Learning-Based Texture Synthesis and Automatic Inpainting Using Support Vector Machines

Xinghui Dong, Junyu Dong, Guimei Sun, Yuanxu Duan, Lin Qi, and Hui Yu

***Abstract*—Texture synthesis methods based on patch sampling and pasting that can generate realistic textures with a similar appearance to a small sample. However, the sample usually has to be used throughout the synthesis stage. In contrast, the learnt representation of the textures are more compact and discriminative, and can also yield good synthesis results. In this paper, we introduce a learnt approach for texture synthesis based on Support Vector Machines (SVM). This approach benefits from the merit of SVM that the sample texture pattern is learnt using a model, and the sample itself can be discarded during the synthesis stage; the approach is also used to synthesize 3D surface textures. Experimental results show that our approach is particularly effective in modeling and synthesizing near-regular or regular textures, which are difficult to achieve using traditional parametric texture synthesis methods. We further apply the proposed approach to constrained texture synthesis, image extrapolation and texture inpainting. For texture inpainting, we especially develop a new method for automatically detecting holes in textures without the requirement of human intervention. Our approach yields promising results for the three tasks.**

***Index Terms*—Texture synthesis, image extrapolation, inpainting, hole-filling, support vector machines (SVM).**

I. Introduction

AS an omnipresent visual experience, texture [1-2] has been used in many industrial automation systems, e.g., chewing robots [3-4], image quality assessment [5], industrial inspection [6], quality control [7] and traffic flow animation [8].

J. Dong is supported by National Natural Science Foundation of China (NSFC) (No. 60702014, 61271405 and 61401413) and the Ph.D. Program Foundation of Ministry of Education of China (No. 20120132110018). (Corresponding author: Junyu Dong).

X. Dong is with the Centre for Imaging Sciences, the University of Manchester, M13 9PT, UK (e-mail: dongxinghui@gmail.com).

J. Dong and L. Qi are with the Department of Computer Science, Ocean University of China, Qingdao, China (e-mail: dongjunyu@ouc.edu.cn).

G. Sun is with Wuxi No.1 Senior Middle School, Jiangsu, China.

Y. Duan is with Nokia SDM Department, Qingdao, 266101, China.

H. Yu is with the School of Creative Technologies, University of Portsmouth, Portsmouth, PO1 2DJ, UK.

Texture can be derived from different sources, e.g., scanned photos and hand-drawn pictures. In computer graphics applications, texture mapping is normally used to wrap around a 2D texture onto the surface of a 3D object. Scanned photos can be directly used for texture mapping; while hand-drawn pictures are not naturalistic even if they are attractive in the aesthetic [9].

Texture mapping is usually utilized to improve visual naturalism in computer game or movie industries where an adequate size of the sample texture is required. When the size of the sample is inadequate, it has to undergo a repeated tiling process. However, unless human users manually process the tiled images, the direct tiling yields obvious artifacts or seams. In this case, texture synthesis techniques [10-13] can be used to automatically obtain an arbitrary size of the texture from a small sample based on its structural content. The synthesized texture has a visually similar appearance to the sample. Also, texture synthesis can be utilized for inpainting an image [9], [14-15] by reconstructing the missing or deteriorated parts of the image.

Traditional texture synthesis methods [9], [15-17] always require a sample texture throughout the synthesis stage. In the case of 3D surface textures, however, the input of these methods could be expensive [18] as multi-channel image representations, e.g., BTF [14], are normally used to represent the complex appearances of these textures. This limitation may reduce the performance of some systems, e.g., mobile applications. In contrast, statistical or parametric texture synthesis methods are able to learn compact representations from the sample and synthesize a new texture using the learnt data,with recent studies on statistical texture synthesis even exploit deep learning techniques [10], [19]. Although statistical methods may synthesize textures using the learnt distributions, the computational cost is expensive compared with patch-based methods [16-17], meanwhile, these methods are difficult to achieve good performances with for near-regular textures. Since near-regular textures normally exhibit repetitive-like structures, an initiative question iswhether or not we can learn the pattern of a certain sample texture while synthesizing new textures without using this sample, as achieved by statistical or parametric texture synthesis methods.

Textures are normally associated with intensity (or color) variations. In other words, texture can be considered as a phenomenon related to the certain spatial layouts of varied intensities or colours, therefore, spatial distributions of the grey levels contained in textures are important to their representation.

According to the Markov Random Field (MRF) theory, the grey level of a pixel is determined by its neighboring pixels. Using this, from the viewpoint of image representation, the grey levels of the neighboring pixels can therefore be used as a feature vector, while the grey level of the central pixel of the neighborhood can be treated as a class label. The Support Vector Machines (SVM) technique [20] is commonly known as a simple yet robust classifier as it is normally involved in classification tasks [21]. In essence, SVM learns a set of mapping functions from an annotated training dataset, thus, the parameters of an SVM model are compact.

Inspired by the successful applications of SVM [20] to various tasks [20-21], we propose a simple texture synthesis approach on the basis of the SVM classifier. Specifically, we intend to learn the texture patterns contained in the sample texture by modeling the grey level distributions using SVM ,and apply the learnt model to synthesizing larger textures. Our method essentially uses the intensity values of the pixels contained in the L-shaped neighborhood of a pixel as its feature vector. The predicted class label obtained using the model is regarded as the intensity value of the pixel: as a learning-based approach, our method does not require the sample for the synthesis procedure once the model has been learned. And unlike traditional learning-based or parametric methods, our method performs particularly well on synthesizing regular or near-regular textures. The proposed method is also applied to the synthesis of 3D surface textures [22-23].

Furthermore, we generalize the proposed method to two specific applications: constrained texture synthesis and image extrapolation. As a natural popularization, the constrained texture synthesis method is also applied to texture inpainting [9], [15] ,in particular, our method can automatically find the holes and perform inpainting without human intervention.

This paper extends the previous work in [24] in the following aspects: (1) it reports texture synthesis experiments in more detail and tests the proposed method using irregular textures, (2) it generalizes the proposed method to three additional tasks, and (3) it compares the proposed method with two traditional methods [15-16], two state-of-the-art convolutional neural network (CNN) based methods [10], [19] and a random forest [25] based method.

The contributions of this paper are fourfold. First, we introduce a parametric approach for learning and synthesizing textures using SVM, with this approach generating a compact texture description by learning an SVM model from raw grey level values. Second, we apply the proposed approach to constrained texture synthesis, image extrapolation and texture inpainting, this further augments the use of the proposed texture synthesis approach. Third, we develop an automatic hole detection approach based on the self-similarity features this approach may also be used for other tasks, e.g. textile defect detection. Fourth, we model 3D surface textures in a parametric way, which allows the surface texture samples to be discarded during texture synthesis and therefore reduces the requirement of memory.

The rest of this paper is organized as follows: in Section II, we review the related work; in Section III, the proposed texture synthesis approach is introduced; in Section IV, we describe two specific applications of the proposed approach: constrained texture synthesis and image extrapolation ; in Section V, as a natural generalization of constrained texture synthesis, texture inpainting is introduced, together with an automatic hole detection approach; In Section VI, we present our results and compare the proposed method with other approaches; finally, we draw conclusions in Section VII.

