Least Distortion Texture Map Construction From Multiple Views

Xiaohua Zhang* Yoshinari Nakanishi* Kiichi Kobayashi*
Hideki Mitsumine** Suguru Saito***
*NHK Engineering Services Inc.
**NHK Science and Technical Research Laboratories
***Tokyo Institute of Technology

Abstract

Texture mapping is an important technique to increase the intricate details covering the 3D models. We propose an algorithm that enables high quality texture map construction through the projection from multiple views onto 3D model. The extracted textures for each polygon was packed onto tessellated rectangular texture map. To solve the discontinuity problem along the edge of triangle, the texture outside each triangle in image space are also stored into texture triangle without increasing the extra memories. The optimal correspondence between the triangle vertices in image space and the triangle vertices in texture space make the constructed texture only have the least warping distortion. The experimental results are presented to verify our algorithm.

1. Introduction

Texture mapping is an important technique to increase the intricate and abundant details covering the colored 3D surface models [1]. Purely synthesized textures are applied to the 3D model in many applications such as driving and flying simulators. The incorporation of 3D model and real scene can provide impact ``visualized reality'' which is required in many applications such as virtual museum, electronic catalogs, architecture design etc. In general, texture map is constructed from real images, accompanying with the parameterized 3D model. Each vertex in the 3D model is attached with texture coordinates in the texture maps.

Several approaches have been proposed to construct texture map from range images or photographs. Marc Soucy et al. proposed a texture-mapping approach for the compression of colored 3D triangulations from several range images [2], while Yu used the same scheme to place the synthetic triangular texture patches into texture map and obtained texture coordinates [3]. It is known that the adjacent triangles in 3D object space will not be adjacent in texture space. If the triangles on image plane are fetched without any extension and directly stored, it will result in aliasing along the edges of triangle patches during rendering 3D models. In many cases, the triangles in image space and texture space are not similar triangles. The least distortion should be considered when mapping triangles from the image space to the texture space.

Bernardini et al. proposed a method to construct accurate digital models of scanned objects by integrating high-quality texture and normal maps with geometric data [4]. Rather than parameterizing the mesh into triangles, they partition the surface into patches to use mip-mapping technique. The occlusions are handled by comparing depth values stored in pre-computed depth buffers.

In this paper, we propose a method to construct texture map from multiple views with known 3D geometric data of object and known camera intrinsic and extrinsic parameters. To render new views without seams between triangles whose

textures are not adjacent in texture space, we use a technique of triangle contraction and extension during texture map construction, which avoids the edge aliasing due to the lack of information along the edges of texture triangles. Least distortion is achieved when warping triangles from the image space to the texture space by considering the optimal vertex correspondence between two spaces. The occlusions are detected on image plane and the depths are computed and compared only when occlusions occurred for this view.

2. OVERVIEW

The computational framework for constructing texture map relates to coordinate system in several spaces such as view spaces, image spaces and texture spaces as shown in Fig. 1. The inputs to this framework are 3D object model that has been triangulated and camera parameters. The 3D model was created using a method proposed in [5] and the camera was calibrated through an approach similar to [6]. The 3D model has its own 3D object space; which is transformed to the view space using the calibrated camera's extrinsic parameters before the texture map construction. Since multiple views are used, each image spaces correspond to each view point while the texture space is independent of view space as illustrated in figure 1.

