Skeletal Texture Synthesis

Pan Gang, Zhang Jiawan, Zhou Xiaozhou, Sun Jizhou
Tianjin University, Tianjin University, University of Alberta, Tianjin University
{pangang@tju.edu.cn, jwzhang@tju.edu.cn, xzhou3@cs.ualberta.ca, jzsun@tju.edu.cn}

Abstract

In this paper, we present an improved synthesis algorithm for both 2D and solid textures. Based on the wavelet-based multi-resolution pyramids and the improved optimization of texture synthesis, a coarse texture was synthesized as a skeleton. By adding detail information to the coarse texture, the higher resolution results can be generated. This skeletal growth-like procedure makes the result more controllable. In addition, to make full use of the spatial relationships among texture pixels, wavelet coefficients rather than traditional RGB channels are used to search the nearest neighbor pixels. Experimental study shows that the method plays well for both 2D and solid texture synthesis.

Keywords--- texture synthesis; solid texture; wavelet; skeletal

1. Introduction

Texture synthesis has been widely recognized as an important research topic in computer graphics, vision, and image processing. There has been a lot of research which was tried to generate textures either to validate texture models or simply to use the result in an application.

Compared with other research, exemplar-based methods, which synthesize a large texture from a small exemplar, is an excellent solution. During the last decade many example-based texture synthesis methods have been proposed. Texture synthesis techniques could be broadly categorized into parametric [6][13], and nonparametric methods. The nonparametric methods could further be classified into pixel-based methods [2][4][15], patch-based methods [3][9], optimization-based methods[8], and appearance-space texture synthesis [11].

In this paper, our goal is to present an optimization texture synthesis method based on wavelet transform, which is accumulation error free. Furthermore, we focus on an improved framework for both 2D and solid texture. Our contributions include two main components: the first component is a framework that effectively synthesizes texture using wavelet technology; the other one consists of an improved optimization texture synthesis algorithm. The key advantages of the improved optimization texture synthesis algorithm are quality and speed. For better wavelet coefficients metric, the quality of the synthesized textures is equal to or better than those generated by previous techniques; meanwhile, the computation speed is much faster than the prior ones because of the reducible iterative synthesis times.

2. Related works

Texture synthesis seeks to generate textures of arbitrary size from a given smaller texture sample. Example-based optimization approach is one of the most important methods to synthesize texture in the recent decades. Most notable Heeger and Bergen[6] iteratively resampled random noise to coerce it into having particular multiresolution oriented energy histograms. De Bonet [2] measured the joint occurrence of texture discrimination features at multiple resolutions and conditions of the similar joint occurrence of features at lower spatial frequencies, but it can produce boundary artifacts if the input texture is not tileable. Wei and Levoy [15] adopted the algorithm to include multi-resolution synthesis. Using Gaussian pyramids they decomposed the texture image into different resolutions and seeked to transform a random noise sample to resemble the sample texture at each resolution using an Efros style neighborhood searching approach. This method works well on stochastic (random) textures but is not suitable for structured textures. Johannes Kopf [7] used 2D texture optimization techniques together with histogram-matching to synthesize solids. It is applicable to a wider variety of textures, but the running time is its main problem.

The speed issue has been addressed by parallel GPU texture synthesis [10][11] which runs much faster than CPU-based algorithms. A further advantage of GPU synthesis is the reduced storage; this is very important for real-time applications as a commodity GPU often has limited texture memory.

Although wavelet transform was first introduced in computer graphics, it has become an important tool in both graphics and image processing applications. In

