Texture Synthesis and Design based on Elements Distribution Creation

Yan Gui[1, 2], Yang Liu[1, 2], Feng Li[1, 2]
{guiyan122@163.com, ly_hn@foxmail.com, lifeng64@139.com }

Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation,
Changsha University of Science and Technology, Changsha, Hunan, P. R. China, 410114[1, 2]

Abstract. The texture synthesis and design starts with an initial elements arrangement and expands it outward by using local and global growth, to obtain the new larger distribution of texture elements. There are two types of synthesized distributions, and their diversity consists in changing the layout of texture elements and decreasing or increasing the number of texture elements according to user's creation. Furthermore, we apply a set of deformation operations to locally change the shapes of texture elements when placing the extracted texture elements into the synthesized distribution, in order to guarantee structure consistency in final synthesized textures. Experimental results show that our method creates a large variety of textures from a given texture sample.

Keywords: Deterministic texture; Texture elements; Element distributions; Texture design; Texture synthesis.

1 Introduction

There has been a plethora of research towards texture synthesis in computer graphics and computer vision, of which example-based texture synthesis methods [1] has become the main threads. To a wide variety of textures ranged from stochastic to structured, we focus on deterministic textures that are formed by spatial repetition of texture elements. It would be very difficult to synthesize the repetitive elements adequately with most existing texture synthesis methods, such as local neighborhood matching-based methods [2-7] and optimization-based methods [8].

Potential solutions [9-15] have been explored through imitating the element distributions in input exemplars. For example, Dischler et al. [9] generated 2D textures or textures on arbitrary surfaces by adding texture particles according to sets of co-occurrences. Barla et al. [10] proposed a method to synthesize 2D arrangements of seed points, and pasted input elements to those locations by local neighborhood matching. Ijiri et al. [11] synthesized 2D distributions by locally growing through rule-based heuristics. However, the above methods [9-11] cannot handle elements with complex shapes which are closely correlated with spatial distributions. Hurtut et al. [12] proposed a statistical model to learn spatial interactions between and within different categories. Passos et al. [13] presented an improved method for arrangement synthesis defined as 2D collection of elements, which provides control over local density of elements. Gui et al. [14] proposed a similar method for periodic pattern of texture analysis and synthesis based on texels distribution. Recently, Huang et al. [15] extend it to texture synthesis on arbitrary surfaces. However, these existing techniques mainly focus on

texture reproduction, which maintains a visual similar to the original sample. It is observed that user manipulations are rarely provided over the fully automatic synthesis process, including the control of the positions and shapes of texture elements, and consistent transition among different texture elements, which can help synthesis a variety of textures.

In this paper, we first extract distributions of texture elements from a large number of deterministic texture samples by constructing their connectivity. And then these distributions can be divided into near-regular or non-regular categories through quantifying the constructed connectivity. We thus can expand an initial elements arrangement in two different ways, by performing local or global growth of texture elements, in order to generate a new larger distribution of texture elements. By default, each texture element in distributions is represented by a discrete point, and the user may also optionally draw a shape contour for it. Once the distribution of texture elements is obtained, we arbitrarily use texture elements to replace each discrete point to synthesize the final textures, which the texture elements are extracted from the input deterministic textures. When pasting texture elements together, we apply a set of deformation operations, such as scaling, rotation, and thin-plate splines (TPS), to change the shapes of texture elements. Such deformation operations used in this paper are helpful to avoid large overlapping and holes between texture elements, in order to ensure structure consistency. As shown further, our method can create a wide variety of textures as we attempt to grant users more and more control to the positions and shapes of texture elements.