II. Related Work

A. Texture Synthesis

In essence, texture synthesis is aimed at the modeling and generation of 2D textures. Ideally, a compact model is preferred by a texture synthesis approach in order to describe the sample. This approach should also be able to yield arbitrary sizes of textures which manifest similar appearance to the sample.

Statistical modeling is normally used in parametric texture synthesis methods, which yields a compact model of the sample. Gatys *et al.* [10] modeled natural textures using the correlations between the feature maps computed at multiple layers of a pre-trained convolutional neural network (CNN). Xie *et al.* [12] introduced a sparse FRAME (Filters, Random field, And Maximum Entropy) model in order to describe natural image patterns. You *et al.* [13] approximated linear dynamic systems (LDS) by a principal component regression (PCR) model. They further kernelized the traditional PCR in order to exploit the nonlinearity of training frames for dynamic texture synthesis. Nevertheless, parametric texture synthesis methods usually fail to produce satisfactory results when synthesizing near-regular or regular textures. This is also the case for the state-of-the-art texture synthesis technique based on deep learning [10].

In contrast, “smart copying” [16-17] based non-parametric approaches are suitable for the synthetization of many textures, including near-regular textures. Recently, Sendik and Cohen-Or [19] calculated a structural energy measure from the correlations of the features extracted using a pre-trained CNN model in order to encode the self-similarity and regularity of a texture. In addition, Dai *et al.* [14] developed an example-based facade texture synthesis algorithm by modeling the tiling of the semantic components of a facade texture, yet an obvious shortfall of these approaches is that samples are always required throughout the synthesis stage. In this context, when 3D surface textures, e.g., brick and knitted textiles, are used, the input of those approaches could be expensive [22] because multi-channel image representations, e.g., BTF [18], are required for describing the complex appearances of these textures.

B. Image Inpainting

Since texture plays important roles in representing object surface characteristics, texture inpainting also attracts the attention of researchers. To restore the missing data by defected regions, image inpainting [18] is applied. Typical methods include texture synthesis techniques based on pixels [9], [15]. Efros and Leung [15] proposed a constrained texture synthesis method, which was applied to filling the known blank area on textures, as such, when the method is used for image extrapolation, it can fill the outer region of a texture. Wei and Levoy [9] further replaced the raster-scan synthesis order using the spiral order to remove the bias towards directions. Although these methods [9], [15] have produced promising results, there still exists some issues: for example, the whole texture has to be

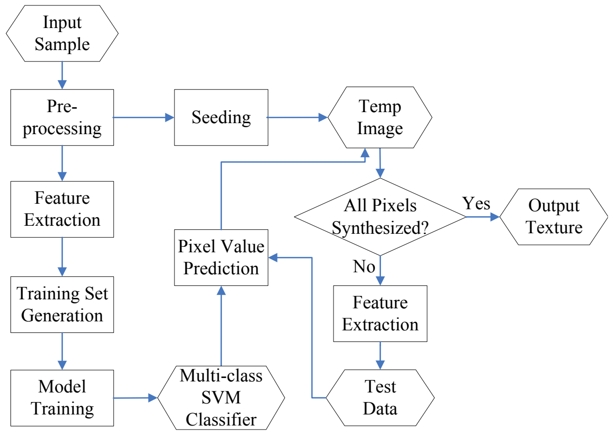


Fig. 1. The pipeline of the proposed 2D texture synthesis method.

searched whenever a pixel in the blank area is synthesized [9], [15]. This demands a high computational complexity as the search process is repeated all the time. In contrast, our approach avoids the exhaustive search by predicting pixel values in the blank area via learning texture patterns using SVM [20].

Texture synthesis approaches normally deal with 2D textures [9-10], [14], [16-17]. However, real surfaces are normally rough and cannot be represented using a single 2D texture image, thus, when 3D surface textures [22-23] are used, they are unable to produce textures rendered at different lighting conditions from the original ones. This limitation prevents those approaches from being applied to the generation of realistic textures in either augmented or virtual reality systems, moreover, it causes problems in traditional texture inpainting, because real-world 3D surface textures require more images as representation [22]. Therefore, we extend the proposed 2D texture synthesis and inpainting methods to the scenario of 3D surface textures.

III. The Support Vector Machines Based Texture Synthesis Approach

Motivated by the success achieved using SVM [20] for various pattern recognition tasks [20-21], we introduce a novel texture synthesis approach based on the strong learning ability of SVM. This approach learns the patterns of a sample texture and applies the learnt model to synthesizing larger textures. The proposed 2D texture synthesis approach is first introduced in this section and then adapted for the synthesis of 3D surface textures.

A. The 2D Texture Synthesis Approach Using SVM

The 2D texture synthesis approach is implemented in five stages: (1) pre-processing the input sample, (2) feature extraction, (3) generating the training data, (4) model training and (5) texture synthesis. Fig. 1 demonstrates the pipeline of the 2D texture synthesis approach.

1) *Pre-processing the Input Sample*

After pre-processing, the number of the grey levels in the sample texture is reduced while the image quality does not obviously decrease. Given the sample texture denoted as S ($S(i,j) \in \{0,1,\dots 255\}$), the reduction operation is described as:

	P	P	P	P	P	P	P	P	P	
	P	P	P	P	P	P	P	P	P	
	P	P	P	P	P	P	P	P	P	
	P	P	P	P	P	P	P	P	P	
	P	P	P	P	X					

Fig. 2. The L-shaped neighborhood that Wei and Levoy [9] proposed. The current pixel to be synthesized (marked as "X") is chosen as the class label. Its neighboring pixels (marked as "P") are used as feature vectors.

$$S(i,j)' = [S(i,j)/4]. \tag{1}$$

This process guarantees that each grey level owns sufficient pixels. Therefore, enough training data can be generated in the feature extraction stage. In total, 64 grey levels are used.

2) *Feature Extraction*

The features computed from the training dataset and the corresponding labels are used to train an SVM model. The class label $C(i,j)$ is chosen as the grey level value of pixels. In addition, the feature vector is required to be representative in order to effectively describe the output pixel. Many texture features have been developed, e.g., 51 types of features that Dong and Chantler [26] examined. When the dimensionality of feature vectors is high, the computational cost required in the training and synthesis stages becomes greater accordingly. Given a certain pixel, its grey level value is dependent on the grey level values within its immediate $N \times N$ neighborhood according to the Markov Random Field (MRF) theory. This is expressed as:

$$S'(i,j) \sim \{S'(i+m,j+n)\}, -\frac{N}{2} \le m,n \le \frac{N}{2}, m,n \ne 0. \tag{2}$$

Thus, features can be directly extracted from the immediate neighborhood of the pixel $I(i,j)$ in the sample texture S' or the resultant image R. The texture synthesis operation normally starts from the top-left pixel in R and then conducted in the raster-scan order. Hence, the grey level values of the pixels located in the L-shaped neighborhood [9] (see Fig. 2) of the pixel $I(i,j)$ are used to obtain a feature vector as:

$$F_L(i,j) = \{S'(i+m,j+n)\}, \tag{3}$$

where $\left(-\frac{N}{2} \le m < 0, -\frac{N}{2} \le n \le \frac{N}{2}\right) \& \left(m = 0, -\frac{N}{2} \le n < 0\right)$. However, the L-shaped neighborhood cannot be applied to the pixels whose location satisfies $(i < \left\lfloor\frac{N}{2}\right\rfloor)\&(j > N)$ or $(i > N)\&(j < \left\lfloor\frac{N}{2}\right\rfloor)$. In these cases, the $1 \times N$ row neighborhood or the $N \times 1$ column neighborhood is used to extract feature vectors. Since the raw pixel values are used as features, more complicated feature extraction operations are avoided.

