Semiregular Solid Texturing from 2D Image Exemplars

Song-Pei Du, Shi-Min Hu, *Member*, *IEEE*, and Ralph R. Martin

Abstract—Solid textures, comprising 3D particles embedded in a matrix in a regular or semiregular pattern, are common in natural and man-made materials, such as brickwork, stone walls, plant cells in a leaf, etc. We present a novel technique for synthesizing such textures, starting from 2D image exemplars which provide cross-sections of the desired volume texture. The shapes and colors of typical particles embedded in the structure are estimated from their 2D cross-sections. Particle positions in the texture images are also used to guide spatial placement of the 3D particles during synthesis of the 3D texture. Our experiments demonstrate that our algorithm can produce higher quality structures than previous approaches; they are both compatible with the input images, and have a plausible 3D nature.

Index Terms—Solid texture, texture generation, semiregular texture

1 Introduction

In computer graphics, texture mapping is widely used to apply 2D texture to the surface of 3D objects to enhance their appearance. However, many real-world objects have interior structure, not just a surface texture. Traditional 2D texturing techniques only provide information about objects' external appearances.

In various applications, solid texturing can provide more realistic results than surface texturing, for example, when rendering objects made from natural materials such as marble or wood, and when we wish to effectively simulate subsurface scattering, or breaking objects. Solid texturing has the further advantage over 2D texturing of not suffering from distortion introduced by surface parameterization. The synthesized solid volume is also mesh independent: the same solid volume can be used to texture varying objects with different shapes and topologies.

Solid textures for computer graphics were first investigated in the 1980s [1], [2]. Recently, interest in solid texture synthesis has revived, and newer methods can generate various kinds of solid textures with high visual quality. While it is straightforward to synthesize solid textures from 3D exemplars, in practice, 3D exemplars are very hard to acquire, and in practice, most modern techniques extend image texture synthesis approaches. The most successful approaches start from 2D images representing three orthogonal views of cross-sections of the texture [3], [4]. While generating impressive results for a wide range of textures, the problem of generating a 3D volume from 2D images is underconstrained, with no unique solution. A good solution is one in which objects exhibit a plausible appearance, with an interior structure in accordance with what our experience of the real world tells us we should see if we cut through the object. These approaches use the given 2D views to generate a texture such that *every* 2D cross-section orthogonal to the coordinate axes is consistent with the input images. Unfortunately, by itself, this is insufficient to produce results with truly plausible 3D structures of the kind we experience in everyday life. The examples in Fig. 1 highlight unnatural results produced by earlier approaches; in particular, they fail to adequately reconstruct the interior structures (see Fig. 1a) or lead to unintended and undesirable diagonal banding (Fig. 1c).

Jagnow et al. [5] propose an alternative solid texturing technique aimed at synthesizing the solid texture of aggregate materials. This seems to be the only work which directly considers synthesis of the interior structure of the volume from plausible 3D particles, and accordingly generates high-quality results. However, the main emphasis in that work is on the statistical distribution of particles, while their local arrangement into structures is ignored. Thus, it is unsuited to textures with regular or semiregular patterns of the kind considered in [6]. In addition, the shapes of all particles need to be manually modeled by users, which makes it inconvenient in practice.

Textures comprising 3D particles embedded in a matrix in a regular or semiregular pattern are common in natural and man-made materials, such as brickwork, stone walls, plant cells in leaves, etc. In this paper, we propose a novel technique which generates such a solid texture from input images of cross-sections of the texture. The following are the key technical contributions: first, instead of requiring particles to be modeled by the user, we provide a novel technique to select proper cross-sections in 2D exemplars and automatically construct a set of 3D particles with interior colors. Second, the arrangement of particles in 3D space is established according to the input 2D exemplars: the output solid texture is generated by choosing and packing

- *S.-P. Du and S.-M. Hu are with the TNList, Department of Computer Science and Technology, Tsinghua University, Information Technology Building (FIT), Beijing 100084, China. E-mail: dusongpei@gmail.com, shimin@tsinghua.edu.cn.*
- *R.R. Martin is with the School of Computer Science and Informatics, Cardiff University, 5 The Parade, Roath Cardiff, Wales CF24 3AA, United Kingdom. E-mail: ralph@cs.cf.ac.uk.*

Manuscript received 4 Sept. 2011; revised 28 Feb. 2012; accepted 4 May 2012; published online 14 May 2012.
Recommended for acceptance by G. Drettakis.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-09-0216.
Digital Object Identifier no. 10.1109/TVCG.2012.129.

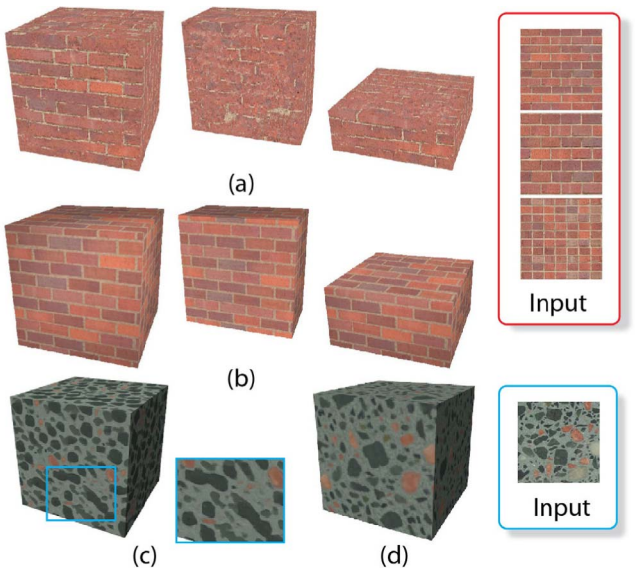

Fig. 1. Artifacts and diagonal structures in previous approaches. (a) Result using [4], and slices through it. (b) Our result. (c) Result using [3] and closeup view showing diagonal structure artifacts. (d) Our result.

particles from the set of precomputed particles. Third, a simple procedure may be optionally used to locally refine particles' positions and shapes. The key point of our method is the emphasis on producing a plausible internal structure in terms of both particle shapes and their structural arrangement. The overall pipeline is shown in Fig. 2.

2 Related Work

We now review directly relevant work on solid texture synthesis. For a complete survey, see [7].

2.1 Procedural Solid Texturing

Early work on solid texture synthesis mostly used procedural methods. A noise function was introduced in [2] for this purpose, and the approach was extended by Worley [8] to support cellular texturing. Recently, Lagae and Drettakis [9] presented a noise function that supports anisotropic filtering of sliced solid noise. Procedural texture synthesis also has a close relationship with model synthesis [10]. The user has to provide a parametric description of the desired texture, which is controlled by varying the parameters. It is difficult in practice to achieve a desired texture.

2.2 Image-Based Solid Texture Synthesis

Recent works use texture examples as a way of making texture synthesis easier for the user. Wei et al. [11] give a detailed survey of example-based texture synthesis.