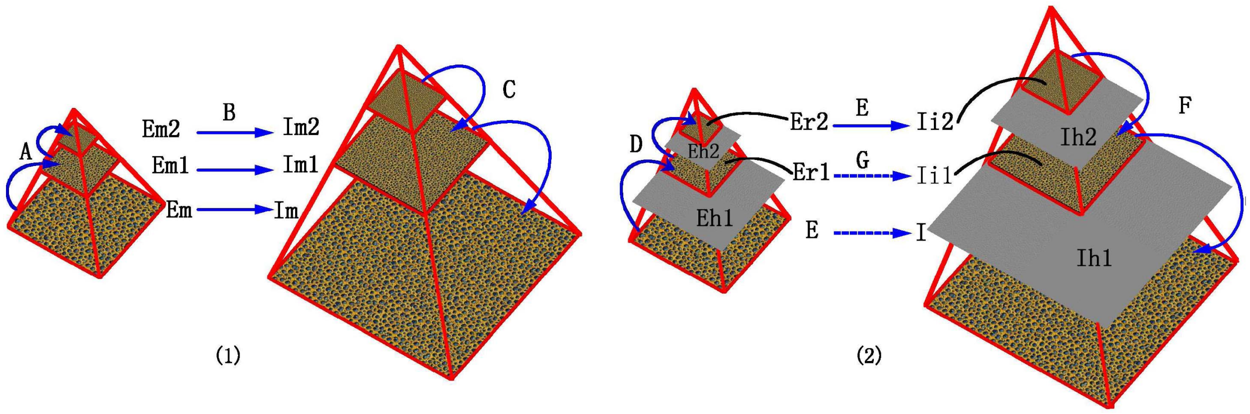

Figure 1: Traditional multi-resolution pyramids method (left) and skeletal texture synthesis method (right). In traditional method, process (A) getting input pyramids with down sampling method, process (B) synthesizing texture, which must be done at every resolution pyramid, process(C) getting feature information from lower-resolution texture. In texture skeletal method, process (D) getting input pyramids with multi-level wavelet transform, process (E) synthesizing texture, which is optional, process(F) reconstructing output pyramids by adding detail information to texture skeleton. It deserves to be specially noted that the high frequency coefficient could be negative value, for better display, this will be shown in gray scale images.

particular, some authors have used wavelet transform for texture synthesis and analysis. Portilla [13] proposed a statistical model for texture representation using a complete complex wavelet transform. Joseph [1] synthesized textures by constructing multi-scale wavelet transform tree. But these methods are all based on statistical simulation of wavelet coefficients and have limitations for synthesizing structural textures. Gallagher and Kokaram [5] introduced wavelet-based multi-resolution pyramids. Given an initial sample texture, the algorithm generates new texture using a nonparametric technique that incorporates the Dual Tree Complex Wavelet Transform (DT-CWT). But this strategy only works on pixel-based texture synthesis method. Tonietto [14] used wavelet coefficients as metric to select nearest patches, which improved the synthesis quality to a certain extent. But this improved patch-based method also leads to errors accumulating.

3. Skeletal Framework

This section describes the framework of skeletal texture synthesis algorithm. Fig.1 left show the traditional multi-resolution method. The traditional method first synthesizes the texture at a coarse resolution, and then up-sample it to a higher resolution via interpolation. This serves as the initialization of the texture at the higher resolution. Also, within each resolution level, the method must run the synthesis algorithm repetitively.

Compared with the traditional one, the right part in Fig.1 indicates the skeletal texture synthesis framework. More precisely a wavelet decomposition of the initial image is computed. Optimization synthesis starts from a coarse level. Each pixel at this coarse level corresponds to a small patch at the finer level. To synthesize a finer result, details are added to the coarse one. This process is visually similar to skeletal growing from small to large. We summarize the algorithms in the following pseudocode.

Algorithm 1: SKELETALTEXTURESYNTHESIS(E)

```
$G_{in}(0) \leftarrow E;$
$i \leftarrow 1;$
while $(i <= MaxLevel)$
    $G_{in}(i), C_{in}(i) \leftarrow WaveletTrans(G_{in}(i-1));$
    $i++;$
end while
$G_{out} \leftarrow InitRandom(G_{in}, MaxLevel);$
$j \leftarrow 0$
while $(!G_{in}^{j} == G_{in}^{j+1})$
    $G_{out}^{j+1} \leftarrow argminE_{Global}(G_{out}, G_{in}^{j});$
    $G_{in}^{j+1} \leftarrow NearestNeighborSearch(G_{out}^{j+1}, G_{in});$
    $j++;$
$G_{out}(MaxLevel) \leftarrow G_{out}^{j};$
$k \leftarrow MaxLevel$
while $(k > 1)$
    $C_{out}(k) \leftarrow Correspond(G_{out}(k), C_{in}(k));$
    $G_{out}(k-1) \leftarrow IWaveletTrans(G_{out}(k), C_{out}(k));$
    $k--;$
end while
return $G_{out};$
```

The proposed method uses wavelet transform to deal with the input exemplar E. In the decomposition scheme, we get $G_{in}(MaxLevel)$ as the real input texture and C_{in} as detail information. With texture skeleton E_r, a low-resolution intermediate texture $G_{out}(MaxLevel)$

was synthesized. $G_{out}(MaxLevel)$ provides smooth information and the skeleton for the objects in the image. Using $G_{out}(MaxLevel)$ and C_{in}, we can reconstruct G_{out}.