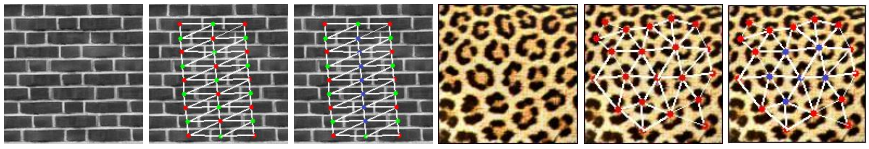
2 Element Distributions Creation

2.1 Connectivity Construction and Analysis

Given a deterministic texture sample, the main task is to model the spatial neighborhood relationships among all texture elements. Each texture element in the texture sample can be represented by its bounding boxes, and the centers of these bounding boxes are used to define the element positions in the texture sample. According to the discrete positions information, the most suitable method used in [11, 14] is to extract the Delaunay triangulation in order to get a connectivity among all texture elements. The connectivity also can be called the distribution of texture elements. In addition, if there are texture elements with different classes in the texture sample, as shown in Figs. 1 (a2) and (b2), they are marked by using different colored points.

To the constructed connectivity, our focus is on exploring whether they are near-regular or non-regular, which are of primary importance to characterize the spatial arrangements. For this purpose, we first define a neighborhood that is composed of all neighboring texture elements of each texture element. In each neighborhood, edges are connected between two adjacent texture elements. The neighborhood can form a ‘ring shape’ when the size of the neighborhood is equal to the number of edges. As Figs. 1 (a3) and (b3) show, the texture elements which having neighborhood with ring shape are marked by blue. Based on these neighborhoods, if they have the same sizes, and their ring shapes have more close appearance similarity that can be measured through the area of the neighborhood, the constructed connectivity is near-regular or regular (Fig. 1 (a2)), else is supposed to be non-regular (Fig. 1 (b2)). Indeed, the key for analyzing the constructed connectivity is to describe the discrete or compact structure information among texture elements, as texture elements in texture samples

are independent each other (Fig. 1 (b1)), or define a partitioning of textures, with each texture element having a non-overlapping, but adjoining spatial extent (Fig. 1 (a1)).



(a1) (a2) (a3) (b1) (b2) (b3)

Fig. 1. Near-regular and non-regular connectivity.

2.2 Local and Global Growth

By considering near-regular or non-regular connectivity, texture synthesis begins with an initial elements arrangement, and expands it outward by placing the new positions, in order to reproduce a new larger elements distribution. Given a regular elements arrangement (Fig. 2 (a)), we need to decompose it into two types of sub-models, including horizontal model (Fig. 2 (a1)) and vertical model (Fig. 2 (a2)), which can be used as the placement rules to guide the extension. For instance, to perform extension by using horizontal model, the position of a new texture element (P_x, P_y) is computed as $P_x = Nrx + ov + xl$ and $P_y = Nry - yr + 1$, where (Nry, Nrx) is the coordinate of and lower right corner of the bounding box; xl and yr represent the shortest distance from the center to the bounding box (Figs. 2 (a1) and (a2)); ov is a user specified spacing between adjacent texels to avoid overlapping. Similarly, the position of a texture element in vertical direction can be computed based on vertical model. As shown in Fig. 2 (b), we reproduce the final element distribution through using the horizontal model and the vertical model alternatively. By default, each texture element in the synthesized element distributions is represented by a discrete point, and the user may also optionally draw a shape contour for texture elements. In addition, the class information for each new placed position can be recorded during the extension.

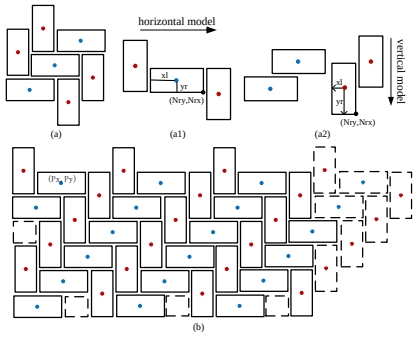


Fig. 2. Regular elements distribution extension.