3) *Training Data Generation*

The feature vector $F_L(i,j)$ and class label $C(i,j)$ extracted at each pixel location (i,j) in the sample texture are concatenated into a training feature set as:

$$TF = \{TF(i,j)\} = \{F_L(i,j); C(i,j)\}. \tag{4}$$

In terms of the three types of feature vectors that we mentioned in the previous subsection, three training feature sets are obtained respectively.

4) *Model Training*

The SVM classifier [20] is used to learn a representation model of the sample texture. In terms of binary classification, given a set of labeled training pairs (u_p, v_p) $(p = 1, \dots, l)$ where u_p is a feature vector ($u_p \in R^n$) and v_p is the

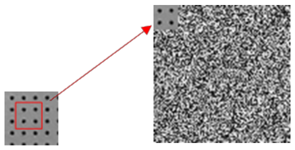


Fig. 3. The seeding process applied during the initialization of the proposed texture synthesis method.

corresponding class label ($v_p \in \{1, -1\}$), SVM [20] solves an optimization problem which is expressed as:

$$\min_{E,b,\varepsilon} \frac{1}{2} E^T E + B \sum_{p=1}^{l} \varepsilon_p, \tag{5}$$

subject to $v_p\left(E^T \emptyset\left(u_p\right) + b\right) > 1 - \varepsilon_p$, $\varepsilon_p \geq 0$. Here, E is the weight vector, b is the bias and $B > 0$ is the penalty coefficient of the error term ε_p. The training feature vectors u_p are transformed into a higher dimensional space using the function $\emptyset$ and a linear separating hyperplane is usually obtained using SVM. Besides, $K\left(u_p, u_q\right) \triangleq \emptyset\left(u_p\right)^T \emptyset\left(u_q\right)$ is the kernel function. We employ the *LibSVM* library [27], which uses the "one-against-one" scheme to design multi-class classifiers. Let m denote the number of classes, and so, $m(m-1)/2$ binary classifiers are built. Each of these classifiers is trained using the data of two classes,then, each binary classification is treated as a voting operation in which votes are cast over all data points u_p. Finally, a point is classified into a class with the maximum number of votes.

A multi-class SVM classifier/model: SM is trained using each training feature set TF as:

$$TF \overset{SVM}{\Longrightarrow} SM. \tag{6}$$

The SM model can be treated as a function of the sample texture S', i.e., $SM = f(S')$, which encodes the patterns of S'. In total, three models are trained.

5) Texture Synthesis

Texture synthesis is fulfilled via predicting the grey level value of pixels using a model. A white noise image R_0 is first created, which is used as the initialization of the resultant image R. A small "seed" patch ($\geq N \times N$) is then randomly cropped from the sample, the patch is placed at the top-left corner within R (see Fig. 3) with the seeded image being referred to as R_0^S. The synthesis operation is conducted in the raster-scan order anda feature vector is extracted as the grey level values in the row, column or L-shaped neighborhood of the pixel to be synthesized (see Section III-A-2 for more details). The intensity value of the pixel is predicted using the corresponding model SM together with this feature vector. The synthesis process is repeated until every pixel in the resultant image has been assigned a new grey level value. The synthesis process can be expressed as:

$$R_0^S \overset{SM}{\Rightarrow} R. \tag{7}$$

It can be seen that the only input of the synthesis process is the size of the resultant image while the sample texture is not required during this process.

B. Synthesis of 3D Surface Textures

Different surface geometry and reflectance characteristics can be found in real-world surface textures. This significantly affects the appearance of a texture when the illumination direction changes. Although the two images shown in Fig. 4 look different,

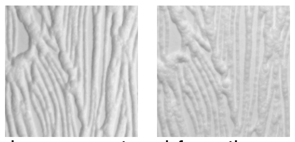
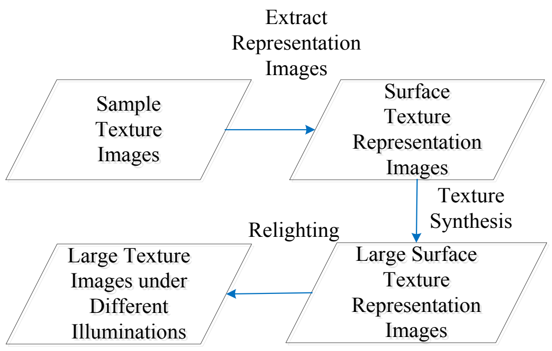


Fig. 4. Two texture images captured from the same surface texture sample [33] at different illumination conditions.


Extract Representation Images
Sample Texture Images
Surface Texture Representation Images
Texture Synthesis
Relighting
Large Texture Images under Different Illuminations
Large Surface Texture Representation Images
Fig. 5. The pipeline for synthesis of 3D surface textures.

they have been captured from the same texture at different illumination directions, nevertheless, most texture synthesis studies focus on 2D textures. The 2D texture synthesis approaches cannot reproduce the model used for relighting textures at different viewpoints and illumination conditions from the original ones when dealing with 3D surface textures.

As far as we know, there are only a few publications [22-23] available in 3D surface texture synthesis. A number of representation images of the sample surface texture, which can be used to represent the complex appearance of the sample, are normally required by existing approaches throughout the synthesis stage: this is due to the fact that these approaches are designed based on searching the most similar pixels or image patches [9], [15-17], [22]. In contrast, the proposed synthesis approach can encode representation images parametrically. Moreover, the generalization of this approach to 3D surface textures is straightforward, hence, the representation images of the sample are not needed through the synthesis process.

The process of 3D surface texture synthesis is conducted in four phases (see Fig. 5 for pipeline). First, the representation images of a sample surface texture are derived. Second, for each representation image, an SVM model is learnt. Third, the model is used to synthesize larger representation images. Finally, these images are relit using the relighting algorithm and novel texture images are derived in different illumination conditions.

1) Three-Dimensional Surface Texture Representation

A large set of images are normally required in order to encode the appearance of 3D surface textures in different viewpoints or lighting conditions [22]. Since it is difficult to apply synthesis approaches to the original texture database, as an alternative solution, building a compact representation model of the sample surface texture becomes important [22]. Given a 3D surface sample texture S_{3D}' and a representation function r, the sample representation images are obtained as:

$$SR = r(S_{3D}'). \tag{8}$$

The 3I [28] and eigen-based [29] methods are used to represent 3D surface textures in varied illumination conditions.

3I Representation For the surface of an object satisfying the

Lambertian reflectance law, Shashua [28] represented the image of a convex object using a linear combination of three representation images captured at three linearly independent lighting directions. The relighting operation is implemented as the product of a surface representation matrix M and a coefficient vector c at a certain illumination direction:

$$i = Mc, \tag{9}$$

where $i = (i_1, i_2, \cdots, i_m)^T$ is the image vector containing pixel values: i_1, i_2, …, i_m. Therefore, the vector c, which contains the coefficients of the linear combination, needs to be solved.