Capturing 3D texture data directly is hard, requiring, e.g., MRI scanners, so the focus has been on synthesizing solid texture from 2D images of textures. Thus, Djamchid and Jean-Michel [12], [13] use a parametric approach based on analyzing the spectral characteristics of the input 2D images, while Heeger and Bergen [14] use a pyramid histogram matching technique to synthesize a volume with similar statistical properties to the input images. Several algorithms take multiple 2D example images representing texture cross-sections in orthogonal views, and generate a volume for which each 2D cross-section parallel to one of the given views is similar to the corresponding input image. Wei [15] did this first, extending exemplar-based techniques for 2D image texture synthesis to 3D. Three interleaved 2D neighborhoods orthogonal to the main axes are determined at each voxel of the output, and the color of each voxel is set by a local optimization process which minimizes the error between the output 2D neighborhoods and their best matches in the input images. Qin and Yang [16] synthesize solid texture based on *basic gray level aura matrices* (BGLAMs), characterizing the co-occurrence probability distributions of gray levels at all possible displacements. BGLAMs are computed for the input 2D exemplars, providing constraints from multiple view directions when sampling the volume to produce the output. Kopf et al. [3] extend global texture optimization methods from 2D [17] to generate solid textures, using neighborhoods in a similar way to [15]. Dong et al. [4] give an efficient GPU-based technique based on a synthesis pipeline first introduced by [18] for 2D. It is spatially deterministic and only synthesizes voxels when necessary. Ma et al. [19] use similar techniques for motion synthesis based on 3D vector fields. This idea is extended in [20] to texture 3D objects guided by user sketching.

These exemplar-based techniques can produce good results, but share a common problem: implausible results are typically produced for textures comprising particles arranged in a background material matrix in regular or semiregular structures, such as stone walls. Zhang et al. [21] introduced a *texton mask* to control the synthesis process to better preserve features for image texture synthesis, and this idea has been employed in 3D texture synthesis [3], [4].

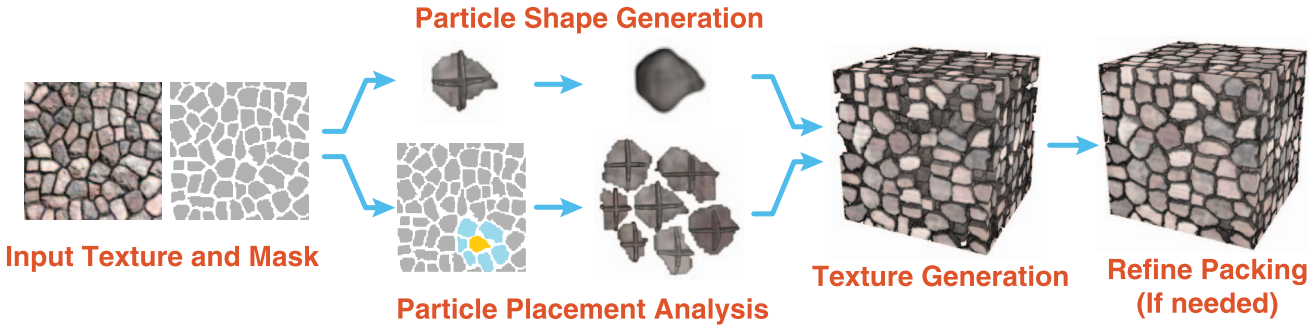

Fig. 2. Pipeline. The shapes and colors of particles are determined from three orthogonal input images. Three-dimensional neighborhood information is analyzed to guide the texture generation process to generate a solid texture with plausible structures. An optional step of particle adjustment may be used after initial texture generation to further improve quality.

While this does better preserve the structuring, and improves the visual quality of the result, implausible regions are still produced.

Jagnow et al. [5] introduced stereological techniques to synthesize solid textures for aggregate materials. The algorithm places particles using a precomputed distribution, and cannot handle 2D image inputs with particles in semiregular patterns. Vector representation allows resolution independence. Wang et al. [22], [23] introduced the vector representation for solid textures composed of intermixed regions, while they did not deal with the problem of synthesizing solid textures. Ma et al. [24] introduced discrete element textures to synthesize repetitive elements according to the input exemplars, while generating impressive results, to synthesize a 3D volume, 3D exemplars must be given.

2.3 Synthesis of Volume Interiors and Natural Objects

Other works focus on texturing the interior of a *specific* object. These approaches can provide realistic results for objects with complex interior structures, but require lengthy user interaction. Furthermore, the texture created is dependent on the specific boundary given. This is less than ideal for texturing *classes* of related objects, and objects with complex topology.

Cutler et al. [25] use a procedural approach to generate the interior of solid models, based on a scripting language; the texture generated is layered. Owada et al. [26] describe a user interface for browsing and designing 3D objects whose internal texture is generated from user-specified 2D cross-section images; a similar approach is also employed by Pietroni et al. [27]. Takayama et al. [28] extend the idea of lapped textures [29] to the 3D case and provide an interface to design anisotropic solid textures. However, synthesis is based on 3D solid texture exemplars generated manually or provided by some other method—typically one of the methods mentioned earlier which generate implausible structures. Takayama et al. [30] use diffusion surfaces to design and represent the structure and color of 3D models. These methods are more concerned with the object to be illustrated and its structure, and again the results are object dependent.

3 Algorithm

Our method starts from three mutually orthogonal 2D example images containing cross-sections of the desired texture. The texture should consist of discrete particles which are embedded in a matrix in some regular or semiregular pattern. With each input image, a binary mask must be provided to indicate the locations of cross-sections of particles. This can be simply generated by thresholding and manually refined by the user, as needed. From the input images, we first reconstruct particles, and analyze their neighborhood relationships. We then iteratively add particles to the volume to be textured, to ensure the result has a plausible interior structure. Sometimes, our approach can result in textures with gaps, and we show how this issue can be remedied by postprocessing the particles' positions and shapes. Our method is capable of producing textures with semiregular patterns while needing little user interaction. We now discuss the steps in our pipeline.

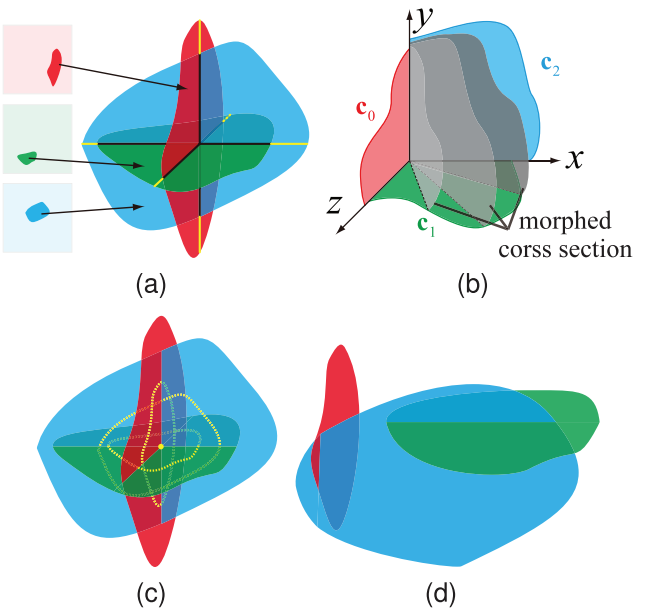

Fig. 3. (a) A triple of three orthogonal cross-sections. Matching error of a triple is the sum of average color differences along each intersection after aligning the cross-sections. (b) Reconstructing the particle shape in one octant by sweeping a morphing plane. (c) The aligned center is constrained to locate in the area enclosed by yellow dash lines. (d) An example showing bad alignment.

3.1 Reconstructing Particles

We first reconstruct the 3D particles used to build the output texture, based on the particles' 2D cross-sections seen in the three orthogonal input images.

