www.ietdl.org

Published in IET Image Processing
Received on 25th May 2010
Revised on 26th December 2010
doi: 10.1049/iet-ipr.2010.0428

ISSN 1751-9659

Vector-field-based deformations for three-dimensional texture synthesis

Y.-L. Su[1] *C.-C. Chang*[2] *Z.-C. Shih*[1] *W.-K. Tai*[3]

[1]*Institute of Multimedia Engineering, National Chiao Tung University, Hsinchu 300, Taiwan*
[2]*Department of Computer Science and Information Engineering, National United University, 1, Lienda, Kungching Li, Miaoli 360, Taiwan*
[3]*Department of Computer Science and Information Engineering, National Dong Hwa University, Shou-Feng, Hualien 974, Taiwan*
E-mail: ccchang@nuu.edu.tw

Abstract: A major drawback of conventional approaches to three-dimensional (3D) texture synthesis is the lack of fine-scale deformations in the synthesised results. This study presents a novel vector-field-based deformation approach to synthesising 3D textures with fine-scale deformations. The pre-process of this approach constructs feature vectors and similarity sets of voxels from an input 3D exemplar to accelerate neighbourhood matching. The synthesis process first introduces 3D vector fields for deforming synthesised results. A 3D pyramid synthesis technique integrated with 3D vector fields is then used to synthesise 3D textures. The proposed approach uses only eight neighbourhoods of each voxel for neighbourhood matching. Experimental results show that the proposed approach efficiently synthesises 3D textures with fine-scale deformations.

1 Introduction

Texture mapping, a technique for adding texture to a surface, is an important tool for modelling complex surfaces. However, distortion and discontinuity are often problematic in texture mapping. Designing a mapping technique to solve these problems is very difficult. An alternative is to use three-dimensional (3D) textures [1, 2] to address the above problems. A 3D texture can be represented as coloured voxels in a 3D space representing a material. The 3D textures have several advantages over 2D textures. First, 3D textures eliminate the need to find a parameterisation for the surface to be textured. Second, 3D textures provide texture information for an entire volume including both the surface and interior.

Procedural approaches [3] use procedures to define and produce 3D textures such as wood, clouds and so on. However, users often have difficulty expressing a desired texture procedurally, and this approach is unsuitable for many materials. This approach also has difficulty synthesising 3D textures with various deformations. Various image-based approaches [4–10] have been developed to synthesise 3D textures from 2D textures. Some methods of synthesising 3D textures [6, 8, 9] use three orthogonal slices for neighbourhood matching, which is useful for many different textures. However, the quality of synthesised results is inconsistent, and fine-scale deformations are often needed to enhance the synthesised results.

The novel vector-field-based deformation approach proposed in this paper synthesises 3D textures with fine-scale deformations from a small 3D exemplar. The proposed 3D texture synthesis approach is based on neighbourhood matching which provides flexible fine-scale deformations. Fig. 1 is a flowchart of the proposed approach. First, a small 3D exemplar is input to the proposed algorithm. The pre-process then generates feature vectors and similarity sets to accelerate neighbourhood matching during the synthesis process. For feature vector generation, a cube of information for each voxel is captured from the input 3D exemplar as a feature vector. Principal component analysis (PCA) is then applied to reduce the dimensions of the feature vector. The PCA is a standard technique for reducing the number of dimensions without excessive data loss. To construct a similarity set, the voxels most similar to those in the input 3D exemplar are identified. Finally, 3D vector fields and 3D pyramid synthesis techniques [11, 12] are introduced to synthesise 3D textures with deformations during the synthesis process. Experimental results show that the proposed approach can generate the desired deformations for 3D texture synthesis.

The remainder of this paper is organised as follows. Section 2 reviews related works. Section 3 presents the proposed approach. Section 4 presents experimental results. Section 5 reports analysis and discussion. Section 6 provides conclusions.

2 Related works

2.1 3D texture synthesis

Recently developed 3D texture synthesis approaches include Jagnow *et al.* [7], who used a stereological approach to

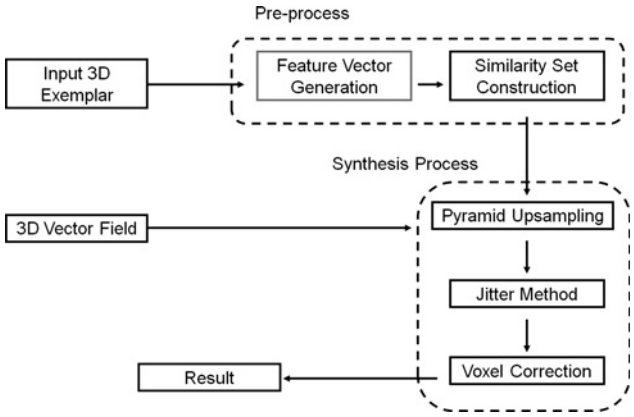

Fig. 1 *Flowchart of the proposed approach*

synthesising 3D textures from 2D textures. Their approach first analyses the materials of spherical particles and then applies it to arbitrarily-shaped particles. This approach also provides a disciplined, systematic method of predicting material structures. Chiou and Yang [4] developed a 3D texture synthesis approach based on 2D textures. Their approach uses a probability model and visual hull concept to generate desired 3D textures automatically. Qin and Yang [9] presented an algorithm for generating 3D textures from input examples. Their approach first creates aura matrix representations and then generates a 3D texture by sampling the aura matrices of an input exemplar. Kopf *et al.* [8] synthesised 3D textures from a 2D texture by extending 2D texture synthesis to synthesise 3D textures and then applying a technique combining non-parametric texture optimisation with histogram matching. Takayama *et al.* [10] proposed an approach to represent 3D objects with spatially varying oriented textures. They extended the patch-based synthesis approach of lapped textures to 3D textures and used this technique to deform non-homogeneous textures when creating solid models. Dong *et al.* [6] presented a 3D texture synthesis approach for pre-computing 3D candidates during pre-processing, synthesising a volume from a set of pre-computed 3D candidates and generating 3D textures on surfaces.

