JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 26, 291-305 (2010)

Short Paper

Texture Tiling on 3D Models Using Automatic Polycube-maps and Wang Tiles*

CHIN-CHEN CHANG+ AND CHEN-YU LIN
+*Department of Computer Science and Information Engineering*
National United University
Miaoli, 360 Taiwan
E-mail: ccchang@nuu.edu.tw
Institute of Computer Science and Engineering
National Chiao Tung University
Hsinchu, 300 Taiwan

In mapping textures onto 3D models, it is essential to eliminate all seams and avoid excessive distortions. To achieve these goals, texture tiling provides an alternative approach for texturing surfaces. In this paper, a texture tiling approach that combines polycube-maps and Wang tiles for 3D models is proposed. The polycube of a 3D model is first constructed automatically and a tiling mechanism is then used to fill the tiles on the polycube. Finally, rectangular cells which are transformed from the polycube are generated and textures are mapped onto the 3D model. The results show that the proposed approach leads to seamless texture mapping on 3D models.

Keywords: computer graphics, texture mapping, texture tiling, polycube-maps, Wang tiles

1. INTRODUCTION

In mapping textures onto 3D models, enhancing the visual appearance of a 3D model is important. Therefore, it is essential to eliminate seams and avoid excessive distortions. To achieve these goals, texture tiling offers an approach for texturing surfaces. In the texture tiling approach, it is only necessary to synthesize textures on the tiles; a process is independent of the surface geometry. Cube maps [1] provide a method for seamless texture mapping. The shape of a 3D model should resemble a cube. Polycube-maps [26] can break this restriction and decrease the distortion. However, users need to be more involved to construct polycubes which takes more time. Therefore, developing techniques for automatically determining appropriate polycubes with no user intervention is critical.

In this paper, a texture tiling approach that combines polycube-maps and Wang tiles for 3D models is presented. The proposed approach automatically constructs a poly-cube which consists of cubes. It achieves seamless texture mapping between the 3D model and

Received February 14, 2008; revised May 21, 2008; accepted July 25, 2008.
Communicated by Tong-Yee Lee.
* This paper was supported by National Science Council of Taiwan R.O.C., under contract No. NSC 96-2221-E-239-027.
+ Corresponding author.

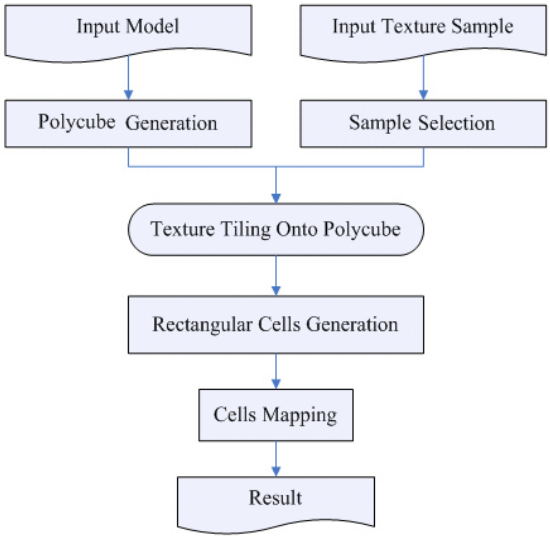


Fig. 1. Flowchart of the proposed approach.

the polycube. The flowchart of the proposed approach is shown in Fig. 1. First, a user inputs a 3D model and a sample texture image. Then, these two inputs are processed separately. The polycube of the input model is constructed. Four diamond-shaped samples are randomly selected from the input texture. Next, the mechanism of Wang tiles is reformulated to seamlessly tile textures onto a polycube. During the rectangular cell generation, the system converts the structure of the polycube to rectangular cells. Finally, texture mapping is performed between the 3D model and the polycube according to the mapping function of each rectangular cell.

The major contributions of this paper are as follows: First, the proposed approach can automatically generate polycube-maps. User intervention can be avoided and the proposed approach can reduce extra time. Second, generated polycube-map parameterization is used to do texture tiling. The desired texture mapping is easily obtained.

The rest of this paper is organized as follows: In section 2, related works of patch-based texture synthesis, polycube-maps and Wang tiles are described. Then the construction of a polycube with tiled textures is presented in section 3. In section 4, the generation of texture mapping is proposed in detail. Sections 5 and 6 give the results and conclusions, respectively.