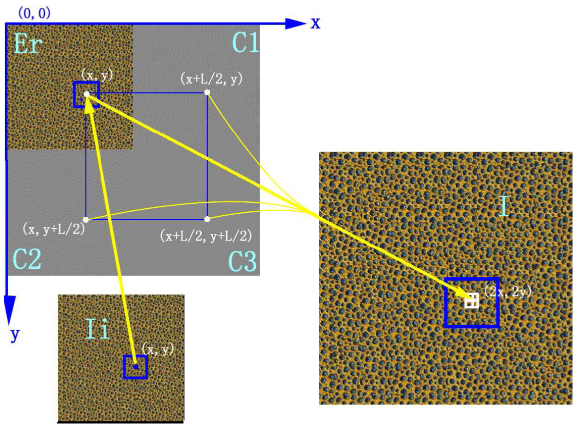

Figure 2: 2D texture recostruction process.

Fig.2 shows the reconstruction process. For each pixel p in $G_{out}(MaxLevel)$, we record the position where it comes from in E_r and its corresponding detail information C_1, C_2 and C_3. C_1, C_2 and C_3 are all wavelet coefficients. With the detail information, we get the final high-resolution texture I by inverse Haar wavelet transform. Every pixel in the $G_{out}(i+1)$ evolves to four pixels in $G_{out}(i)$.

Similar to 2D texture reconstruction, the high-resolution target solid S can be synthesized with detail information H and low-resolution solid $S_{out}(MaxLevel)$. We hold H in three directions for voxels of $S_{out}(MaxLevel)$. But the high-level solid voxels and the detail information are not one-to-one. In other words, some voxels in S could not be generated using the detail information directly. For those "missing" voxels, we adopt a strategy as follows: assuming that we want to synthesize a $128\times128\times128$ solid, for every one of the three directions, using the proposed 2D texture reconstruction, we can get 64 slices whose dimensions are 128×128.We mix the slices of the three directions to get the target. For every direction, each slice toward this direction is used twice in succession, since with inverse Haar wavelet transform, every voxel should map to two for each direction. Then voxel is synthesized by mixing the values in the three directions.

4. Improved Optimization Texture Synthesis

Optimization-based texture synthesis could be summarize as follows. First, initialize the pixel values of output image I randomly chosen from exemplar E. Then, the goal is to iteratively increase the similarity between E and I. Every iterative step tries to minimize the energy function[12] which measures the difference between E and I.

$$E_{global}(I,(e)) = \sum_t \min Dist(I_t, \{e\}) \tag{1}$$

E_{global} is the total energy of I compared with E, $D(I_t, \{e\})$ is the minimum distance between certain neighborhood and all possible neighborhoods from E.

There are two main phases in the iterative process: optimization and search. In the optimization phase, the texel value of I is changed based on the search results. For each pixel p, we calculate its values by weighted average of all the neighborhoods including pixel p. In the search phase, the goal is to optimize Eq.(1) by finding the best matching in E for every neighborhood I_t. The new neighbor index is held in an array to be used for the optimization phase. This is a standard nearest neighbor search in a high-dimensional space, and it determines the running time of the method.

The critical part of the texture synthesis algorithm is nearest patch searching. The former techniques usually apply RGB, specular, shininess or displacement channels to compare patches and build a list of candidate patches which satisfy an error criterion. It searches the input image for all possible patches, and picks the patch with smallest distance as the nearest one. The distance is computed for the patches as follows:

$$Dist(I_t, e_i) = \left[\frac{1}{A}\sum_{n=1}^{A}\left(p_{I_t}^n - p_{e_i}^n\right)^2\right]^{1/2} \tag{2}$$

Where A is the number of pixels in the patch, and $p_{I_t}^n$ represents the value of the n-th pixel in the neighborhood I_t. The pixel's values can be either grayscale or RGB triplets.

In our method, the criterion to compute the neighborhoods' distance was replaced by wavelet-based one, since the wavelet transform has nice localization properties in both the spatial and the frequency domains. To compute the distance between two neighborhoods, wavelet coefficients are selected as target channels.

$$Dist(I_t, e_i) = \sum_{\psi=R,G,B}\left[\sum_{n=1}^{A}\left(C_{I_t}^n - C_{e_i}^n\right)_{\psi}^2\right] \tag{3}$$

Where A is the number of pixels in a patch, and $c^n_{I_t}$ represent the values of the n-th wavelet coefficient in the neighborhood I_t.