To a non-regular elements arrangement (Fig. 3 (a)), we first need to provide a set of discrete points and then use Lloyd's method [10, 11] to obtain a random distribution of

positions (Fig. 3 (b)). Since the obtained distribution is close to uniformity because each point is located in the center of the Voronoi region, so we further perform an adjustment. Given a neighborhood $\omega(e_{ref})$ taken from the initial elements arrangement and a neighborhood $\omega(e_{tar})$ obtained from the synthesized random distribution, where e_{ref} and e_{tar} are the current selected position of the texels (as shown in Figs. 3 (a) and (b)). The adjustment process falls into three stages.

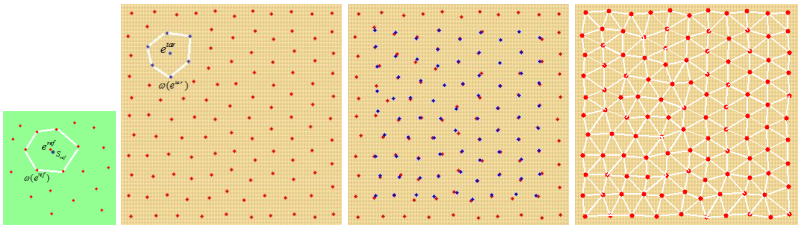
Step 1: we find a matched e_{ref} for the current selected e_{tar}. It is the one whose neighborhoods $w(e_{ref})$ has the most similar neighborhood condition to that of the selected position e_{tar}. We first sort position point $e_i^{ref} \in \omega(e^{ref})$ and $e_j^{tar} \in \omega(e^{tar})$ in counter-clockwise order simply, and then the differences can be measured by using following err function:

$$Err(\omega(e^{ref}), \omega(e^{tar})) = \sum_{i,j} w_1 \theta(e_i^{ref}, e_j^{tar}) + w_2 L(e_i^{ref}, e_j^{tar}), \quad i, j \in \{1, ..., N\} \tag{1}$$

where, N is the number of texture elements in neighborhoods; $\theta(e_i^{ref}, e_j^{tar}) = \left|\theta_i^{ref} - \theta_j^{tar}\right|$ measures differences in angles. θ_i^{ref} is angle between x-axis and edges (e_i^{ref} , e^{ref}) ; similarly to define θ_j^{tar}. $L(e_i^{ref}, e_j^{tar}) = \left|L_i^{ref} - L_j^{tar}\right|$ measures differences in length of edges. L_i^{ref} and L_j^{tar} are the length of edges (e^{ref}, e_i^{ref}) and (e^{tar}, e_j^{tar}) respectively. w_1 and w_2 are weights to balance the differences in angles and edge lengths. We can find the best matched e_{ref} when minimizing the error function (Eq. 1).

Step 2: we compute the reference shift vector S_{ref} of the corresponding e_{ref}, which is the distance from the position e_{ref} to the barycenter of the neighborhoods $w(e_{ref})$.

Step 3: we translate the selected position e_{tar} by $S_{tar} = (S_{ref} A_{tar}) / (n A_{ref})$, where A_{tar} and A_{ref} are the area of the neighborhoods $w(e_{tar})$ and $w(e_{ref})$ respectively, n is the size of the neighborhoods $w(e_{tar})$. The translated positions are marked by using blue (Fig. 3 (c)).



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

Fig. 3. Non-regular elements distribution adjustment.

When all available positions in D_{tar} have been translated through repeating Step 1 to Step 3, we can obtain the final elements distribution (Fig. 3 (d)), having the stochastic property. On the other hand, we can control the density of texture elements in the synthesized elements distribution through increasing and decreasing the number of the initial discrete points set. Note that the class information in such elements distribution can be ignored.