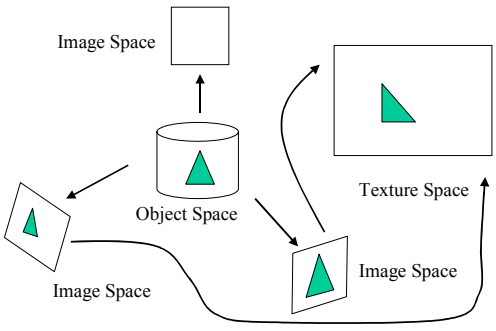


Fig.1. The flow of texture map construction

Suppose the triangle vertices order is counter-clockwise and only when the front of a triangle faces to the viewpoint it is possibly visible. Not all triangular polygons of 3D triangulation model are visible in the view space from a specific viewpoint. The visibility test for a triangle in view space must be carried out in the obstacle-free case. If the triangle passed the visibility test, it is projected onto 2D image space. However, even a triangle passed this test, it is still invisible if it is occluded by another triangle which is more nearer to the camera. The occlusion test is very important to take the correct textures. It should be noted that the visibility test is carried out in the 3D view space, while the occlusion test is carried out in 2D image space corresponding to the specific viewpoints. If a triangle passed both the visibility test and occlusion test, now it could be used to compute the texture of this triangle.

A 3D triangle may be visible from multiple view points, so there are several image triangles corresponding to this 3D triangle whose texture can be taken by weighted blending multiple image triangles or taken from only one image triangle which has the largest area compared with other triangles.

During texture space tessellation, the size of this texture triangle and the location should be decided to store the texture of a 3D triangle. The size is computed as the longest length of the image triangle with the largest area. To manipulate texture space easily, the size is extended to a power of 2. The texture triangles are arranged from the largest size to the smallest size.

Since the adjacent triangles in 3D object space will not be adjacent to each other in texture space, the discontinuities will occurs along the edges of rendered triangles. We use a technique of contracting texture triangle and extending image triangle to solve this problem. The transformation from image space to texture space often cause some shape changes, which will distort the rendered result. We use an optimal vertices correspondence between image triangle and texture triangle to alleviate the unpleased distortions.

3. VISIBILITY TEST AND OCCLUSION TEST

3.1. Visibility Test

We first detect if the front face of triangle is visible or not when there is no obstacle between the viewpoint and the triangle in 3D view space. We suppose a camera has view field less than 180°

and all vertices of any triangle are inside this view field. For a triangle $\Delta P_0P_1P_2$, its normal vector is computed as $\vec{N} = \vec{P_0P_1} \times \vec{P_0P_2}$. Instead of using the view direction along camera optical axis, the ray from viewpoint to the triangle is used for detecting the visibility. The inverse ray vector from viewpoint to one of vertex, for example, vertex P_0 is $\vec{R}$. If the dot product $\vec{N} \cdot \vec{R}$ is less than zero, the triangle is visible or else invisible.

3.2. Occlusion Test

Even if a triangle passed the above visibility test, it may not be visible if it is occluded by others triangles. The occlusion test is required for assuring that the texture of a triangle is obtainable or not. The occlusion test is a time-costly task, so we use a strategy to exclude the triangles that are far from the current triangle on 2D image space. For a 3D triangle, suppose its projection on an image plane is A as shown in Fig. 2. We first detect if it has cross intersection with other triangles on the same 2D image space.

For a 2D image triangle A, its boundary box S is computed as shown in figure 2 (a). Any triangle that does not intersect with this boundary box is excluded. This exclusion is completed by only using comparisons without any arithmetic computation.

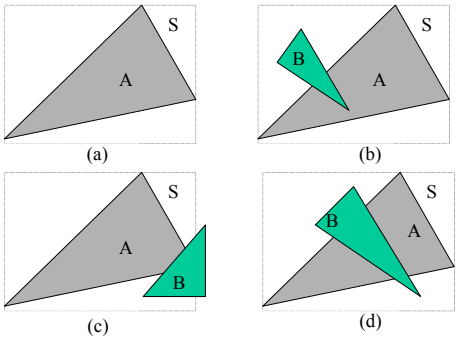


Fig. 2. The occlusion test

Only if a triangle B intersects with the boundary box S of triangle A, it may intersect with triangle A. The intersection of triangle A with B only has two cases: (1) The triangle A contains at least one of vertices of triangle B, or triangle B contains at least one of vertices of triangle A as shown in figure 2 (b) and (c); (2) Any vertex of A or B is not inside triangle B or A, but the edges of both triangles intersect with each other as shown in figure 2(d). Since the judgment of point-in-triangle needs less computation than that of edge-intersection, we detect the first case, if failed, then detect the second case. If the vertex of triangle A is not inside any triangle and any triangle does not contain the vertex of triangle A and the edge of triangle A does not intersect with the edges of any triangle, then the triangle A does not intersect with any triangle on the 2D image plane. So it is not occluded by any triangle in 3D view space, it is visible. If triangle A intersects any triangle B on 2D image plane, and the corresponding 3D triangle of A is far from the viewpoint than the corresponding 3D triangle of B, the triangle A is invisible.

