Texture Synthesis on Subdivision Surfaces

Weiyin Ma[1] and Zhuangzhi Wu[1,2]
[1]City University of Hong Kong
[2]On leave from Beijing University of Aeronautics and Astronautics
mewma@cityu.edu.hk, mezzwu@cityu.edu.hk

ABSTRACT

This paper presents an approach for seamless texture synthesis on subdivision surfaces. Based on an input sample texture and a preferred scale, the initial control mesh of the subdivision surface is first subdivided to a resolution that is appropriate for further processing. The texture is then applied to the polygonal model of the control mesh through texture synthesis. Image processing algorithms are further applied to smoothly connect textures of neighboring faces. The entire seamlessly connected surface texture is finally mapped to the limit surface using a subset of rules for geometry subdivision. Several examples are provided to demonstrate the proposed approach using Loop subdivision surfaces.

Keywords: Texture synthesis, texture mapping, subdivision surfaces, Loop subdivision.

1. INTRODUCTION

For many design and graphics applications, such as digital mockup for product design and digital sculpture modeling for art work representation, one often uses surface textures to enhance visual effect of the created virtual model. In literature, one may find two general classes of algorithms for texture synthesis, i.e. procedural methods [8, 18] and texture from sample methods [4, 14, 19, 21]. Procedural methods can be very efficient in memory utilization and texture rendering. However, existing procedural-based methods can only process a specific class of materials such as wood and marble. Texture from sample methods can synthesize a wide variety of textures, as long as sample textures are provided. The method discussed in this paper falls in the class of texture synthesis methods from sample textures. While most of the reported approaches work with polygonal models. This paper presents an alternative approach for texture synthesis on subdivision surfaces based on a known sample texture.

Subdivision-based modelling was first introduced to the CAD and graphics community in early 1980's for defining free-form curves and surfaces starting from an initial control mesh through recursive refinement [1, 3]. In the limit, it produces a smooth curve or surface. Today, one may find rich families of subdivision surfaces widely used in modelling and graphics applications [17, 22]. Given a 2D sample texture and a 3D subdivision surface, we want to cover the entire surface with the sample texture in a seamless fashion. In literature, one may find various approaches and excellent results have been introduced recently on texture synthesis. All existing algorithms can be classified into two general classes according to the target object on which the sample texture will be synthesized, namely texture synthesis on a 2D rectangular domain [4, 5, 10, 12, 20] or on a 3D surface [14, 16, 19, 21]. Most of the algorithms reported so far are for texture synthesis on a 2D rectangular domain. Among the algorithms for texture synthesis on 3D surfaces, most of the algorithms are extended from the counter part for 2D rectangular domains. One may also classify existing texture synthesis algorithms according to the nature of the algorithms, such as pixel-based texture synthesis [5, 19, 20, 21], patch based texture synthesis [4, 10, 12, 14, 16], or hybrid algorithms. Pixel-based algorithms synthesize the source sample texture to the target object pixel by pixel, while patch-based algorithms synthesize the source sample texture to the target object area by area.

Pixel-based algorithms synthesize the target texture line by line and pixel by pixel for each line based on the source sample texture. For each pixel to be synthesized, a neighborhood window of N pixels with fixed topology around the target pixel is used for comparison with the source sample texture. The window will move within the source sample texture and the best position is identified where the difference between the neighborhood window of the synthesized target texture and that of the source texture is minimized. The corresponding parameters of the pixel in the source texture are then used as that for the pixel to be synthesized. The difference between the target window and the moving source window can be defined as the sum of differences between the corresponding pixels of the two windows. The difference between the corresponding pixels can be evaluated as the distance between the two pixels in RGB

coordinates with certain norm. The quality of the synthesis results will usually depend on the size N of the window, the size of typical features of the sample texture being synthesized, and the type of sample texture itself. The window size should usually be larger than the features of the source texture to be synthesized and is often varies from 5×5 pixels upwards. The larger the size of the window, the better the synthesis result, but the computation time would substantially increase with the increase of the size of the evaluation window. Pixel-based algorithms for texture synthesis work well for stochastic textures and may also produce satisfactory results for some structured textures. The computation time of these algorithms is often slow. The final synthesized texture is usually blurred compared with the source texture. Typical examples can be found in [5, 20] for texture synthesis on 2D rectangular patches and in [19, 21] on 3D surfaces.

Patch-based algorithms synthesize textures area by area. During the synthesis procedure, a patch of the original source texture is selected and pasted onto the target object. The process is continued until the entire surface of the target object is covered. The selection of a particular patch from the source texture can be based on optimal pattern matching for structured textures, or just randomly for stochastic textures. The patch size depends on the size of the source texture and can be as larger as possible. To remove visual discontinuities, various techniques are introduced for smoothly bridging the textures along patch boundaries. Depending on the type of the target object, patch-based algorithms can also be further classified into two classes for 2D rectangular domains [4, 10, 12] or for 3D surfaces [14, 16].Compared with pixel-based approaches for texture synthesis, patch-based approaches usually preserve the original structure of the source sample texture and the processing speed is extremely fast.