2. RELATED WORKS

2.1 Texture Synthesis

Efros and Freeman [9] presented a patch-based approach for texture synthesis. This technique outperforms pixel-based methods for regular textures. Liang *et al.* [19] developed a patch-based sampling scheme for texture synthesis based on a source texture to build blocks. Hertzmann *et al*. [14] proposed a new framework for processing images by

example, called "image analogies." The output textures have higher visual quality but still have some undesirable artifacts. Wu and Yu [30] presented a feature-matching and deforming algorithm for texture synthesis. Kwatra *et al*. [16] developed a texture synthesis algorithm based on graph cuts. In their approach, patches from a source texture were transformed and copied to the output and then stitched together along optimal seams to generate an output texture. Dong *et al*. [6] proposed a patch-based texture synthesis algorithm that cuts and stitches irregularly shaped patches (IRSPs) to generate output textures.

Wang tiles [4, 27, 28] are a tiling set consisting of a set of square tiles. The edges of a tile are assigned different colors each of which corresponds to one sample. All shared edges should have matched colors. Grunbaum and Shepherd [12] provided an algorithm to tile a plane with a finite set of Wang tiles aperiodically. They create large non-repetitive textures. Culik [5] proved that thirteen tiles are enough to tile aperiodically. Stam [25] was the first to consider non-periodic Wang tiles for texture synthesis. He applied it to the rendering of water surface. Cohen *et al*. [4] further investigated this approach and invented an automatic method. They presented a simple stochastic system which non-periodically tiled a large texture with a small set of Wang tiles. Lai and Tai [17] presented an approach to synthesize the transition texture to be tiled on a terrain. They used Wang tiles to present a good-looking profile of successions on a terrain for tiling transition textures.

Kwatra *et al*. [15] presented an optimization-based technique to progressively optimize the textures. This approach merges local similarity measures into a global metric to optimize the whole texture. Dong *et al*. [7] presented an optimization-based approach for tile-based texture synthesis. Their approach generates an ω-tile set of high pattern diversity and quality and improve the quality of Wang tile based texture synthesis.

2.2 Texture Mapping

Texture mapping mostly follows the multi-chart approach. It focuses on partitioning, parameterization and packing. Carr and Hart [2] and Cignonoi *et al*. [3] assigned a patch consisting of a single or pairs of triangles. Each patch can be parameterized with low distortion. However, this approach produces seams on patch boundaries. Other approaches [11, 18, 20, 22, 24] considered large patches and parameterized each patch; all unsuccessfully. In order to avoid this problem, several researchers [18, 21, 23] cut the surface where the seam is less visible.

Cube maps [1] achieved a seamless texture mapping, but then the 3D model's shape must be similar to a cubic shape. Tarini *et al*. [26] extended this concept to arbitrary meshes and provided a new mechanism, called polycube-maps. With this, a user defines the shape of the polycube roughly approximating the 3D model. Then, the polycube is warped to approximate the model. The vertices of the model are projected onto the polycube. Finally, the polycube is warped inversely and the projections are optimized. For texture mapping, a user first roughly approximates the 3D model with a polycube. The dual space of the polycube is defined. Each cell of the dual space is centered in a corner of the polycube. Finally, the projection functions of the cells are obtained. This dual partition decreases distortion because the projection function is varied according to the cell configuration. Wang *et al*. [29] developed the polycube splines which are built based on polycube maps. Their modeling techniques can be applied in solid modeling and shape

computing. Fu and Leung [10] combined polycube-maps with Wang tiles [4]. They used the algorithm of polycube-maps and reformulated the texture tiling mechanism of Wang tiles for 3D models.

3. POLYCUBE WITH TILED TEXTURE

3.1 Automatic Polycube Construction

A polycube is composed of axis-aligned unit cubes which are attached face to face. First, a bounding box of the input model is established. Next, a user can adjust to a suitable parameter to set the size of the unit cube. According to this size, the proposed technique uniformly subdivides the bounding box into unit cubes. Then, the triangle-cube intersection algorithm [8] which examines the intersection between triangles of the model and unit cubes is applied to construct a polycube of the model. There are three phases. In the first phase, there are a trivial-accept and three trivial-reject tests to eliminate easy cases. The second phase detects the triangle edges that penetrate any face of a cube. The third phase examines whether the cube corners poke through the interior of a triangle. The remaining cubes do not intersect the model.

