Method of Direct Texture Synthesis on Arbitrary Surfaces

Fu-Li Wu, Chun-Hui Mei, and Jiao-Ying Shi

State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, P.R. China
E-mail: {flwu, chmei, jyshi}@cad.zju.edu.cn

Received May 6, 2003; revised March 2, 2004.

Abstract A direct texture synthesis method on arbitrary surfaces is proposed in this paper. The idea is to recursively map triangles on surface to texture space until the surface is completely mapped. First, the surface is simplified and a tangential vector field is created over the simplified mesh. Then, mapping process searches for the most optimal texture coordinates in texture sample for each triangle, and the textures of neighboring triangles are blended on the mesh. All synthesized texture triangles are compressed to an atlas. Finally, the simplified mesh is subdivided to approach the initial surface. The algorithm has several advantages over former methods: it synthesizes texture on surface without local parameterization; it does not need partitioning surface to patches; and it does not need a particular texture sample. The results demonstrate that the new algorithm is applicable to a wide variety of texture samples and any triangulated surfaces.

Keywords texture synthesis, texture mapping, mesh simplification, computer graphics

1 Introduction

Computer graphics applications often use texture to present the surface detail on coarse geometry model. Textures are extensively used to increase the realism of surfaces. In the past few years, texture synthesis algorithms\[1-10\] have achieved success in generating a synthesized texture on a plane that looks like the original 2D texture sample. However, these methods are difficult to be extended to cover arbitrary 3D surfaces with texture sample, because arbitrary surfaces do not have a smooth global parameterization on plane. Moreover, globally consistent texture mapping can introduce either distortions or discontinuities for the texture over the surface which has an arbitrary geometry, or both. Recently, some algorithms are presented in [11-19] for generating texture over arbitrary triangulated surface, using a given sample of 2D texture. We focus on these recent progresses and classify them into two classes. One is texture synthesis method\[13-15,18,19\], the other is texture mapping method\[11,12,16,17\], as described below.

Texture Synthesis

A number of algorithms have been proposed for the problem of synthesizing textures on arbitrary surfaces. Turk\[14\], Wei and Levoy\[13\], and Ying et al.\[15\] proposed algorithms for directly synthesizing new textures on arbitrary surfaces, which extend Wei and Levoy's texture synthesis method\[3\] by generalizing the search strategy. However, these algorithms are slow, and they can be accelerated by using tree-structured vector quantization. Furthermore, they are only suitable for a special kind of texture samples.

Recently, Tong et al.\[18\] synthesized bi-directional texture functions (6D functions capturing spatially-varying reflectance and texture information) on surfaces by generalizing the method of Wei and Levoy\[13\] of local neighborhood flattening and comparison and that of Turk\[14\]. Zhang et al.\[19\] proposed an approach to synthesize progressively-variant textures on surfaces, based primarily on studies by Hertzmann et al.\[8\] and Turk\[14\]. But these methods require significant computation time.

Texture Mapping

The traditional approach for texturing a surface is to map a 2D texture pattern on it. However, global parameterization for arbitrary surface is difficult, and unfolding the surface on plane is not so easy, as mapping texture to arbitrary surfaces will introduce some form of distortion. Even if we can unfold the surfaces, there is in general no perfect way to perform such a flattening without introducing some form of distortion or discontinuity. Based on recent progress, some methods\[11,12,16,17\] are proposed to texture arbitrary surface with mapping.

Neyret and Cani\[11\] texture surface isotropically by tiling a small collection of triangulated texture samples that match together along borders. The triangulated surface is tiled at desired scale into areas in which these textured samples are

*Correspondence
The method does not introduce discontinuities, and has less distortions. However, this approach works only for isotropic texture, and it requires pre-computing for the input texture samples obeying specific boundary conditions. Praun et al.\cite{12} proposed a method to iteratively paste irregular textured patches onto surface. The idea is to predefine a tangential vector field over the surface. To paste texture onto surface, a surface patch is grown and parameterized over texture space. As the previous algorithms, the method is only applicable to a specific class of textures. User is required to pre-produce irregular texture patch samples. Moreover, the local overlap of two texture samples is not to be sensitive to discontinuities. Soler et al.\cite{16} presented a hierarchical texture mapping algorithm for mapping texture onto arbitrary surface. This algorithm is an extension of Efros’ approach\cite{7}. Their idea is to progressively cover the surface by texture patches from input image. Starting with large patches, they recursively split these patches into smaller ones, which are mapped to texture space. The mapping minimizes the texture fitting error with already textured neighboring patches. The method records texture coordinates in the original texture sample for each triangle. However, they do not offer a mechanism to control locally over the texture orientation, and they do not blend neighbor textures for reducing the discontinuities on patch boundaries.

Soler et al.\cite{16} presented a hierarchical texture mapping algorithm for mapping texture onto arbitrary surface. This algorithm is an extension of Efros’ approach\cite{7}. Their idea is to progressively cover the surface by texture patches from input image. Starting with large patches, they recursively split these patches into smaller ones, which are mapped to texture space. The mapping minimizes the texture fitting error with already textured neighboring patches. The method records texture coordinates in the original texture sample for each triangle. However, they do not offer a mechanism to control locally over the texture orientation, and they do not blend neighbor textures for reducing the discontinuities on patch boundaries. Dischler et al.\cite{17} proposed an algorithm for synthesizing texture onto surfaces. They first extract “texture particles” from the texture sample by color thresholding and then paste “texture particles” on surfaces. But their method does not work for anisotropic and highly structured texture samples.

Summarizing the above mapping algorithms, we conclude with the following characteristics: 1) it is required to pre-produce particular texture samples\cite{11,12,17}; 2) it is required to partition interactively the initial surface to some surface patches\cite{11,12,16} with local parameterizations to texture space.

The aim of this paper is to directly synthesize texture on arbitrary surface with texture sample. Here is an overview of the process in our method. 1) Firstly the initial triangulated surface is simplified to a base mesh. 2) After user specifies tangential vectors at a few seed triangles, we interpolate vectors at the remaining triangles to construct a tangential vector field, which indicates the desired growing orientation of the texture on surface. According to the vector field, each triangle is mapped to texture space and has an initial texture coordinate while its shape is unchanged. 3) According to the vector field, we recursively map triangles to texture space until the whole surface is completely mapped using breadth-first searching strategy. The mapping process searches the suitable texture coordinates for each triangle in the sampling texture space, and at the same time the texture match is optimized between each newly mapped triangle and its already textured neighboring triangles. 4) The new texture of each triangle is blended with its neighboring triangles’ textures at the edge. We pack all new textures of triangles to a texture atlas. 5) In rendering phase, the simplified mesh is subdivided to construct a finer mesh, which matches the resolution of the original mesh.

2 Algorithm Description

2.1 Simplification of Mesh

The mesh simplification technology is important because highly detailed geometric models are rapidly becoming commonplace in computer graphics. It helps the storage, remeshing and transmission of meshes. The purpose of simplifying mesh is to reduce the computing time for texture synthesizing and to get better synthesizing results. Hoppe\cite{20} proposed a progressive mesh representation (PM) from an arbitrary mesh by unifying 2 adjacent vertices $v_1$ and $v_2$ into a single vertex $v$. Thus, an initial mesh $M_0$ can be simplified into a coarser mesh $M^0$ by applying a sequence of $n$ successive edge collapse transformations. Garland and Heckbert\cite{21} developed a surface simplification algorithm which can rapidly produce high quality approximations of polygonal models. The algorithm uses iterative contractions of vertex pairs to simplify models and maintains surface error approximations using quadric matrices. In this paper, we use Garland’s method to simplify the initial mesh.