Among various patch-based methods reported so far for texture synthesis, Efros and Freeman [14] reported an approach called image quilting for patch-based texture synthesis on rectangular domains. Individual regular square patches are first randomly extracted from the source sample texture and pasted onto the target domain with boundary overlap. A minimum-error-boundary-cut is then performed within the overlap region such that the two neighboring patches are connected smoothly with minimum error along the cutting boundary curve. Kwartra et al. [10] presented another approach for patch-based texture synthesis on rectangular domains. The approach is a further generalization of the image quilting method using arbitrarily shaped patches and it is called graphcut textures. Once the texture is initialized, the algorithm finds new patch locations so as to refine the texture. Both the patch shape and size are determined based on a minimum cost graph cut method. Liang et al. [12] on the other hand presented a patch-based method for texture synthesis on rectangular domains based on patch pasting and boundary blending. During the process for patch pasting, an appropriate patch is selected from the source sample texture and pasted onto the target domain with pattern matching. The boundary edges are further smoothed with linear blending that assures smooth transition between neighboring patches after synthesis. For patch-based texture synthesis on 3D surfaces, Praun et al. [16] presented lapped textures on 3D surfaces. With this approach, a user specifies a tangential vector field over the surface for controlling the texture orientation. The input sample texture is then repeatedly pasted onto the surface by flattening a local area of the input 3D mesh and parameterizes the patch in the texture space. In theory, all patch-based texture synthesis algorithms for 2D rectangular domains can be extended for 3D surfaces. Magda and Kriegman [14], e.g., presented an algorithm similar to the 2D version of image quilting [5], but the shape of a patch in case of 3D surfaces is a triangle patch.

In literature, one may also find a few publications on texture mapping on subdivision surfaces [2, 15]. The method of [2] involved texture creation on the initial control meshes and further texture mapping onto subdivision surfaces. The initial texture on the control mesh was created using a combination of techniques such as 3D painting, solid textures and procedural textures with scalar fields. The created texture on control mesh is further mapped onto the subdivision surface through subdivision with texture coordinates. The method of [15] on the other hand relies on a global parametrization of the entire control mesh in the form of a pelt with seam texture blended along cutting boundaries, the so called model pelting and texture blending. The approach presented in this paper is a kind of patch-based approach for texture synthesis on 3D surfaces from a sample texture. Based on the input sample texture and a preferred scale, the initial control mesh of the subdivision surface is first subdivided to a resolution that is appropriate for patch-based texture synthesis. The synthesized texture is then applied to the polygonal model of the control mesh face by face. Image processing algorithms are further applied to smoothly connect textures of neighboring faces. The entire seamlessly connected surface texture can then be mapped to any fine resolution of the subdivision surface or to the limit surface using a subset of rules for geometry subdivision similar to that of [2]. Compared with existing approaches, the proposed method exhibits several advantages. Since the algorithm works with a coarse control mesh and maps the texture to fine resolution meshes while performing subdivision operations, it is extremely memory efficient and the rendering process is pretty fast. There is no need either to preprocess the sample texture. Apart from the selection of the input parameters, the entire synthesis process is fully automatic and produces quality textures with fast synthesis speed.

2. OVERVIEW OF THE SYNTHESIS METHOD

Based on the dimension of the given texture and a preferred scale, we first subdivide the initial control mesh of the subdivision surface to a resolution that is appropriate for the synthesis procedure. A patch-based texture synthesis method is then applied to synthesize a piece of texture for every face of the polygonal model. To further improve the texture smoothness across patch boundaries, all textures are blended across the patch boundaries. Synthesized textures after smooth blending are stored in one or a set of texture atlas for further processing. The polygonal model can then be processed together with the texture atlas. One can map the synthesized texture to any refined meshes or to the limit surface at any resolutions using a subset of rules for geometry subdivision. Fig. 1 summarizes the entire procedure for texture synthesis on Loop subdivision surfaces.

- Fig. 1(a) illustrates a sample texture to be used for texture synthesis.
- Fig. 1(b) illustrates an initial control mesh M_0 of a subdivision surface.
- The initial control mesh is refined to resolution M_i shown in Fig. 1(c) that is appropriate for texture synthesis with the given sample texture and a texture synthesis scale.
- Fig. 1(d) shows the control mesh M_i with synthesized texture after texture synthesis using a patch-based approach. During the procedure for texture synthesis, an atlas *A* is created for storing the synthesized textures.
- If necessary, one may obtain a refined mesh M_j shown in Fig. 1(e) by subdividing M_i (*j*-*i*) times with the texture atlas *A*.
- One may also directly map the texture for any resolution to the limit surface mesh as shown in Fig. 1(f) for $\tilde{M}_i$ and Fig. 1(g) for $\tilde{M}_j$.