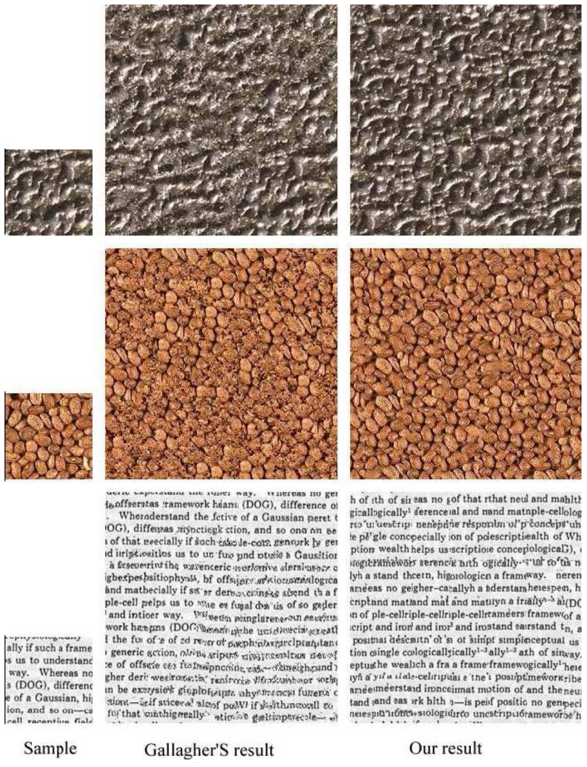

Figure 3: Comparisons between our method and Gallagher's.

Through wavelet transform the high-frequency coefficients C_h (detail coefficients) can represent spatially local phenomena such as edges. While we all know that the human visual system is very sensitive to edges, corners etc, this makes this criterion more suitable for textures containing regular noticeable features. Our results also showed that this criterion can get rather better neighborhood matching.

5. Results

We implement the proposed approach by using C# language on a PC with a 2.8GHz CPU and 1GB RAM. Fig.3 shows comparison between our method and Gallagher's. Fig.6 shows some representative results. Some solids synthesized by our method are effortlessly mapped on a variety of 3D objects with non-trivial geometry and topology.

The results show that our method can apply to most kinds of exemplars. For the low-resolution synthesis process, there are two ways that could be used to speedup the searching process. First, the low resolution exemplar E_r replace the original one E, that is, the smaller exemplar, the smaller searching space. Second, the feature of texture in E_r is similar to, but smaller than that in E, smaller neighborhoods should be used in the searching process. Thus, less number of pixels will be calculated. We benefit from it not only speeding up the synthesis process, but also making our method more fit for texture containing large scale structures.

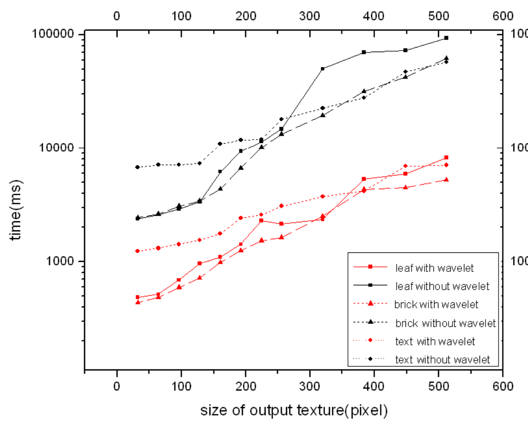

Figure 4: Time comparison of our method with 1 level wavelet transform and Kwatra's method for 2D texture synthesis. The exemplars are 128 × 128, and the neighborhood is 13 × 13.

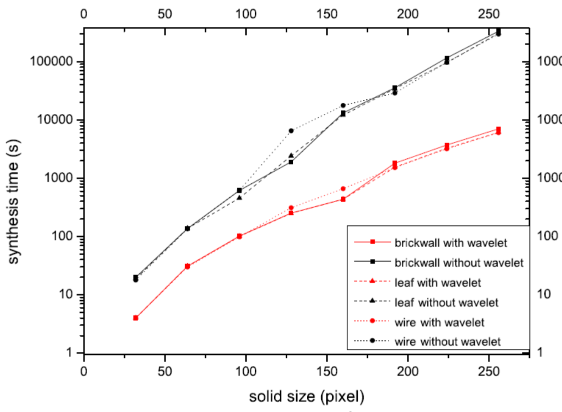

Figure 5: Time comparison of our method with 1 level wavelet transform and Kopf's method for solid texture synthesis. The exemplars are 128×128, and the neighborhood is 13 × 13.

