7] introduced a method with the unique ability to restrict synthesis to a subset of the voxels. They synthesized a volume from a set of pre-computed 3D candidates. Their pre-computed 3D candidates improved synthesis efficiency and significantly reduced the dependency chain required to compute voxel colors. Their approach generates good results efficiently. However, if the three exemplars do not define a coherent 3D volume, the quality of the synthesized result will be poor.

Chen and Wang [3] presented a method for synthesizing solid textures from 2D exemplars. They analyzed the relevant factors for further improvements of the synthesis quality. They adopted an optimization framework with the k-coherence search and the discrete solver for 3D texture synthesis. Their approach was integrated with two kinds of histogram matching methods, position and index histogram matching. They effectively

cause the global statistics of the synthesized 3D textures to match those of the exemplars.

## 2.2 Grey Level Co-occurrence Matrix

The grey level co-occurrence method provides a second-order approach for generating features [9, 12]. Given a spatial window within the grey level image, the GLCP method computes the conditional joint probabilities,  $C_{ij}$ , of all pairwise combinations of grey levels  $i$  and  $j$ , given the inter-pixel displacement vector  $(\delta, \theta)$ , which represents the relationship of the pixel pairs, where  $\delta$  is the interpixel distance, and  $\theta$  is the orientation. The set of GLCPs are defined as

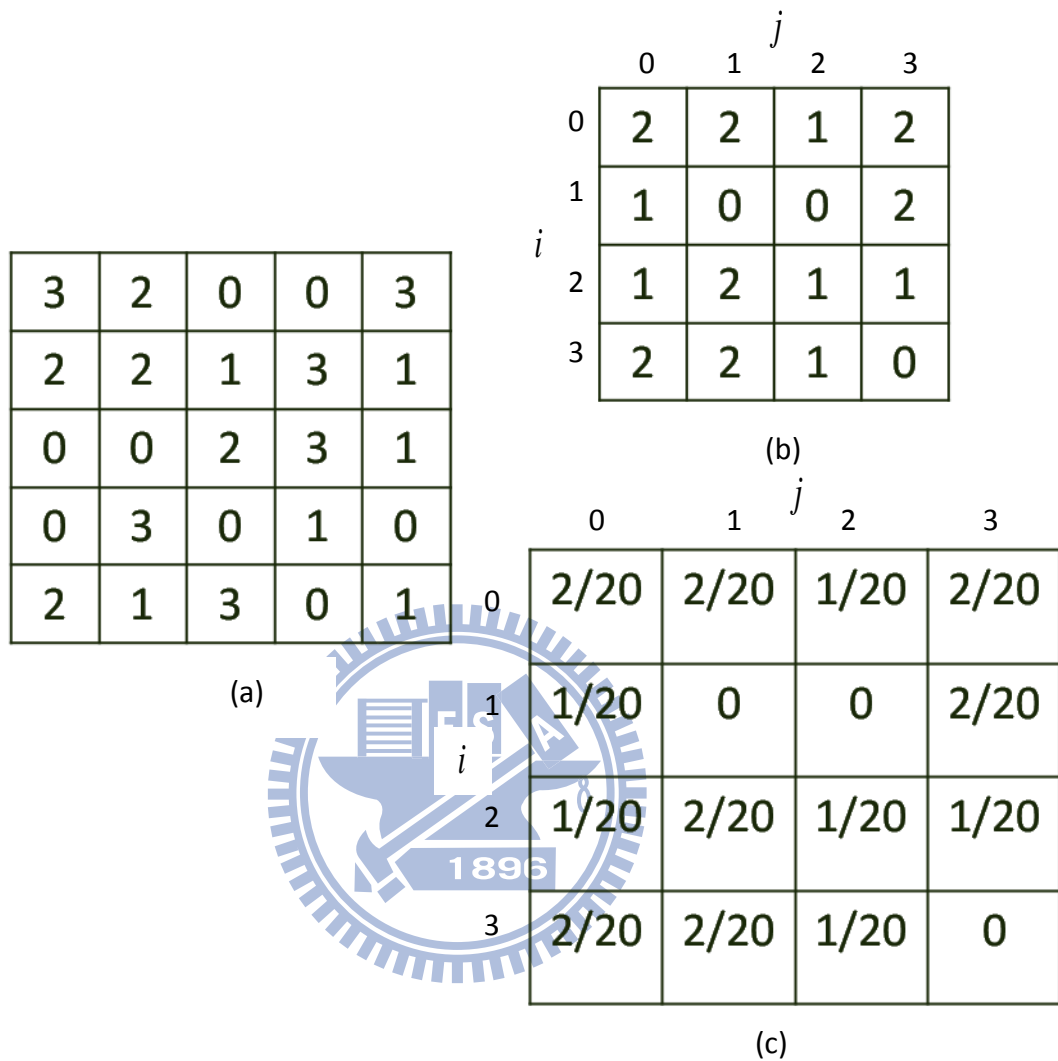
$$C_{ij} = \frac{P_{ij}}{\sum_{i,j=0}^{G-1} P_{ij}},$$



where  $P_{ij}$  is the frequency of occurrence between two grey levels,  $i$  and  $j$ , for a given displacement vector  $(\delta, \theta)$ , for the given window size;  $G$  is the number of quantized grey levels. The probabilities are stored in a grey level co-occurrence matrix (GLCM), where index  $(i, j)$  in the matrix is probability  $C_{ij}$ . Statistics are applied to the GLCM to generate texture features which are assigned to the center pixel of the image window. For example, as shown in Fig. 2.2, we use a  $5 \times 5$  window size for the grey image and  $G$  is set as 4 (Fig. 2.2(a)). The GLCM is generated with the inter-pixel displacement vector (1,



0°) (Fig. 2.2(b)) and then GLCP (Fig. 2.2(c)) is computed.



**Figure 2.2** The process for computing grey level co-occurrence probabilities

- (a) a  $5 \times 5$  window size for the grey image and  $G$  is set as 4
- (b) GLCM is generated with the inter-pixel displacement vector  $(1, 0^\circ)$
- (c) GLCP is computed from (b)

Haralick et al. [9] first presented a texture analysis method, grey level co-occurrence

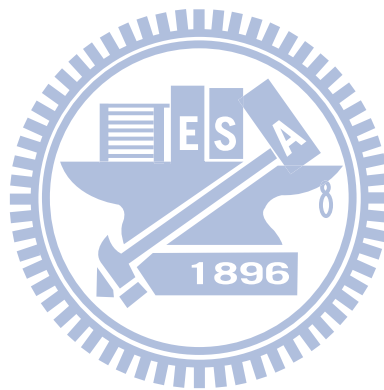
matrix (GLCM). The GLCM method is a way of extracting second order statistical texture features. The approach has been used in various of applications [2, 12, 13, 18]. The key idea is to describe the probability of any grey level occurring spatially relative to any other grey level, within a given image window.

Jobanputra and Clausi [12] extended the GLCM method. They proposed a Gaussian weighting scheme. Co-occurring pixel pairs closer to the center of the image window are assigned larger co-occurring probabilities according to a Gaussian distribution. The generated features are referred to as weighted GLCM texture features.

Jobanputra and Clausi [13] presented a weighted GLCM method for preserving boundaries for image texture segmentation. They selected suitable GLCM parameters for improved boundary preservation. From their tests, weighted GLCM features provide improved boundary preservation and segmentation accuracy at a computational cost.

Chen [2] presented a GLCM method for face detection. Face detection is one of the most important researches in face recognition. They extracted texture features from a color image by wavelet transform, and then extracted the suitable statistics by the GLCM method. They used the statistics to execute the face detection process.