Recent advances in 3D texture synthesis enable synthesis of arbitrarily-sized 3D textures using a small exemplar. Unfortunately, several approaches have only proven suitable for particle textures whereas others do not provide sufficient deformation for 3D texture synthesis. Therefore developing fine-scale deformation for 3D texture synthesis is a prerequisite for improving the appearance of synthesised results.

2.2 Texture synthesis with deformation

Several approaches have been proposed for synthesising 2D textures with deformations. The approach developed by Ashikhmin [13] applied a texture synthesis algorithm to natural textures. This simple and efficient implementation allowed users to input interactive deformations during the synthesis process using a painting-like interface. Lefebvre and Hoppe [12] developed a texture synthesis algorithm based on neighbourhood matching to achieve parallelism when deforming synthesis textures. Their approach included a coordinate up-sampling step and a correction approach. They also introduced a method of enhancing the resolution of coarse synthesised results. Turk [14] presented another method of synthesising surface textures. In their approach, a hierarchy of points from low- to high density is created over a given surface. The points are then connected to form a hierarchy of meshes. The user then specifies a vector field over the surface that indicates texture deformation. Lefebvre and Hoppe [15] created a framework for exemplar-based texture synthesis with anisometric deformation. This method transforms an input texture from a space of pixel colours into a space of appearance vectors before performing texture synthesis in the transformed space. Zhou *et al.* [16] proposed a geometric method of synthesising texture for arbitrary manifolds, which enabled interactive and versatile editing and animation so that users could specify vector fields used to deform the synthesis direction.

The above synthesis approaches provide simple methods of deformation for 2D texture synthesis. However, developing fine-scale deformations for texture synthesis remains challenging.

3 Proposed approach

Fig. 1 illustrates the proposed approach for synthesising 3D textures from a small 3D exemplar. The pre-process of the proposed approach constructs the feature vectors and similarity sets from the small 3D exemplar to accelerate neighbourhood matching in the synthesis process. In the synthesis process, 3D vector fields are used to deform the synthesised results. The 3D pyramid synthesis technique is then introduced to synthesise 3D textures. Pyramid upsampling increases the texture sizes for different levels. The jitter method provides deterministic randomness by perturbing the textures. The voxel correction uses neighbourhood matching to ensure that the results are consistent with the input small 3D exemplar.

3.1 Pre-process

Traditionally, 3D texture synthesis using point-wise colours as features for neighbourhood matching requires a large neighbourhood size and substantial data. Replacing point-wise colours with appearance vectors [15] improves the quality of texture synthesis. Appearance vectors are more continuous and have lower dimensions than point-wise colours for neighbourhood matching. The proposed approach thus transforms point-wise colours in a colour space into feature vectors in an appearance space. These information-rich feature vectors are then used to synthesise 3D textures.

After inputting a small 3D exemplar V, colours in $m \times m \times m$ grids of each voxel in V are used to construct the feature vectors needed to form an appearance-space

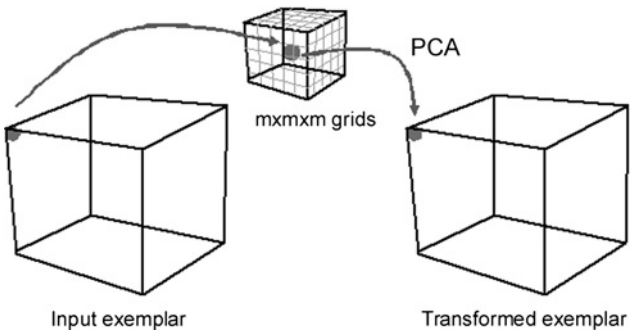

Fig. 2 *Feature vector generation from an input 3D exemplar to a transformed exemplar*

exemplar V', where m is a user-defined parameter (Fig. 2). Each voxel in the appearance-space exemplar V' contains a feature vector with $m \times m \times m \times 3$ dimensions ($m \times m \times m$ for grids and 3 for RGB). Then, PCA is performed to map V' onto a transformed exemplar $\tilde{V}'$.

For similarity set construction, the proposed method constructs a similarity set for each voxel in $\tilde{V}'$ to enhance neighbourhood matching in the synthesis process. The k most similar voxels in $\tilde{V}'$ for each voxel p are identified by using the k-coherence search method [17]. Moreover, based on the coherence synthesis principle [13], searching for candidates from the $n \times n \times n$ neighbourhoods of voxel p in $\tilde{V}'$ accelerates the synthesis process, where n is a user-defined parameter. Therefore the proposed approach searches for the k most similar voxels from the $n \times n \times n$ neighbourhoods of voxel p in $\tilde{V}'$ to construct a similarity set $C^l_{1\ldots k}(p) = \{C^l_1(p), C^l_2(p), \ldots, C^l_k(p)\}$, where l is a pyramid level, $C^l_1(p) = p$, and k is a user-defined parameter. This technique can accelerate neighbourhood matching because it does not need to search each voxel in $\tilde{V}'$ for neighbourhood matching during synthesis process. After finding $C^l_1(p)$, no voxel in the $n \times n \times n$ neighbourhoods of $C^l_1(p)$ can be $C^l_2(p)$. This eliminates the local minimum problem. Similarly, no voxel in the $n \times n \times n$ neighbourhoods of $C^l_{k-1}(p)$ can be $C^l_k(p)$ for voxel p until $C^l_{1\ldots k}(p)$ is constructed.