These three known lighting vectors: l_1, l_2 and l_3 corresponding to three representation images, respectively, are expressed using the lighting matrix L:

$$L = (l_1, l_2, l_3) = \begin{bmatrix} l_{1x} & l_{2x} & l_{3x} \\ l_{1y} & l_{2y} & l_{3y} \\ l_{1z} & l_{2z} & l_{3z} \end{bmatrix}. \tag{10}$$

Since the 3I method uses three images of the sample taken at an illumination slant angle and three different tilt angles [28] as representation images, the image data matrix can be written as:

$$I = \begin{bmatrix} i_{11} & i_{12} & i_{13} \\ i_{21} & i_{22} & i_{23} \\ \vdots & \vdots & \vdots \\ i_{m1} & i_{m2} & i_{m3} \end{bmatrix}, \tag{11}$$

which is also the surface representation matrix M. In I or M, each column contains the vector of a representation image. The scaled surface normal matrix is computed as:

$$N_a = IL^{-1}, \tag{12}$$

where L^{-1} can be calculated using the SVD (Singular Value Decomposition) method [22].

Regarding a lighting vector at an illumination direction:

$$l = \left(l_x, l_y, l_z\right)^T = (\cos\tau\sin\sigma, \sin\tau\sin\sigma, \cos\sigma)^T, \tag{13}$$

the new image i is obtained using:

$$i = IL^{-1}l. \tag{14}$$

According to Equation (9), we know $c = L^{-1}l$. Therefore, Equation (14) can be written as:

$$i = Mc = Ic. \tag{15}$$

This formula shows that the linear combination of three images can be used to represent an image in terms of a lighting vector.

Normally, three different images, with regard to a 3D surface texture, acquired using three different illumination tilt angles (τ) and the same slant angle (σ), are utilized by the 3I method as representation images (see Fig. 9). The new image is computed using Equation (15) based on τ and σ [22].

Eigen-Based Representation The base images in the eigen space are utilized for the eigen-based approach [29] to represent 3D surface textures (see Fig. 10). These base images are derived by applying SVD [22] to sample textures. The image data I can be described as:

$$I = U_I W_I V_I^T. \tag{16}$$

where each eigen vector of II^T, with regard to the singular value in W_I, corresponds to a column in U_I. V_I contains a set of coefficients for linear combinations and can be employed to build eigen-based images. W_I can be written as:

$$W_I = diag(w_1, w_2, w_n), \tag{17}$$

where $diag()$ is a function that can be used to obtain diagonal elements of a matrix; w_i is the singular value of I and $w_i \geq w_{i+1}$. Since singular values decrease rapidly and the first few eigenvectors encode the majority of the image data, the original W_I can be approximated by:

$$W_I{}' = diag(w_1, w_2, \cdots, w_k, 0, \cdots, 0), \tag{18}$$

where k is the number of the singular values retained. An approximation of the image matrix I is derived using:

$$I' = U_I W_I{}' V_I^T. \tag{19}$$

The image data matrix can be expressed as:

$$I = M_1 M_2, \tag{20}$$

where M_1 is the surface relighting representation matrix. If M_2 is known and a lighting model is assumed, M_1 can be solved using SVD. If we do not know M_2 or do not want to assume a lighting model, M_1 and M_2 can be derived by using SVD to analyze I. In this context, M_1 can be written as:

$$M_1 = U_I W_I{}'. \tag{21}$$

Since the last $n - k$ columns of $U_I W_I{}'$ are zeros, in essence, M_1 is an $m \times k$ matrix, likewise, M_2 can be obtained as a $k \times n$ matrix because the last $n - k$ rows of V_I^T can be assigned zeros. As a result, a set of k eigen base images are derived. The coefficients of the linear combination of these base images are provided by M_2. These coefficients can be used to generate the original images in I using:

$$I = (i_1, i_2, \cdots, i_n) = M_1 M_2, \tag{22}$$

where i_1, i_2, …, i_n are image data vectors, representing the original images captured at different illumination directions.

When different coefficients from those contained in M_2 are used, however, we can obtain a novel image under an arbitrary illumination direction using the linear combinations of those base images. In this study, 3D surface textures were represented using three (k=3) base images.

2) *Training SVM Models*

The representation images of a 3D surface texture are used for synthesis. An individual SVM model is learnt for each representation image SR_k (k=3 for both the 3I [28] and eigen-based [29] methods) ,this process is the same as that used for 2D textures (see Equations (3), (4) and (6)).

3) *Representation Image Synthesis*

Synthesis is conducted on each 3I [28] or eigen-based [29] representation image separately ,this stage is equal to the synthesis operation used for 2D textures. It is noteworthy that, however, the seeding image patches (see Fig. 3) in each representation image have to be sampled from the same pixel location within these images. As a result, we retained the spatial correspondence between multiple representation images.

4) *Relighting the Synthesized Representation Images*

After the synthesis operations are complete for all representation images of a 3D surface texture, the synthesized representation images are relit in different illumination conditions. The relighting process is described in Section III-B-1. For more details, please refer to [22].

IV. Constrained Texture Synthesis and Image Extrapolation Based on SVM

Constrained texture synthesis and image extrapolation were originally introduced by Efros and Leung [15];the former was used to fill the blank region on a texture while the latter was used to fill the outer region of an image. The approach was

Algorithm 1: The algorithm for constrained texture synthesis

```
(1) We choose a 64×64 texture patch from the original texture.
        If the original texture is larger than 64×64 pixels
            a 64×64 patch that is close to the "blank" area on the texture
            is randomly selected as the training sample.
        else
            the whole texture image excluding the blank region is used
            as the training sample.
(2) We extract features from the patch and produce the training data.
(3) We train an SVM model using the training data.
(4) We fill the blank region with synthesized pixels.
```

revised by Wei and Levoy [9] via applying a spiral synthesis order in order to avoid the bias to directions. Nevertheless, the image must be exhaustively searched for both approaches, otherwise, the selected match for the neighborhood of the pixel to be synthesized may be not optimal. The proposed synthesis approach has shown the merit that the sample can be discarded throughout the synthesis stage, we hence apply this approach to constrained texture synthesis and image extrapolation. This constrained synthesis method can perform texture inpainting (hole-filling) with only a small part of the image for training an SVM model, and discard this part after the model is learnt. Algorithm 1 describes the proposed constrained texture synthesis approach in detail. For image extrapolation, we extract features based on the linear neighborhood instead of the L-shaped neighborhood, so the pixels in the extrapolation region can be synthesized in clockwise.

V. Automatic Hole Detection Based on Self-similarity and Texture Inpainting Using SVM

Since texture inpainting can be treated as filling the "holes" contained in a texture image, this task should start with locating holes. We propose a new algorithm for detecting holes in a texture image using self-similarity features. The detected holes are inpainted using the proposed constrained texture synthesis method. To be specific, a pixel in the hole region, which is called "reference pixel", is first located, then, the *Euclidian* distance is calculated between the square neighborhood of each pixel and that of the reference pixel. All of the distance values make up a self-similarity map of the input image (for simplicity, we refer to the distance as similarity). Next, each distance value is weighted by the standard deviation computed over the neighborhood of the corresponding pixel. Otsu's threshold [30] is applied to the self-similarity map and the hole region is obtained. Finally, the proposed constrained texture synthesis approach is used to inpaint the hole region.