4. TEXTURE SPACE TESSELLATION

For any 3D triangle in the view space, it may be seen from several viewpoints. The texture of this triangle is taken from these views if the projected 2D image triangle in a view space passed the visibility test and occlusion test. The acquired texture of a triangle is stored in the texture space. For the texture of a 3D triangle, its size and location in the texture space should be decided before storing.

In our implementation, the constructed texture map consists of several texture image files. Most renderers such as developed by using OpenGL require the widths and heights of texture image files are powers of 2, for example, $512 = 2^9$ pixels (from now on, all units are in pixels).

Given a 3D triangle T^{3D}, its projections onto several image planes are computed. Let the image triangles that passed the visibility test and occlusion test are $T_1^I, T_2^I, \cdots, T_n^I$. Triangle $T_k^I (1 \le k \le n)$ has the largest area among these image triangles, which has more texture information than others. The length of longest edge of triangle T_k^I is calculated as e, which is

used to decide the size of texture triangle in texture space. To allocate the texture triangles easily in texture space, the length e is extended to d as a power of 2. Any two of texture triangles having the same size d combines to a square. However, when stored in the texture space, the second triangle has a size $d-1$. Note that the length of the longest edge e of projected image triangle is unpredictable, it should be limited in a definite range, such as [8,128] to avoid being too large or too small.

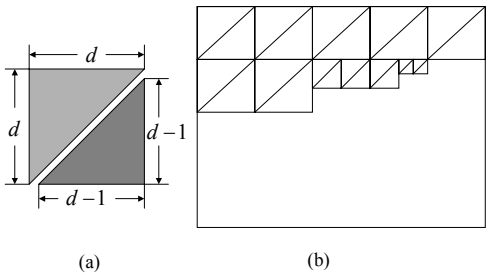


Fig.3. The texture space tessellation

The 3D triangles are scanned in the order of sizes of their texture triangles, from the largest one to the smallest one. The texture space tessellation is illustrated in Fig.3 (a): two triangles form a square and (b): the allocation of texture triangles.

5. LEAST DISTORTION

The distortion in a rendered image may have a variety of sources. The texture triangles in the texture space are isosceles right triangles, while the triangles in the image space may be arbitrary. The optimal correspondence between the vertices of triangles in both spaces can reduce the distortion of mapping from the image space to the texture space. Since triangles being adjacent in 3D object space are not adjacent to each other in the texture space, the rendered edges of triangles are aliased in the resulted new views. This is because the renderer takes the pixels outside a triangle in the texture space when rendering a pixel near to or on the triangle edge in the image space, while the pixels outside a triangle have no any relation with this triangle - the discontinuity between triangles in the texture space. Note that the optimal correspondence and triangle extension are only applied to the image triangles that passed the visibility test and occlusion test.

5.1. Optimal Vertices Correspondence

The vertex correspondence between a triangle in image space and a triangle in texture space is arbitrary according to the vertices order in the original 3D object model. This correspondence is not an optimal one and will cause heavy distortion when mapping from the image space to the texture space. For example, the correspondence of a vertex with obtuse angle to a vertex with 45° will cause the lost of texture information. We can refer to the figure 4 that the vertex correspondence $(A_i, B_i, C_i) \leftrightarrow (C', A', B')$ in the image and texture spaces is not a best correspondence. However, by observing the shape of triangles in figure 4, perhaps the optimal correspondence is $(A_i, B_i, C_i) \leftrightarrow (A', B', C')$.