3.2 Synthesis process

3.2.1 Defining 3D vector fields: The proposed approach introduces 3D vector fields to deform the synthesised results. First, a 3D space containing three orthogonal axes is fixed at each point. Mathematical formulas are then applied to deform the three axes such that the synthesised results change with the vector fields. The vector field is the same size as the synthesised result. Fig. 3 shows a 3D vector field with orthogonal axes at each point and a space size of $5 \times 5 \times 5$.

After defining a 3D vector field, the vector field A_l for each level l is obtained by downsampling the 3D vector field. The inverse vector field A_l^{-1} is then computed from A_l for each level. For pyramid upsampling and voxel correction, fields A_l and A_l^{-1} are used at each level of the synthesis process.

3.2.2 Pyramid upsampling: For the pyramid upsampling shown in Fig. 4, the proposed approach synthesises one voxel

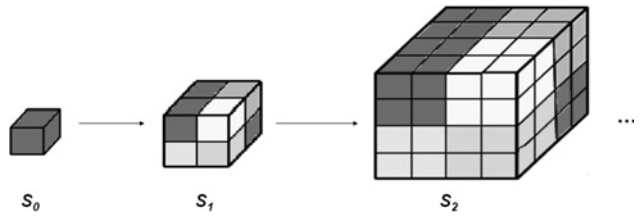

Fig. 4 *Synthesis from one voxel to a $w \times w \times w$ 3D texture*

to a $w \times w \times w$ 3D texture, $S_0 - S_L$, where $L = \log_2 w$. This step synthesises a 3D texture S in which each voxel $S[p]$ stores the coordinates of voxel p. After a voxel is constructed, its coordinates are assigned values of (1, 1, 1). The coordinates of parent voxels for the next level are then upsampled. Each of the eight children is assigned parent coordinates plus a child-dependent offset as follows

$$S_l[p] = S_{l-1}[p - \Delta] + h_l \Delta \cdot A_l(p) \tag{1}$$

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

where h_l is the regular output spacing of exemplar coordinates in level l, and $h_l = 2^{(\log_2 w - l)}$; Δ is the relative locations of the eight children; A_l is the vector field used to compute the spacing distance for level l.

3.2.3 Jitter method: The upsampled coordinates are then jittered to achieve deterministic randomness. The upsampled coordinates at each level are perturbed by

$$S_l[p] = S_l[p] + J_l(p) \tag{2}$$

where $J_l(p) = h_l H(p) r_l$ is a jitter function produced by a hash function $H(p)$: $Z^2 \rightarrow [-1, +1]^2$ and a user-defined per-level parameter r_l.

3.2.4 Voxel correction: The voxel correction [15] modifies the jittered coordinates to recreate neighbourhoods

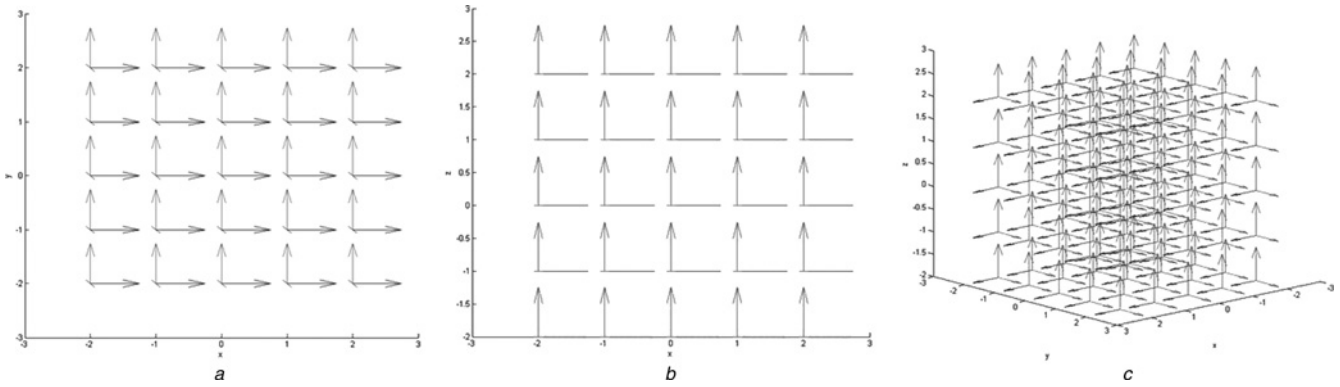

Fig. 3 *$5 \times 5 \times 5$ 3D vector field with orthogonal axes on the*

a *XY*-plane
b *XZ*-plane
c With three orthogonal axes at each point

doi: 10.1049/iet-ipr.2010.0428

similar to those in the transformed exemplar $\tilde{V}'$. Voxel correction includes the two following steps:

The first step of the voxel correction is to find the eight neighbourhoods of voxel p. The inverse vector field A_l^{-1} is then used to infer the eight warped neighbourhoods of voxel p (Fig. 5*a*). Hence, for each voxel p, the appearance vectors of its eight warped neighbourhoods at the current level are gathered and represented as a warped neighbourhood vector $\tilde{\boldsymbol{N}}_{S_l}(\boldsymbol{p})$ according to the following formula

$$\tilde{N}_{S_l}(p) = \left\{ \tilde{V}'[S[p + \tilde{\phi}(\Delta)] \middle| \Delta = \begin{pmatrix} \pm 1 \\ \pm 1 \\ \pm 1 \end{pmatrix} \right\} \tag{3}$$

where $\tilde{\varphi}(\Delta) = (\phi(\Delta)/||\phi(\Delta)||)$ maintains its rotation but removes any scaling and $\varphi(\Delta) = A_l^{-1}(p) \cdot \Delta$.

