

# Texture Synthesis by Support Vector Machines

Junyu Dong<sup>1</sup> Yuanxu Duan<sup>1</sup> Guimei Sun<sup>2</sup> Lin Qi<sup>1</sup>

<sup>1</sup>Department of Computer Science, Ocean University of China [dongjunyu@ouc.edu.cn](mailto:dongjunyu@ouc.edu.cn)

<sup>2</sup>Wuxi No.1 Senior Middle School, Jiangsu, China [sunguimeiwuxi@163.com](mailto:sunguimeiwuxi@163.com)

## Abstract

We introduce a simple texture synthesis method based on Support Vector Machines (SVM). Although SVM has been effectively used for various pattern recognition tasks, there is no report available on directly applying SVM for texture synthesis. The advantage of using SVM is that the sample can be simply modeled by a linear model and is not required during the synthesis stage. In addition, the method can be further extended to synthesize 3D surface texture or Bidirectional Texture Functions. Our experimental results show that the method can successfully model and synthesize semi or highly structured textures, which can be difficult subjects for previous texture synthesis methods based on parametric models.

## 1. Introduction

Research into texture synthesis is normally concerned with learning and generation of 2D images of texture. Existing approaches either employ parametric texture models or apply non-parametric algorithms [2][6][10]. As suggested in a detailed review by [6], both global and local sampling strategies can be used for during synthesis process. An ideal synthesis algorithm should be able to use a compact model to describe the sample texture and can produce the output texture has identical perceptual characteristics to the sample but with arbitrary sizes. In addition, representation of real-world 3D surface texture, e.g. Bidirectional Texture Functions (BTF) can also be synthesized [1][5].

Parametric methods normally employ statistical models and can provide compact representation for the sample. However, for texture synthesis they may have problems to synthesize semi-structured or highly structured textures. In contrast, recent approaches based on non-parametric models, e.g. algorithms based on “smart copying” [2][6], can produce good results for many types of texture, especially for semi-structured texture. The disadvantage is, however, they require the input sample during the whole synthesis process. If the subjects are 3D surface textures (such as

brick, woven or knitted textiles, embossed wallpapers etc.), whose appearances require multi-dimensional representations such as BTF, then the input to these methods can be too expensive [5].

The support vector machine (SVM) has been widely used for classification and regression [12][13]. Essentially, it is a supervised learning method that employs a linear model and generates input-output mapping functions from a set of labeled training data. Thus, the model only requires few parameters. Our intuitive idea is to employ the SVM to model the input sample texture and then use the model for synthesizing the output texture.

To our best knowledge, there is no publication available regarding the direct implementation of SVM in texture synthesis. Furthermore, no parametric models like SVM have been used for the synthesis of 3D surface texture or bidirectional texture functions. Thus, the main contribution of this paper is: (1) to introduce a parametric method for modeling and synthesizing texture based on support vector machines, and (2) both 2D texture and 3D surface textures (or bidirectional texture functions) can be synthesized, meaning that 3D surface textures can also be modeled in a parametric manner.

## 2. Two-dimensional texture synthesis based on support vector machine

In general, texture synthesis is concerned with the generation of a large texture image from a small sample. We first introduce the SVM-based algorithm for synthesizing 2D texture. It comprises four phases: (1) feature extraction, (2) generation of training data set, (3) SVM training, and (4) prediction of output pixel value. The method only requires few parameters and can discard the sample during synthesis process. Figure1 shows the flowchart of the whole process.

### 2.1 Pre-processing and feature extraction

Firstly, we reduce noise in the sample so that the model training will not be affected by noise. We then decrease the number of grey levels in the sample image without affecting the image quality. The reduction of

the grey level resolution can guarantee there are enough pixels for each grey level and provide enough training data. In this paper, the number of grey levels of each image is 64.