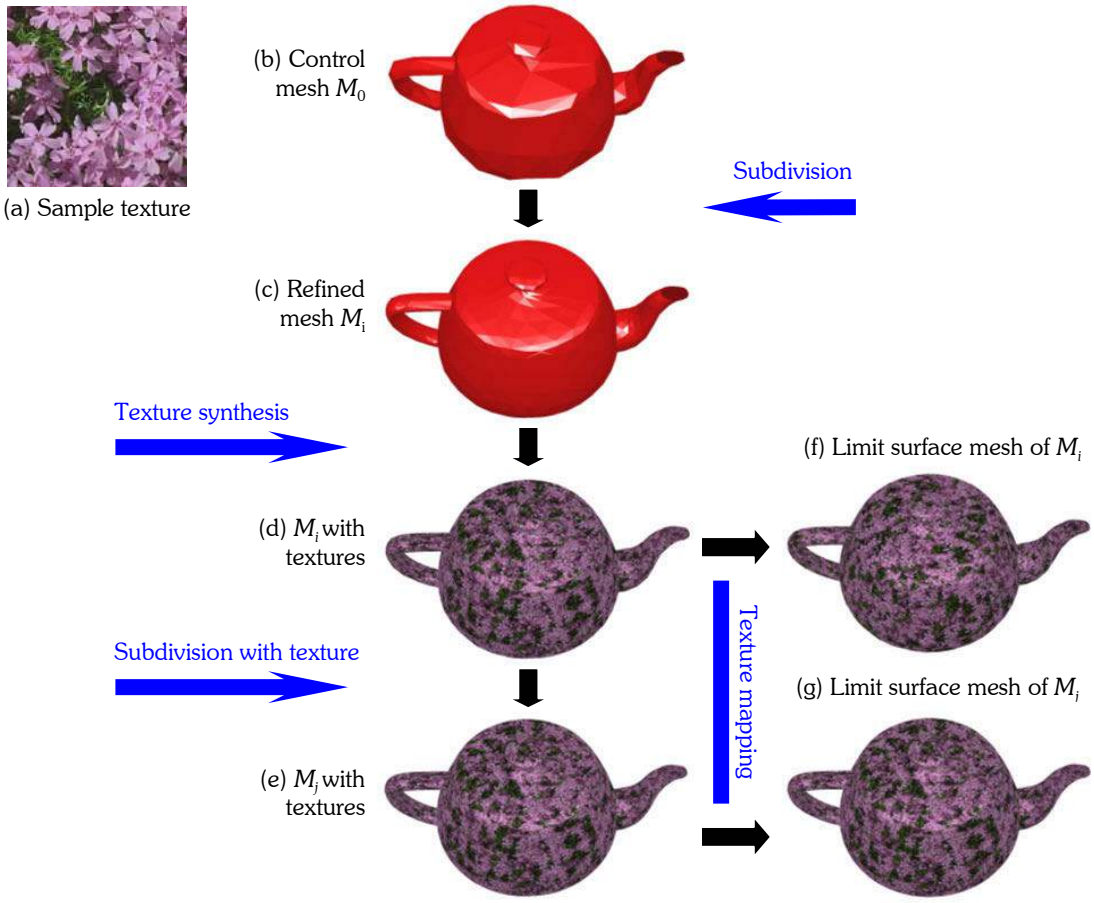

Fig. 1. Texture synthesis on Loop subdivision surfaces

3. KEY ALGORITHMS FOR TEXTURE SYNTHESIS

The general procedure for texture synthesis on subdivision surfaces is summarized as the following algorithm. For every face *F*, the algorithm first parameterizes the face in the sample texture space and assigns a texture *T* with corresponding texture coordinates. An edge-blending algorithm is then applied for every face *F*. The synthesized texture after edge blending is stored in a texture atlas for further processing.

- Pre-processing: Refine the initial control mesh M_0 to an appropriate resolution M_i for texture synthesis. Specify a preferred texture orientation and a texture scale *f*.
- Local parameterization: For each face $F \in M_i$, parameterize *F* (in 3D space) to F' (in 2D texture space) according to local face orientation and texture scale *f*. After local parameterization, assign the selected texture patch *T* to *F*.
- Edge blending: For each face, blend the boundary of F' in texture domain *T* for obtaining visually smooth effect.
- Texture atlas creation: All smoothly blended textures for individual faces will be stored in one or a set of texture atlas for further processing.

For global control of the texture orientation on individual faces, we use a vector field commonly used for texture synthesis [5, 16, 19, 21]. Each face of the control mesh is assigned a vector *V* that is defined as the projection of the corresponding direction of a vector field at that position onto the corresponding face, also called face orientation. For the present implementation, a constant vector field is used. The constant direction field can be entered interactively before synthesizing the textures of all faces. Fig. 2 illustrates two examples with assigned vector for each individual faces for texture orientation.