The second step of the voxel correction is to find the most similar voxel in transformed exemplar $\tilde{V}'$ to replace voxel p. For each warped voxel neighbourhood I ($I = 1-8$) of voxel p, the most similar k voxels $I_1, I_2, \ldots, I_k$ can be obtained from the similarity set of the warped voxel neighbourhood I. Fig. 5*b* shows an illustration where $I = 1$ and $k = 3$. The warped relationship by vector field A_l between voxels I and p is then used to infer the candidate voxels $I_{1p}, I_{2p}, \ldots, I_{kp}$ for voxel p. With candidate voxels $I_{1p}, I_{2p}, \ldots, I_{kp}$ for voxel p, the warped neighbourhood vectors $\tilde{N}_{S_l}(I_{1p})$, $\tilde{N}_{S_l}(I_{2p}), \ldots, \tilde{N}_{S_l}(I_{kp})$ are computed as in the first step. Therefore for voxel p, there are $8 \times k$ candidate voxels and thus $8 \times k$ warped neighbourhood vectors $\tilde{N}_{S_l}(u)$, where u is a candidate voxel. The proposed approach identifies the candidate voxel that has the best match by comparing $\tilde{N}_{S_l}(u)$ s with $\tilde{N}_{S_l}(p)$. It then replaces voxel p with this voxel.

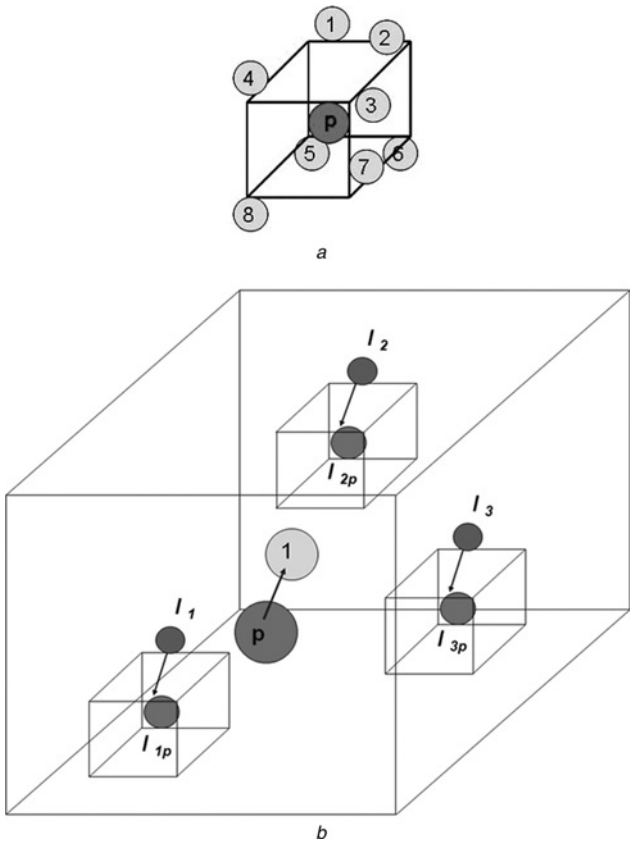

Fig. 5 *Voxel correction includes*

a Eight warped neighbourhoods of voxel p
b Most similar three voxels ($k = 3$) I_1, I_2 and I_3 of warped neighbourhood voxel $I = 1$ are used to infer candidate voxels I_{1p}, I_{2p} and I_{3p} for voxel p

4 Results

Several experiments were conducted to evaluate the effectiveness of the proposed approach. The proposed algorithm was implemented in a MATLAB environment running on a PC with a 2.66 GHz Core2 Quad CPU and 4.0 GB of memory.

For feature vector generation, m was set to 5, and PCA was used to reduce the 375-dimension feature vectors to eight-dimension feature vectors. Moreover, to eliminate border effect problems, the proposed approach discards the volume of two voxels (roughly half of the five voxels) on each border. For similarity set construction, k was set to 3 and n was set to 7. For the synthesis process, r_l was set to 0.7. All input 3D exemplars were $64 \times 64 \times 64$, and the resulting 3D textures were $128 \times 128 \times 128$. The vectors fields in the experiments were circular deformation on the XY-plane; zigzag deformation on the XY-plane; slant deformation in the 3D space; and slant deformation on the XY-plane. Fig. 6 shows examples of $5 \times 5 \times 5$ 3D vector fields.

Fig. 7*a* shows the first input 3D exemplar with the small and compact pattern. Fig. 7*b* shows the synthesised results for circular deformation on the XY-plane. Fig. 7*c* shows the synthesised results for zigzag deformation on the XY-plane. The XY-plane was deformed in two different directions. Fig. 8*a* shows that the second input 3D exemplar was a structural 3D texture. Fig. 8*b* shows the synthesised results for 3D slant deformation. Slant patterns exist on all planes and inside the volume result. The synthesised results were continuous on all planes. Fig. 8*c* shows the synthesised results for slant deformation on the XY-plane with no deformations in the XZ- and YZ-directions. These synthesised results were continuous at the cross-sections of different planes.

To maintain objectivity, the proposed approach was compared with the conventional approach [6] since both are based on the 3D pyramid synthesis technique. Since the previous approach synthesises 3D textures from a 2D texture without deformations, 3D textures were generated by the proposed approach without deformations for comparison purposes. The input was a 128×128 2D sample with a particle-like 3D texture containing few colours. Fig. 9 shows the synthesised results. The synthesised results obtained with the proposed approach are of at least comparable quality. Furthermore, the synthesised results also indicate that the proposed approach achieves slightly better preservation of features compared to the conventional approach. Finally, the synthesised results for the proposed approach reveal relatively less blurring.