A. Detecting the Reference Pixel Automatically

Normally, the intensity values in the hole region of the texture image are more uniform than those that are out of this region. In terms of texture self-similarity [31-32], if we have a pixel in the hole region as the reference pixel, we can find other pixels in this region because their neighborhoods are more similar to that of the reference pixel than those of the non-hole pixels. Therefore, we first locate the reference pixel and then calculate a self-similarity map with reference to this pixel.

In [31], the reference pixel is manually selected by users according to their interested region, whereas we wish to automatically obtain it. Since illumination has little effect in the hole area, the distribution of the intensity in the area is even. Thus, the reference pixel is the one which receives the least influence by the illumination. If we calculate the variance of the pixel values in each $M \times M$ neighborhood, the location where the variance is minimal will be the reference pixel in the hole region. The method can be expressed as:

$$P^{*}_{ref} = min_P \sum_{P' \in N_P} Var\left(I(P')\right), \quad (23)$$

where P^{*}_{ref} is the reference pixel of the image I which contains a hole, N_P is the $M \times M$ square neighborhood around the pixel P, and $Var()$ is the function for computing the variance [32].

The detected reference pixel is the least sensitive pixel to the illumination in the image. Thus, all pixels in the hole region own similar intensity values. Due to this, the pixel with the minimal variance lies in the hole region.

B. Obtaining the Hole Region Based on Self-similarity

The *Euclidean* distance is first calculated between the square neighborhood of each pixel P contained in the image I and that of the reference pixel P^{*}_{ref}. This is expressed as:

$$d\left(N_P, N_{P^{*}_{ref}}\right) = \sqrt{\sum_p (N_P(p) - N_{P^{*}_{ref}}(p))^2}. \quad (24)$$

All of these distance values are comprised of a self-similarity map of I. Each value is weighted by the standard deviation computed over the corresponding neighborhood, then, the threshold for the weighted self-similarity map is acquired using Otsu's method [30]. Finally, this threshold is applied to the self-similarity map. The hole region contains the pixels whose self-similarity value is lower than the threshold.

1) Obtaining the Self-similarity Map

Given a pixel $I(i,j)$ in a texture image I, the similarity $Sim(i,j)$ between the neighborhood of the reference pixel and the neighborhood of this pixel is calculated. All the similarity values computed make up a self-similarity map, for simplicity, we use the distance as similarity here. Although the distance is dissimilarity in essence, this process does not affect detection results while it does boost the computational speed. The computation of the self-similarity map is expressed as:

$$Sim(i,j) = \sqrt{\sum_{m,n=-N/2}^{N/2} [I(i+m, j+n) - I(i_r+m, j_r+n)]^2}, \quad (25)$$

where $Sim(i,j)$ is the self-similarity at the pixel location (i,j), N is the size of the neighborhoods involved (N is assigned the same value as M used in the reference pixel selection stage), and (i_r, j_r) represents the location of the reference pixel [32].

2) Deriving the Hole Region Using the Self-similarity Map

Since the self-similarity map represents the similarity between the pixels in an image and the reference pixel, there must be a threshold T for the self-similarity map, which can be used to separate the hole region from the normal texture region. For an automatic threshold selection, a straightforward choice is Otsu's approach [30]; we first applied it to the self-similarity map. The hole label map $HL(i,j)$ can be derived according to:

$$HL(i,j) = \begin{cases} 0, & Sim(i,j) > T \\ 1, & Sim(i,j) \leq T \end{cases}, \quad (26)$$

where "1" means the hole pixel while "0" indicates the normal texture pixel. After conducting extensive experiments, however,

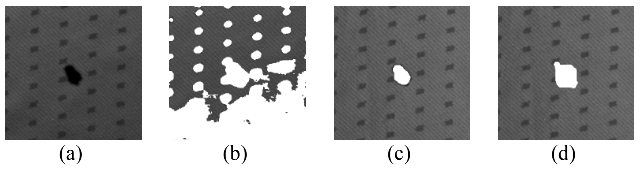


Fig. 6. (a) A texture image containing a hole; (b) the result obtained by applying Otsu's threshold [30] to the self-similarity map, where the hole is not correctly located; (c) the result derived by applying the Otsu's threshold obtained from the weighted self-similarity map to the original self-similarity map; and (d) the result with an enlarged hole.

we found that this operation failed to produce acceptable results on some texture images. The threshold obtained was often larger than what it actually was, Figs. 6 (a) and (b) show a defected texture image and the result obtained by applying Otsu's threshold [30] to the self-similarity map respectively; it can be seen that many non-hole regions are wrongly detected.

Considering the hole region is relatively homogeneous, the self-similarity map should be weighted by a certain statistic in terms of the homogeneity. Hence, the standard deviation $std(i,j)$ computed over the $N \times N$ neighbourhood of each pixel location (i,j) is used to weight the self-similarity value $Sim(i,j)$ at this location. The weighting process is described as:

$$Sim(i,j)' = std(i,j) \times Sim(i,j). \tag{27}$$

The threshold T' is obtained using Otsu's method [30] on the weighted self-similarity map. In this experiment, we used T' as the approximate threshold for the self-similarity map $Sim(i,j)$ where equation (26) was utilized together with T'. It has been shown that the approximation is a better choice than the threshold T, this benefits from the fact that the homogeneity of the hole region is enhanced by applying the standard deviation to weighting the self-similarity map. Therefore, the threshold T' produces better results than T. Fig. 6 (c) shows the result obtained using T'.

However, since the neighborhood of the pixels that are in the distance of $\lfloor N/2 \rfloor$ pixels from the boundary of the hole region contains not only the hole region but also the texture region, the self-similarities at those pixels' positions are larger than they should be. This results in the fact that the hole region is incorrectly located and the detected region is smaller than its actual size. Thus, we enlarge the hole region from inner to outer for $\lfloor N/2 \rfloor$ pixels after it has been detected. Besides, there may be a "cottony" phenomenon or the shadow nearby the border of the hole or the border itself is coarse. We therefore further enlarge the hole region for several pixels. These processes yield a larger hole region than its actual size. However, it guarantees that the hole region is completely detected. Fig. 6 (d) shows the final resultant image with the enlarged hole region.

After hole detection is complete, a hole label map HL is built. In this map, the location of all the pixels in the hole is marked with "1" while the position of the other pixels is labeled as "0".