In our experiments we use one level wavelet transform. The synthesis time depends on size of input texture, the output and the size of the neighborhood. Fig.4 and Fig.5 show the time comparison between our method with one level wavelet transform and the former ones without wavelet. From these time curves, our method reduces the time cost greatly, especially when the output is large. To get a solid large enough to contain the object, Kopf's method [7] has to synthesize a set of Wang cubes to get the result solid. This strategy usually brings repeated texture in the surface of the object. For the time cost reduction, our method can get solid large enough to contain the object in acceptable time.

6. Conclusions and future work

This section describes the framework of skeletal texture synthesis algorithm. Fig.1 left show the traditional multiresolution method. The traditional method first synthesizes the texture at a coarse resolution, and then up-sample it to a higher resolution via interpolation. This serves as the initialization of the texture at the higher resolution. Also, within each resolution level, the method must run the synthesis algorithm repetitively.

In this paper, we proposed wavelet-based framework for the exemplar-based texture synthesis algorithm. We have successfully applied texture skeleton synthesis method to improve the former algorithm. Our method tries to deal with exemplar from a different angle, we can get additional information about pixels' spatial relationships. An exemplar does not have only RGB, specular, shininess or displacement, but also wavelet coefficient. By wavelet transform the texture has nice localization properties in both the spatial and the frequency domains. With the former method, high-dimensional vectors make the algorithm too slow; to solve this problem we have different path. First, we use some algorithms to reduce the dimension, for example PCA, but this needs extra time; Second, select effective channels to integrate redundancy channel, without additional time cost. Furthermore, we are excited about the possibilities of using the method on both 2D and 3D texture synthesis.

Interesting directions for future research include further improving the quality and the speed of synthesis. For example, we plan to further optimize the overlap area between the contiguous neighborhoods. We also plan to use GPU to further accelerate the method. Another promising and important direction is to develop a parallel algorithm.

7. Acknowledgment

This work has been supported by National Natural Science Foundation of China under Grant No.60673196 and No.60776807, Key Topic of 863 Program under Grant No.2006AA12A105, the Natural Science Foundation of Tianjin under Grant No.07F2030, Key Projects in the Science and Technology Pillar Program of Tianjin under Grant No.08ZCGYSF00401 and No.08ZCKFGX04100.

References

[1] Ziv Bar-joseph, Ran El-yaniv, Dani Lischinski, and Michael Werman. Texture mixing and texture movie synthesis using statistical learning. IEEE Transactions on Visualization and Computer Graphics, 7:120–135, 2001.

[2] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In Proceedings of SIGGRAPH 1997, Computer Graphics Proceedings, Annual Conference Series, pages 361–368. ACM, ACM Press, 1997.

[3] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In Proceedings of SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series, pages 341–346. ACM, ACM Press, 2001.

[4] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In Proceedings of IEEE ICCV 1999, Computer Vision Proceedings, Annual Conference Series, pages 1033–1039. IEEE, 1999.

[5] C. Gallagher and A. Kokaram. Nonparametric wavelet based texture synthesis. In Proceedings of IEEE ICIP 2005, Image Processing Proceeings, Annual Conference Series, pages 462–465. IEEE, 2005.

[6] David J. Heeger and James R. Bergen. Pyramidbased texture analysis/synthesis. In Proceedings of SIGGRAPH 1995, Computer Graphics Proceedings, Annual Conference Series, pages 229–238. ACM, ACM Press, 1995.

[7] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. ACM Transactions on Graphics, 26(3):1–9, July 2007.

[8] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. ACM Transactions on Graphics, 24(3):795–802, July 2005.

[9] Vivek Kwatra, Arno Sch¨odl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. ACM Transactions on Graphics, 22(3):277–286, July 2003.

[10] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. ACM Transaction on Graphics, pages 777–786, 2005.

[11] Sylvain Lefebvre and Hugues Hoppe. Appearancespace texture synthesis. ACM Transaction on Graphics, 25(1):541–548, January 2006.

[12] Darwyn R. Peachey. Solid texturing of complex surfaces. In Proceedings of SIGGRAPH 1985, Computer Graphics Proceedings, Annual Conference Series, pages 279–286. ACM, ACM Press, 1985.

[13] Javier Portilla, Eero, and P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. International Journal of Computer Vision, 40:49–71, 2000.

[14] Leandro Tonietto, Marcelo Walter, and Claudio Rosito Jung. Patch-based texture synthesis using wavelets. In Proceedings of SIBGRAPI 2005, pages 383–389, 2005.