Finally, the proposed approach was theoretically compared with the previous approach [18], which is currently the standard approach to 3D texture synthesis. The previous approach considers information on three orthogonal 2D slices and uses optimisation techniques with histogram matching to preserve global statistical properties. The proposed approach, however, is a novel technique for real 3D space texture synthesis that uses information-rich appearance vectors and cubic neighbourhoods during neighbourhood matching. The previous approach also has simple deformations such that (a) a different exemplar may be specified for each orthogonal view of the volume, and (b) it directly constrains the colours of specific voxels in the volume. Conversely, the proposed approach uses various vector fields for 3D texture synthesis with detailed deformations. Finally, the previous approach

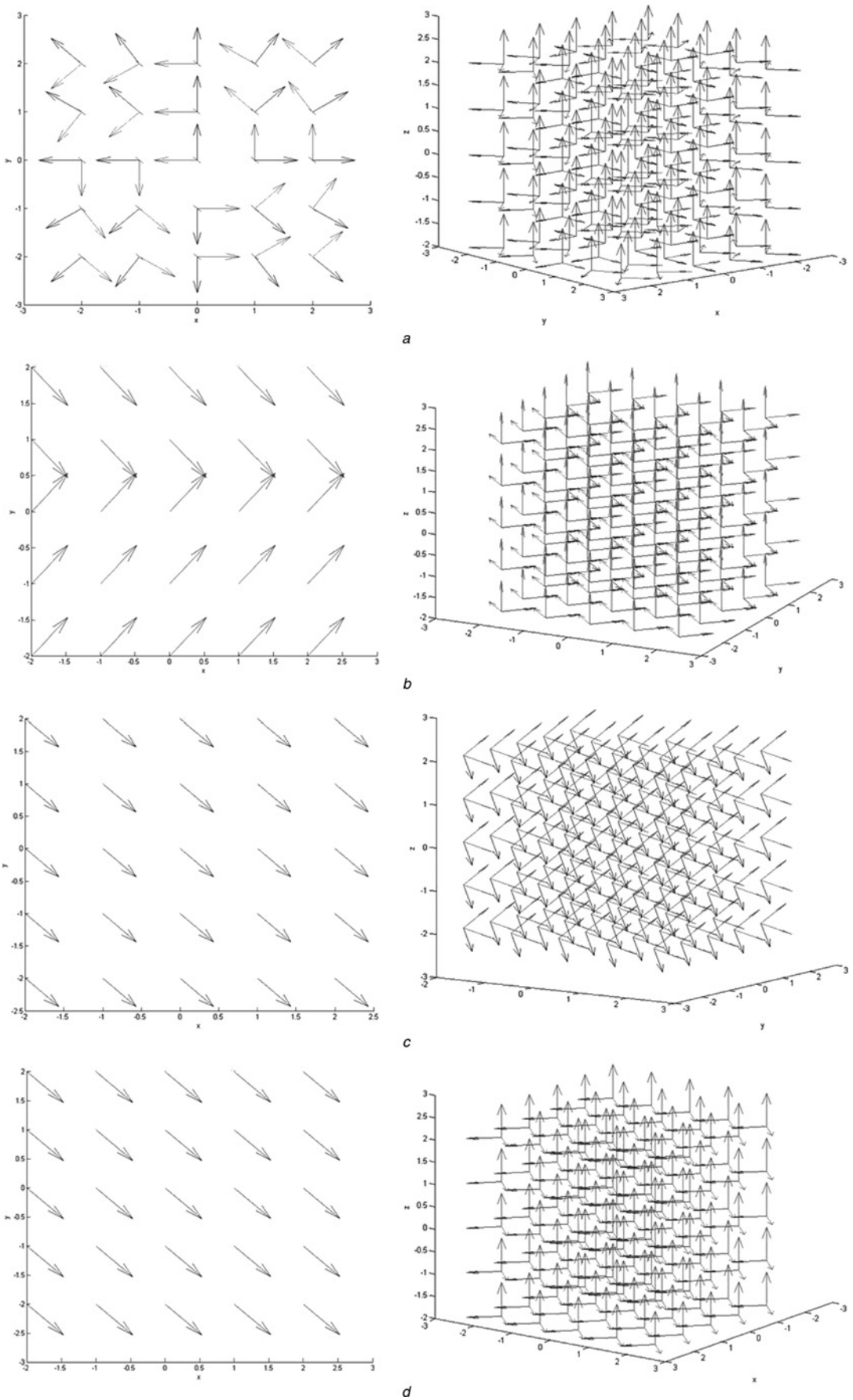

Fig. 6 *Shows examples of 5 × 5 × 5 3D vector fields*

a 3D vector field with a circular pattern on the *XY*-plane: the *XY*-plane and three axes are shown at each point
b 3D vector field with zigzag deformation on the *XY*-plane: the *XY*-plane and three axes are shown at each point
c 3D vector field with 3D slant deformation: one axis on the *XY*-plane and three orthogonal axes are shown at each point
d 3D vector field with slant deformation on the *XY*-plane: one axis on the *XY*-plane and three orthogonal axes are shown at each point