3 Texture Elements Placement

3.1 Texture Elements Extraction

Existing state-of-the art image cutout techniques [16-18] can be used to extract individual texture elements. However, none of these segmentation methods can work well for arbitrary texture samples. In this setting, input texture sample is represented by a graph $G =< P, E >$, with each pixel as one node $p \in P$ and pairwise adjacent pixels as edge $< p, q > \in E$. Based on appearance similarity between texture elements, their extraction is to optimize the following energy function:

$$E(f) = \sum_{p \in P} D_p(f_p) + \sum_{p<q \in N} V_{p,q}(f_p, f_q) + \sum_{i,j \in H} U'_{i,j}(f_i, f_j) \quad (2)$$

where the data item $D_p(f_p)$ measures the conformity of node p assigned label f_p, the smooth item $V_{p,q}(f_p, f_q)$ is charged for adjacent nodes with different labels, and the repetition energy item $U'_{i,j}(f_i, f_j)$ measures the labeling smoothness on the similar nodes; N is an neighborhood system, joining adjacent nodes in 4-neighbor (or 8-neighbor); H is an extended neighborhood systems based on the repetitive similarity. Unlike the optimization model in RepSnapping, the repetition energy item $U'_{i,j}(f_i, f_j)$ can be modified based on a robust appearance similarity δ among the repeated texels, which is defined as the following function:

$$U'_{i,j}(f_i, f_j) = \mu \mid f_i - f_j \mid \cdot \exp(-\beta \cdot \delta^2(i, j)), \quad < i, j > \in H \quad (3)$$

where, μ is a trade-off weight, and β is a constant. Inspired by the texture samples with their own pattern feature, the appearance similarity measurement $\delta(i, j)$ in Eq. (3) is developed by considering both colors and pattern features:

$$\delta(i, j) = \left\| c_i - c_j \right\| + \left\| T_i - T_j \right\| \quad (4)$$

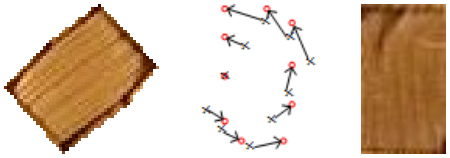
where, $\| c_i - c_j \|$ is used to measure the difference in color between nodes i and j; c_i and c_j are the average color of nodes i and j respectively ; $\| T_i - T_j \|$ describes the difference in pattern feature between nodes i and j; T_i and T_j are the average feature vector of nodes i and j respectively, which is computed by using Gabor wavelet transform [19] in multiple scales and orientations. And the values of these differences are normalized in Eq. (3). The above energy function (Eq. 2) can be minimized by using max flow-min cut algorithm. As illustrated in Fig. 4, only with less user interactions (Figs. 4 (a1) and (b1)), all texture elements in the texture samples can be extracted simultaneously (Figs. 4 (a2) and (b2)).

(a1) (a2) (b1) (b2)

Fig. 4. Texture elements extraction.

3.2 Texture Elements Deformation

Since the extracted texture elements keep their original shapes, large overlapping and holes between adjacent texture elements can be found when performing texture elements placement, which no guarantee that texture structures is continuous in final textures. In order to alleviate such effects, deformation operations are applied to change the shapes of texture elements. Generally, we observed that it is easy to control texture elements with regular shape, which mainly consists in changing the size and orientation of the texture element by using linear transformation operations, such as scaling and rotation, to keep the structure consistency. Indeed, our motivation is to transform the irregular shapes of texture element. Given a reference shape and a texture element to be deformation, we first need to sample contour points sets respectively (Fig. 5 (b)), and then we determine the corresponding pairs of contour points between them by using shape matching [20]. As figure 5 shows, when the corresponding pairs of contour points are constructed, the texture element (Fig. 5 (a)) can be deformed to a rectangular region (Fig. 5 (c)). The TPS mapping used in our method not only maintains the shape contour, but also creates smooth and consistent warped content in the interior region of texture elements. Note that we use graph-cuts based segmentation method [5] to merge the conflicting regions and we use example-based completion algorithm [21] to fill holes explicitly.



(a) (b) (c)

Fig. 5. Thin plate spline mapping.