When applying the texture, one might wish to scale the texture during the procedure for texture synthesis. For the moment, a single scale parameter *f* is used and the scale is directly applied to related faces during local parameterization. After local parameterization, the corresponding faces in 3D space and in the 2D texture space are similar with a scale factor *f*. The selection of scale *f* is restricted by the size of the sample texture and the size of individual faces in 3D space.

Both the face orientation and texture scale will affect the parameterization of the corresponding face onto the texture space. The texture will be aligned along the face orientation and scaled when parameterizing a face onto the texture space. The actual position for texture parameterization can be optimized, but one may simply select a random position for stochastic textures. After parameterization, every face *F* in M_i should have a texture *T* which is a portion of the original sample texture. In general, the texture of adjacent faces at this moment may not be smooth across face/patch boundaries. Pixels near patch boundaries are further blended to remove possible visible seams along boundary edges. The blending process can be straight forward if multi-textures are used when rendering individual faces [14]. For subdivision surfaces, as it is necessary to process the texture when performing further subdivision operations, we directly modify the texture using a blending function. After edge blending, every face has its own texture and a texture atlas [7, 11] is created with all individual face textures for dynamic rendering.

The following section provides further details regarding local parameterization, edge blending, and the set up of a texture atlas for further processing.

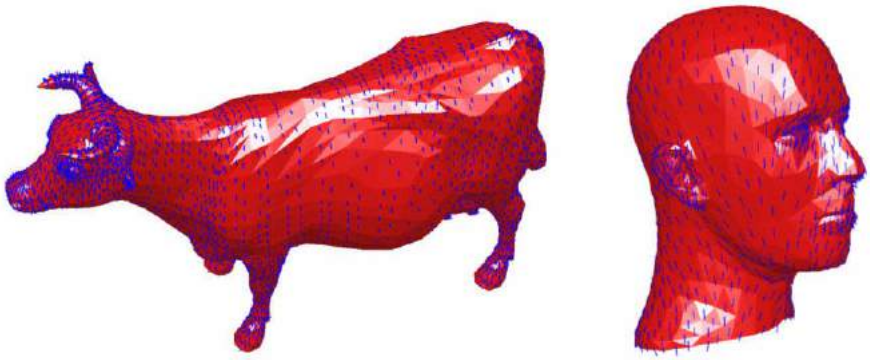

Fig. 2. Face orientation for texture alignment

3.1 Local parameterization

In order to synthesize a texture for every face *F*, we need to parameterize *F* to the 2D texture space. Ideally, the mapping between the triangulated surface and the planar triangulation should be isometric, preserving both angles and distances. As a texture scale parameter *f* and a texture orientation of F have been assigned beforehand, the following procedure can be used for parameterizing *F* (see Fig. 3):

Assumption: Give an arbitrary (triangle) face $\Delta(q_1,q_2,q_3)$ in 3D space and let e_1, e_2 and e_3 be the edge lengths of the triangle. Let $F=\Delta(p_1,p_2,p_3)$ be the corresponding triangle in texture space with texture coordinates $p_i=(s_i,t_i)$, for $i=1,2$ and 3.

1. Parameterize *F*
   (1) Assign $p_1=(0,0)$ and $p_2=\left(e_3/f,0\right)$.
   (2) Assign $p_3=(s_3,t_3)$ such that $\Delta(p_1,p_2,p_3)$ is similar to $\Delta(q_1,q_2,q_3)$.
   (3) Based on point p_1, rotate $\Delta(p_1,p_2,p_3)$ with an angle α clockwise, where α is the angle between the vectors $\langle q_1,q_2\rangle$ and the projected field vector *V* counter-clockwise.
2. Assign a texture memory *T* to *F*
   (1) Extract the bounding box *T* of the triangle $\Delta(p_1,p_2,p_3)$.
   (2) Move the bounding box *T* with $\Delta(p_1,p_2,p_3)$ such that its lower left corner is in coincident with the origin of the texture space.
   (3) Assign memory to *T*.