Lu [18] presented a GLCM method for image segmentation. The key idea is to perform image segmentation by combining the analytic result of color and texture. In the textural part, they proposed an approach to solve the problem in GLCM. Using fixed window size is unable to accurately segment textures in the image. Therefore, adaptive windows are used to analyze features of the image textures.



# Chapter 3

## 3D Texture Synthesis

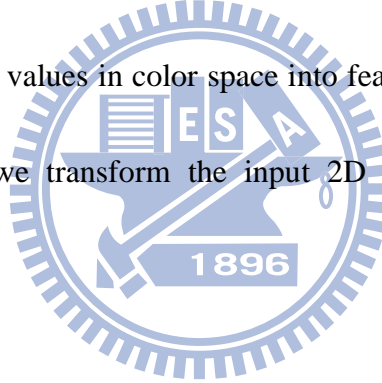
In this chapter, we present our approach for synthesizing 3D textures. In Section 3.1, we describe how to obtain the feature vectors. In Section 3.2, we use similarity sets to accelerate neighborhood matching. And then we compute 3D-candidates which keep the color coherence of the three orthogonal slices. In Section 3.3, we introduce 3D pyramid texture synthesis to our system. It is divided into three parts: the upsampling step, the jitter step, and the correction step. The upsampling step is to increase the synthesized 3D texture sizes from coarse level to fine level, and each voxel in parent level generates eight voxels in children level. The jitter step is to perturb the textures to achieve deterministic randomness. The correction step is to use neighborhood matching to make the results more similar to the exemplar.

### 3.1 Feature Vector Generation

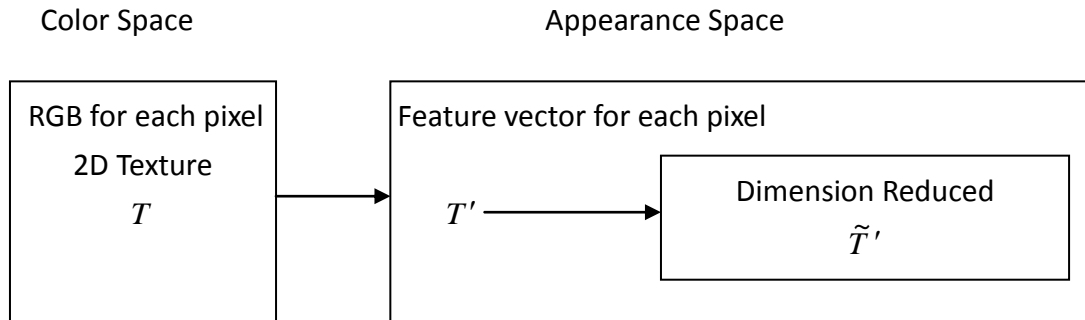
First, we extract the features from an input 2D texture. The feature extraction approach is divided into two parts: color feature extraction method and the grey level co-occurrence method.

### 3.1.1 Color Features

3D texture synthesis using RGB color values for neighborhood matching needs larger neighborhood size and numerous data. Appearance vectors [17] are continuous and low-dimensional than RGB color values for neighborhood matching. Therefore, we transform the texture data values in color space into feature vectors in appearance space. As shown in Fig. 3.1, we transform the input 2D texture  $T$  into the transformed exemplar  $T'$ .



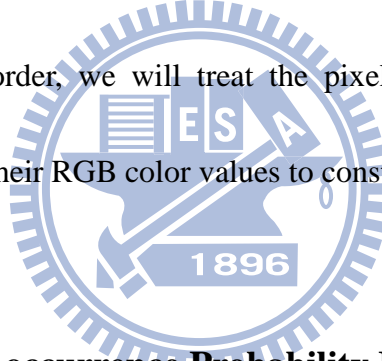
According to Lefebvre and Hoppe [19], we take the RGB color values in  $5 \times 5$  windows to construct appearance vectors for each pixel of an input texture  $T$ . The transformed exemplar  $T'$  consists of the feature vectors at each pixel. There are 75 dimensions (25 for grids and 3 for RGB) for each pixel in  $T'$ . We perform PCA to reduce the dimensions to obtain a transformed exemplar  $\tilde{T}'$ . It means that we project the exemplar  $T'$  using PCA to obtain the transformed exemplar  $\tilde{T}'$ . We reduce the 75-dimensional to 8-dimensional feature vectors.



**Figure 3.1** Overview of color feature extraction

We suppose that the data on each side is connected with them on the opposite side.

For the pixels on the border, we will treat the pixels on the opposite border as its neighbors, and then take their RGB color values to construct feature vectors.

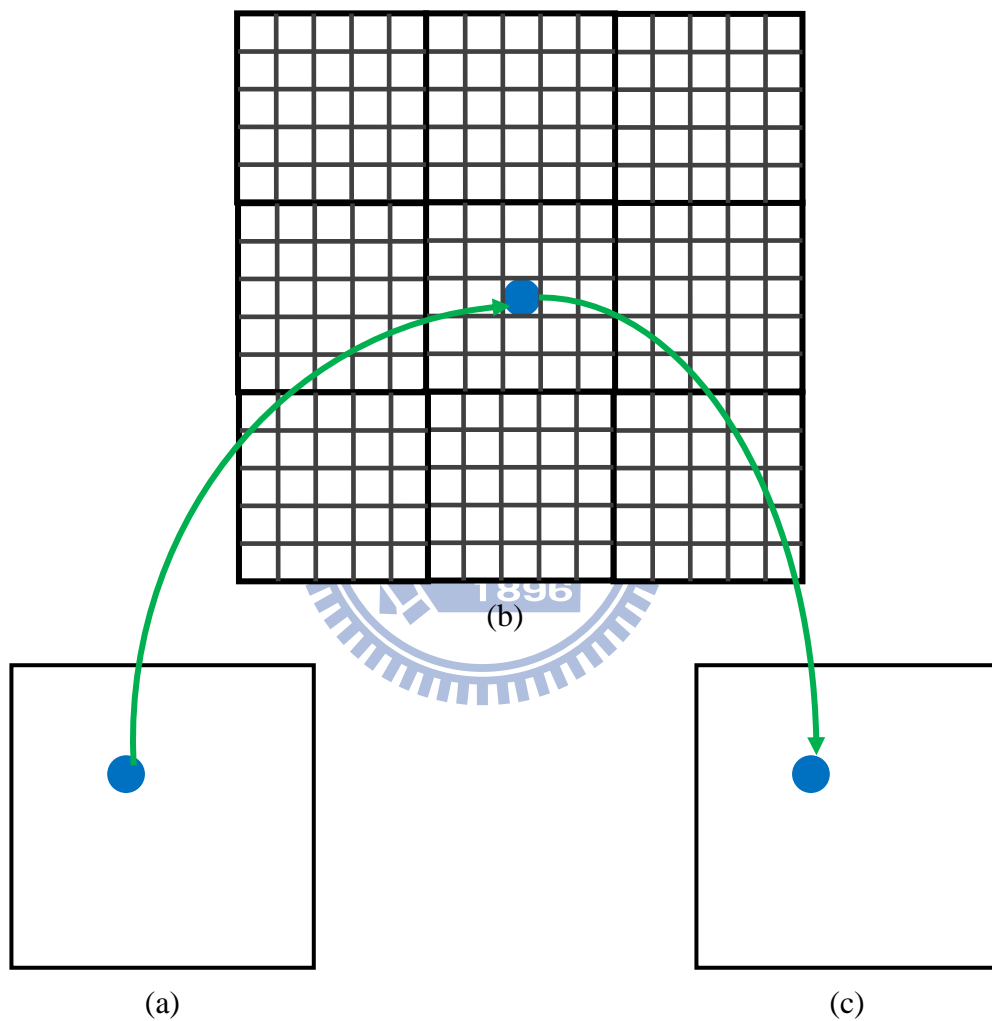


### 3.1.2 Grey Level Co-occurrence Probability Features

Here we convert the input 2D texture to a grey image, and extract GLCP features from the grey image. Extracting GLCP features from a grey image requires the parameters as follows: (1) window size  $(n_x, n_y)$ , (2) image quantization  $(G)$ , (3) displacement vector  $(\delta, \theta)$  and (4) statistic selection.