First, the cross-sections for each particle present in each image are extracted, giving three sets P_i, $(i = 1, 2, 3)$ of cross-sections from different views. As in general the cross-sections may not be registered, we do not know which particle cross-sections are associated and how they are aligned. We use a brute force approach to this problem. We construct *triples* of cross-sections, one selected from each of $P_i (i = 1, 2, 3)$, respectively. There are $P_1 P_2 P_3$ possible triples in total. To decide which triples most plausibly represent particles, we should find those triples of aligned cross-sections which have minimal summed matching error between each pair of cross-sections. In practice, we constrain the distance between the aligned center and the centroid of the each cross-section to be no more than half of the minimal distance between the boundary and the centroid. As shown in Fig. 3c, in each cross-section, the aligned center must thus be located in the area enclosed by yellow dashed lines. This quickly provides good alignment and helps to avoid unhelp cases, e.g., where cross-sections have little or even no intersection (as in Fig. 3d). Although this may reject some plausible alignments, given the randomness and richness of cross-sections, other suitable similar particles can generally be generated from other candidate triples and alignments.

The matching error takes into account both color and shape. It is set to the average color difference in RGB space along the intersection line between the pair of cross-sections; to penalize cross-sections with incompatible shapes, the color difference is set to its maximum value in places along the intersection line where one cross-section

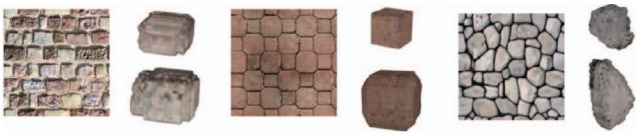

Fig. 4. Generated particles.

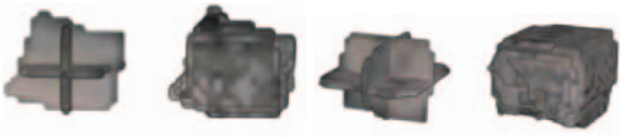

Fig. 5. Two particles constructed from candidate triples with large error. In each example, the candidate triple is shown on the left and the constructed particle is on the right.

has a different diameter to the other (see the yellow line segments in Fig. 3a). The lower the matching error, the more likely the triple corresponds to a meaningful particle. Users can explicitly set an error threshold to avoid implausible shapes; in practice, we retain triples whose error is less than 20 percent of the maximum error which occurs—this worked well for the examples in the paper.

Various approaches can be used for modeling the 3D shapes of particles given three orthogonal 2D *profiles*. Dischler and Ghazanfarpour [31] assume the particles to be generalized cylinders, and model 3D shape by sweeping a generatrix curve around some base curve. Jagnow et al. [32] extend this approach to support multiple morphed curves swept as generatrix curves; this can lead to more natural results. We employ a modified version of the latter, using the three 2D cross-sections in a triple to reconstruct that particle's shape as well as its color.

Consider the positive octant. Two of the cross-sections are arbitrarily chosen as sweeping planes (the red and blue planes in Fig. 3b, denoted $\mathbf{c}_0$ and $\mathbf{c}_2$), while the third is used as a base plane (the green plane in Fig. 3b, denoted $\mathbf{c}_1$). The general idea is to sweep $\mathbf{c}_0$ to $\mathbf{c}_2$ around the y axis.

More specifically, we first construct a morph sequence between $\mathbf{c}_0$ and $\mathbf{c}_2$. Doing so is simple if $\mathbf{c}_0$ and $\mathbf{c}_2$ are both convex: we represent $\mathbf{c}_0$ and $\mathbf{c}_2$ in polar coordinates, and continuously morph the radius. For nonconvex shapes, as the topology of the cross-section is equivalent to a disk, some other shape interpolation technique must be used to construct the morph; we follow [33]. The morph sequence of cross-sections is then scaled in the zx plane to agree with the boundary given by $\mathbf{c}_1$. The volume swept by the morphed cross-sections along the boundary of $\mathbf{c}_1$ represents the shape of the particle in this octant.

For every point p in the swept volume, we can find three corresponding points in the three cross-sections, respectively: two morphed points p_0, p_2 in $\mathbf{c}_0$, $\mathbf{c}_2$ and one projected point p_1 in $\mathbf{c}_1$. The color of p is set to the color of that p_i, $i = 1, 2, 3$ whose color is closest to the mean color of the three.

The above does not work directly if $\mathbf{c}_1$ is not star-shaped with respect to the origin. In this case, if either of $\mathbf{c}_0$ and $\mathbf{c}_2$ is star-shaped, we choose it as the base plane instead; otherwise, we just discard this particle as too complicated.

Shapes of some generated particles are illustrated in Fig. 4. The reconstructed particles are used to synthesize the volumetric structure in the next step, using a selection process.

The cross sections $\mathbf{c}_0$ and $\mathbf{c}_2$ generally do not have exactly the same size along the y axis. In our algorithm, the shape is continuously morphed from $\mathbf{c}_0$ to $\mathbf{c}_2$ along the y axis. Note, however, the matching error rejection step discards shapes which are too inconsistent, so most selected candidate triples lead to plausible shapes without artifacts. In Fig. 5, starting with the rightmost texture in Fig. 4, we show the cross-sections and particles generated from them, for of the two candidate triples with largest matching errors. Even these particles have plausible shapes.

3.2 Placement of Particles

To generate a plausible texture, the global placement of particles in the result should be in agreement with the observed cross-sections in the input images. Jagnow et al. [5] use a stereological approach to estimate the 3D distribution of particles from the distribution of profiles in 2D images. However, this only considers macroscopic statistics of the particles, and is not sufficient to preserve systematic structures like regular patterns. We employ an alternative approach. We first analyze the input exemplars to gather information about possible neighboring particles for each particle, represented in a *candidate neighbor set* $\mathcal{N}_{\mathcal{P}}(p)$ for each particle p. Each element of $\mathcal{N}_{\mathcal{P}}(p)$ is a pair $(q, \mathbf{d})$ indicating that particle q is an acceptable neighbor for p for at a spatial translation $\mathbf{d}$.

The texture is built by iteratively adding particles to the volume according to information in these precomputed candidate neighbor sets. Dischler et al. [34] use a similar image texturing method which considers co-occurrences of neighboring particles from 2D exemplars, but the method has had to be extended for volume synthesis as we do not have explicit 3D information.

We find each candidate neighbor set by first computing 2D candidate neighbor sets as follows. Given the 2D cross-section $\mathbf{c}$ of some particle in an input image, we find the neighboring cross-sections near $\mathbf{c}$. The 2D candidate neighbor set $\mathcal{N}_{\mathcal{C}}(\mathbf{c})$ for $\mathbf{c}$ is the set of pairs $(\mathbf{c}', \mathbf{d}')$ where $\mathbf{c}'$ is some other cross-section in the same input image as $\mathbf{c}$ and $\mathbf{d}'$ is the translation between their centers (see Fig. 6). In practice, we only consider the k nearest neighboring cross-sections. If all cross-sections were circles of the same size, to get one-ring neighbors, k would be 6. To allow for irregular particle shapes and hence a more variable number of close neighbors, we increase k somewhat, using $k = 10$ in all

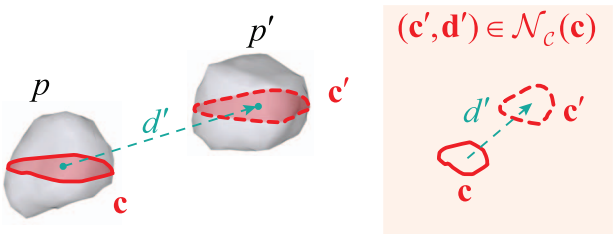

Fig. 6. Left: given particle p, p' is one of its 3D candidate neighbors. Right: it is chosen by considering 2D candidate neighbors of its corresponding cross-section $\mathbf{c}$ within the input exemplars.