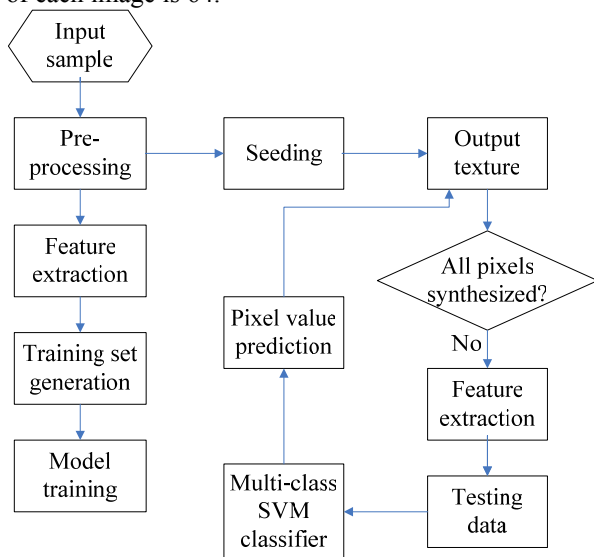


Fig. 1 the flowchart of the 2D synthesis algorithm

When SVM is used for pattern recognition tasks, the input is normally a feature vector and the output is a class label. For texture synthesis, we use the pixel intensity as the class label. Thus, the input feature vector should be able to describe the characteristics of the output pixel. Since the output texture can be assumed to follow Markov Random Field [10], i.e. the value at certain pixel location only depends on its neighborhood, an obvious choice is to extract features from its immediate neighborhood. In this paper, we use the “L” shaped neighbor as used in [11], since the synthesis process starts from the top left location of the image and in a raster order.

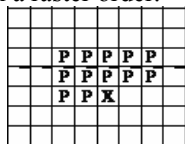


Fig. 2 The neighborhood defined by Wei and Levoy [11]. It uses local populated neighbor pixels (marked as “P”). The current output pixel is marked as X.

Many well-known texture features can be used, e.g. spatial and transformation features. However, high dimensionality of feature vectors requires a large amount of computation during training and prediction. We simply use all pixel values in the defined neighborhood to form a feature vector. In this way, expensive computation of additional feature extraction is also avoid.

## 2.2 Model training

We first need to generate a training data set so that it can be used to train the SVM model. We simply scan the whole sample image in a raster order and take each pixel value as its class label, while using pixels in its L-shaped neighborhood as shown in Figure 2 to form the corresponding feature vector. In Figure 2, the pixel value marked by “X” is used as the class label, while all pixel values marked by “P” are used to form the feature vector of pixel “X”.

With the training data set, we then train the multi-class SVM classifier and use the cross-validation method to optimize the parameters.

## 2.3 Synthesis

Once the model is produced, it can be used for texture synthesis by predicting the pixel value i.e. the class label in the output image. We first generate a white noise image with a pre-defined size as the initial output. Then we randomly select a small patch from the sample image and seed the patch in the output image. We require the seeded patch containing at least two texture elements, so that the feature vector of the first pixel to be synthesized can have elements coming from original texture. The seeding process is shown in Figure 3.

The output texture is synthesized in a raster order, i.e. pixels are generated in a left-right and top-down manner. As shown in Figure 1 and Figure 2, all pixel values in the L-shape neighbor are used to form a feature vector and the SVM model is used to determine the value of the centered pixel. This process is repeated until all pixels are assigned values.

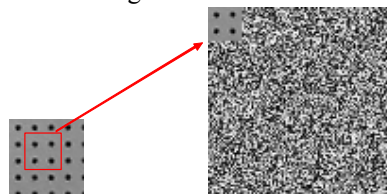


Fig. 3 The seeding process.

## 3. Synthesis of 3D surface texture

Real-world surface textures normally comprise rough surface geometry and various reflectance properties, which can produce dramatic effect on the appearances of the sample texture under varied illumination directions. For example, Figure 4 shows two example images of a 3D surface texture illuminated from two directions. However, the majority of research into texture synthesis is concerned with 2D images of texture. If the subjects are 3D surface textures (such as brick, woven or knitted textiles, embossed wallpapers etc.), then 2D techniques cannot provide the information required for rendering under other than the original illumination and viewpoint conditions.

Only a small number of publications are available regarding synthesis of 3D surface texture [5]. Existing methods all require the set of sample representation images during synthesis process, as the basic 2D synthesis algorithms are based on “smart copy” ideas and employ local sampling strategies. Instead, our SVM-based synthesis algorithm can explicitly model the representation image set in a parametric way, and the extension from 2D to 3D synthesis is straightforward. Thus, the sample data set is not required during synthesis.

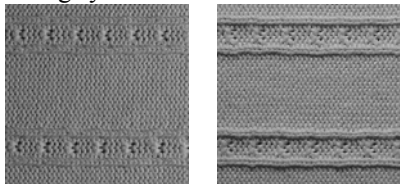


Fig. 4 Two images of a 3D surface texture imaged under differing illumination. The texture is from an Aran jumper.