For window size  $(n_x, n_y)$ , we use a window size of  $15 \times 15$  to compute GLCP features. Because large window sizes are necessary to gather sufficient data to

characterize local texture regions, and small window sizes will result in poorly sampled co-occurring probabilities and will produce incoherent features [13]. As shown in Fig. 3.2, we compute GLCP features for the window size  $15 \times 15$ .



**Figure 3.2** (a) grey level image generated from the input 2D texture

(b)  $15 \times 15$  window for each pixel (c) GLCP features computed from the window.

For image quantization ( $G$ ), according to Clausi [5], the larger values of  $G$  are deemed excessive. Therefore, we set  $G$  as 32 for our experiments. For displacement vector  $(\delta, \theta)$ , we select the suitable displacement vector. The interpixel distance is set as 1 and the orientations are set as  $0^\circ, 45^\circ, 90^\circ, 135^\circ$ . Because we consider the characteristic of an input 2D texture for each direction and consider the influence of the distance of the pixel pairs.

For statistic selection, we will select the suitable statistic. Many of the statistics suggested by Haralick [9] produce highly correlated texture features which are not desirable. Clausi [5] studied the relationship of the statistical parameters and concluded that entropy, contrast and correlation compose a preferred set of statistical parameters. Therefore, we use three grey level shift invariant statistics in our approach as follows:

$$\text{Entropy} : - \sum_{i,j=0}^{G-1} C_{ij} \log C_{ij} ,$$

$$\text{Contrast} : \sum_{i,j=0}^{G-1} C_{ij} (i - j)^2 ,$$

$$\text{Correlation} : \frac{\sum_{i,j=0}^{G-1} (i - \mu_x)(j - \mu_y) C_{ij}}{\sigma_x \sigma_y} ,$$



where  $\mu_x$  and  $\mu_y$  are the means in the the x- and y-directions, respectively;  $\sigma_x$  and  $\sigma_y$  are the standard derivations in the the x- and y-directions, respectively. For entropy statistics, it means the level of spatial disorder of gray levels in the GLCM. For contrast statistics, it means the contrast of spatial disorder of gray levels in the GLCM. For correlation statistics, it means the linear dependency of gray levels on those of neighboring pixels in the GLCM. Therefore, we extract 12 GLCP features for our experiments (4 orientations for each statistical parameter).

### 3.1.3 Weighted Color and GLCP Features

After color and GLCP features are extracted, we propose weighted color and GLCP features for 3D texture synthesis. Different weights are assigned to color and GLCP features for a given input 2D texture. The weighted color and GLCP features are defined as

$$WF = (Weight_{CF} \times CF, Weight_{GLCPF} \times GLCPF),$$

where  $Weight_{CF}$  and  $Weight_{GLCPF}$  are the weights, and they are changed according to the characteristics of the input 2D texture.  $CF$  and  $GLCPF$  are the color feature vectors and GLCP feature vectors, respectively.

## 3.2 Similarity Set and 3D-Candidate Construction

Based on [1, 24], we find the  $k$  most similar candidates in the exemplar for each pixel  $p$ , and then search from the candidates for neighborhood matching. The method can accelerate neighborhood matching because we do not have to search from each pixel in the exemplar for neighborhood matching. Therefore, we have to construct a similarity set to record the  $k$  candidates similar to each pixel.

Furthermore, we repeat the input texture thrice as three directional exemplars  $T_x$ ,  $T_y$ ,  $T_z$ . By Dong et al. [7], we construct a small set of 3D-candidates for each pixel of the three exemplars to build the relation between the three exemplars. A 3D-candidate is defined by three  $5 \times 5$  slices (2D neighborhoods). A suitable candidate should be consistent across the crossbar which is the strip that is intersected by two slices. Therefore, we minimize the color disparity between the strips shared by interleaved slices.

## 3.3 3D Pyramid Synthesis

### 3.3.1 Pyramid Upsampling

The 3D pyramid synthesis approach [7, 10] synthesizes 3D textures from coarse

level to fine level. There are  $l+1$  levels in the whole synthesis process,  $l=0 \sim \log_2 m$ , where  $m$  is the size of the target 3D texture.

In the proposed approach, we synthesize from one voxel to a 3D texture,  $S_0 \sim S_l$ . A 3D texture  $S$  is synthesized from a voxel. Each voxel  $S[y]$  stores three coordinate, indicating  $\tilde{T}_x'$ ,  $\tilde{T}_y'$ ,  $\tilde{T}_z'$  exemplar's pixel for each direction, respectively. In our experiments, we build a voxel and assign the values (1,1), (1,1), (1,1) to it as triple coordinates. For next level, we upsample the coordinates of parent voxels. We replace eight children for each parent voxels to the scaled parent coordinates plus child-dependent offset as

$$S_l[ijk]_x = S_{l-1}\left[\left\lceil \frac{i}{2} \right\rceil, \left\lceil \frac{j}{2} \right\rceil, \left\lceil \frac{k}{2} \right\rceil\right]_x + h_l(j \bmod 2, k \bmod 2)$$

$$S_l[ijk]_y = S_{l-1}\left[\left\lceil \frac{i}{2} \right\rceil, \left\lceil \frac{j}{2} \right\rceil, \left\lceil \frac{k}{2} \right\rceil\right]_y + h_l(i \bmod 2, k \bmod 2)$$

$$S_l[ijk]_z = S_{l-1}\left[\left\lceil \frac{i}{2} \right\rceil, \left\lceil \frac{j}{2} \right\rceil, \left\lceil \frac{k}{2} \right\rceil\right]_z + h_l(i \bmod 2, j \bmod 2)$$

$$\Delta_x \in \left\{ \begin{pmatrix} 0 \\ -0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ -0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ -0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ -0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0 \\ 0.5 \\ 0.5 \end{pmatrix} \right\}$$

where  $h_l$  is the regular output spacing of exemplar coordinates, and  $ijk$  signifies the location of voxel  $v$ .  $S_l[ijk]_x$  is the new coordinate value at location  $ijk$  with the 3D texture of level  $l$  for exemplar  $\tilde{T}_x$ .  $S_l[ijk]_y$  is the new coordinate value at location  $ijk$  with the 3D texture of level  $l$  for exemplar  $\tilde{T}_y$ .  $S_l[ijk]_z$  is the new coordinate value at location  $ijk$  with the 3D texture of level  $l$  for exemplar  $\tilde{T}_z$ .

### 3.3.2 Jitter

After we upsample the coordinates from a voxel, we jitter our upsampled result in order to achieve deterministic randomness. We use the upsampled coordinates plus a jitter function value to perturb it at each level. The jitter function  $J_l(v)$  is defined by a hash function  $H(v): Z^2 \rightarrow [-1, +1]^2$  and a parameter  $r_l$  that is specified by user.

$$S_l[v] = S_l[v] + J_l(v)$$

$$J_l(v) = h_l H(v) r_l$$

### 3.3.3 Voxel Correction

After we jitter the coordinates for achieving deterministic randomness, we make the coordinates similar to those in the exemplars  $T_x$ ,  $T_y$ ,  $T_z$ . We take the jittered results to recreate neighborhood. There are feature values for each pixel after constructing feature vectors. For every voxel  $v$ , we respectively collect the feature values of its neighborhood

to obtain the neighborhood vectors for each direction. We search the most similar pixel from the transformed exemplars to make the result similar to the exemplars  $T_x$ ,  $T_y$ ,  $T_z$ .