We hope the vertex with larger angle of triangle in the image space corresponds to the vertices with large angle of triangle in the texture space and vertex with smaller angle corresponds to the vertices with smaller angle. This completed by comparing the lengths of edge since the longest edge has the largest opposite angle. This avoids the time-costly angle computations.

5.2. Contraction And Extention

We used a technique to reduce the distortion caused by the triangle edge aliasing. The technique is to contract each texture triangle in texture space and correspondently extend each image triangle in image space. The contraction and extension are applied after vertex optimal correspondence. The concepts of contraction and extension are illustrated in Fig. 4.

When the size and location of a texture triangle has been decided, its region is also decided. If we just fetch the texture from image space and store it in this texture triangle, we lost the pixel information around the image triangle, which will results in edge aliasing when rendering the object. As shown in figure 4(a), the computed current texture triangle is ΔABC. We hope to store both the texture of image triangle and the neighboring pixel information around the image triangle. We

contract the triangle ΔABC to the shadowed triangle $\Delta A'B'C'$ in the figure, so that the texture is stored in $\Delta A'B'C'$ while the neighboring pixel information is stored in the gap between ΔABC and $\Delta A'B'C'$.

The distance C_d between the correspondent edges of ΔABC and $\Delta A'B'C'$ called contraction distance should be carefully chosen. If the distance is too small, there is no enough space for storing the neighboring pixel information and then the edge aliasing still exists. If the distance is too large, there is too large gap space for storing the neighboring pixels while only the pixels that are very near to the image triangle should be stored. On the other hand, since the size of texture triangle ΔABC is already decided, if the contraction distance is too large, the region of $\Delta A'B'C'$ will become too small to store the triangle texture and will possibly cause heavy distortion in the rendered results. Theoretically, the minimum contraction distance in texture space is related with the interpolation method for sampling used by the renderer. In the case of nearest neighbor point sampling, the distance is $\sqrt{2}/2$ and in the case of bilinear interpolation, the distance is $\sqrt{2}$, while the cubic interpolation needs contraction distance of $2\sqrt{2}$. The computation of texture triangle contraction in the texture space is simple, since we known the vertices of ΔABC and the contraction distance.

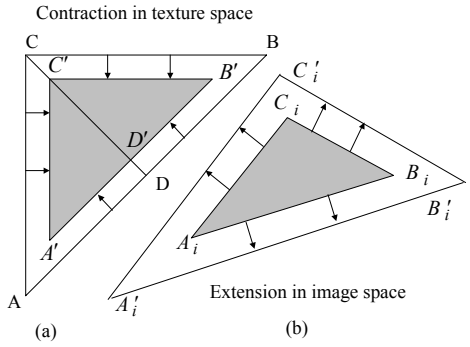


Fig.4. The contraction in texture space and extension in image space

Let the image triangle in image space corresponding the texture triangle in the texture space is $\Delta A_iB_iC_i$. Since the triangle ΔABC has been contracted to the triangle $\Delta A'B'C'$ in the texture space, the texture taken from the image triangle $\Delta A_iB_iC_i$ should be stored in the texture triangle $\Delta A'B'C'$. In the image space, the triangle $\Delta A_iB_iC_i$ is extended to the triangle $\Delta A_i'B_i'C_i'$, so the texture in the region between these two triangles is taken and stored in the gap between triangles ΔABC and $\Delta A'B'C'$ in the texture space.

The computation of image triangle extension is a little complicated than that of texture triangle contraction, since we do not know the extension distance. The correspondence between triangles in the image space and the texture space help us finding the vertices of $\Delta A_i'B_i'C_i'$ using the concept of barycentric coordinates.