4 Experimental Results and Discussions

Figure 6 demonstrates the capability of our method for creating a large variety of textures from a small sample (Figs. 6 (a1)-(d1)). If the initial elements arrangement conforms to element distribution of texture samples, the resulting textures have an appearance similar to

the texture samples (Figs. 6 (a2) and (b2)). Otherwise, the designed textures show various appearances (Figs. 6 (a3)-(a6) and Figs. 6 (b3)-(b6)). For last two examples, our presented method changes the elements density interactively, and the elements density increase gradually. In addition, we can interactively replace the initial elements arrangement (Fig. 6 (c6)) and manipulate the scale or orientation of texture elements in order to generate various outputs (Figs. 6 (d5) and (d6)).

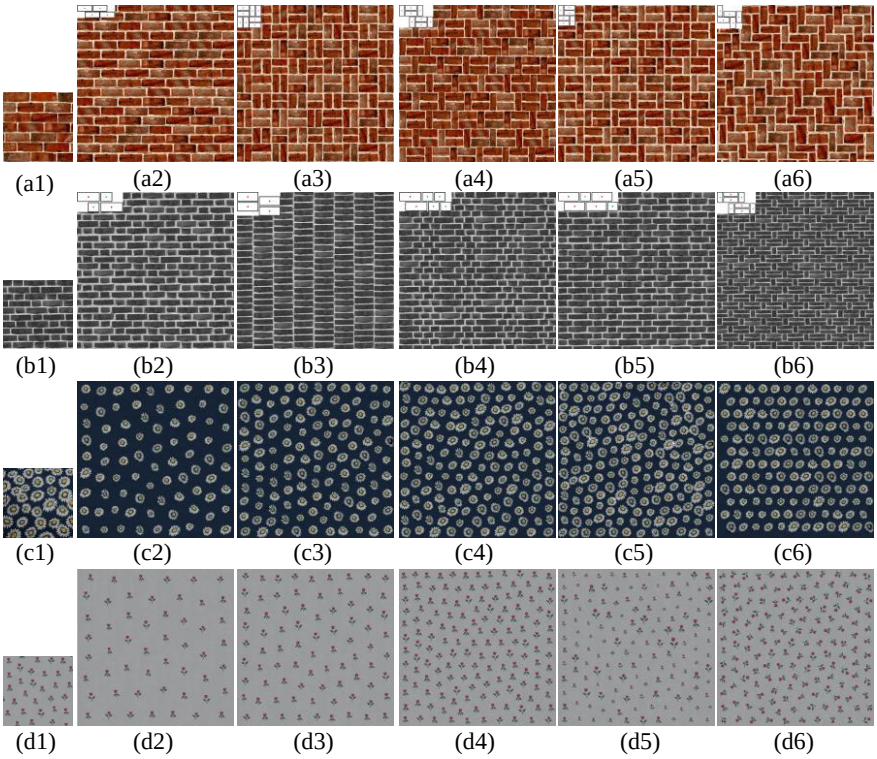


Fig. 6. More results of our texture design and synthesis.

In Fig. 7, we compare our approach with existing techniques of texels distribution based [14], appearance-space synthesis [7], near regular texture synthesis [6], Graph-cut [5], image quilting [4], patch-based [3] and jump-map based [2]. The texture sample in Fig. 7 (a) consists of oblique blocks, which has a near regular structural layout but irregular color appearance in individual blocks. So far, we have not yet seen an existing texture synthesis algorithm that preserves structural regularity as well as structure continuity in the synthesized texture. The results synthesized by these existing techniques are illustrated in Fig. 7 (c)-(d). We can find that the quality of the texture produced with patch-based methods [3-7] is superior to that of pixel-based method [2]. However, structure misalignments still remain because there are no exact copies at the overlapping region of adjacent patches. Our interactive controllable technique consists in deforming each single texture element by using TPS firstly. As demonstrated in Figs. 7 (b1) and (b2), we can see that our method performs better than these

existing techniques. The structural regularity and structure continuity are preserved in our designed results.