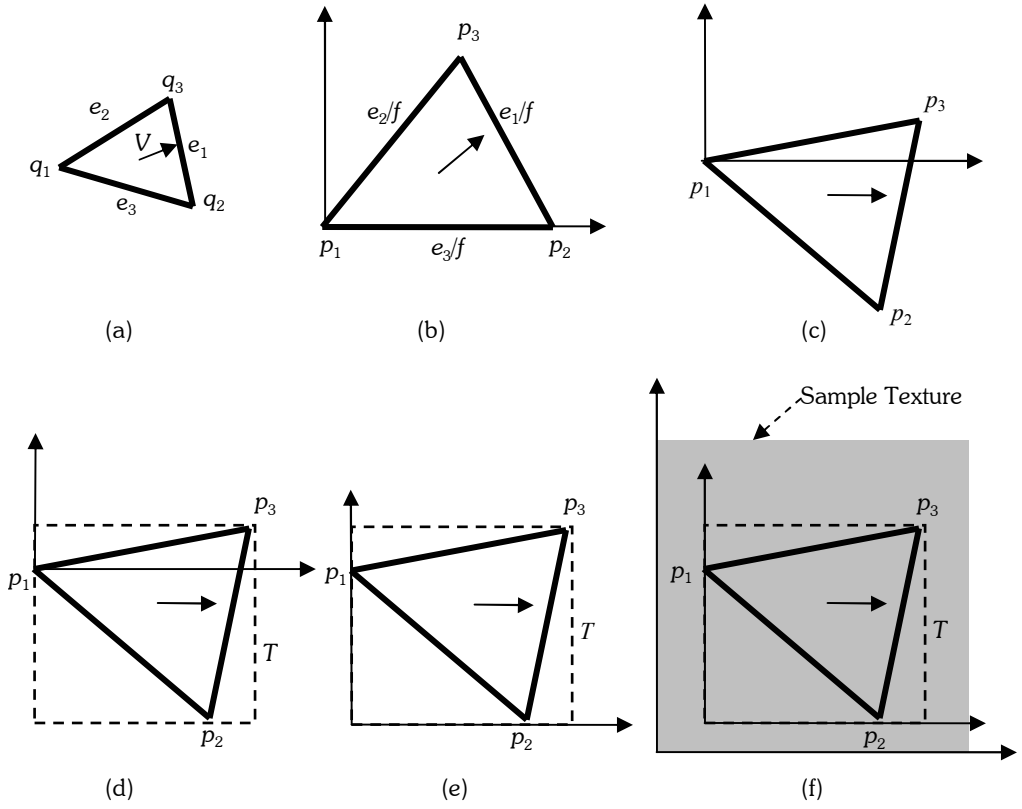

Fig. 3. The procedure of the parameterization of *F*

After local parameterization, each face *F* has a corresponding texture memory *T* located at the lower left corner of the sample texture. The texture *T* will then be repositioned within the sample texture such that it would best match the texture of neighboring triangles that have been allocated earlier. In [14], finding a good match normally involves comparing some boundary region of the candidate texture *T* to the boundary region of the neighbors already textured. This is generally a time-consuming processing. Considering the type of texture to be synthesized and the next step for edge blending, we randomly set a place in the sample texture space for the texture *T* for every face *F* (see Fig. 3f), and then copy the texture patch to the texture memory *T*. This produces comparable quality results while eliminates the expensive process for texture comparison.

3.4 Edge Blending

After local parameterization, every face *F* has a texture *T* which is a portion of the original sample texture. In general, the textures of adjacent faces have edge discontinuities. To eliminate such visual artifacts, the pixels near the boundary edges should be blended. In [14], multi-texture was used when rendering each face. The blending process is straightforward and there is no need to pre-process the texture beforehand. In our case, as it is necessary to subdivide the mesh with texture in subsequent operations, an alternative approach is used to blend the texture off-line and store them in a texture atlas for further processing.

We consider an arbitrary face $F=\Delta(A,B,C)$ in 3D space and denote its texture patch as $F'=\Delta(A',B',C')$ in the sample texture space. For each edge of F', such as $B'C'$, we need to blend its adjacent neighborhood 3 to 7 pixels (blue blending area). The blending process is as follows (See Fig. 4 for an illustration):

1. Given $F=\Delta(A,B,C)$ in 3D and the corresponding texture patch $F'=\Delta(A',B',C')$.
2. Find adjacent face F_1 of F sharing the same edge BC and get the corresponding texture patch F_1' of F_1.
3. Determine F'' being the extension of F_1' in the parametric space and determine F_1'' being the extension of F' in the parametric space such that F' and F'' are similar triangles and so do F_1' and F_1''.
4. Modify the boundary texture of face $F'=\Delta(A',B',C')$ along boundary $B'C'$ 3 to 7 pixels by blending the textures of F' and F''. In a similar way, the textures of F_1' should also be modified along boundary $B'C'$ 3 to 7 pixels by blending the textures of F_1' and F_1''.

The blending for boundary texture modification is defined as a linear blending across boundary $B'C'$ such that the texture will be smoothly connected.