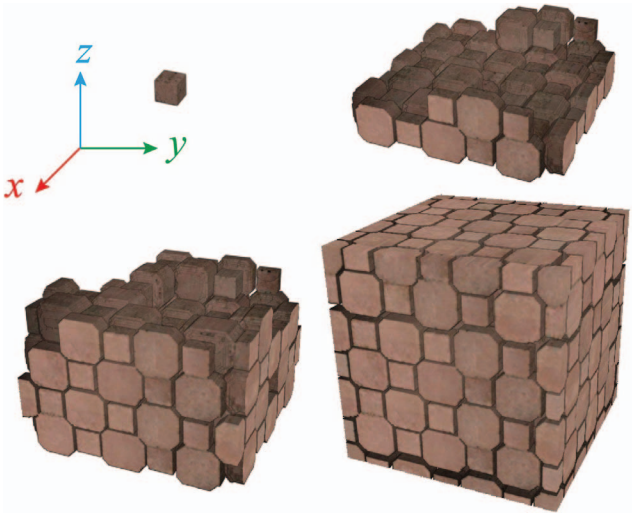

Fig. 7. Tiling snapshots using the quasi-layered order.

examples shown here. We can consider $\mathbf{d}'$ to be a 3D translation where the component perpendicular to the image is zero. Thus, for some particle p, let its set of three 2D cross-sections be $\mathcal{C}(p)$. The set of particles corresponding to cross-section $\mathbf{c}$ is given by $\mathcal{P}(\mathbf{c}) = \{p \mid \mathbf{c} \in \mathcal{C}(p)\}$. Then, the candidate neighbor set $\mathcal{N}_{\mathcal{P}}(p)$ is given by

$$\mathcal{N}_{\mathcal{P}}(p) = \bigcup_{\mathbf{c}\in\mathcal{C}(p)} \bigcup_{(\mathbf{c}',\mathbf{d}')\in\mathcal{N}_{\mathcal{C}}(\mathbf{c})} \{(p', \mathbf{d}') | p' \in \mathcal{P}(\mathbf{c}')\}.$$

Intuitively, $\mathcal{N}_{\mathcal{P}}(p)$ contains those particles which could be neighbors of particle p, together with their positional relationship.

To synthesize the overall volume V, we use a list L_c to store all candidate neighbors of every particle which has been added to V. We start by putting a random particle p into the volume V at the volume center, and add all its candidate neighbors from $\mathcal{N}_{\mathcal{P}}(p)$ to L_c. Then, we iteratively take an element $(p', \mathbf{d}')$ from L_c (chosen as described later) and as long as p' does not overlap with any existing particles, it is added to V, and L_c is updated as a result; otherwise, p' is discarded and we consider the next possibility. Usually, volume synthesis is performed within a cube, with wrap-around coordinates—any particle which goes outside the cube reappears on the opposite side (this allows us to generate a tilable texture), so, eventually, there is no space to add further particles. The algorithm stops when L_c is empty, that is, no further particles can be added to the volume. See Algorithm 1. A key issue is how to choose the next particle to take from L_c. A simple approach is to sort all L_c according to distance between the particle and the volume center, and to take the particle closest to the center at each iteration. This causes the set of added particles to grow isotropically in space, and also results in closely packed particles. Another approach is to add particles in quasi-layered order, which is appropriate if the particles are arranged in regular or semiregular patterns.

Algorithm 1. Texture Synthesis
Require: $\mathcal{C}$ {set of all particles}
$\mathcal{N}_{\mathcal{P}}(p)$ {candidate neighbor set for each $p \in \mathcal{C}$}
Ensure: A synthesized volume V with plausible structures
$V \leftarrow \emptyset$, $L_c \leftarrow \emptyset$
Add one randomly chosen particle p to V
Add each element in $\mathcal{N}_{\mathcal{P}}(p)$ to L_c
while L_c is not empty **do**
  $(p, \mathbf{d}) \leftarrow$ the front element of L_c in some chosen order
  Remove $(p, \mathbf{d})$ from L_c
  if p has no overlap with any particles in V **then**
    Add p to V
    Add each element in $\mathcal{N}_{\mathcal{P}}(p)$ to L_c
  end if
end while
return V

Suppose p is a particle placed in the volume. Each candidate particle in $\mathcal{N}_{\mathcal{P}}(p)$ is computed based on one of the three texture planes $x-y$, $y-z$, or $z-x$. For quasi-layered order, we add the particles layer by layer: we first tile an initial $x-y$ plane, and the volume is then grown vertically along the z axis. Using such a layer-based approach is more natural for regular or semiregular patterns, such as brick-work or a stone wall. The candidates in L_c are the union of those in $L_{c,x-y}$, $L_{c,y-z}$, and $L_{c,z-x}$, each containing candidates computed from one texture plane. During tiling process, we preferentially add particles from $L_{c,x-y}$ first, which makes the volume tend to grow horizontally along in x and y directions. After adding a new particle, the three subsets are all updated. The order within a subset such as $L_{c,x-y}$ is determined by sorting them according to distance from the center of that plane. We lazily take the first suitable particle found. The selection process is thus very efficient, and also successfully preserves the texture pattern of the input exemplars. Fig. 7 illustrates steps of the tiling process using quasi-layered order.

3.3 Coloring the Matrix

The matrix material between the particles usually has little regular structure, so we modify the parallel technique introduced by Dong et al. [4] to synthesize the color of matrix. We only build candidates for masked out pixels of the input exemplar images, and directly "correct" the nonparticle voxels. Typically, we use a neighborhood size of 5 and perform two passes of corrections.

Putting all these ideas together, we can now fill the volume with colored texture. Some examples are shown in Figs. 9 and 12.

3.4 Packing Refinement

The volume filling constructed by Algorithm 1 may contain undesirable gaps: "unlucky" choices made for earlier particles may result in no suitable particles remaining to fill the gaps, and furthermore, the spatial relations of 3D particles derived from the exemplars may be somewhat imprecise. This is particularly a problem for textures densely packed by particles with irregular shapes. The quality of the results can be improved by two optional extra steps: adjusting particles' positions, and adjusting their shapes.

To adjust the particle's positions, we employ a simple repulsion scheme. For two infinitesimal point particles r and s, we define a repulsive force $\mathbf{F}_{rs}$ acting on r due to s to be

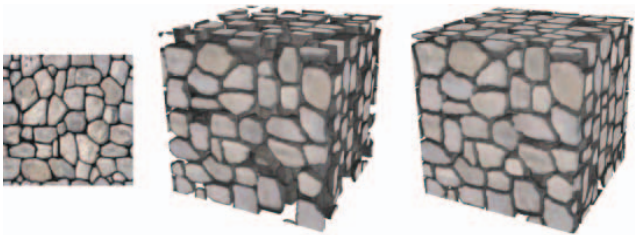

Fig. 8. Left: input exemplar. Middle: initial texture with gaps. Right: final texture after particle adjustment.

$$\mathbf{F}_{rs} = -\frac{\sigma}{d_{rs}}\mathbf{v}_{rs},$$

where $\mathbf{v}_{rs}$ is the unit vector from r to s and d_{rs} is their distance. For two extended particles p and q, we generalize the above equation to give the force $\mathbf{F}_{pq}$:

$$\mathbf{F}_{pq} = \int_{r\in p, s\in q} -\frac{\sigma}{d_{rs}}\mathbf{v}_{rs},$$

so in a discrete setting, the force $\mathbf{F}_p$ on particle p is

$$\mathbf{F}_p = \sum_{q\neq p} \mathbf{F}_{pq}.$$

Computing a precise value for $\mathbf{F}_p$ is expensive and not worthwhile. In practice, we sample the volume using a 64^3 grid, and compute $\mathbf{F}_p$ over a 17^3 neighborhood region. We iteratively update the particles' positions based on these repulsive forces, moving p in the direction of $\mathbf{F}_p$. The distance moved is limited by an exponentially decreasing threshold. Typically, we perform no more than 20 iterations.