[15] Li-YiWei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, pages 479–488, New York, 2000. ACM, ACM Press.

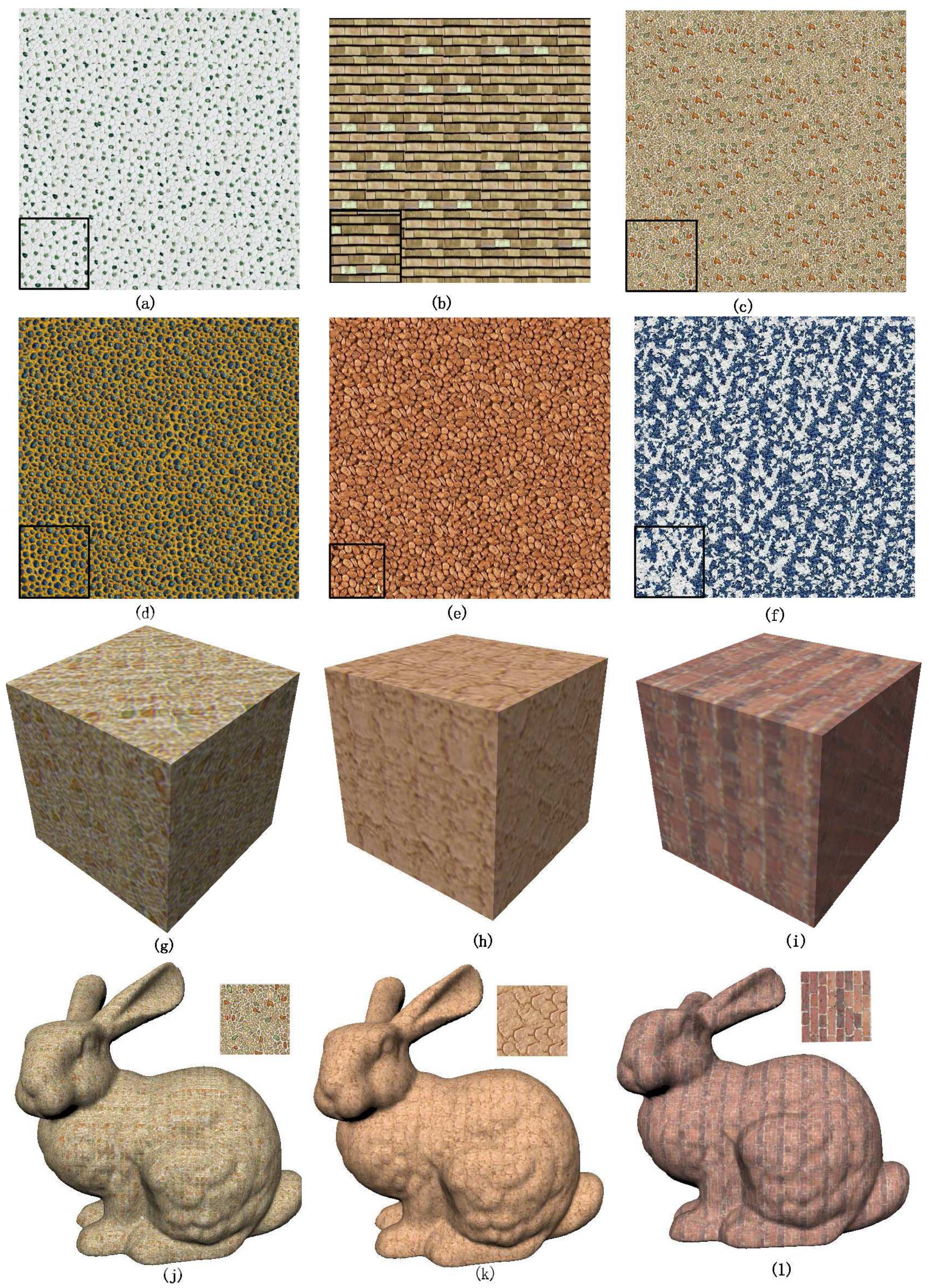

Figure 6: Texture synthesis results by our method with 1 level wavelet transform. The exemplars and corresponding results are listed as (a)flower pattern, (b)brick, (c)leaf, (d)animal skin,(e)peanut, (f) RGan cool. (g) to (i) in the third line are solid cube textures, (j) to (l) in the fourth line are corresponding solid textures carved by Stanford bunny.