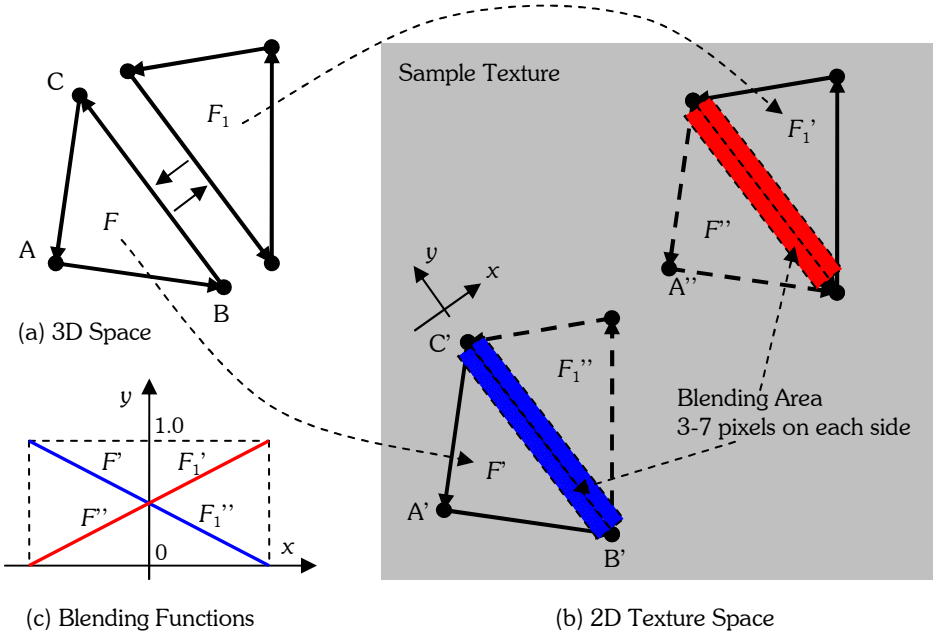

Fig. 4. The blending process across an edge

3.5 Atlas Generation

After edge blending, each face has its own rectangular texture, i.e. the minimum bounding box of triangular texture. In order to store the texture for further processing, such as for subdivision and texture mapping to the limit surface, we create a single texture bitmap called a texture atlas. To determine the size of the atlas, we use a method described in [7]. Let $R^2 = \sum_{\text{All } i} S_i$ be the total area of all the bitmaps of individual faces and let L=1.2*R, the size of the atlas is then defined as L*L. To pack all individual textures in the atlas, the algorithm further sorts all bitmaps by their height, and lines them up horizontally. We then fold the line to fit the atlas within the L*L area. As the rendering function glTexImage2D() of OpenGL requires that the width and height parameters of the texture should not be greater than 2048. We may need to use multiple 2048*2048 atlas in order to store all the textures for OpenGL rendering depending on the size of the target object for texture synthesis.

3.6 Mesh Refinement with Texture

After texture synthesis on the control mesh, one may map the synthesized texture to a refined mesh or a limit surface mesh using a method for texture mapping on subdivision surfaces reported in [2]. The method extends the coordinates [x, y, z] of vertices in 3D space to 5D space with coordinates [x, y, z, s, t], where [s, t] are the texture coordinates of the corresponding vertices. The texture on a coarse mesh can then be carried on to refined meshes by subdividing the mesh with texture coordinates in 5D with the same subdivision rule. For mapping the texture to any level of limit surfaces, one may first subdivide with texture coordinates to that level and then simply use the texture coordinates of the refined control mesh as that for the limit surface mesh.

4. RESULTS AND DISCUSSIONS

The proposed texture synthesis method works well for a variety of textures. Fig. 5 illustrates some sample textures used for testing the synthesis method [4]. These textures include isotropic texture patterns (Fig. 5(e)-(h)), anisotropic textures (Fig. 5(b)-(d)), and highly structured textures (Fig. 5(a)-(d)). Fig. 6 illustrates some examples in synthesizing highly structured and anisotropic textures on subdivision surfaces. Fig. 7 illustrates some other examples in synthesizing isotropic texture patterns on subdivision surfaces. As the entire texture synthesis approach involves only direct texture parametrization and linear edge blending, it is extremely fast. For all the presented examples, the CPU time is within one or a few seconds for processing and it is then on the fly for dynamic rendering. Fig. 8 illustrates an example using a bunny model for texture scale control. The scale used for the example of Fig. 8 are f=0.01, 0.005 and 0.00317 for (a), (b), and (c) respectively. In theory, one may apply an arbitrary scale for texture synthesis. Note that the actual selection of the scale depends on the size of the 2D texture, internal features and the size of the 3D model. Combined with methods for texture synthesis on 2D rectangular domains for texture extension and mesh subdivision before applying the texture synthesis algorithm, one may achieve arbitrary scale control for texture synthesis on subdivision surfaces.

The texture synthesis method presented in this paper can be directly applied to face splitting subdivision schemes, such as Loop and Catmull-Clark surfaces [1, 13]. For triangle control meshes, the synthesis process is straight forward. For other type of meshes, one may first triangulate the faces for initial parameterization. When the atlas is ready, the step for texture mapping is the same for any kind of meshes. For all the examples shown in Figs. 6-8, Loop subdivision surfaces are used. For non-face-splitting subdivision schemes, such as Doo-Sabin, $\sqrt{2}$ and $\sqrt{3}$ subdivision surfaces [3, 9, 17, 22], the initial parameterization procedure is the same, but image processing techniques need to be used for texture decomposition and recomposition when performing subdivision with synthesized texture coordinates.