C. Inpainting for 2D Textures

We use the constrained texture synthesis method introduced in Section IV to fill the hole detected. During each synthesis pass, a hole pixel location is filled and the label map is updated. This process is repeated until all of the "1" positions in the label

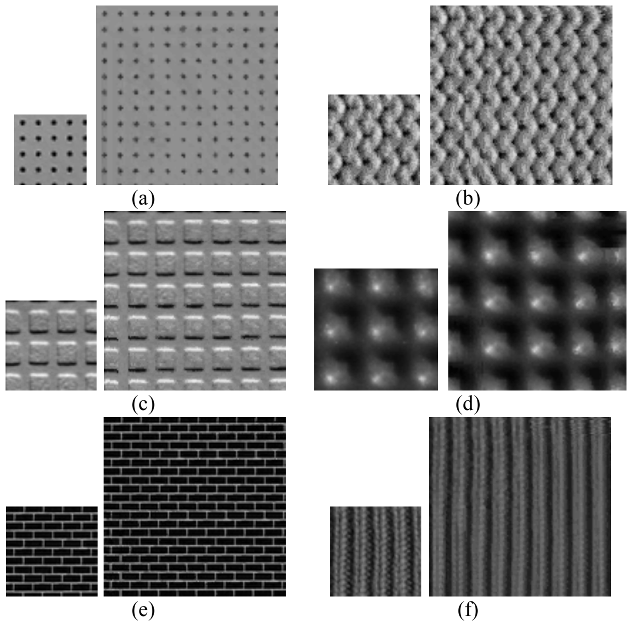


Fig. 7. Six sets of successful 2D texture synthesis results using near-regular or regular textures. In each sub-figure, the left image is the sample texture while the right image is the resultant texture.

TABLE I
THE MEANS AND STANDARD DEVIATIONS OF THE FSIM [34] AND VSI [35] VALUES COMPUTED BETWEEN SAMPLE TEXTURES AND THE CORRESPONDING SYNTHESIZED TEXTURES DERIVED USING THE PROPOSED SVM-BASED METHOD AND FIVE BASELINES.

Method	SVM	Quilting [16]	Nonparametric [15]
FSIM	**0.91±0.11**	0.71±0.08	0.58±0.18
VSI	**0.95±0.06**	0.83±0.06	0.75±0.11
Method	CNN [10]	Deep Corr. [19]	RF
FSIM	0.73±0.07	0.62±0.11	0.61±0.20
VSI	0.86±0.05	0.78±0.07	0.76±0.13

map have been updated.

D. Three-Dimensional Surface Texture Inpainting

We further extend the 2D texture inpainting method to 3D surface textures: the 3I method [28] is used. The inpainting operation is first performed on each base image separately, then, the surface albedo and gradient maps are extracted from the inpainted images to represent Lambertian surfaces [22]. Finally, relighting is conducted on these maps in order to generate novel images in given illumination conditions.

VI. EXPERIMENTAL RESULTS

In Sections III, IV and V, we introduced the SVM-based 2D texture and 3D surface texture synthesis, constrained texture synthesis, image extrapolation and texture inpainting methods. In this Section, we report the results derived using these methods (more results can be found in the supplemental material).

A. Dataset

The *PhoTex* dataset [33] was used for 2D and 3D surface texture synthesis experiments. This dataset was introduced in order to provide photometric data for texture analysis. The surface texture images contained in the *PhoTex* dataset were

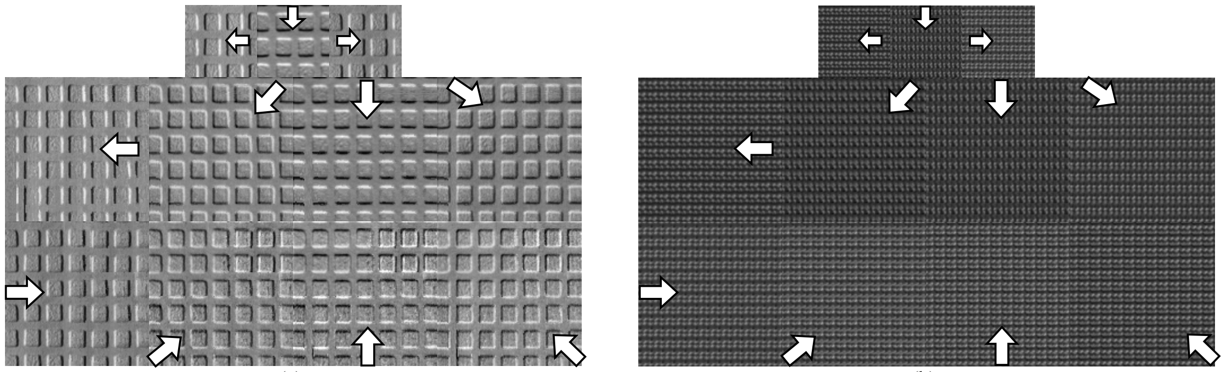

(a) (b)

Fig. 9. The 3D surface texture synthesis results derived when 3I representation images [28] are used. In each sub-figure, the first row displays three sample texture images while the second and third rows shown eight different resultant images. Block arrows imply illumination directions.

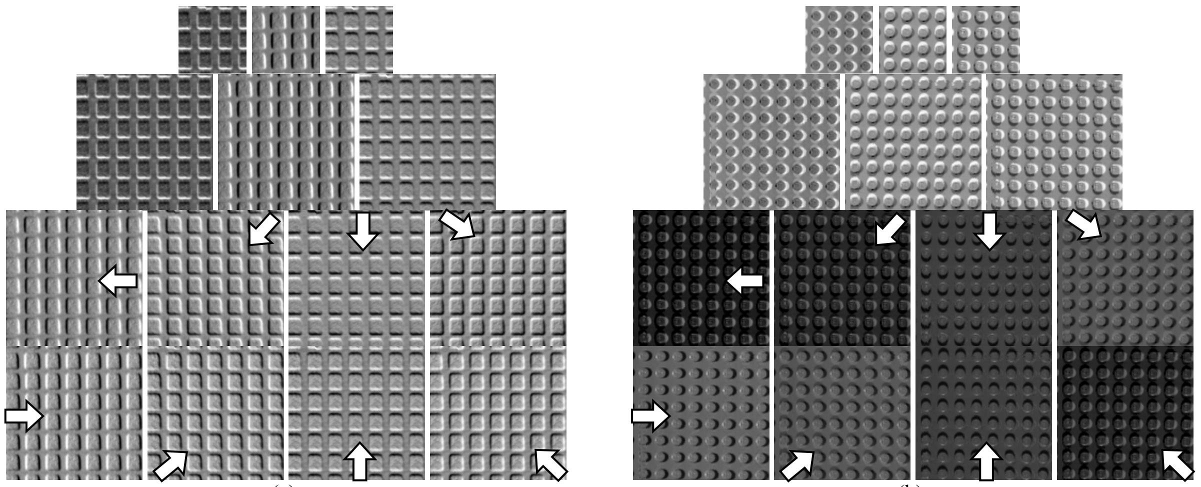

(a) (b)

Fig. 10. The 3D surface texture synthesis results obtained when eigen-based representation images [29] are used. In each sub-figure, the first row shows the sample base images; the second row displays the synthesized base images; and the third and fourth rows present the relighting results. In this figure, block arrows indicate illumination directions.

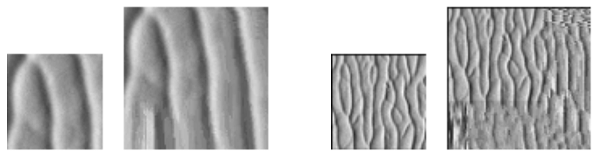

(a) (b)

Fig. 8. Two sets of failed 2D texture synthesis results using irregular textures. In each sub-figure, the left image is the sample texture while the right one is the resultant texture.

acquired under controlled illumination conditions and constant viewpoint.