After adjusting particles' positions, we may further improve the texture by filling undesirable gaps, by pushing each particle boundary toward its center's Voronoi cell. To avoid deforming the particle too much, we use an approximately uniform scaling approach. In direction $\mathbf{d}$, the maximum allowable scaling factor f_{d} is determined by the requirement that the particle's boundary should not be too close to its Voronoi boundary. The overall uniform scaling factor f_u for the particle is the minimum of f_{d} over each direction orthogonal to a Voronoi cell boundary. We set the final scaling factor in direction $\mathbf{d}$ to $\min((1+\alpha)f_u, f_{\mathrm{d}})$. Increasing α fills gaps more, but causes more deformation of the particle's shape; a suitable default value is $\alpha = 0.1$.

For simplicity, we implement a discrete version of the Voronoi diagram on a grid, and compute the nearest particles for each grid node to give the discrete Voronoi cell for each particle. The allowable closeness of particles to their Voronoi boundaries can be automatically calculated as half the average thickness of the matrix from image exemplars, or can be set by the user. After scaling particles, particles are necessarily located inside the Voronoi cells, which also removes any overlap of particles caused by the position adjustment step. Fig. 8 shows an example of the overall adjustment process.

4 Results and Discussions

4.1 Results

We have tested our algorithm using a wide range of 2D exemplars, using three orthogonal images with corresponding masks as input. The time taken for synthesis depends on the number of particles generated in the solid texture cube. With a discrete resolution of 128^3, the time taken for estimating and placing particles is about 1 second for tens of particles, and no more than 10 seconds for about 3,000 particles, using a quad core 2.27 GHz CPU. The total time including the optional refinement step is no more than 5 minutes.

Our algorithm can generate three kinds of texture: those with regular, semiregular, or irregular patterns. Here, we demonstrate our results and show the strengths of our methods over existing techniques.

The main advantage of our approach over earlier methods is its ability to generate solid textures containing particles placed in a regular or semiregular pattern, as shown in Fig. 9. Co-occurrence of 3D particles is well

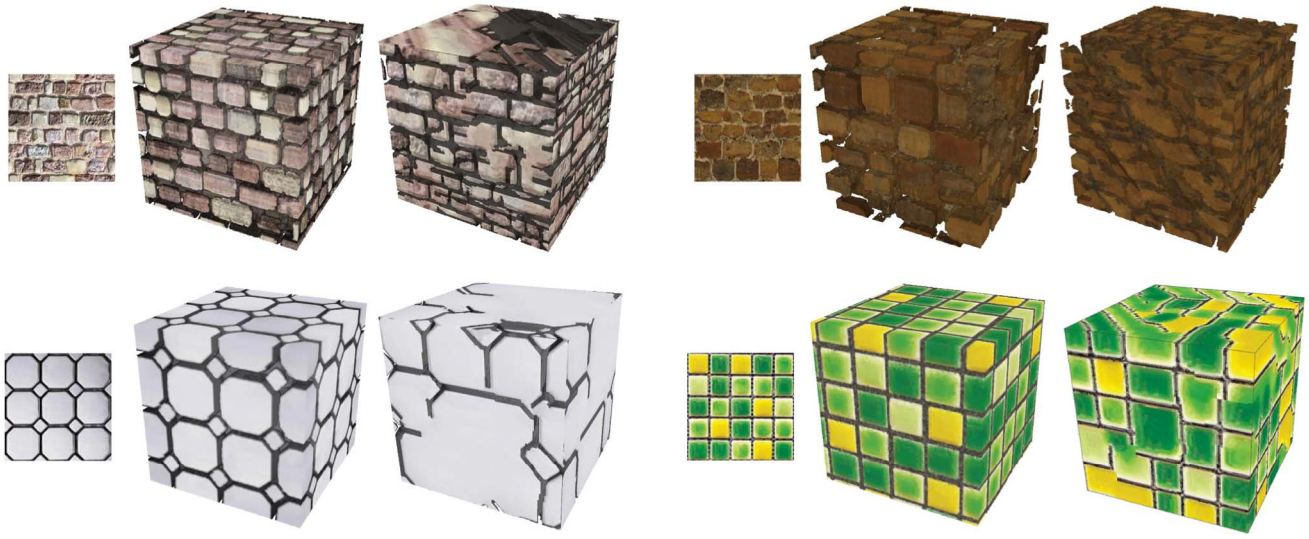

Fig. 9. Synthesized regular and semiregular solid textures; in each example, our result is shown on the left and the result based on [4] is on the right. The first three cases are rendered with a transparent matrix to help see the internal structure.

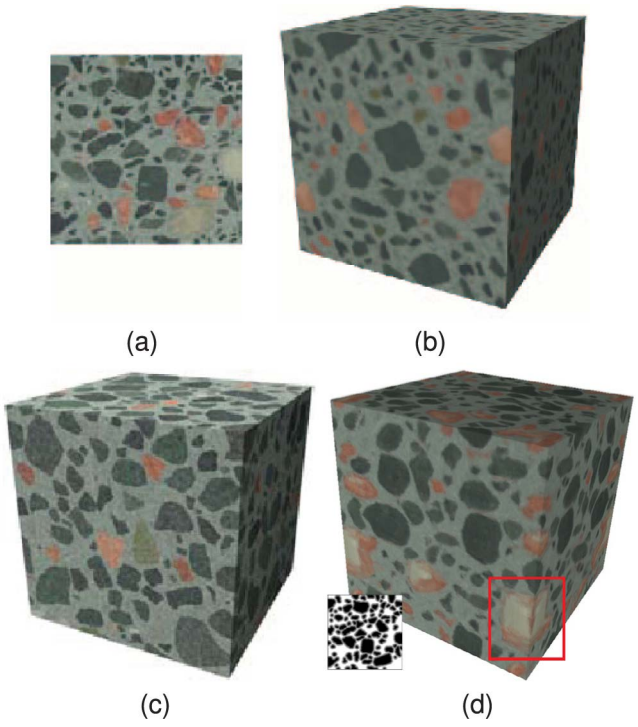

Fig. 10. Comparative results. (a) Input image. (b) Our result. (c) Result based on [5]. (d) Result based on [4].

captured by the candidate neighbor sets, allowing generation of an appropriate volumetric structure.

While perfectly regular structures can be fairly simply generated by procedural techniques, it is tedious to have to devise a new procedure for each structure. This paper contains examples with five different regular patterns, each of which would need a different procedure. In addition, construction of plausible particle shapes for use in a procedural approach, with variation in shapes and color, is not a trivial task; it may also be hard to adjust the procedure to suit variable sized particles. A further disadvantage is that small repetition units may lead to noticeable repetition in generated output. One main advantage of our method is when synthesizing semiregular textures like the first row in Fig. 9. No existing methods can produce satisfactory results for such cases, and procedural based methods would also find it difficult to do so.

We have also compared our methods with the approach in [4], using 7×7 neighborhoods, four passes and a four level pyramid. As shown in Fig. 9, through some 2D slices look similar to the input exemplars, the interior structures are not well preserved.