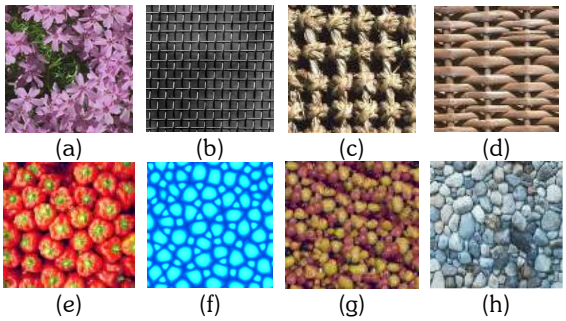

Fig. 5. Sample textures: (a)-(d) highly structured texture patterns; (b)-(d) anisotropic texture patterns; and (e)-(h) isotropic texture patterns

Fig. 6. Texture synthesis on subdivision surfaces with highly structured textures and anisotropic textures.

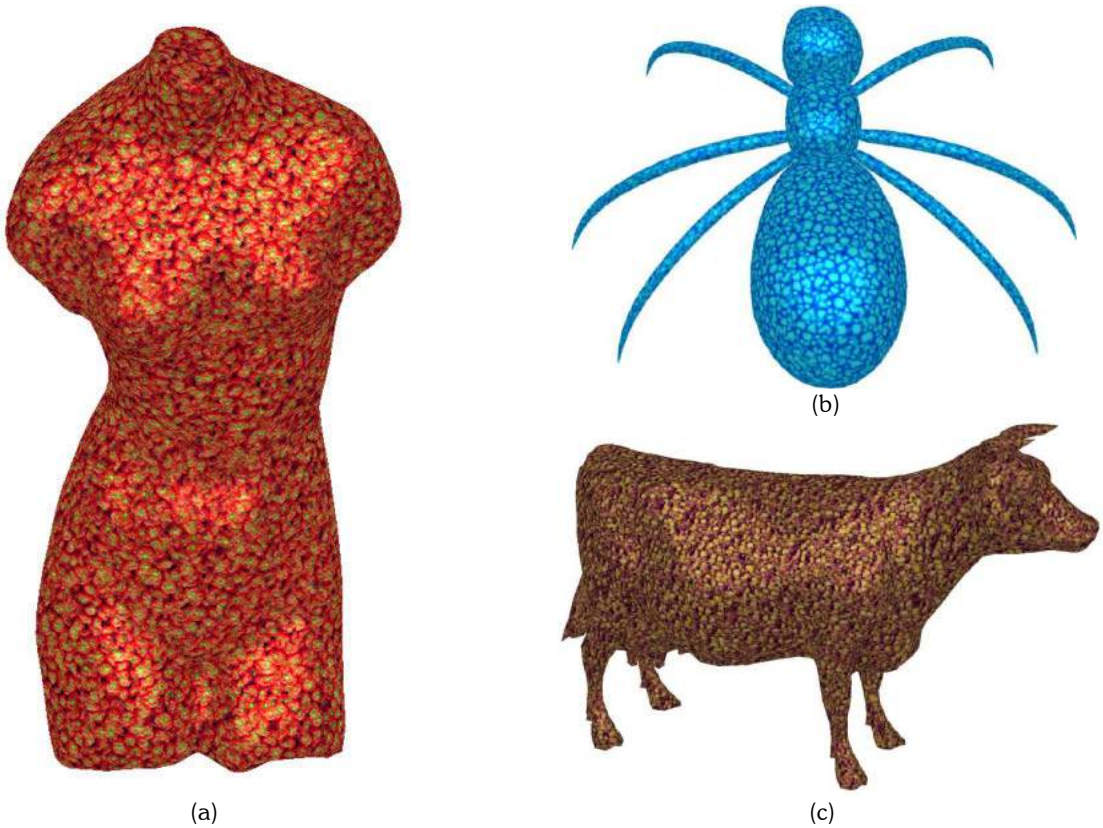

Fig. 7. Texture syntheses on subdivision surfaces with isotropic textures

(a) (b) (c)