In neighborhood matching, 4 diagonal locations in each direction are taken for voxel  $v$  to obtain the neighborhood vectors  $N_{s_l}(v)_x$ ,  $N_{s_l}(v)_y$ ,  $N_{s_l}(v)_z$ :

$$N_{s_l}(v)_x = \left\{ \tilde{T}_x[S[v + \Delta_x]] \Delta_x = \begin{pmatrix} 0 \\ \pm 1 \\ \pm 1 \end{pmatrix} \right\}$$

$$N_{s_l}(v)_y = \left\{ \tilde{T}_y[S[v + \Delta_y]] \Delta_y = \begin{pmatrix} \pm 1 \\ 0 \\ \pm 1 \end{pmatrix} \right\}$$

$$N_{s_l}(v)_z = \left\{ \tilde{T}_z[S[v + \Delta_z]] \Delta_z = \begin{pmatrix} \pm 1 \\ \pm 1 \\ 0 \end{pmatrix} \right\}$$

In order to improve convergence without increasing the size of the neighborhood vector, we average the voxels nearby voxel  $v + \Delta_x$  according to [12]. We average the appearance values from 3 synthesized voxels nearby  $v + \Delta_x$  as the new feature value to replace the voxel  $v + \Delta_x$ . And then, we use the new feature values at 4 diagonal voxel to construct neighborhood vector.  $N_{s_l}(v; \Delta_x)$  is the averaged feature value at voxel  $v + \Delta_x$ .  $N_{s_l}(v; \Delta_y)$  is the averaged feature value at voxel  $v + \Delta_y$ .  $N_{s_l}(v; \Delta_z)$  is the averaged feature value at voxel  $v + \Delta_z$ .

$$N_{s_l}(v; \Delta_x) = \frac{1}{3} \sum_{\Delta' = M\Delta_x, M \in \Psi_x} \tilde{V}'[S[v + \Delta_x + \Delta'] - \Delta']$$

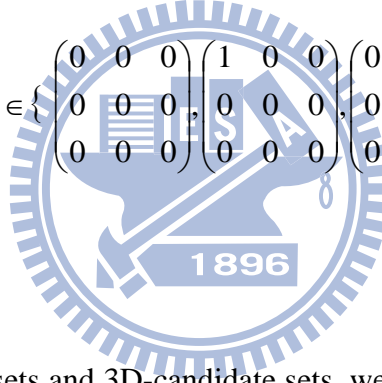
$$\Psi_x \in \left\{ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right\}$$

$$N_{s_l}(v; \Delta_y) = \frac{1}{3} \sum_{\Delta' = M\Delta_y, M \in \Psi_y} \tilde{V}'[S[v + \Delta_y + \Delta'] - \Delta']$$

$$\Psi_y \in \left\{ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right\}$$

$$N_{s_l}(v; \Delta_z) = \frac{1}{3} \sum_{\Delta' = M\Delta_z, M \in \Psi_z} \tilde{V}'[S[v + \Delta_z + \Delta'] - \Delta']$$

$$\Psi_z \in \left\{ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right\}$$



Using the similarity sets and 3D-candidate sets, we utilize the 4 voxels ( $i_x = 1 \sim 4$ ) for each direction nearby voxel  $v$ . For the neighborhood voxel  $i_x$  ( $i_x = 1 \sim 4$ ) in direction  $x$ , we can obtain the most similar 3 pixels ( $i_{x1}, i_{x2}, i_{x3}$ ) in exemplar  $T_x$  for voxel  $i_x$  from the similarity set. Then, we transform the offset between voxel  $i_x$  and voxel  $v$  from 3D space to 2D space to infer the candidates ( $i_{x1v}, i_{x2v}, i_{x3v}$ ) in exemplar  $T_x$  for voxel  $v$ . To keep color consistence in three directions, we use the 3D-candidate set to infer the other two coordinates ( $i_{(x \rightarrow y)v}, i_{(x \rightarrow z)v}$ ) in exemplars  $T_y$  and  $T_z$ . Additionally, directions  $y$  and  $z$  can be done by the same step.

Finally, we can obtain a set of triple candidates  $TC_{1..k}$  which point towards pixels in exemplars  $T_x, T_y, T_z$ . We compute the neighborhood vectors  $N_{s_l}(TC_{1..k})_x, N_{s_l}(TC_{1..k})_y, N_{s_l}(TC_{1..k})_z$  by the averaged feature values from the 4 nearby pixels. And Then, we compute the total difference among  $N_{s_l}(TC_{1..k})_x, N_{s_l}(TC_{1..k})_y, N_{s_l}(TC_{1..k})_z$  and  $N_{s_l}(v)_x, N_{s_l}(v)_y, N_{s_l}(v)_z$  to replace the triple coordinate for voxel  $v$  with the best matching triple candidate.



# Chapter 4

## Implementation and Results

In Section 4.1, we describe the implementation of our method. Then, we show synthesis results and comparisons in Section 4.2.

### 4.1 Implementation

Our system is implemented on a PC with 3.00GHz and 3.00GHz Core2 Extreme CPU and 8.0GB of system memory. We use MATLAB to implement our method. The input 2D texture size is  $128 \times 128$ , and the output 3D texture size is  $128 \times 128 \times 128$ .

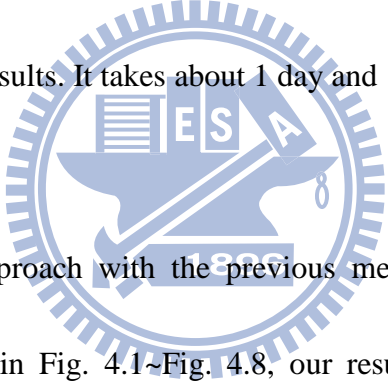
In feature vector generation, we use a  $5 \times 5$  window for extracting color features and a  $15 \times 15$  window for extracting GLCP features for each pixel from the input 2D texture. Furthermore, we set image quantization  $G$  as 32 and displacement vector  $(\delta, \theta)$  as  $(1, 0^\circ)$ . Then, we use  $7 \times 7$  neighborhood for constructing similarity sets. In 3D pyramid



synthesis step, the parameter for jitter step is set to 0.7.

## 4.2 Results

It takes about 2 minutes and 22 seconds to generate weighted color and grey level co-occurrence probability features. It takes different times to construct similarity sets based on characteristics of an input 2D texture. However, it does not exceed 1.5 hours. And it takes about 3 hours 20 minutes to construct a 3D-candidate set. Once the feature vectors and similarity sets are constructed, they are used in syntheses process with different target sizes for results. It takes about 1 day and 1 hour for synthesis process.



We compare our approach with the previous method which only extracts color features. As we can see in Fig. 4.1~Fig. 4.8, our results preserve more features and structures better than the results synthesized by the previous approach. This is because we can obtain more information than the previous method. The previous method only extracts color features to generate feature vectors from input 2D texture, but our approach extracts not only color features but also GLCP features. The grey level co-occurrence method is one of the statistical methods. GLCP features describe the probability of any grey level occurring spatially relative to any other grey level within a given window for each pixel. From the distribution, we can obtain the information of various characteristics

for each pixel from input 2D texture. For irregular textures, color features are hard to record various characteristics accurately for each pixel. However, GLCP features record accurately the various characteristics because of the statistical relatedness among pixel pair within a given window for each pixel.