doi: 10.1049/iet-ipr.2010.0428

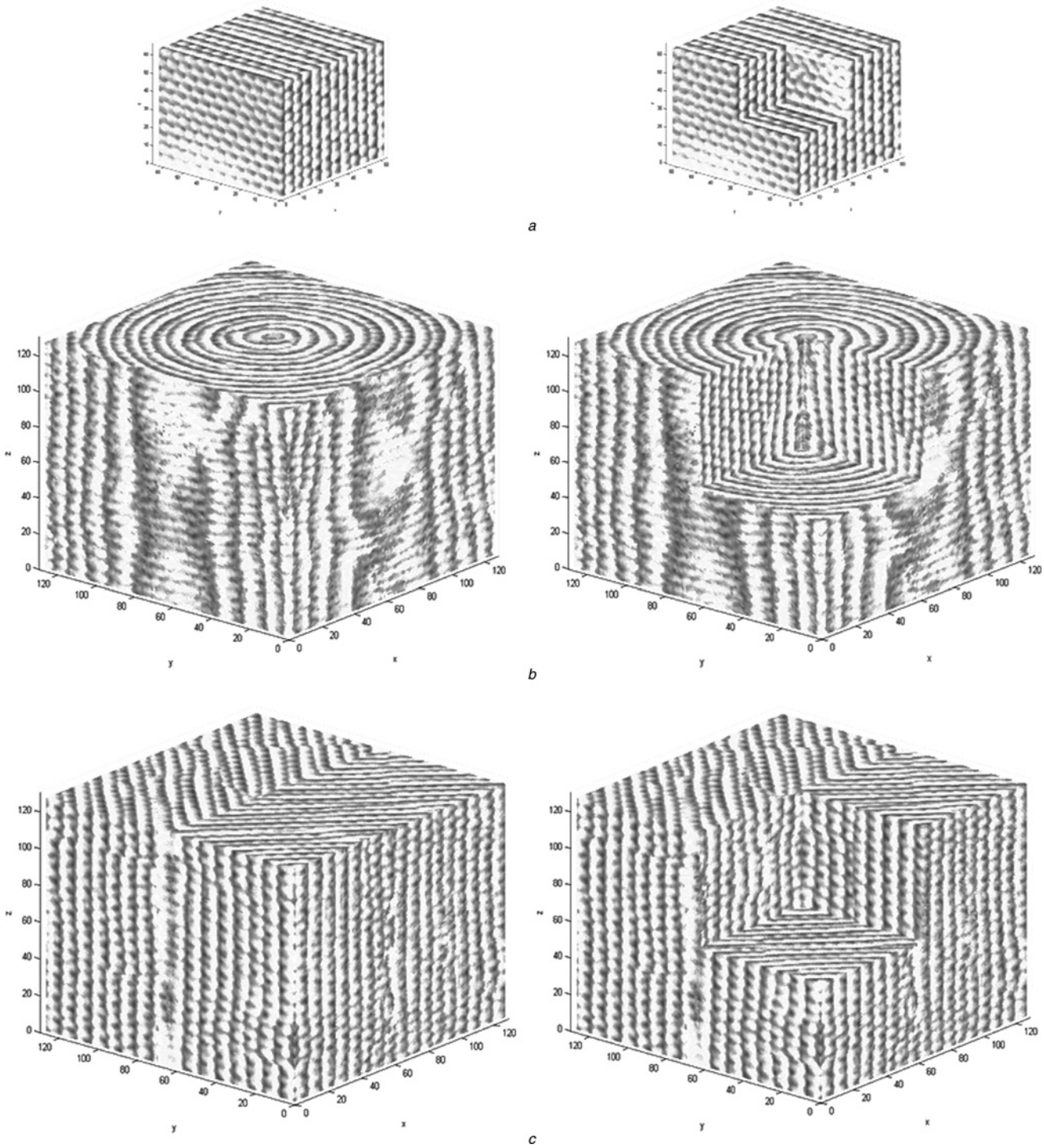

Fig. 7 *Showing the synthesised results*

a Input 3D exemplar and cross-section at $X = 32$, $Y = 32$ and $Z = 32$ for the input 3D exemplar
b Resulting 3D texture with circular deformation on the XY-plane and cross-section at $X = 64$, $Y = 64$ and $Z = 64$ for the resulting 3D texture with circular deformation on the XY-plane
c Resulting 3D texture with zigzag deformation on the XY-plane and cross-section at $X = 64$, $Y = 64$ and $Z = 64$ for the resulting 3D texture with zigzag deformation on the XY-plane

does not always yield high-quality synthesised results. The proposed approach partially solves this problem by using a 3D vector field.

5 Analysis and discussion

Most previous approaches to 3D texture synthesis consider the information on three orthogonal 2D slices and do not capture information from a 3D space. Therefore they only deform textures on the slices, and they have difficulty performing deformation in 3D space. The method of real 3D space texture synthesis proposed in this paper uses vector fields for deformable 3D texture synthesis.

The experiments clearly show the parameters needed for the proposed algorithm to obtain the desired synthesis results. Based on related works, the