The synthesis of 3D surface texture comprises following stages: (1) extracting representation images of the sample surface texture, (2) training SVM models for each representation image, (3) generating new representation images using SVM, and (4) relighting the synthesized representation to generate new texture images under varied illumination directions.

### 3.1 Representation of 3D surface texture

Three-dimensional surface textures require a great number of images to capture its appearance under varied illumination or view points [1]. It is important to extract a compact representation of the sample texture, as it is impractical to operate directly on the original data set [5]. Following [4] and [5], we use the 3I and Eigen methods to represent 3D surface texture under many different lighting directions.

The 3I method uses three images of the sample texture taken at an illumination slant angle of  $45^\circ$  and tilt angles of  $0^\circ$ ,  $90^\circ$  and  $180^\circ$  [9]. The Eigen method uses 3 or 6 base images in eigen-space to represent 3D surface textures. The base images can be obtained by performing Singular-Value Decomposition (SVD) on the sample image space. The advantage of using an eigen-space approach is that we may synthesize textures with complex reflectance functions, although specular spikes will require large numbers of base images [5].

### 3.2 Training

Since the subjects to be synthesized are the representation images, we simply train a model for each sample image in the representation data set. This is identical to the training stage in 2D texture synthesis. For the 3I representation method, three images are modeled separately. For the eigen representation, each

base image is modeled using support vector machines. After training, representation images are modeled separately.

### 3.3 Synthesis of representation images

Since each of the 3I or Eigen representation images can be treated as a 2D texture, synthesis can be performed separately. This is again identical to the synthesis process in 2D texture synthesis. However, we must force the seeding patches in all representation images originated from the same location in the sample images. In this way, the spatial correspondence between different representation images can be maintained.

### 3.4 Relighting

Once the result representation images are synthesized, they can be relit using different illumination directions. For the 3I representation, a linear combination of the result images can generate new texture under arbitrary illumination directions [5]. For the synthesized eigen base images, an interpolation between coefficients can produce new textures under given illumination angles.

## 4. Experimental results

In our experiments, we use texture samples from the PhoTex texture database [7] and BTF data from Yale University texture database [4]. The LibSVM package is used for SVM implementation [3].

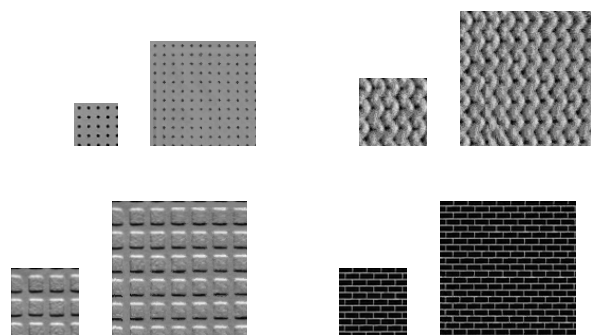


Fig. 5 2D texture synthesis results. The small image is the input sample, while the larger one is the output.

### 4.1 Results of 2D texture synthesis

Figure 5 shows the 2D texture synthesis results. It can be seen that the proposed algorithm can generate good results for semi or highly structured textures. The synthesized results are comparable with those produced by “smart copying” technique such as image quilting method proposed by [2][5], but the texture is actually modeled by a simple linear model and the sample is not required during synthesis. However, it should also be noted the current method has problems to synthesis random textures.

### 4.2 Results of 3D surface texture synthesis

The proposed method can also successfully synthesize 3D surface textures. Figure 6 shows the results produced by using the 3I representation. Figure 7 shows the results produced by synthesizing and relighting eigen based images. For the comparison purpose, in Fig. 8 we show synthesis results produced by [5], which is based on Efros' image quilting method and also uses eigen base images as input. Same sample textures are used in Fig. 7 and Fig. 8.

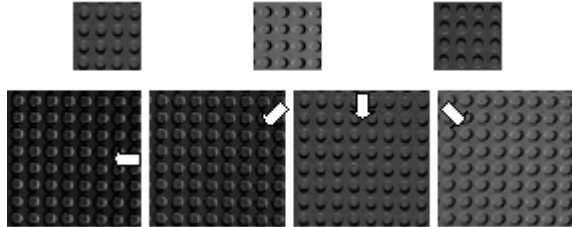


Fig. 6 3D surface texture synthesis results by using the 3I representation. Block arrows shows illumination directions.

