Solid Texture Synthesis using Position Histogram Matching

Jiating Chen*,†‡§, Bin Wang*‡§

*School of Software, Tsinghua University, Beijing, China 100084
†Department of Computer Science and Technology, Tsinghua University, Beijing, China 100084
‡Key Laboratory for Information System Security, Ministry of Education, Beijing, China 100084
§Tsinghua National Laboratory for Information Science and Technology, Beijing, China 100084
chenjt04@gmail.com, wangbins@tsinghua.edu.cn

Abstract

In the past, several approaches have been proposed to produce high quality solid textures. Unfortunately, they often suffer from several synthesis artifacts, such as color blurry, bad texture structures, introducing aberrant voxel colors and so on. In this paper, we present a novel algorithm for synthesizing high quality solid textures from 2D exemplars. We adopt an optimization framework with the k-coherence search and the discrete solver for solid texture synthesis. The texture optimization approach is integrated with a new kind of histogram matching, Position Histogram Matching, which effectively causes the global statistics of the synthesized solid textures to match those of the exemplars. Experimental results show that our synthesized results do not suffer from color blurry, and most texture structures are preserved well, outperforming the previous solid texture synthesis algorithms in terms of the synthesis quality.

1. Introduction

Texture mapping has been recognized as an important tool in modeling surface details without explicit modeling for geometry or material properties. However, defining a distortion-free and discontinuity-free mapping is challenging and sometimes even impossible for some complex objects. In contrast, when applying a solid texture onto an object, it allows carving the object out a block of the texture volumetric data, avoiding the problems of distortion and discontinuity. Furthermore, once a solid texture is available, it can be used to texture arbitrary objects not only on object surface, but also throughout the entire object volume.

A wide variety of solid textures have been produced by recent work on solid texture synthesis from 2D exemplars. Wei [1] adapted 2D neighborhood matching synthesis schemes to 3D volumes, but got unfavorable results, which are rather blurring. Qin and Yang [2] generated solid textures by capturing the co-occurrences of grayscale levels in the neighborhoods of 2D images. Since the color channels are closely correlated, independent channel synthesis usually leads to visual artifacts. Jagnow et al. [3] proposed a solid texture synthesis method based on stereology techniques, in which large texture structures are preserved by modeling for different particle shapes. However, it tends to omit a number of fine structures due to the segmentation. Kopf et al. [4] extended 2D texture optimization techniques to synthesis solid textures, by introducing color histogram matching approach to improve the synthesized results. Dong et al. [5] generated solid textures by limiting the synthesis domain to a subset of the voxels around the object surface and performing a parallel spatially deterministic synthesis algorithm on GPU. Though both of them produce impressive results in many cases, some obvious problems have not been alleviated. For example, fine grain details are missing, and some distinct texture structures enclosed by the red panes are missing, as illustrated in Figure 1.

Based on an optimization framework[6] with k-coherence search [7] and discrete solver [8], our approach aims at introducing the position histogram matching in the re-weighting scheme, which ensures that all the voxel colors are copied equiprobably from the exemplars, thus both neighborhood matching and histogram matching are achieved. Since we use k-coherence search in the nearest neighborhood search phase, the computation time is greatly shortened. As expected, experimental results show that our algorithm outperforms or at least is comparable to the previous solid texture synthesis algorithms in terms of the synthesis quality.

Figure 1. Some quality problems for the synthesized results generated by previous methods.
2. High Quality Solid Texture Synthesis

Similar to the approach of Kopf et al. [4], the energy function that measures the differences between the solid texture and the exemplar is defined as

\[
E(s, e) = \sum_{v} \sum_{i\in\{x,y,z\}} \|s_{v,i} - e_{v,i}\|^2
\]
\[
= \sum_{v} \sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} w_{v,i,u} \|s_{v,i,u} - e_{v,i,u}\|^2
\]

Here \(e\) denotes the input exemplar, \(s\) denotes the synthesized solid, \(s_{v,x}, s_{v,y}\) and \(s_{v,z}\) are the neighborhoods of \(v\) in the slices orthogonal to the \(x, y\) and \(z\) axes, \(N_i(v)\) denotes the neighborhood of the voxel \(v\) in the slice perpendicular to the \(i\)-th axis, and \(\gamma = 0.8\). More specific issues are discussed in [4].

Our algorithm is multi-resolution and computes the output solid textures from the lowest resolution to the highest resolution. It is achieved by solving Equation (1). Before synthesis process, our algorithm requires a preprocessing step for computing a k-coherence similarity-set for each input pixel. The synthesis process begins with the lowest resolution volume where the initial value of each voxel is randomly chosen from the exemplar. When synthesizing a higher resolution, we first initialize it by upsampling from the already synthesized lower resolution result, and then perform synthesis on this level in an iterative way, alternating between the optimization phase and the search phase, until the synthesized solid achieves good quality.

2.1. Optimization Phase

Kopf et al. [4] adopts the least square solver to minimize the energy. The updated voxel is defined as

\[
s'_{v} = \sum_{i\in\{x,y,z\}} \sum_{u\in N_i(v)} W_{v,i,u} e_{u,i,v}
\]

Instead we use the discrete solver [8]. For each updated voxel, the pixel in the set \(s(v) = \{e_{u,i,v} | u \in \{x, y, z\}, v \in N_i(v)\}\) that most reduces the energy function is chosen for the updated voxel. In practice, we first calculate a prospective value \(s_v\) using Equation (2), and then select a texel \(e_{u,i,v}\) from the set \(s(v)\) that is most similar to \(s_v\) for the updated voxel. Since each voxel is copied directly from the input exemplar, the blur issue can be avoided.