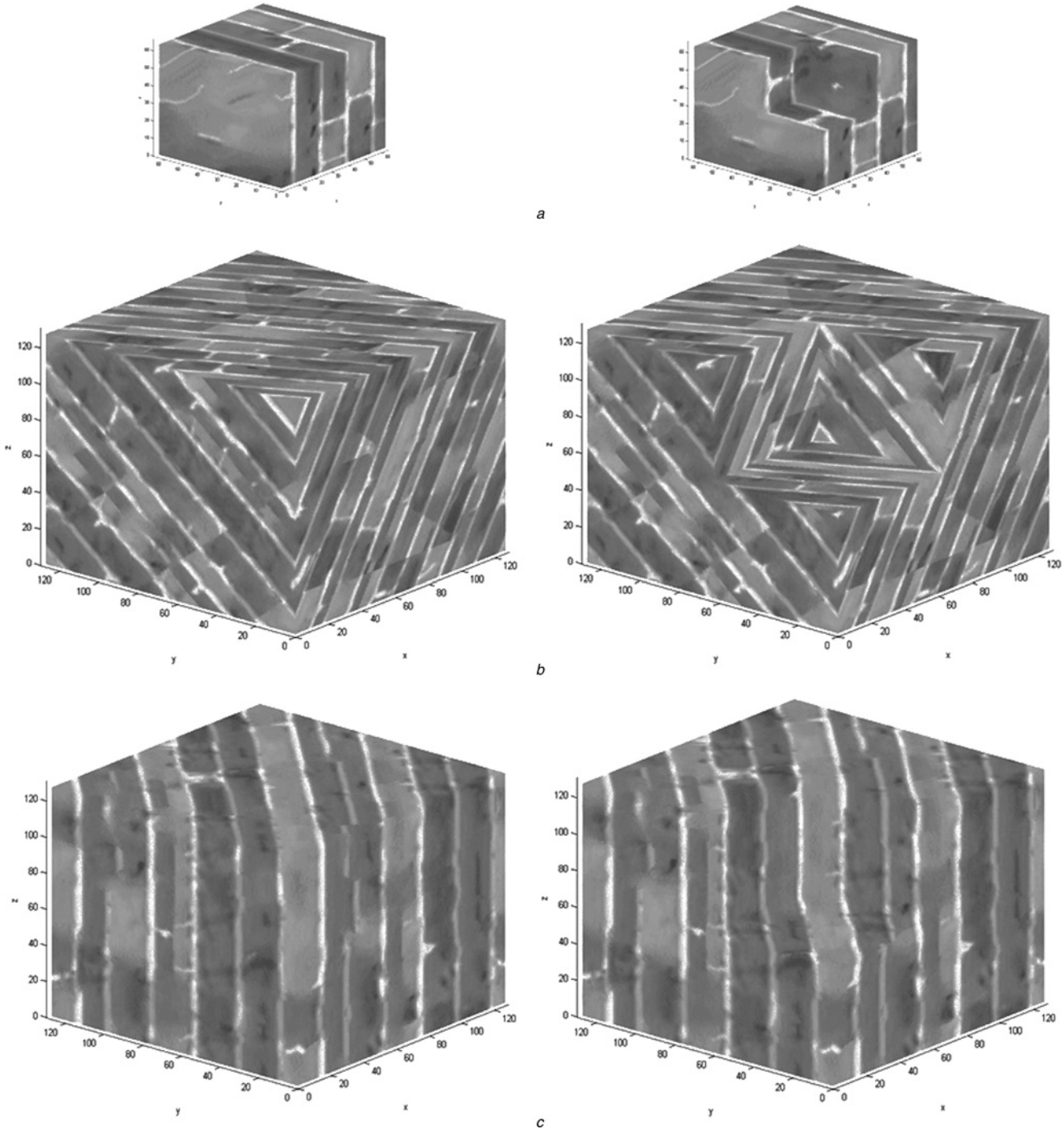

Fig. 8 *Showing the synthesised results*

a Input 3D exemplar and cross-section at $X = 32$, $Y = 32$ and $Z = 32$ for the input 3D exemplar
b Resulting 3D texture with 3D slant deformation and cross-section at $X = 64$, $Y = 64$ and $Z = 64$ for the resulting 3D texture with 3D slant deformation
c Resulting 3D texture with slant deformation on the XY-plane and cross-section at $X = 64$, $Y = 64$ and $Z = 64$ for the resulting 3D texture with slant deformation on the XY-plane

parameters are set to obtain the desired results by heuristics. The pre-process uses $5 \times 5 \times 5$ grids to generate feature vectors at each voxel, reduces 375D feature vectors to 8D feature vectors by PCA, sets $7 \times 7 \times 7$ grids for constructing similarity sets, and sets $k = 3$ for computing the most similar voxels for each voxel. Using lower parameter values decreases the quality of the synthesis results whereas using higher parameter values increases computation cost without improving synthesis results. The synthesis process sets the jitter step parameter to 0.7. Any other values of this parameter do not contribute the better results.

The proposed approach varies the three axes when defining a 3D vector field, and the resulting texture changes with the vector fields. For example, designing a circular field creates a texture with a circular pattern. Thus, users can easily deform the synthesised results. Moreover, the proposed approach can synthesize 3D textures with fine-scale deformations because the size of the user-defined 3D vector field is the same as that of the synthesised results and the 3D pyramid synthesis