For texture inpainting, we collected a set of 20 2D texture images. Each of these images contains at least one hole, those images were then used for the hole detection and texture inpainting experiments.

B. Two-Dimensional Texture Synthesis

We compare the SVM-based texture synthesis approach with its counterparts. To our knowledge, there is no computational measure for quantitatively evaluating the quality of texture synthesis. Image quality assessment (IQA) aims to evaluate the quality of an image variant by comparing it with the original image, similarly, we must compare the synthesized texture with the sample texture if we intend to assess the quality of the synthesized texture, therefore, we applied two IQA measures to the assessment of texture synthesis results. Specifically, we computed the Feature-Similarity (FSIM) index [34] and Visual Saliency-Induced (VSI) index [35] between the sample and synthesized textures in order to measure the texture synthesis quality.

1) Baselines

Image Quilting We used 21×21 patches for the texture synthesis method that Efros and Freeman [16] proposed.

Non-parametric Sampling The 19×19 search window was used for the approach that Efros and Leung [15] developed.

CNN The implementation that Gatys *et al.* [10] published was utilized along with default parameters.

Deep Corr. We utilized the source code that Sendik and Cohen-Or [19] published with the default parameters.

Random Forest Inspired by the success of Random Forests (RF) [25] on postal address block location [36], we implemented texture synthesis based on this technique by referring to the SVM-based texture synthesis method.

2) Results

Table I shows the means and standard deviations of the FSIM

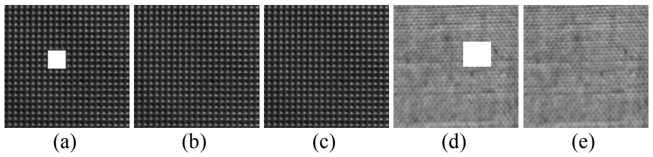


Fig. 11. The SVM-based constrained texture synthesis results: (a) and (d) shows the original texture images; (b) and (e) are the resultant images generated in the raster-scan order with regard to the images shown in (a) and (d) respectively; (c) is the resultant image derived in the spiral order corresponding to the image in (a).

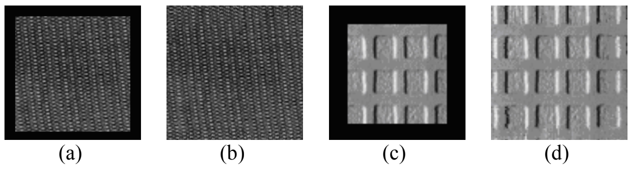


Fig. 12. Results generated by the SVM-based image extrapolation approach: (a) and (c) show the texture images with superimposed black borders while (b) and (d) show the resultant images.

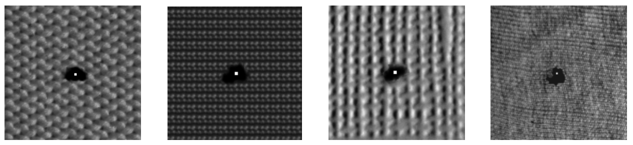


Fig. 13. Four resultant images with the reference pixels labeled. Note that we label the reference pixel using a small, white region in order to display it more clearly.

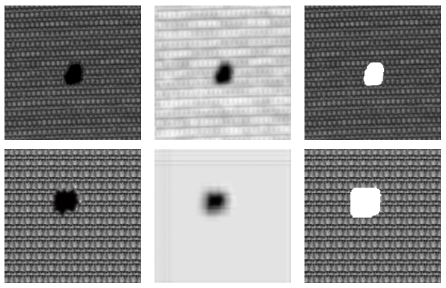


Fig. 14. (Left): the original texture images with a hole; (middle) the corresponding self-similarity maps; and (right) the resultant images obtained using our hole detection algorithm in terms of the original images. The white region indicates the final detected hole region.

[34] and VSI [35] values calculated between sample textures and the resultant textures synthesized using the proposed SVM-based method and the five baselines. As can be seen, the synthesis results obtained using our method are better than those produced using the baselines no matter which performance measure is used. Especially, the CNN-based method [10] cannot produce promising results for regular textures. This finding is similar to that Gatys *et al.* [10] observed. In addition, Figs. 7 (a)-(f) show the synthesis results of six different near-regular or regular textures respectively. As can be observed, the textures synthesized using the proposed approach do not contain obvious artifacts.

Fig. 8 further shows two sets of failed results when an irregular texture is synthesized at two scales respectively. These failed results may be attributed to the fact that the features that we used were not adequate for representing irregular textures. In addition, the L-shaped neighborhood is asymmetric which may have made the synthesis process sensitive to texture direction.

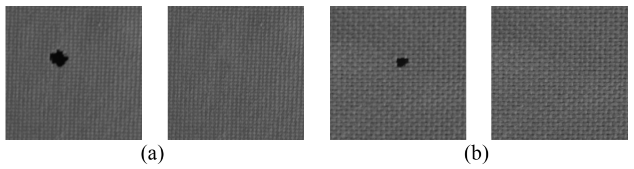


Fig. 15. Two groups of 2D texture inpainting results. In each group, the left side shows an original texture image with a hole while the right side displays a resultant image inpainted using the proposed method.

C. Texture Synthesis for 3D Surface Textures

We used two types of 3D surface texture representations. Fig. 9 shows two groups of synthesis results obtained using the 3I representation [28]. Also, Fig. 10 shows two sets of results derived by synthesizing and relighting Eigen-based images [29]. It is shown that our method can synthesize 3D surface textures.

D. Constrained Texture Synthesis and Image Extrapolation

We tested the constrained texture synthesis method in both the raster-scan and spiral orders [9]. Since the results produced using the spiral order are very close to those obtained using the raster-scan order (see Figs. 11 (a)-(c)), the raster-scan order was mainly used. Fig. 11 (e) presents the result derived using the proposed method on the image displayed in Fig. 11 (d). It can be observed that our approach performs well for constrained texture synthesis, furthermore, two sets of image extrapolation results obtained using the proposed method are displayed in Fig. 12. Here, it is shown that our method also produces good results for image extrapolation.

E. Hole Detection and Texture Inpainting

In this experiment, we first performed the hole detection operation, Fig. 13 displays four resultant images with the reference pixel labeled in white. Obviously, all reference pixels locate in the hole regions. Fig. 14 shows two groups' resultant images obtained using our hole detection method. It is shown that our method is able to locate the hole regions in those defective textures, furthermore, the proposed constrained texture synthesis method was used to inpaint the detected holes. Fig. 15 presents original 2D texture images and the corresponding inpainted images. As it can be seen, there is no obvious artifact in the inpainted images.

VII. Conclusions and Future Work

We have proposed a texture synthesis approach based on Support Vector Machines (SVM) [20]. The parameters of an SVM model are learnt from the input texture, which can be discarded throughout the synthesis process. We further applied this method to three additional tasks: constrained texture synthesis, image extrapolation and texture inpainting. Notably, only a small patch of the original texture is required for training an SVM model in order to learn its texture pattern. For texture inpainting, we developed an automatic hole detection method using self-similarity features, the learnt SVM model can be used to fill the hole region or extrapolate the outer region of the input image using a pixel-by-pixel scheme. The merit of the proposed approach is that the pixels in the synthesized region are generated by the prediction operation based on the learnt model rather than by exhaustively searching for the best match in the input texture , in addition to this, the proposed approach was used to synthesize and inpaint 3D surface textures [22-23]. Our approach was compared with five different baseline methods in the scenario of 2D texture synthesis, with the experimental results showing that the proposed approach performed better than the five methods.