The input 2D texture in Fig. 4.1(a) is a stochastic and marble-like texture. And the input 2D texture in Fig. 4.2(a) is a particle-like texture. They only contain two kinds of colors, and they are vivid. Fig. 4.1(b) shows the result synthesized by the previous approach and Fig. 4.1(c) shows our result. As we can see from the results, the quality of our method is almost the same as that of the previous method. And our result is continuous and not the duplication of the input 2D texture. As long as there are few complete particle patterns in the input 2D texture, our approach and the previous method can synthesize the desired results, as shown in Fig. 4.2(b) and Fig. 4.2(c). From Fig. 4.1 and Fig. 4.2, our approach and the previous method can preserve features and structures.

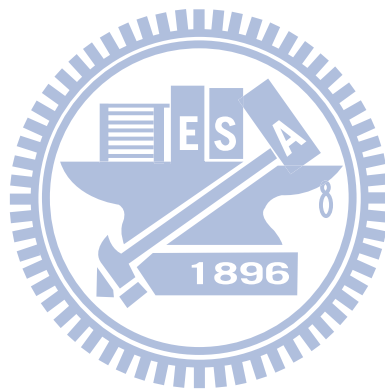
The input 2D texture in Fig. 4.3(a) is a stochastic and marble-like texture. The input 2D texture in Fig. 4.4(a) is a kind of stochastic textures. The input 2D texture in Fig. 4.5(a) is a stochastic and camouflage-like texture. In Fig. 4.3(c), our result preserves more white features in whole area. But in the previous result (Fig. 4.3(b)), it preserves

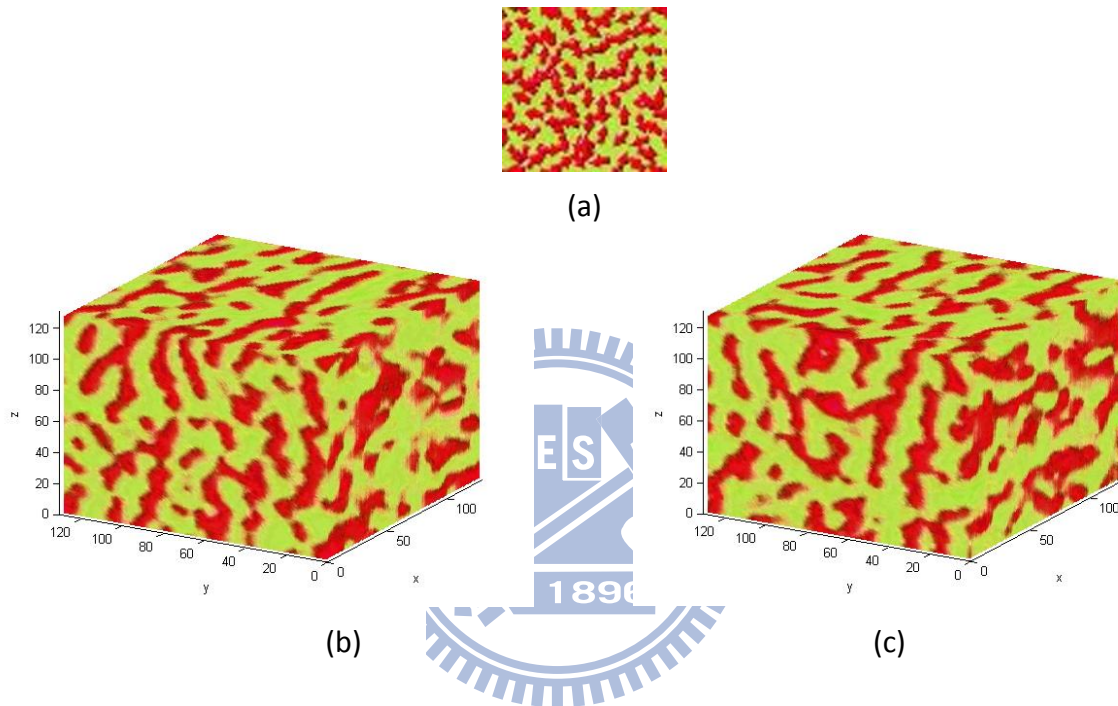
less white features. And our result is more colorful than that of the previous method. In Fig. 4.4(c), our result preserves complete features in whole area. But the previous result (Fig. 4.4(b)), it does not preserve features as we can see in Fig. 4.5(c), our result preserves more features than that of the previous method (Fig. 4.5(b)). From Fig. 4.3 and Fig. 4.4 and Fig. 4.5, we can preserve more features than the previous method.

The input 2D textures in Fig. 4.6(a), Fig. 4.7(a), and Fig. 4.8(a) are grey level textures. They are stochastic patterns as we can see in Fig. 4.6(b), our result preserves more black features in whole area. But the previous result (Fig. 4.6(b)) preserves less black features. In Fig. 4.7(c), color variation of our result is better than that of the previous method (Fig. 4.7(b)). Our result is information-rich. The previous result does not keep the features as we can see in Fig. 4.8(c), our result preserves white features in whole area. But the previous result (Fig. 4.8(b)) preserves less white features. From Fig. 4.6, Fig. 4.7 and Fig. 4.8, the quality of our approach is better than that of the previous method.

We can see in Fig. 4.1 and Fig. 4.2, the results of our approach and the previous method are similar. However, we can see in Fig 4.3 and 4.8, the results of our approach are better than the previous method. This is because they are regular textures in Fig. 4.1(a)

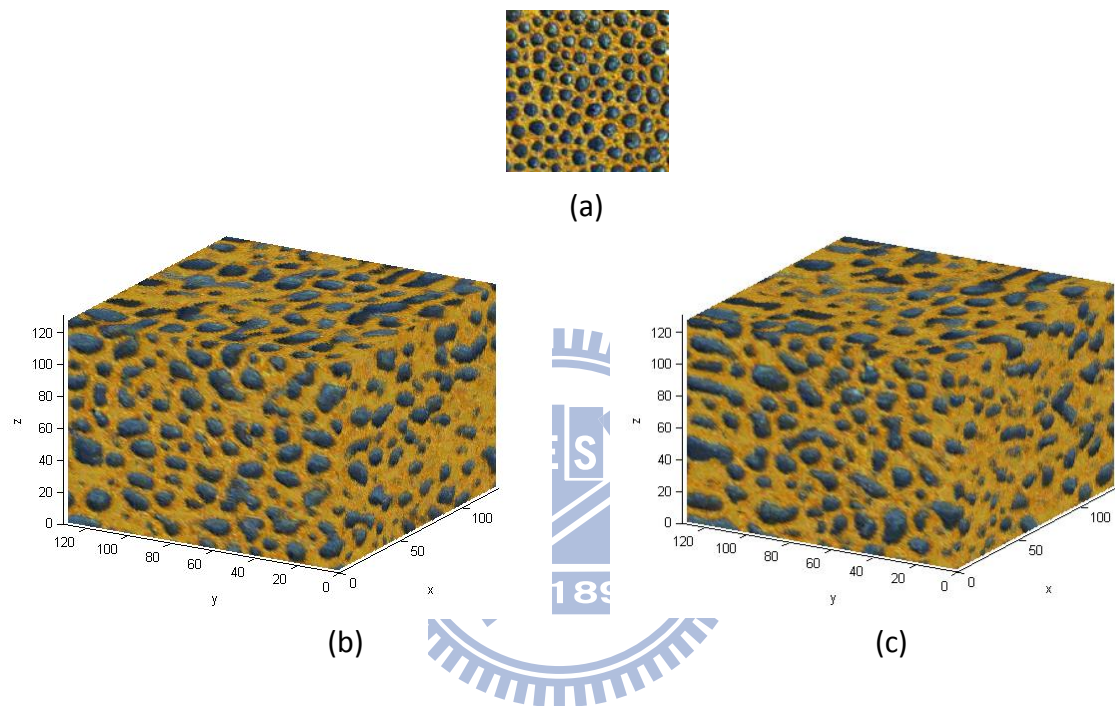
and Fig. 4.2(a), so we can synthesize good results by only using color features. But in Fig 4.3(a) and 4.8(a), they are irregular textures, but color features can not record the information of various characteristics accurately. GLCP features can solve the above problem because of the statistical relatedness among pixel pair within a given window for each pixel. Therefore, our approach can synthesize well than previous method.