doi: 10.1049/iet-ipr.2010.0428

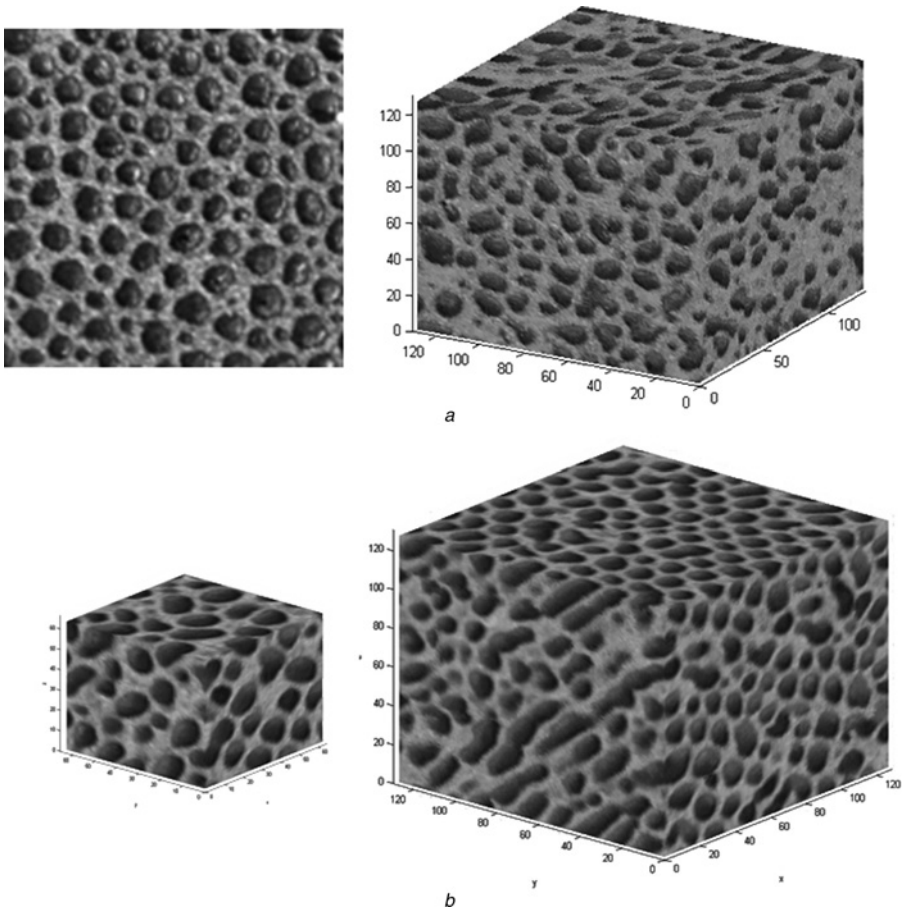

Fig. 9 *Shows the synthesised results*

a Resulting 3D texture synthesised by the conventional approach
b 3D texture synthesised by the proposed approach

scheme is based on neighbourhood matching. This improves the visual appearance of the synthesised 3D texture and is also applicable in 3D texture editing.

There are several reasons for undesired results using the proposed approach: (a) the input 3D exemplar is a particle texture with different sizes and colours of particles, most of which are of the same colour as the background; (b) the input 3D exemplar is a homogeneous texture with a colour similar to that in the overall exemplar; and (c) the input 3D exemplar is structural and contains larger patterns. In this case, the feature in the 3D exemplar is too large for synthesis.

6 Conclusions

The novel vector-field-based deformation approach for 3D texture synthesis proposed in this paper generates the desired deformations for 3D texture synthesis. The main contributions of this paper are as follows. (a) A method of 3D texture synthesis from an input 3D exemplar is proposed. The proposed method synthesising a 3D texture from an input 3D exemplar is based on neighbourhood matching, which enables fine-scale deformations. (b) A 3D vector field for 3D texture synthesis improves the appearance of the resulting 3D texture.

Further research is needed to develop 3D textures with flow fields [18] that produce different synthesised results over time. Further, since the proposed approach is time-consuming, further studies are needed to develop an acceleration algorithm.

7 Acknowledgments

The authors would like to acknowledge the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract no. NSC 98-2221-E-239-020-.

8 References

1 Peachy, D.R.: 'Solid texturing of complex surfaces', *ACM SIGGPRACH 1985*, 1985, **19**, (3), pp. 279–286
2 Perlin, K.: 'An image synthesizer', *ACM SIGGPRACH 1985*, 1985, **19**, (3), pp. 287–296
3 Ebert, D.S., Musgrave, F.K., Peachey, K.P., Perlin, K., Worley, S.: 'Texturing & modeling: a procedural approach' (Academic Press, 2002, 3rd edn.)
4 Chiou, J.W., 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 Dischler, J.M., Ghazanfarpour, D., Freydier, R.: 'Anisotropic solid texture synthesis using orthogonal 2D views', *Eurographics 1998*, 1998, **17**, (3), pp. 87–95
6 Dong, Y., Lefebvre, S., Tong, X., Drettakis, G.: 'Lazy solid texture synthesis'. Eurographics Symp. on Rendering 2008, 2008
7 Jagnow, D., Dorsey, J., Rushmeier, H.: 'Stereological techniques for solid textures', *ACM SIGGRAPH 2004*, 2004, **23**, (3), pp. 329–335
8 Kopf, J., Fu, C.W., Cohen-Or, D., Deussen, O., Lischinski, D., Wong, T.T.: 'Solid texture synthesis from 2D exemplars', *ACM SIGGRAPH 2007*, 2007, **26**, (3)
9 Qin, X., Yang, Y.H.: 'Aura 3D textures', *IEEE Trans. Vis. Comput. Graph.*, 2007, **13**, (2), pp. 379–389

10 Takayama, K., Okade, M., Ijirl, T., Igarashi, T.: 'Lapped solid textures: filling a model with anisotropic textures', *ACM Trans. Graph. (SIGGRAPH 2008)*, 2008, **27**, (3)
11 Heeger, D.J., Bergen, J.R.: 'Pyramid-based texture analysis synthesis', *ACM SIGGRAPH 1995*, 1995, **14**, (3), pp. 229–238
12 Lefebvre, S., Hoppe, H.: 'Parallel controllable texture synthesis', *ACM SIGGRAPH 2005*, 2005, **24**, (3), pp. 777–786
13 Ashikhmin, M.: 'Synthesizing natural textures'. ACM SIGGRAPH Symp. on Interactive 3D Graphics, 2001, pp. 217–226
14 Turk, G.: 'Texture synthesis on surfaces', *ACM SIGGRAPH 2001*, 2001, **20**, (3), pp. 347–354
15 Lefebvre, S., Hoppe, H.: 'Appearance-space texture synthesis', *ACM SIGGRAPH 2006*, 2006, **25**, (3)
16 Zhou, K., Huang, X., Wang, X., *et al.*: 'Mesh quilting for geometric texture synthesis', *ACM Trans. Graph. (SIGGRAPH 2006)*, 2006, **25**, (3), pp. 690–697
17 Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., Shum, H.Y.: 'Synthesis of bidirectional texture functions on arbitrary surfaces', *ACM SIGGRAPH 2002*, 2002, **21**, (3), pp. 665–672
18 Kwatra, V., Adalsteinsson, D., Kim, T., Kwatra, N., Carlson, M., Lin, M.: 'Texturing fluids', *IEEE Trans. Vis. Comput. Graph.*, 2007, **13**, (5), pp. 939–952

doi: 10.1049/iet-ipr.2010.0428