For textures with irregularly distributed particles, our algorithm can successfully preserve the global appearance of the input exemplars and provide results comparable to those in [5], while doing so more efficiently (see Fig. 10). Excluding the time for manually modeling particles, the time spent by [5] varies from a few minutes to an hour, depending on the particle density, size, and complexity. The most time consuming part of our method is step based on the algorithm in [4] to synthesize the color of the matrix, which takes no more than 1 minute. The rest of our algorithm takes less than 20 seconds. In addition, our algorithm constructs the particles automatically which

Fig. 11. Comparison to a random particle placement approach. Left: randomly placed particles, showing large gaps and overlaps. Middle: after particle deformation, particle shapes are significantly distorted, and unlike the input texture. Right: our result.

makes it more practical; we also produce specific colors for particles and the matrix, rather than a constant color with noise as in [5]. Fig. 10 also provides a comparison to the method in [4], which results in undesired color blending in some regions.

For irregular textures, we note that simply placing particles at random positions cannot produce satisfactory results, as it will result in gaps and overlaps. We may try as an alternative to place points in locations leading to a desired Voronoi diagram to get dense packing, but if the shapes and sizes of (the given) particles have large variations, determining suitable locations for Voronoi cells which match the set of particles is a difficult task. In practice, large gaps will still occur; on the other hand, deforming the particle to fit its Voronoi cell may cause its shape to differ too much from its original shape. Fig. 11 shows such a result based on such an approach for the texture in Fig. 8, using the same number of particles as computed by our algorithm and randomly placing them in the volume. Even after deforming the particles, the result looks very different to the input exemplar.

Further results of our method are shown in Fig. 12; using the same input exemplars, comparisons to the output produced by the methods in [3] and [4] are shown in Fig. 13.

4.2 Particle-Aware Texture Editing

The textures we produce have particles embedded in a matrix. This nature can be taken into account when deforming objects having such textures. For example, we can model the particles as quite rigid, while the matrix is some more readily deformable material. Under a nonrigid transform, the shapes of the particles should deform little (or not at all), and most of the deformation should take place in the matrix. Fig. 14 demonstrates a simple case of deforming a textured cube by bending. The particle centers are transformed first, and a small local deformation applied to each particle; the rest of the deformed cube is again filled with matrix.

Another case of texture editing is shown in Fig. 15, we directly change the particles' color according to some existing solid texture to produce various results.

4.3 Limitations and Applicability

Our algorithm works best when the input 2D images have particles arranged in a fairly regular cellular-like structure, like a stone wall or pile of pebbles. There are several cases in which our method does not provide high-quality results: although the shapes of inferred particles are plausible, their placement may not be as expected.

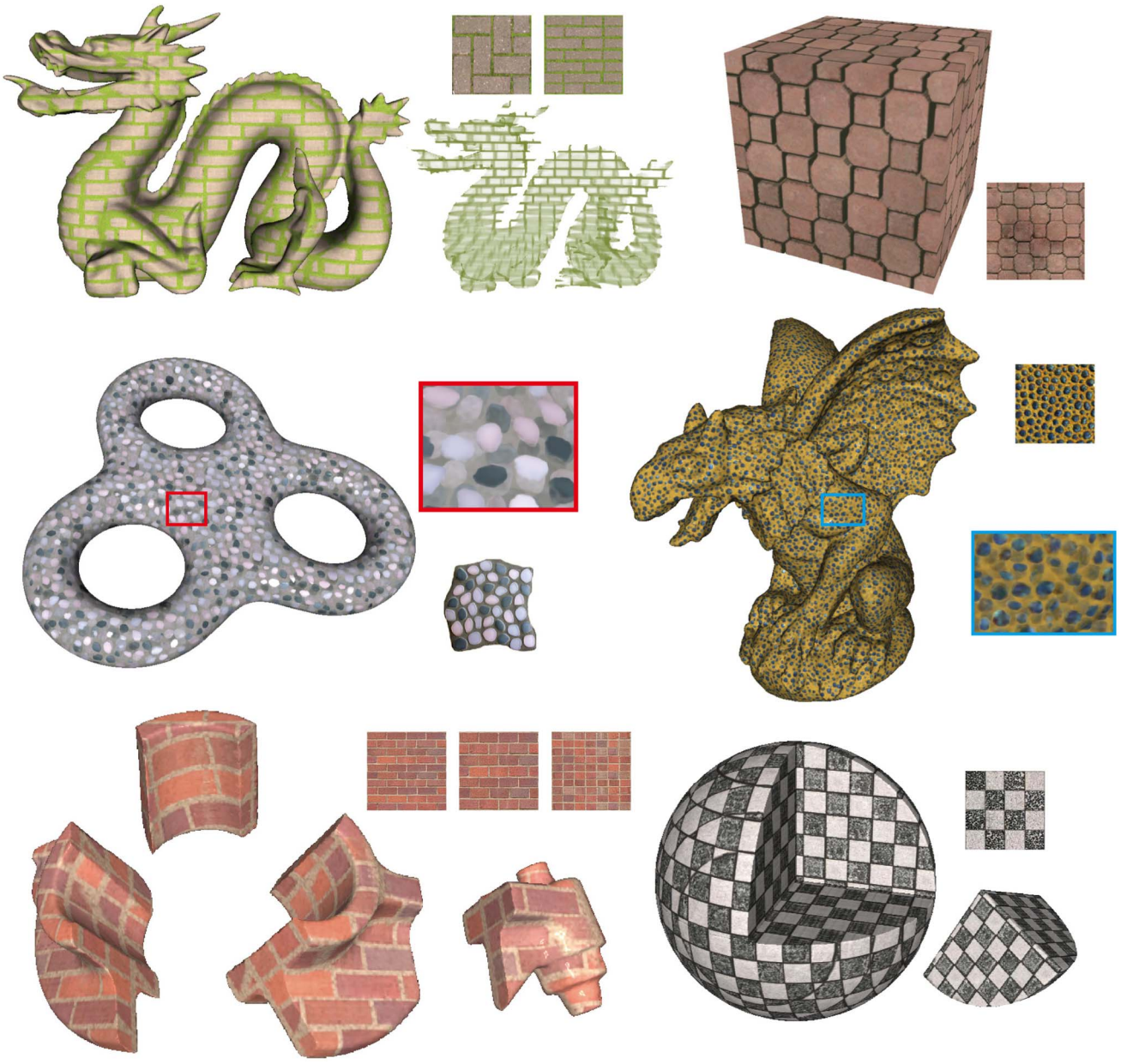

Fig. 12. Further solid textures synthesized using our approach. For the dragon model, the image at the lower right uses transparent bricks to show the volumetric structure. The closeup view of the gargoyle model shows the particles embedded in a translucent matrix.

First, if the three input exemplars do not correspond to an intuitive volume structure, packing may be rather disorganized, as shown in the first row of Fig. 16. In such cases, it is in fact rather unclear what a "natural" answer actually would be.

Second, we pack the particles using only nearest neighbor information, which may not be sufficient, if the three input exemplars imply some longer range organization. For example, the three input exemplars may be composed of particles packed in concentric rings (see the second row in Fig. 16), leading to an expected 3D structure comprising concentric spheres of particles. Further work is needed to handle such cases. Indeed, in general, synthesis of solid texture from 2D exemplars is an underconstrained problem, and 2D exemplars of three orthogonal views are not able to provide enough information to reproduce the volume. As another example, consider particles arranged in a body-centered cubic structure. It would be tricky to synthesize such a solid texture from three 2D exemplars.

Overall, the strength of our method lies in producing plausible results when the "natural" result is a semiregular structure of some kind.