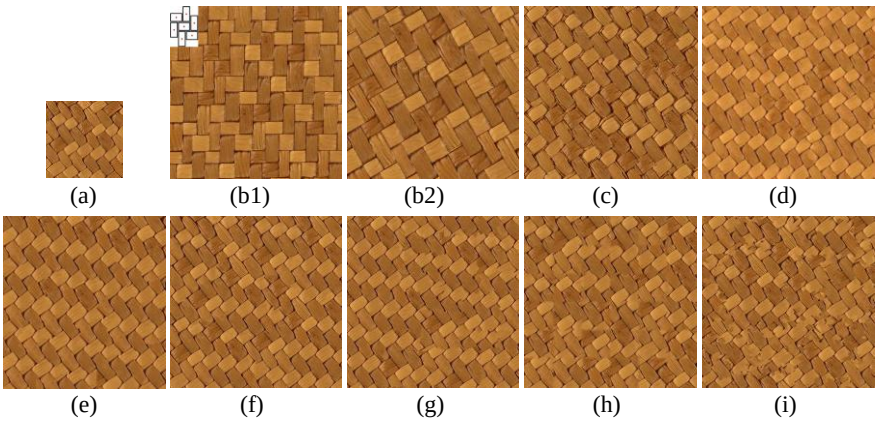


Fig. 7. Comparison with existing texture synthesis techniques [2-7, 14].

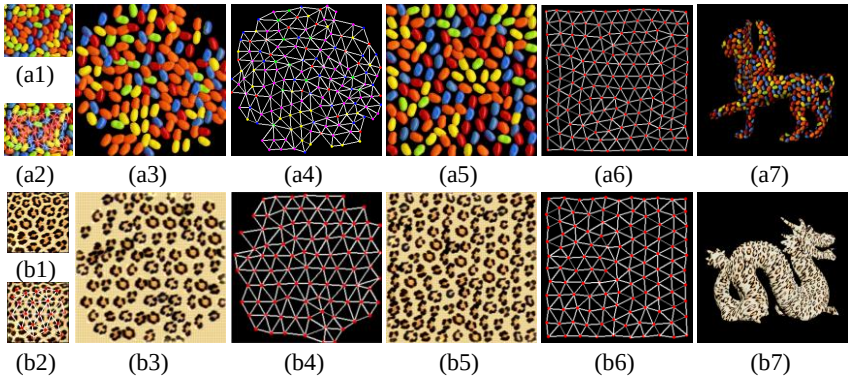


Fig. 8. A comparison with element distributions based texture synthesis [14].

Figure 8 shows two examples in which the element distributions based texture synthesis [14] fail to obtain an optimized stochastic distribution but which is handled well by our proposed method. In [14], neighborhood comparison they applied is the difficulty of expanding the distribution of positions infinitely because the error accumulates in each local growth step. As shown in Fig. 8 (a4), the synthesized distribution deviates greatly from the original distribution in samples, which cause large overlapping between adjacent texture elements (Fig. 8 (a3)). Although the synthesized distribution is desired shown in Fig. 8 (b4), the repetition in the resulting large textures (Fig. 8 (b3)) is not satisfying because the randomness. The advantage of our approach is that we obtain the new element distribution by controlling the total number of texture elements interactively (Figs. 8 (a6) and (b6)), so that our method produce high quality synthesis results (Figs. 8 (a5) and (b5)). In addition, we paste the synthesized textures onto arbitrary surfaces to add non-geometric details (Figs. 8 (a7) and (b7)).

5 Conclusions

The main contribution of this paper is to introduce an interactive and controllable scheme for semantic texture elements recombination whose success relies on a key factor: a flexible texture synthesis and design procedure. More specifically, more informative element distributions are produced according to user's need and creation. By using all segmented texture elements and various element distributions, our technique can create a wide variety of textures, while the existing texture synthesis methods does not have such ability.