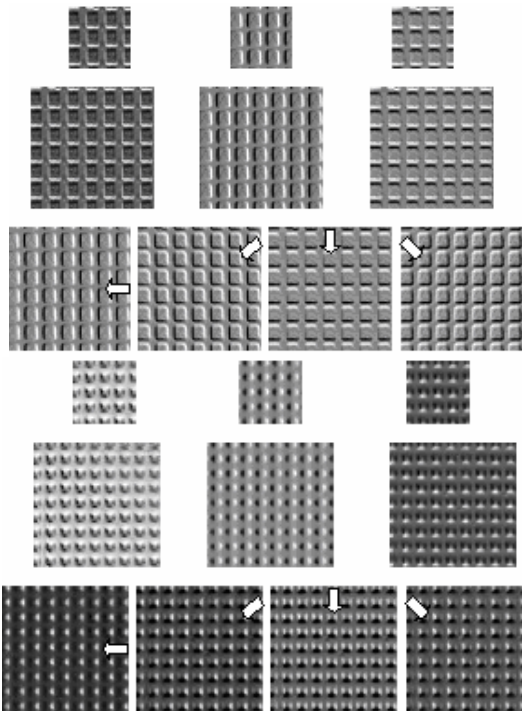


Fig. 7 3D surface texture synthesis results by using eigen base images as representation. For each group, the top row is the sample eigen base images. The second row shows the synthesized base images. The bottom row shows the relighting results.

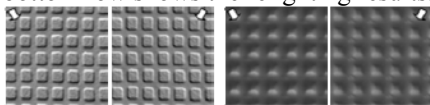


Fig. 8 3D surface texture synthesis results produced by [5], which is based on the Image Quilting method.

## 5. Conclusion

We introduce a simple texture synthesis method based on Support Vector Machines (SVM). After feature extraction and model training, the input sample can be modeled by a linear model and is not required during the synthesis stage. In addition, the method can be further extended to synthesize 3D surface texture or Bidirectional Texture Functions. Our experimental results show that the method can successfully model and synthesize semi or highly structured textures. Future work will focus on the improvement of synthesis results for random textures.

## References

- [1] Dana, K. J.; Van Ginneken, B.; Nayar, S. K. and Koenderink, J. J. Reflectance and Texture of Real-World Surfaces. *ACM Trans. on Graphics*, Vol. 18, No. 1, pp. 1-34, 1999.
- [2] Efros, A. A. and Freeman, W. T. Image Quilting for Texture Synthesis and Transfer. In *Proc. of the 28th conf. on Computer graphics and interactive techniques*, pp. 341-346, 2001.
- [3] Fan, P.-H. Chen, and C.-J. Lin. Working set selection using the second order information for training SVM. *Journal of Machine Learning Research* 6, 1889-1918, 2005..
- [4] Koudelka, M.L., Magda, S., Belhumeur, P.N. & Kriegman, D.J. Acquisition, Compression, and Synthesis of Bidirectional Texture Functions , In *Proc. of the 3rd International Workshop on Texture Analysis and synthesis*, pp59-64, , Oct. 2003
- [5] Dong, J. and Chantler, M. Capture and synthesis of 3D surface texture. *International Journal of Computer Vision*, Vol. 62: Nos. 1-2, pp177-194, April-May 2005.
- [6] Liang, L.; Liu, C.; Xu, Y.; Guo, B. and Shum, H. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, Vol. 20, No. 3, July, pp. 127-150, 2001.
- [7] <http://www.macs.hw.ac.uk/texturelab/resources/databases/Photex/index.htm>.
- [8] Portilla, J. and Simoncelli, E.P. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, Vol. 40, Issue 1, pp. 49-71, 2000.
- [9] Sashua, A. Geometry and Photometry in 3D visual Recognition. Ph.D. thesis, MIT, 1992.
- [10] Zhu, S. C.; Wu, Y. and Mumford, D. Filters, random fields and maximum entropy (FRAME). *International Journal of Computer Vision* 27(2), March/April, pp. 1-20, 1998.
- [11] Wei, L. and Levoy, M. Fast texture synthesis using tree-structured vector quantization. In *Computer Graphics Proceedings of SIGGRAPH 2000*. pp.479-88.
- [12] Vapnik V N. The Nature of Statistical Learning Theory. NY: Springer-Verlag, 1995.
- [13] Cristianini N, Shawe-Taylor J. Kernel Methods for Pattern Recognition. Cambridge University Press, 2004.