5 Conclusion

This paper has given a novel technique for synthesizing solid textures from 2D cross-section views of semiregular structures containing 3D particles. Plausible 3D particle shapes are ensured by reconstructing them from compatible cross-sections. Analyzing the input texture tells us how to place particles in 3D to build up a texture in which the structure is well preserved. Experiments show that our algorithm can produce more realistic semiregular solid textures from 2D exemplars than previous approaches.

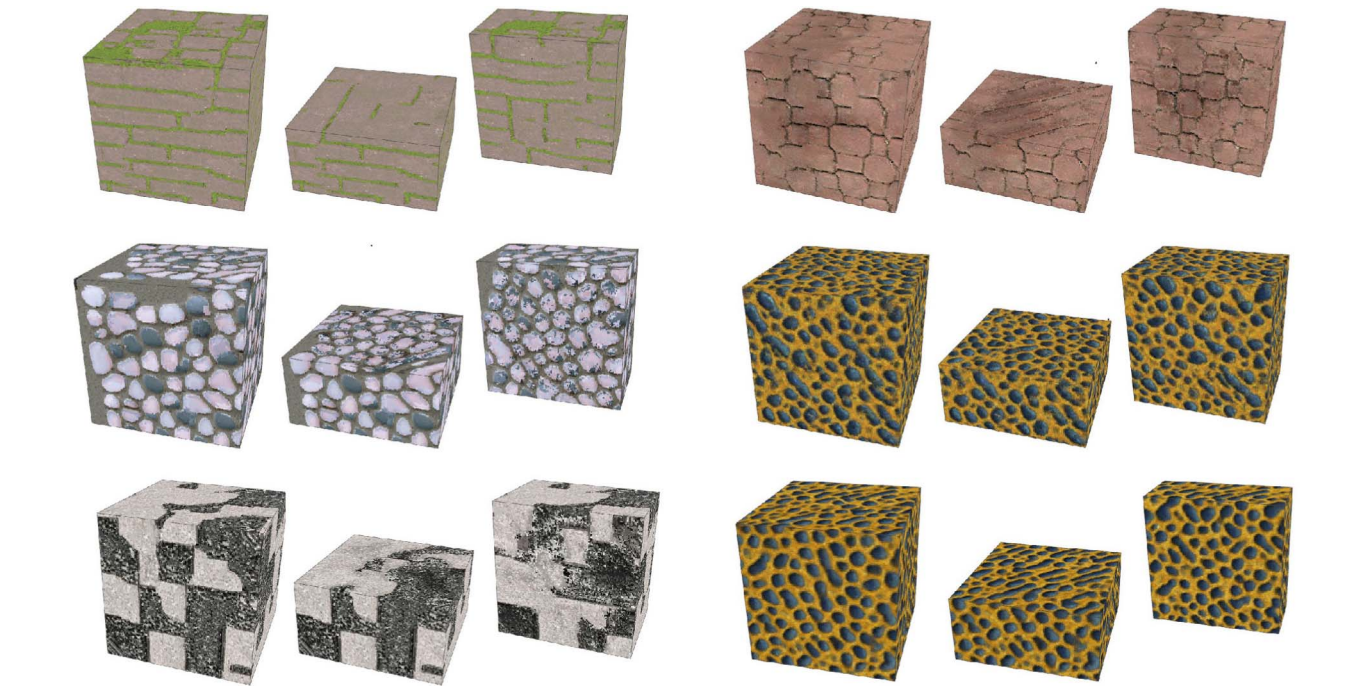

Fig. 13. Examples textured using the same 2D input data as in Fig. 12, but other texturing methods. The first five examples use the method in [4], while the example at the bottom right uses the method in [3]. Each example shows a textured cube, and two slices through it which reveal the internal texture structure.

Synthesizing a 3D volume from 2D exemplars always requires assumptions about missing information. Given a 2D image texture, people rely on their experience to imagine the corresponding volume. While there is no universally correct approach to generating solid textures from exemplars, we use intuitively reasonable methods to reconstruct the shapes of particles, and to determine their placement. As the placement of particles is based on *local* neighborhood information, future work might improve upon this by considering longer range, and even global structures. Higher levels of interdependence between different textures could also be employed to generate complex materials, composed of different objects made from different textures—such as a tree with wood inside and bark outside. We also hope to extend our method to handle other types of high-dimensional textures like bidirectional texture functions [35], and to combine our method with volumetric deformation techniques [36], [37] for data visualization, edit propagation schemes [38], [39] for supporting intuitive stroke-based inputs, and texture compression and filtering [40].

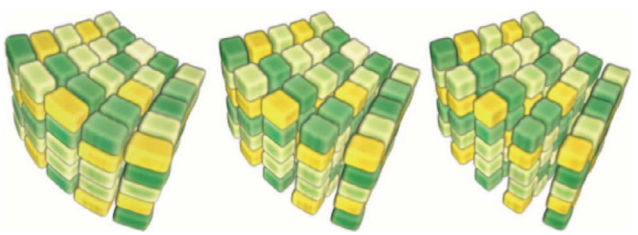

Fig. 14. Deformation: (left) particles and matrix equally rigid, (middle) and (right) weak and strong preservation of particle shapes. The matrix is omitted to show the interior structure.

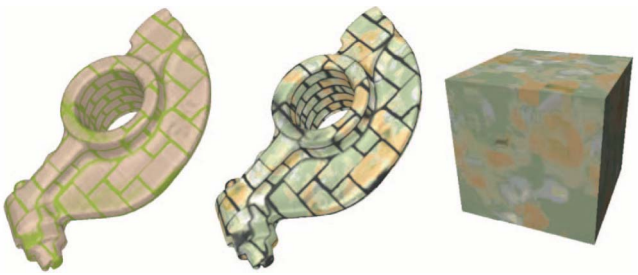

Fig. 15. Change particles' and matrix color: (left) original texture, (middle) particles' color is changed according to existing solid texture(-right), which is synthesized by Kopf et al. [3].

ACKNOWLEDGMENTS

This work was supported by the National Basic Research Project of China (Project Number 2012CB316400), the Natural Science Foundation of China (Project Numbers 61120106007, 60970100). Shi-Min Hu is the corresponding author.

Fig. 16. Failures. In each case, the three orthogonal input exemplars are copies of the same image on the left. Plausible particle shapes are generated, but placement fails to produce a satisfactory overall structure.

REFERENCES

[1] D.R. Peachey, "Solid Texturing of Complex Surfaces," *SIGGRAPH Computer Graphics,* vol. 19, pp. 279-286, 1985.

[2] K. Perlin, "An Image Synthesizer," *SIGGRAPH Computer Graphics,* vol. 19, pp. 287-296, 1985.

[3] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong, "Solid Texture Synthesis from 2D Exemplars," *ACM Trans. Graphics,* vol. 26, pp. 2:1-2:9, 2007.

[4] Y. Dong, S. Lefebvre, X. Tong, and G. Drettakis, "Lazy Solid Texture Synthesis," *Computer Graphics Forum,* vol. 27, pp. 1165-1174, 2008.

[5] R. Jagnow, J. Dorsey, and H. Rushmeier, "Stereological Techniques for Solid Textures," *ACM Trans. Graphics,* vol. 23, pp. 329-335, 2004.

[6] Y. Liu, W.-C. Lin, and J. Hays, "Near-Regular Texture Analysis and Manipulation," *ACM Trans. Graphics,* vol. 23, pp. 368-376, 2004.

[7] N. Pietroni, P. Cignoni, M. Otaduy, and R. Scopigno, "Solid-Texture Synthesis: A Survey," *IEEE Computer Graphics and Applications,* vol. 30, no. 4, pp. 74-89, July/Aug. 2010.