Acknowledgments. The project was supported by the National Science Foundation of China (No. 61402053) and by the Scientific Research Fund of Hunan Provincial Education Department (No. 16C0046).

References

[1] L. Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, State of the Art in Example-based Texture Synthesis, Eurographics 2009, State of the Art Report, EG-STAR.

[2] Zelinka S. and Garland M.: Towards Real-time Texture Synthesis with the Jump Map. EGRW '02 Proceedings of the 13th Eurographics workshop on Rendering, pp. 99--104, (2002).

[3] Liang L., Liu C., Xu Y., Guo B., and Shum H. Y.: Real-time texture synthesis by patch-based sampling. ACM Transaction on Graphics, 20, (3), pp. 127--150 (2001).

[4] Efros A. and Freeman W.: Image quilting for texture synthesis and transfer. In SIGGRAPH 2001, Computer Graphics Proceedings, pp. 341--346 (2001).

[5] V. Kwatra, A. Schiodl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: image and video synthesis using graph cuts, ACM Transactions on Graphics, 22(3), pp.277--286 (2003).

[6] Liu Y. X., Lin W. C., and Hays J.: Near-regular texture analysis and manipulation. In SIGGRAPH '04: ACM SIGGRAPH, pp. 368--376 (2004).

[7] Lefebvre S. and Hoppe H.: Appearance-space texture synthesis. In: SIGGRAPH '06: ACM SIGGRAPH, pp. 541--548 (2006).

[8] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example based synthesis, ACM Transactions on Graphics, 24(3), pp.795--802 (2005).

[9] J. M. Dischler, K. Maritaud, B. Levy, D. Ghazanfarpour. Texture particles, Computer Graphics Forum, 21, pp.401--410 (2002).

[10] P. BARLA, S. BRESLAV, J. THOLLOT, F. SILLION, L. MARKOSIAN. Stroke pattern analysis and synthesis, Computer Graphics Forum, 25, pp.663--671 (2006).

[11] T. IJIRI, R. MECH, T. IGARASHI, G. S. P. MILLER. An example-based procedural system for element arrangement, Computer Graphics Forum, 27(2), pp.429--436 (2008).

[12] T. HURTUT, P. E. LANDES, J. THOLLOT, Y. GOUSSEAU, R. DROUILLHET, J. F. COEURJOLLY, Appearance-guided synthesis of element arrangements by example, In Proceedings of the Symposium on Non-Photorealistic Animation and Rendering, pp.51--60 (2009).

[13] V. ALVES DOS PASSOS, M. WALTER, M. SOUSA. Sample-based synthesis of illustrative patterns, In Computer Graphics and Applications, PG 10, pp.109--116 (2010).

[14] Y. Gui, L. Z. Ma. Periodic Pattern of Texture Analysis and Synthesis based on Texels Distribution, The Visual Computer, 26(6-8), pp.951--964 (2010).

[15] J. Huang, L. Zhang, Y. Gui. Surfaces Texture Synthesis Based on Texel Distribution, Journal of Chinese Computer Systems, 2015 (Chinese).

[16] Y. Li, J. Sun, C. K. Tang, and H. Y. Shum. Lazy snapping , ACM Transactions on Graphics, 23(3), pp.303--308 (2004).

[17] C. Rother, V. Kolmogorov, and A. Blake. Grabcut, interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics, 23(3):309--314 (2004).

[18] H. Huang, L. Zhang, H. C. Zhang, RepSnapping: efficient image cutout for repeated scene elements, Computer Graphics Forum, 30(7), pp.2059--2066 (2011).

[19] B. S. MANJUNATH, AND W. Y. MA, Texture features for browsing and retrieval of image data [J], IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(8), pp.837--842 (1996).

[20] Belongie S., Malik J., and Puzicha J.: Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(4), pp. 509--522 (2002).

[21] A. Criminisi, P. Prez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting, IEEE Transactions on Image Processing, 13(1), pp.1200--1212 (2004).