The Cartesian coordinates of vertices A', B' and C' are represented through their barycentric coordinates $A'(u_a, v_a, w_a), B'(u_b, v_b, w_b)$ and $C'(u_c, v_c, w_c)$ related to the triangle ΔABC in the texture space:

$$\begin{cases} A' = u_a A + v_a B + w_a C \\ B' = u_b A + v_b B + w_b C \\ C' = u_c A + v_b B + w_b C \end{cases} \quad (1)$$

With the correspondence and the assumption that texture triangle is an orthogonal projection of the image triangle, the Cartesian coordinates of vertices of triangle $\Delta A_iB_iC_i$ are represented through the same barycentric coordinates related to the triangle $\Delta A_i'B_i'C_i'$ in the texture space, which can be written in matrix form:

$$\begin{pmatrix} A_i \\ B_i \\ C_i \end{pmatrix} = \begin{pmatrix} u_a & v_a & w_a \\ u_b & v_b & w_b \\ u_c & v_c & w_c \end{pmatrix} \begin{pmatrix} A_i' \\ B_i' \\ C_i' \end{pmatrix} \quad (2)$$

Now it is simple to find the extended triangle $\Delta A_i'B_i'C_i'$, since we known the Cartesian coordinates of vertices A_i, B_i, C_i and all barycentric coordinates in the matrix.

For the implementation of texture map construction, one could extend the image triangle without obvious computation of the vertices of $\Delta A_i' B_i' C_i'$. If the mapping between $\Delta A' B' C'$ in texture space and $\Delta A_i B_i C_i$ is computed, and using the barycentric coordinates of each pixel inside the triangle ΔABC represented by the vertices of triangle $\Delta A' B' C'$, not the triangle ΔABC, the Cartesian coordinates of corresponding pixel in the image space can be computed from the same barycentric coordinates related to the triangle $\Delta A_i' B_i' C_i'$, not the triangle $\Delta A_i B_i C_i$. This implementation has only little difference with that of using image triangle extension, but the concept is the same.

5.2. Mapping From Image Space to Texture Space

We use an inverse mapping function to fill the texture triangle by scanning each pixel in the triangle and fetch its texture from the corresponding image triangles. The mapping function is computed using barycentric coordinates. As described before that a 3D triangle may be seen from multiple view points, so it is better to combine the textures from multiple image spaces.

For a texel in the texture triangle, the combination is a weighted sum of textures from multiple image triangles. The weights are computed as the cosine of normal vector $\vec{N}$ of 3D triangle and the inverse view ray $\vec{R}$. This weight computation is carried out in 3D view space. Calculating the inverse view ray for each point inside the 3D triangle is time-costly. An option is to use the normalized areas of projected image triangles in each image space of a 3D triangle as the weighting coefficients. The value of a texel in the texture space and weighting coefficients of visible image triangles are computed using formulae in Eq.(3).

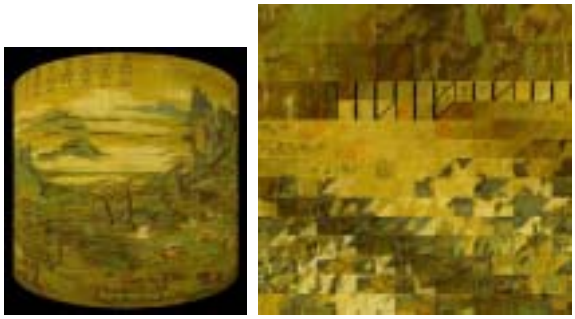
$$t = \sum_{i=0}^{m} w(i) P(i)$$
$$w(i) = A(i) / \sum_{i=0}^{m} A(i) \qquad (3)$$

where m means the 3D triangle is visible in $m+1$ views, while t is the computed texel value inside a texture triangle, $P(i)$ is the pixel value in the ith visible image triangle, $w(i)$ is the computed weighting coefficient for the ith visible image triangle and $A(i)$ is the area of the ith visible image triangle.

6. EXPERIMENTAL RESULTS

Two experimental results are shown in this section. One used synthesized data and the other used real data. In both experiments, the contraction distance is set to 1.5 since our renderer uses bilinear interpolation for sampling. During texture map construction, the vertices correspondence is optimized.