[8] S. Worley, "A Cellular Texture Basis Function," *Proc. 23rd Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 291-294, 1996.

[9] A. Lagae and G. Drettakis, "Filtering Solid Gabor Noise," *ACM Trans. Graphics,* vol. 30, pp. 51:1-51:6, 2011.

[10] P. Merrell and D. Manocha, "Model Synthesis: A General Procedural Modeling Algorithm," *IEEE Trans. Visualization and Computer Graphics,* vol. 17, no. 6, pp. 715-728, June 2011.

[11] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the Art in Example-Based Texture Synthesis," *Proc. Eurographics Conf.,* 2009.

[12] G. Djamchid and D. Jean-Michel, "Spectral Analysis for Automatic 3-D Texture Generation," *Computers and Graphics,* vol. 19, pp. 413-422, 1995.

[13] G. Djamchid and D. Jean-Michel, "Generation of 3D Texture Using Multiple 2D Models Analysis," *Computer Graphics Forum,* vol. 15, pp. 311-323, 1996.

[14] D.J. Heeger and J.R. Bergen, "Pyramid-Based Texture Analysis/Synthesis," *Proc. 22nd Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 229-238, 1995.

[15] L.-Y. Wei, "Texture Synthesis by Fixed Neighborhood Searching," PhD dissertation, Stanford, CA, USA, 2002.

[16] X. Qin and Y.-H. Yang, "Aura 3D Textures," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 2, pp. 379-389, Mar. 2007.

[17] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture Optimization for Example-Based Synthesis," *ACM Trans. Graphics,* vol. 24, pp. 795-802, 2005.

[18] S. Lefebvre and H. Hoppe, "Parallel Controllable Texture Synthesis," *ACM Trans. Graphics,* vol. 24, pp. 777-786, 2005.

[19] C. Ma, L.-Y. Wei, B. Guo, and K. Zhou, "Motion Field Texture Synthesis," *ACM Trans. Graphics,* vol. 28, pp. 110:1-110:8, 2009.

[20] G. Zhang, S. Du, Y. Lai, T. Ni, and S. Hu, "Sketch Guided Solid Texturing," *Graphics Models,* vol. 73, pp. 59-73, 2011.

[21] J. Zhang, K. Zhou, L. Velho, B. Guo, and H. Shum, "Synthesis of Progressively-Variant Textures on Arbitrary Surfaces," *ACM Trans. Graphics,* vol. 22, pp. 295-302, 2003.

[22] L. Wang, K. Zhou, Y. Yu, and B. Guo, "Vector Solid Textures," *ACM Trans. Graphics,* vol. 29, pp. 1-8, 2010.

[23] L. Wang, Y. Yu, K. Zhou, and B. Guo, "Multiscale Vector Volumes," *ACM Trans. Graphics,* vol. 30, pp. 167:1-167:8, 2011.

[24] C. Ma, L.-Y. Wei, and X. Tong, "Discrete Element Textures," *ACM Trans. Graphics,* vol. 30, pp. 62:1-62:10, 2011.

[25] B. Cutler, J. Dorsey, L. McMillan, M. Müller, and R. Jagnow, "A Procedural Approach to Authoring Solid Models," *ACM Trans. Graphics,* vol. 21, pp. 302-311, 2002.

[26] S. Owada, F. Nielsen, M. Okabe, and T. Igarashi, "Volumetric Illustration: Designing 3D Models with Internal Textures," *ACM Trans. Graphics,* vol. 23, pp. 322-328, 2004.

[27] N. Pietroni, M.A. Otaduy, B. Bickel, F. Ganovelli, and M. Gross, "Texturing Internal Surfaces from a Few Cross Sections," *Computer Graphics Forum,* vol. 26, pp. 637-644, 2007.

[28] K. Takayama, M. Okabe, T. Ijiri, and T. Igarashi, "Lapped Solid Textures: Filling a Model with Anisotropic Textures," *ACM Trans. Graphics,* vol. 27, pp. 1-9, 2008.

[29] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped Textures," *Proc. 27th Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 465-470, 2000.

[30] K. Takayama, O. Sorkine, A. Nealen, and T. Igarashi, "Volumetric Modeling with Diffusion Surfaces," *ACM Trans. Graphics,* vol. 29, pp. 180:1-180:8, 2010.

[31] J.-M. Dischler and D. Ghazanfarpour, "Interactive Image-Based Modeling of Macrostructured Textures," *IEEE Computer Graphics Applications,* vol. 19, no. 1, pp. 66-74, Jan./Feb. 1999.

[32] R. Jagnow, J. Dorsey, and H. Rushmeier, "Evaluation of Methods for Approximating Shapes Used to Synthesize 3D Solid Textures," *ACM Trans. Applied Perception,* vol. 4, pp. 5:1-5:27, 2008.

[33] M. Alexa, D. Cohen-Or, and D. Levin, "As-Rigid-As-Possible Shape Interpolation," *Proc. 27th Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 157-164, 2000.

[34] J.-M. Dischler, K. Maritaud, B. Levy, and D. Ghazanfarpour, "Texture Particles," *Computer Graphics Forum,* vol. 21, pp. 401-410, 2002.

[35] K. Xu, J. Wang, X. Tong, S. Hu, and B. Guo, "Edit Propagation on Bidirectional Texture Functions," *Computer Graphics Forum,* vol. 21, pp. 401-410, 2002.

[36] Y. Wang, C. Wang, T. Lee, and K. Ma, "Feature-Preserving Volume Data Reduction and Focus+Context Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 17, no. 2, pp. 171-181, Feb. 2011.

[37] X. Zhao, B. Li, L. Wang, and A. Kaufman, "Texture-Guided Volumetric Deformation and Visualization Using 3D Moving Least Squares," *The Visual Computer,* vol. 28, pp. 193-204, 2012.

[38] X. An and F. Pellacini, "AppProp: All-Pairs Appearance-Space Edit Propagation," *ACM Trans. Graphics,* vol. 27, pp. 40:1-40:9, 2008.

[39] K. Xu, Y. Li, T. Ju, S. Hu, and T. Liu, "Efficient Affinity-Based Edit Propagation Using K-D Tree," *ACM Trans. Graphics,* vol. 28, pp. 118:1-118:6, 2009.

[40] C.-F. Hollemeersch, B. Pieters, P. Lambert, and R. Van de Walle, "A New Approach to Combine Texture Compression and Filtering," *The Visual Computer,* vol. 28, pp. 371-385, 2012.

Song-Pei Du received the BS degree in computer science from Tsinghua University in 2009. He is currently working toward the PhD degree in the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics, geometric modeling and image processing.

Shi-Min Hu received the PhD degree from Zhejiang University in 1996. He is currently a professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is associate editor-in-chief of *The Visual Computer* (Springer), and on the editorial boards of *Computer-Aided Design and Computer & Graphics* (Elsevier). He is a member of the IEEE and the ACM.

Ralph R. Martin received the PhD degree in 1983 from Cambridge University. He is currently a professor at Cardiff University. He has published more than 200 papers and 12 books, covering such topics as solid and surface modeling, intelligent sketch input, geometric reasoning, reverse engineering, and various aspects of computer graphics. He is a fellow of the Learned Society of Wales, the Institute of Mathematics and its Applications, and the British Computer Society. He is on the editorial boards of *Computer Aided Design*, *Computer Aided Geometric Design*, *Geometric Models*, *the International Journal of Shape Modeling*, *CAD and Applications*, and the*International Journal of CADCAM*.