2.2. Search Phase

We adopt the k-coherence search in search phase. A candidate-set is built for each output voxel by taking the union of all similarity-sets of the neighborhood pixels, and then the best match for each voxel neighborhood is obtained by searching through the candidate-sets. Initial experiments show that it’s still time consuming for the neighborhood comparison in a full dimensional space with \(8 \times 8\) neighborhood. A good tradeoff between the quality and speed for the nearest neighborhood search is achieved by reducing the full dimensionality from 192 to about 20 dimensions with the Principal Component Analysis (PCA).

2.3. Position Histogram Matching

Kopf et al. [4] preserves global statistics by using a re-weighting scheme with color histogram matching in the following way:

\[
w'_{v,i,u} = \frac{w_{v,i,u}}{1 + \sum_{b=1}^{k} \max(0, H_{s,j}(b_j(e_{u,i,v})) - H_{e,j}(b_j(e_{u,i,v})))}
\]

Here \(H_{s,j}\) and \(H_{e,j}\) denote the \(j\)-th histogram of the synthesized result and the exemplar. \(H(b)\) denotes the value of bin \(b\) in a histogram, and \(b_j(c)\) specifies the bin containing color \(c\) in the histogram \(H_{s,j}\) and \(H_{e,j}\). There are two conspicuous limitations existing in color histogram matching. First it works only for color but not for general structure information, as demonstrated in Figure 2. Both color histograms of Results 1 and 2 keep in step with that of the exemplar, but some obvious texture structures are missing in Result 1. Secondly it even fails to preserve color histograms sometimes. For example in Figure 1, the texture structures of white color enclosed by red panes are missing in Kopf et al.’s result. One possible reason is that texture channels are improperly decorrelated. For instance, the weight of a texel \(A(r, g, b)\) expected to be kept in the re-weighting scheme, is reduced by Equation (3) actually because one or two bins among \(\{b_1(A), b_2(A), b_3(A)\}\) in the result histogram have a larger count than those in the exemplar histogram.

We first give a primary illustration on the notion of the position histogram. It is defined as a 2D grid, with the same size as the exemplar. Each of the grid unit records the frequency, appearing in the result volume, of the corresponding pixel in the exemplar, as illustrated in Figure 3. In the position histogram, the frequency grows with the increase of brightness. In order to distinguish the place where the frequency is 0, we paint it red. In other words, in the red part the corresponding pixels in the exemplar do not appear in the result. In Figure 3 the black texture structures corresponding to the red part of the position histogram are missing in the synthesized result for Result 1. And for Result 2, the frequency of most part of the position histogram is approximately the same. Therefore, almost all the texture structures in the exemplar can be found in the result volume.

Our main purpose of introducing position histogram matching is to ensure that most of the pixels in the exemplar can be found in the result, and they are required to
we reduce the weight in order to make it harder to choose pixel $p$ for the updated voxel, or else it would converge to the same position, preventing the pixels appearing equiprobably in the result.

3. Results

We implement our synthesis algorithm by using a three-level synthesis pyramid, with $8 \times 8$ neighborhood at the lower two levels and $6 \times 6$ neighborhood at the highest level, and $k = 5$ for the k-coherence search. It takes only about 5 iterations to obtain good quality results at the higher two levels. Due to the fewer iterations and the k-coherence search, synthesizing a $128^3$ solid texture takes less than 10 minutes on a 2.2 GHz CPU. Therefore, the synthesis time is shorter than that of Kopf et al. [4], and it is independent of the size and richness of the exemplars.

In Figure 4, we show some of our synthesized results. Our technique works well for a wide range of textures varying from isotropic to anisotropic, from low-frequency to high-frequency, from fine-grain-detailed to strong-large-structured, and so on. Figure 5 shows comparisons with several previous approaches. In comparison with Kopf et al.’s [4], our results do not suffer from color blurring, as well as fine grain details and most texture structures are well preserved. More importantly, large texture structures can be efficiently preserved even without the aid of the feature maps by our method, which is demonstrated in the second row. While all the pixels from the exemplar are restricted to have the same probability to appear in the synthesized result, little extra effort should be made to preserve texture structures. Therefore, we can conclude that our method does produce high quality solid textures better for a wider textures than previous methods.

4. Conclusions

In this paper, we present a novel algorithm for synthesizing high quality solid textures from 2D exemplars. Based on the optimization framework, our algorithm enables most of the pixels in the exemplars to appear in the result volume equiprobably by using the position histogram matching. Experiments prove that our method is efficient enough to preserve not only the color histogram but also the various texture structures in the synthesized result. Experimental results demonstrate that our method outperforms or at least is comparable to the previous solid texture synthesis algorithms in terms of the synthesis quality.
Figure 4. Results of our high quality solid texture synthesis.

Figure 5. Comparisons with previous methods.

Acknowledgments

This work is supported by National Science Foundation of China (Grant Nos. 90818011, 60773143 and 90715043), National High-Tech Research & Development Program of China (Grant No. 2007AA040401), and National Basic Research Program of China (Grant No. 2004CB719400).

References