Fig. 8. Scale control for texture synthesis

5. CONCLUSIONS

This paper proposes a general approach for texture synthesis on subdivision surfaces. The approach is based on techniques for texture synthesis on 3D mesh models and texture mapping to refined meshes using the same rule for geometry subdivision. The approach used in the paper falls in the class of patch-based texture synthesis on 3D surfaces and it is realized through direct image parametrization and linear edge blending, hence it is extremely fast. With the proposed method, one can work on a mesh with appropriate resolution for texture synthesis and there is no need to handle meshes with a large size at fine resolutions. By using a mesh with proper resolution and combined with texture synthesis on 2D rectangular domains, one can use an arbitrary scale for texture synthesis on subdivision surfaces. The method can be applied to most of the commonly used subdivision schemes. While all examples are produced using Loop subdivision surfaces with a triangle control mesh, the approach can be directly applied or extended for texturing other subdivision schemes. In case of an arbitrary control polyhedron, the mesh can be temporarily triangulated for texture parametrization, all other steps are then the same after initial parametrization. When performing subdivision with synthesized texture, the approach is straight forward for face-splitting schemes, such as Loop and Catmull-Clark. For non-face-splitting schemes, such as Doo-sabin, $\sqrt{2}$ and $\sqrt{3}$ subdivision surfaces, image processing techniques can be used for texture recomposition. In theory, most existing methods for texture synthesis may be extended for texturing subdivision surfaces.

6. ACKNOWLEDGEMENTS

The work presented in this paper was supported by City University of Hong Kong through a Strategic research Grant (7001296) and by the Research Grants Council of Hong Kong through a CERG grant (CityU 1131/03E). Special thanks are extended to the original owners of the textures and the mesh models for making them available on the Internet. The teapot model, the cow model, the Venus model and the ant model are downloaded from ftp://ftp.research.microsoft.com/users/hhoppe/data/thesis/, http://graphics.cs.uiuc.edu/~garland/, http://www.mrl.nyu.edu/~biermann/subdivision/ and http://www.3dcafe.com/asp/meshes.asp, respectively.

7. REFERENCES

[1] Catmull, E., and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6), 1978, pp. 350–355.

[2] DeRose, T., M. Kass and T. Truong. Subdivision surfaces in character animation. *ACM SIGGRAPH on Computer Graphics*, 1998, pp. 85-94.

[3] Doo, D., and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6), 1978, pp. 356–360.

[4] Efros, A. and W. T. Freeman. Image quilting for texture synthesis and transfer. *ACM SIGGRAPH on Computer Graphics*, 2001, pp. 341-346.

[5] Efros, A. and T. Leung. Texture synthesis by non-parametric sampling. *Proc. IEEE International Conference Computer Vision*, Corfu, Greece, September 1999, pp. 1033-1038.

[6] Gorla, G., V. Interrante and G. Sapiro. Texture synthesis for 3D shape representation. *IEEE Transactions on Visualization and Computer Graphics*, 9(4), 2003, pp. 512-524.

[7] Igarashi, T. and D. Cosgrove. Adaptive unwrapping for interactive texture painting. *ACM Symposium on Interactive 3D Graphics*. 2001.

[8] Jagnow, R., J. Dorsey and H. Rushmeier. Stereological techniques for solid textures, *ACM SIGGRAPH on Computer Graphics*, 2004, pp. 329-335.

[9] Kobbelt, L., $\sqrt{3}$ -subdivision, *ACM SIGGRAPH on Computer Graphics*, 2000, pp. 103-112.

[10] Kwatra, K., A. Schödl, 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.

[11] Levy, B., S. Petitjean, N. Ray and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3), pp. 362-371.

[12] L. Liang, C. Liu, Y. Xu, B. Guo and H.-Y. Shum. *Real-Time Texture Synthesis By Patch-Based Sampling*. MAR-TR-2001-40, Technical Report, March 2001.

[13] Loop, C. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis, University of Utah, Department of Mathematics, 1987.

[14] Magda, S. and D. Kriegman. Fast texture synthesis on arbitrary meshes. *Proceedings of the 14th Eurographics workshop on Rendering*, Leuven, Belgium, 2003, pp. 82-89.

[15] Piponi, D. and G. Borshukov. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *ACM SIGGRAPH on Computer Graphics*, 2000. pp. 471-478.

[16] Praun, E., A. Finkelstein and H. Hoppe. Lapped textures. *ACM SIGGRAPH on Computer Graphics*, 2000, pp. 465-470.

[17] Sabin, M. A., Recent progress in subdivision – a survey. In N. Dodgson, M.S. Floaterand M. Sabin (eds.), *Advances in Multiresolution for Geometric Modelling*, Springer, 2004.

[18] Turk, G. Genereating textures on arbitrary surfaces using reaction-diffusion. *ACM SIGGRAPH on Computer Graphics*, 1991, Vol. 25, pp. 289–298.

[19] Turk, G. Texture synthesis on surfaces. *ACM SIGGRAPH on Computer Graphics*, 2001, pp. 347-354.

[20] Wei, L.-Y. and M. Levoy. Fast texture synthesis using tree-structured vector quantization. *ACM SIGGRAPH on Computer Graphics*, pp 479-488.

[21] Wei, L.-Y. and M. Levoy. Texture Synthesis over Arbitrary Manifold Surfaces. *ACM SIGGRAPH on Computer Graphics*, 2001, pp 355-360.

[22] Zorin, D., P. Schröder. Subdivision for Modeling and Animation, ACM SIGGRAPH Course Notes, 2000.