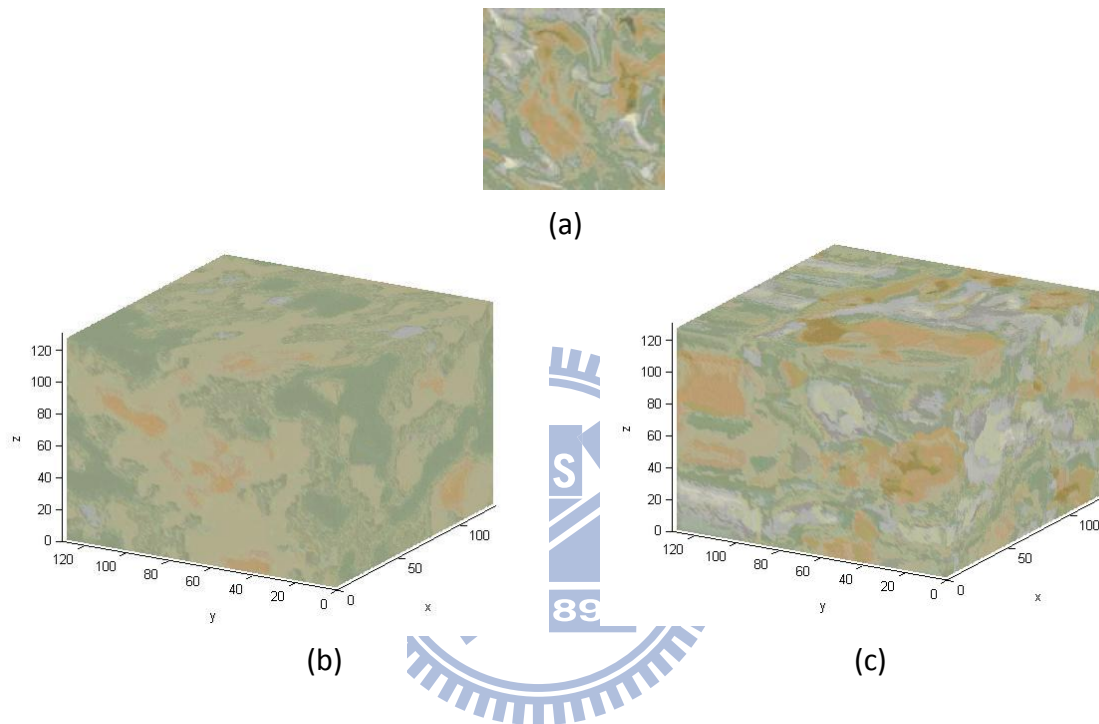
**Figure 4.1** The comparison between the previous method and our method

- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method



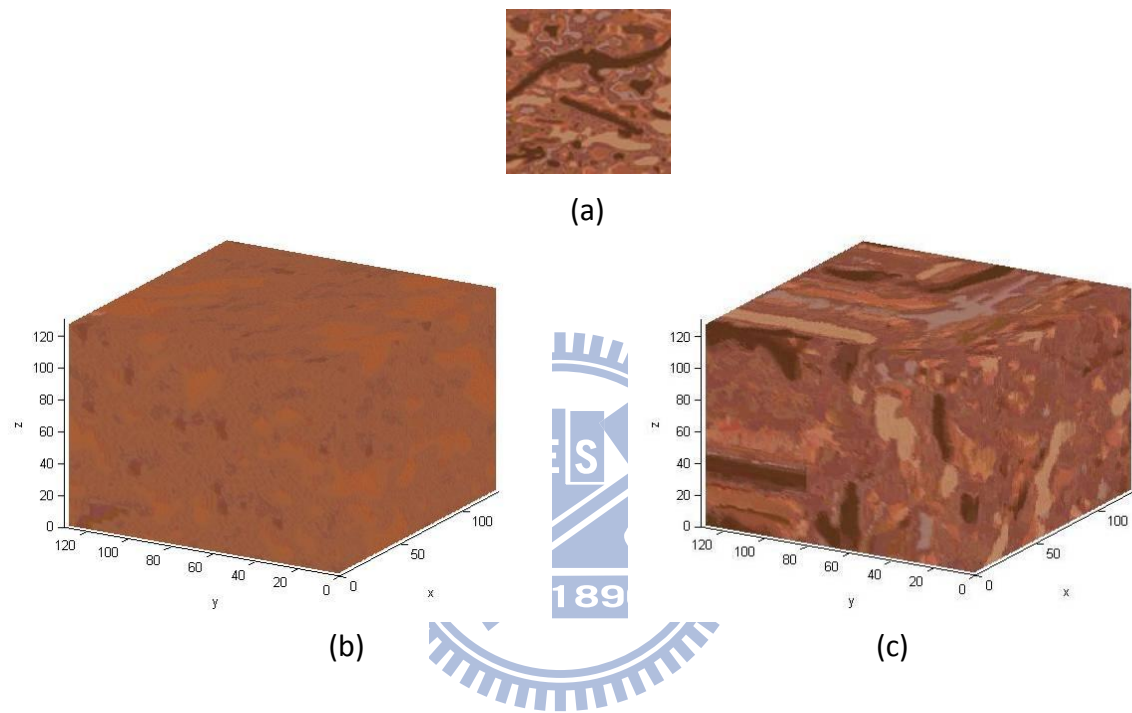
**Figure 4.2** The comparison between the previous method and our method

- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method



**Figure 4.3** The comparison between the previous method and our method

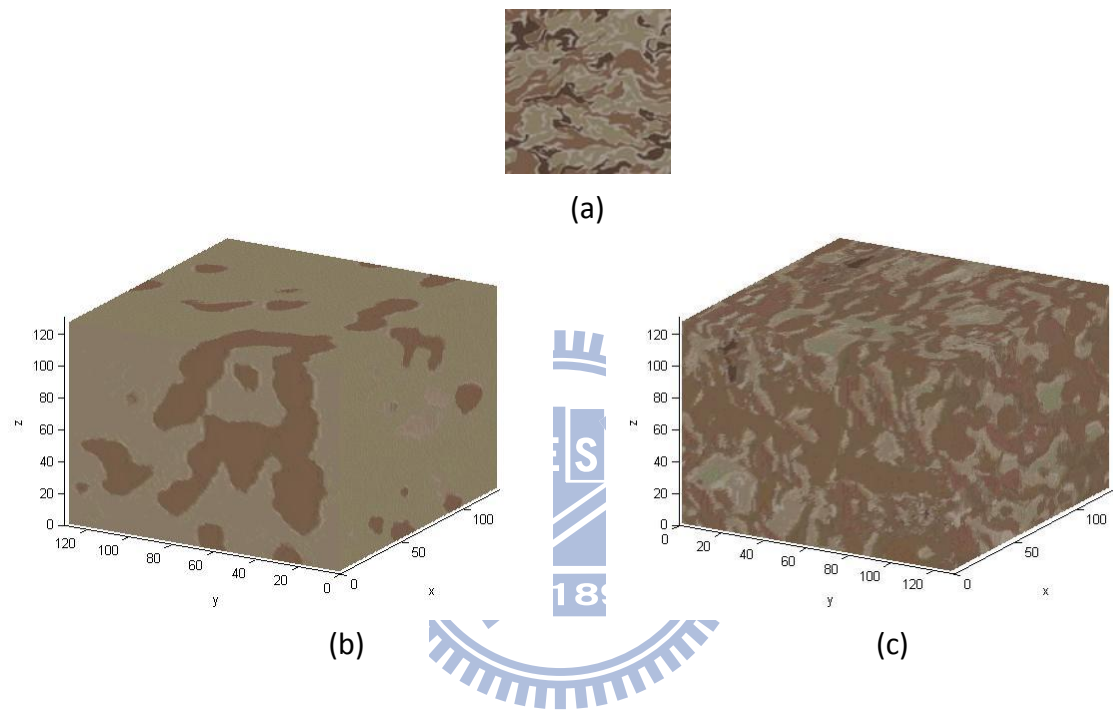
- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method