Experiment with synthesized data: We synthesized a cylinder object and render it from 6 different viewpoints with designated camera parameters. The cylinder object was covered with a painting through texture mapping. One of rendered original images is shown in Fig. 5(a). The dimension of texture file is 512×512. The contraction distance is 1.5. One of the constructed texture map is shown in figure 5(b). Two of rendered images using the constructed texture map are shown in the same figure. Figure 5(c) is the image rendered in the same direction with (a), while (d) is the image rendered after the cylinder object was rotated 30° to the right.

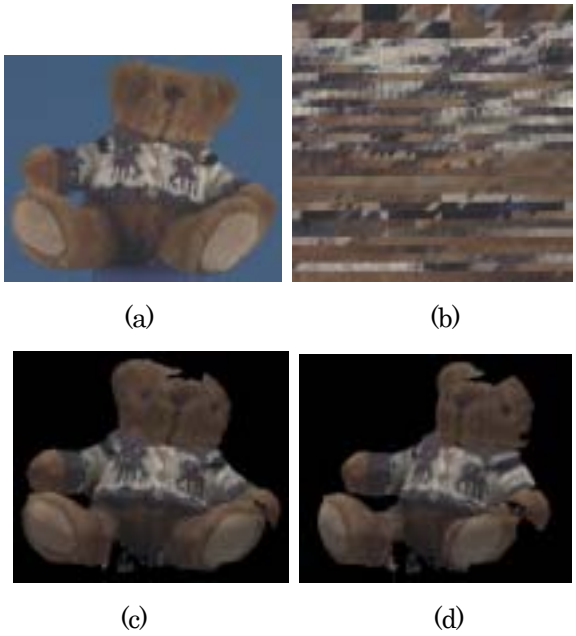


(a) (b)

(c) (d)

Fig.5. The result with synthesized data

Experiment with real data: The experimental result with real data is shown in Fig. 6. The 3D mesh model was established through a method proposed in [5] and the camera parameters were calibrated using a method similar to [6]. The object was set on a rotary table whose rotation axis was also calibrated. The images were taken while rotating the table, one of original images is shown in figure 6(a). We used 6 images separated with 60° interval to construct the texture map, one of texture images whose dimensions are 512×512 is shown in (b). The contraction distance also is 1.5. Figure 6(c) shows the rendered image in the same direction with (a), while (d) is the rendered image after the object was rotated 30° to the left.



(a) (b)

(c) (d)

Fig.6. The result with synthesized data

We also rendered the objects with texture map constructed without triangle contraction and extension and without optimal vertex correspondence. By visually observating the rendered results, they contain more distortions and noises than the results shown above.

7. Conclusion

We described a means for constructing texture map from multiple views using texture triangle contraction and image triangle extension combined with optimal vertex correspondence. The distortions and noises in the rendered results are reduced when using the constructed texture map. Not only the texture inside a triangle but also the texture around the traingle are stored in the texture map, while the storage space was not increased and remained the same and the visual quality of rendered images was improved.

REFERENCES

1) P.S.Heckbert, ``Fundamentals of texture mapping and image warping'', Master's thesis, Computer Science Division, University of California, 1989.
2) Marc Soucy, Guy Godin and Marc Rioux, ``A texture-mapping approach for the compression of colored 3D triangulations'', The Visual Computer, Vol.12, pp.503-514, 1996.
3) Yizhou Yu, Andras Ferencz and Jitendra Malik, ``Compressing texture maps for large environments'', SIGGRAPH'2000, New Orleans, Louisana, July 2000
4) Fausto Bernardini, Ioana M.Martin and Holly Rushmeier, ``High-quality texture reconstruction from multiple scans'', IEEE Trans. on Visualization and Computer Graphics, Vol.7, No.4, pp.318-332, 2001.
5) K. Kobayashi, Y. Nakanishi, X. Zhang, M. Tadenuma, H. Mitsumine, and S. Saito: ``High resolution 3D surface measurement from multiple viewpoint images'', NICOGRAPH 2000, pp.143-150, 2000.
6) Z. Zhang, ``A flexible new technique for camera calibration'', IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.22, No.11, pp.1330-1334, Nov. 2000. Also refer to his website: http://research.microsoft.com/users/zhang/