However, shortfalls of the proposed synthesis approach could be seen when used to synthesize irregular textures, despite it effectively synthesizing semi- or highly regular textures. In addition to improving the results derived using irregular textures, future work may explore the possibility of texture synthesis by combining both SVM and image patch based representation methods. Recently, the importance of the spatial relationship between local image regions to texture representation has been highlighted [26], consequently, the spatial relationship between image patches will be required in order to capture global texture patterns.

References

[1] H.-H. Loh, J.-G. Leu, and R.C. Luo, "The Analysis of Natural Textures Using Run Length Features," *IEEE Transactions on Industrial Electronics*, vol. 35, no. 2, pp. 323-328, 1988.

[2] H. Potlapalli and R. C. Luo, "Fractal-Based Classification of Natural Textures," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 142-150, 1998.

[3] C. Cheng, W. Xu, and J. Shang, "Distributed-Torque-Based Independent Joint Tracking Control of a Redundantly Actuated Parallel Robot With Two Higher Kinematic Pairs," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1062-1070, 2016.

[4] C. Sun, W. L. Xu, J. E. Bronlund, and M. Morgenstern, "Dynamics and Compliance Control of a Linkage Robot for Food Chewing," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 377-386, 2014.

[5] G. Yue, C. Hou, K. Gu , S. Mao, and W. Zhang, Biologically Inspired Blind Quality Assessment of Tone-Mapped Images, *IEEE Transactions on Industrial Electronics*, vol. 65, no. 3, pp. 2525-2536, 2018.

[6] C. Boukouvalas, J. Kittler, R. Marik, and M. Petrou, "Color Grading of Randomly Textured Ceramic Tiles Using Color Histograms," *IEEE Transactions on Industrial Electronics*, vol. 46, no. 1, pp. 219-226, 1999.

[7] M. J. Ferreira, C. P. Santos, and J. Monteiro, "Cork Parquet Quality Control Vision System Based on Texture Segmentation and Fuzzy Grammar," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 756-765, 2009.

[8] Q. Chao, Z. Deng, J. Ren, Q. Ye and X. Jin, "Realistic Data-Driven Traffic Flow Animation Using Texture Synthesis," *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[9] L. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," *Proc. the 27th annual conference on Computer graphics and interactive techniques*, pp. 479-488, 2000.

[10] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture Synthesis Using Convolutional Neural Networks," *Proc. Advances in Neural Information Processing Systems*, 28, 2015.

[11] M.L. Koudelka, S. Magda, P.N. Belhumeur, and D.J, Kriegman, "Acquisition, Compression, and Synthesis of Bidirectional Texture Functions," *Proc. the 3rd International Workshop on Texture Analysis and synthesis*, pp. 59-64, 2003.

[12] J. Xie, W. Hu, S. C. Zhu, and Y. N. Wu, "Learning Sparse FRAME Models for Natural Image Patterns", *International Journal of Computer Vision*, vol. 112, pp. 221-238, 2015.

[13] X. You, W. Guo, S. Yu, K. Li, J. C. Príncipe, and D. Tao, "Kernel Learning for Dynamic Texture Synthesis," *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4782-4795, 2016.

[14] D. Dai, H. Riemenschneider, G. Schmitt, and L. Van Gool, "Example-based Facade Texture Synthesis and Inpainting," *Proc. IEEE International Conference on Computer Vision*, pp. 1065-1072, 2013.

[15] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," *Proc. the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1033-1038, 1999.

[16] A. A. Efros and W. T, Freeman," Image Quilting for Texture Synthesis and Transfer," *Proc. the 28th conference on Computer graphics and interactive techniques*, pp. 341-346, 2001.

[17] L. Liang, C. Liu, Y. Xu, B. Guo, and H, Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transactions on Graphics*, vol. 20, pp. 127-150, 2001.

[18] K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. J, Koenderink, "Reflectance and Texture of Real-World Surfaces," *ACM Transactions on Graphics*, vol. 18, pp. 1-34, 1999.

[19] O. Sendik and D. Cohen-Or, "Deep Correlations for Texture Synthesis," *ACM Transactions on Graphics*, vol. 36, no. 5, 2017.

[20] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[21] N. Cristianini and J. Shawe-Taylor, *Kernel Methods for Pattern Recognition*, Cambridge University Press, 2004.

[22] J. Dong and M, Chantler, "Capture and synthesis of 3D surface texture," *International Journal of Computer Vision*, vol. 62, pp. 177-194, 2005.

[23] J. Dong, G. Sun, and G. Chen, "Conversions between three methods for representing 3D surface textures under arbitrary illumination directions," *Image and Vision Computing*, vol. 26, pp. 1561-1573, 2008.

[24] J. Dong, Y. Duan, G. Sun, and L. Qi, "Texture synthesis by Support Vector Machines," *Proc. International Conference on Pattern Recognition*, pp. 1-4.

[25] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[26] X. Dong and M. J. Chantler, "The Importance of Long-Range Interactions to Texture Similarity," LNCS 8047, pp. 425-432, 2013.

[27] R. Fan, P. Chen, and C. Lin, "Working set selection using the second order information for training SVM," *Journal of Machine Learning Research*, vol. 6, pp. 1889-1918, 2005.

[28] A, Shashua, *Geometry and Photometry in 3D visual Recognition*, Ph.D. Dissertation, MIT, 1992.

[29] R. Epstein, P. W. Hallinan, and A. L, Yuille, "5±2 eigenimages suffice: an empirical investigation of low-dimensional lighting models," *Proc. the Workshop on Physics-Based Modeling in Computer Vision*, pp. 108-116, 1995.

[30] N. Otsu, "A threshold selection method from gray-level histogram," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, pp. 62-66, 1979.

[31] J. Dong, L. Qi, J. Ren, and M. Chantler, "Self-Similarity Based Editing of 3D Surface Textures," *Proc. the 4th International Workshop on Texture Analysis and Synthesis*, pp. 71-76, 2005.

[32] L. Qi, L. Zhang, J. Dong, and Z. Yu, "Self-similarity Based Classification of 3D Surface Textures," *Proc. 2008 International Congress on Image and Signal Processing*, vol. 2, pp. 402-406, 2008.

[33] http://www.macs.hw.ac.uk/texturelab/resources/databases/Photex/index.htm

[34] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: a feature similarity index for image quality assessment," *IEEE Trans Image Process*., vol. 20, no. 8, pp. 2378-2386, 2011.

[35] L. Zhang, Y. Shen, and H. Li, "VSI: A Visual Saliency-Induced Index for Perceptual Image Quality Assessment," *IEEE Transactions on Image Processing*, vol. 23, no. 10, pp. 4270-4281, 2014.

[36] X. Dong, J. Dong, H. Zhou, J. Sun, and D. Tao "Automatic Chinese Postal Address Block Location Using Proximity Descriptors and Cooperative Profit Random Forests," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4401-4412, 2018.