**Figure 4.4** The comparison between the previous method and our method

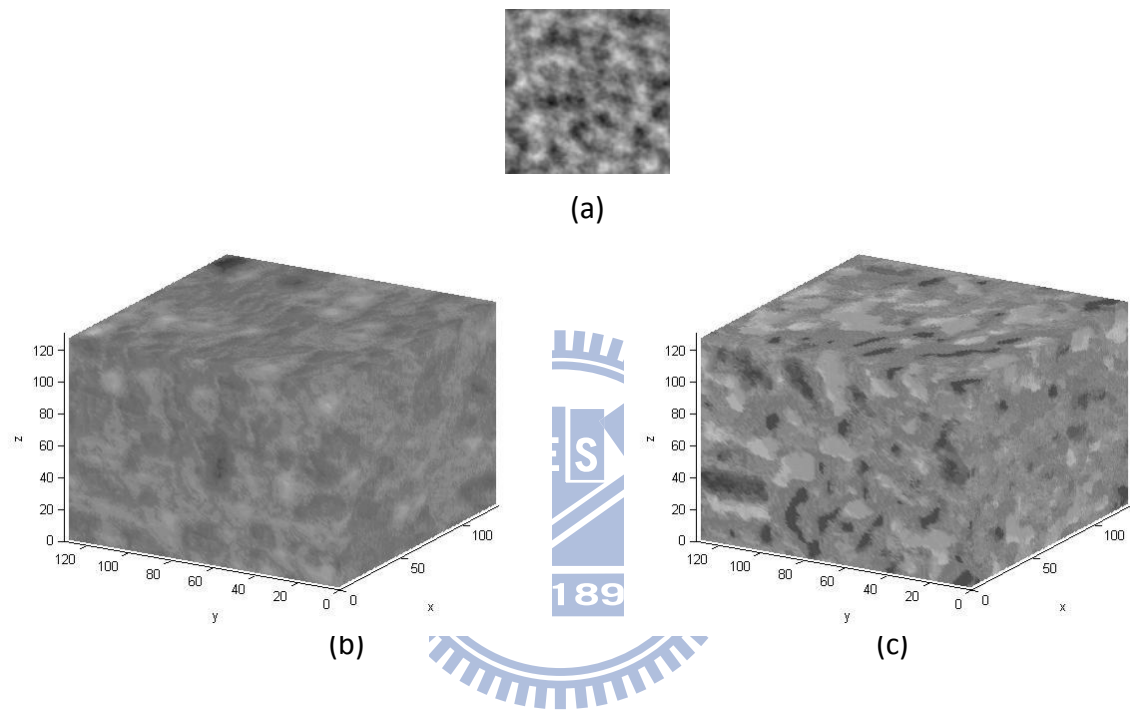
- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method





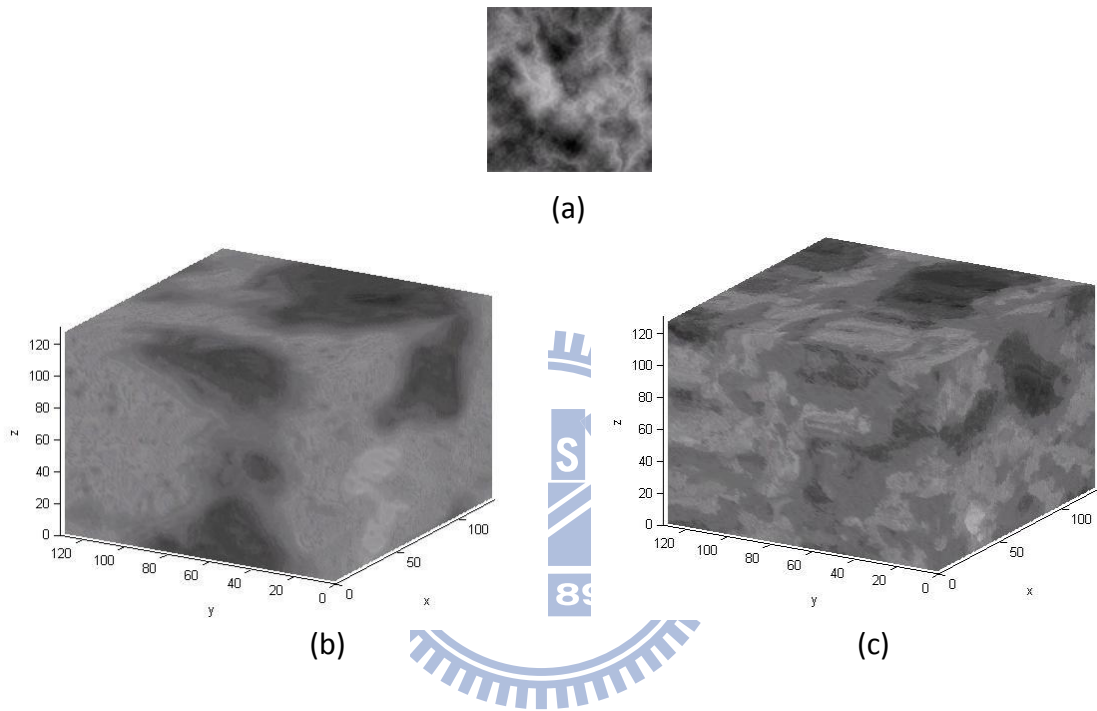
**Figure 4.5** The comparison between the previous method and our method

- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method



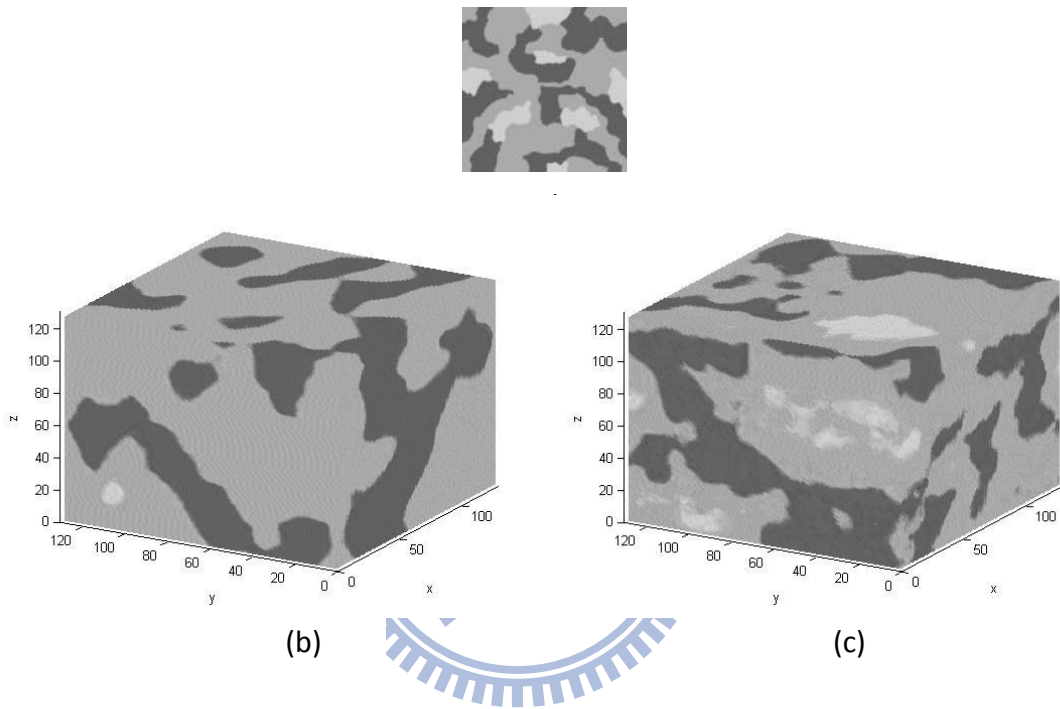
**Figure 4.6** The comparison between the previous method and our method

- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method



**Figure 4.7** The comparison between the previous method and our method

- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method



**Figure 4.8** The comparison between the previous method and our method

- (a) input 2D texture
- (b) output 3D texture by the previous method
- (c) output 3D texture by our method

# Chapter 5

## Conclusions and Future Works

We have presented a 3D texture synthesis method from a 2D texture using weighted color and grey level co-occurrence probabilities. We use the feature vectors consisting of color and grey level co-occurrence probabilities, instead of traditional RGB values for more accurate neighborhood matching. Then, we assign different weights for color and GLCP features. The proposed approach can synthesize desired results for a wide range of textures.

In the future, we will apply our method to synthesize textures changing with time in the 3D space [16]. We will try another algorithm to extract the suitable features from an input 2D texture. For the use of grey level co-occurrence probabilities, we will determine an adaptive window size for each pixel for varying textures.

# References

- [1] Ashikhmin, M., "Synthesizing natural textures", *ACM SIGGRAPH Symposium on Interactive 3D graphics*, pp. 217-226, 2001.
- [2] Chen, J. B. (supervised by Tsai Y. H.), "Face detection based on local color texture", *Master's Thesis*, Department of Computer and Information Science, Hsuan Chuang University, 2006.
- [3] Chen, J., and Wang, B. "High quality solid texture synthesis using position and index histogram matching", *The Visual Computer*, vol.26, no.4, pp. 253-262, 2009.
- [4] Chiou, J. W. (supervised by Yang C. K.), "Automatic 3D solid texture synthesis from a 2D image", *Master's Thesis*, Department of Information Management, National Taiwan University of Science and Technology, 2007.
- [5] Clausi, D., "An analysis of co-occurrence texture statistics as a function of grey level quantization", *Canadian Journal of Remote Sensing*, vol. 28, no. 1, pp. 45-62, 2002.
- [6] Dischler, J. M., Ghazanfarpour, D., and Freydier, R., "Anisotropic solid texture synthesis using orthogonal 2D views", *EUROGRAPHICS 1998*, vol. 17, no. 3, pp. 87-95, 1998.
- [7] Dong, Y., Lefebvre, S., Tong, X., and Drettakis, G., "Lazy solid texture synthesis", *Eurographics Symposium on Rendering*, vol. 27, no.4, 2008.

- [8] Ebert, D.S., Musgrave, F.K., Peachey, K.P., Perlin, K. and Worley, S., *Texturing & Modeling: A Procedural Approach*, third ed. Academic Press, 2002.
- [9] Haralick, R., Shanmugam, K., and Dinstein, I., "Textural features for image classification", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 3, pp. 610-621, 1973.
- [10] Heeger, D. J., and Bergen, J. R., "Pyramid-based texture analysis synthesis", *ACM SIGGRAPH 1995*, vol. 14, no. 3, pp. 229-238, 1995.
- [11] Jagnow, D., Dorsey, J., and Rushmeier, H., "Stereological techniques for solid textures", *ACM SIGGRAPH 2004*, vol. 23, no. 3, pp. 329-335, 2004.
- [12] Jobanputra, R., and Clausi, D., "Texture analysis using Gaussian weighted grey level co-occurrence probabilities", *1st Canadian Conference on Computer and Robot Vision*, pp. 51-57, 2004.
- [13] Jobanputra, R., and Clausi, D., "Preserving boundaries for image texture segmentation using grey level co-occurring probabilities", *Pattern Recognition*, vol. 39, pp. 234-245, 2006.
- [14] Kopf, J., Fu, C. W., Cohen-Or, D., Deussen, O., Lischinski, D., and Wong, T. T., "Solid texture synthesis from 2D exemplars", *ACM SIGGRAPH 2007*, vol. 26, no. 3, 2007.
- [15] Kraevoy, V., Sheffer, A., and Gotsman, C., "Matchmaker: Constructing constrained

- texture maps”, *ACM SIGGRAPH 2003*, vol. 22, no. 3, pp. 326-333, 2003.
- [16] Kwatra, V., Adalsteinsson, D., Kim, T., Kwatra, N., Carlson, M., AND Lin, M.,  
“Texturing fluids”, *IEEE Transactions on Visualization and Computer Graphics*,  
vol.13, no. 5, pp.939-952, 2007.
- [17] Lefebvre, S., and Hoppe, H., “Appearance-space texture synthesis”, *ACM SIGGRAPH  
2006*, vol. 25, no. 3, 2006.
- [18] Lu, W. T. (supervised by Tsai Y. H.), “Image segmentation based on color and texture  
features”, *Master’s Thesis*, Department of Computer and Information Science, Hsuan  
Chuang University, 2007.
- [19] Peachy, D. R., “Solid texturing of complex surfaces”, *ACM SIGGRAPH 1985*, vol.  
19, no. 3, pp. 279-286, 1985.
- [20] Perlin, K., “An image synthesizer”, *ACM SIGGRAPH 1985*, vol. 19, no. 3, pp.  
287-296, 1985.
- [21] Pietroni, N., Cignoni, P., Otaduy, M., and Scopigno, R., “A survey on solid texture  
synthesis”, *IEEE Computer Graphics and Applications*, pp. 1-1, 2009.
- [22] Qin, X., and Yang, Y. H., “Aura 3D textures”, *IEEE Transactions on Visualization and  
Computer Graphics*, vol. 13, no. 2, pp.379-389, 2007.
- [23] Takayama, K., Okade, M., Ijiri, T., and Igarashi, T., “Lapped solid textures: Filling a  
model with anisotropic textures”, *ACM SIGGRAPH 2008*, vol. 27, no. 3, 2008.



- [24] Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., & Shum, H., “Synthesis of bidirectional texture functions on arbitrary surfaces”, *ACM SIGGRAPH 2002*, vol. 21, no. 3, pp. 665-672, 2002.
- [25] Turk, G., “Texture synthesis on surfaces”, *ACM SIGGRAPH 2001*, vol. 20, no. 3, pp. 347-354, 2001.
- [26] Wei, L.Y., “Texture synthesis by fixed neighborhood searching”, *PhD’s Dissertation*, Stanford University, 2002.
- [27] Wei, L.Y., and Levoy, M., “Texture synthesis over arbitrary manifold surfaces”, *ACM SIGGRAPH 2001*, vol. 20, no. 3, pp. 355-360, 2001.
- [28] Ying, L., Hertzmann, A., Biermann, H., and Zorin, D., “Texture and shape synthesis on surfaces”, *Eurographics Workshop on Rendering 2001*, vol. 12, pp. 301-312, 2001.