After the polycube of the model is constructed, the model is on its inside. Let a cell equal in size to a unit cube, be centered in a corner of the unit cubes, and intersect the polycube. Since some unit cells do not intersect the model, some textures which are tiled on the surfaces of the polycube cannot be mapped on the model, as shown in Fig. 2 (a). To solve this problem, non-intersectional cells are detected and removed to correct the structure of the polycube before tiling textures. First, each intersectional cube of the polycube is traced. For each cube, eight cells converted from the cube are examined to determine if they are intersecting the model. The cube is removed from the structure of the polycube if no non-mapping cell intersects it. Fig. 2 (b) is a corrected structure from Fig. 2 (a).

Finally, when the model is not inside the polycube, they intersect each other.

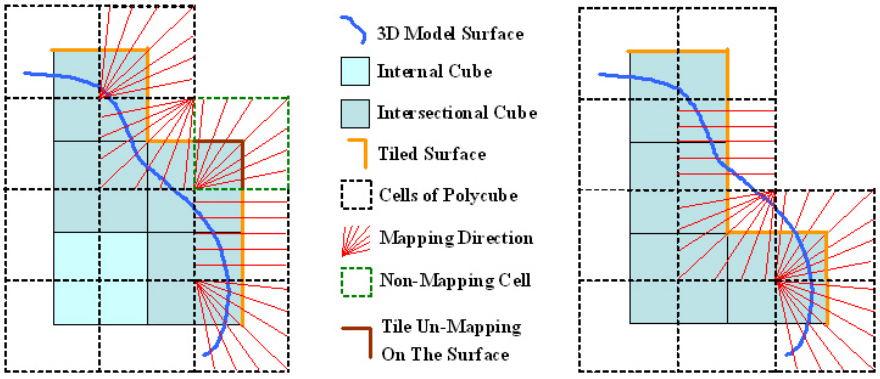


Fig. 2. (a) 2D analogue: relation between cell mapping and the 3D model; (b) 2D analogue: correction of the polycube structure.

3.2 Texture Tiling on Polycube

Wang tiles [4] are a set of square tiles with colored edges. The concept of Wang tiles is extended to tile textures on the surfaces of the polycube. The proposed algorithm consists of two parts. One is the edge coloring which arranges each square surface on the polycube to correspond to four samples; the other is the tile construction which synthesizes four samples to form a tile of each square surface.

The edge coloring technique is based on the approach of Fu and Leung [10]. First, there are four diamond samples randomly selected from the input texture. Then, all edges of the surfaces on the polycube are divided into three groups, namely, X, Y, and Z, according to three axial directions. For each group, two samples are randomly selected from four samples. Then each edge of the surface corresponds to a sample which has been randomly selected from its group samples.

In order to tile textures on the polycube, the concept of Wang tiles is slightly modified by rotating samples. An example is shown in Fig. 3. A sample is divided into upper and lower portions. When it is tiled on the surface of a unit cube, the lower half is on the top side of the blue surface and the upper half is on the right side of the yellow surface. A sample is appropriately rotated 90 degrees counterclockwise for maintaining seamless tiling when tiling on the yellow surface. Also, samples are not rotated when synthesizing tiles on the front, left, back, and right surfaces, as shown in Fig. 4. When tiles are synthesized on the top and bottom surfaces, a sample which corresponds to the edge along the X-axis is rotated 180 degrees, as shown in Fig. 5. And a sample which corresponds to the edge along the Z-axis is rotated 90 or 270 degrees counterclockwise. Therefore, each edge on the surface of the polycube corresponds to a suitable sample. This sample is rotated when tiled on the surface. Textures are tiled on the surface by using this algorithm.

Textures are tiled on each square surface of the polycube based on the approach of Cohen *et al*. [4]. The graph cuts algorithm [16] is used to synthesize a tile from four diamond samples which correspond to the edge of the square surface. Hence, texture tiling can be accomplished on the surface of the polycube. Fig. 6 shows (a) a Laruana model, (b) an intermediate result which was the polycube of the model with unit cubes and (c) the polycube and the model with tiled textures intersecting each other.

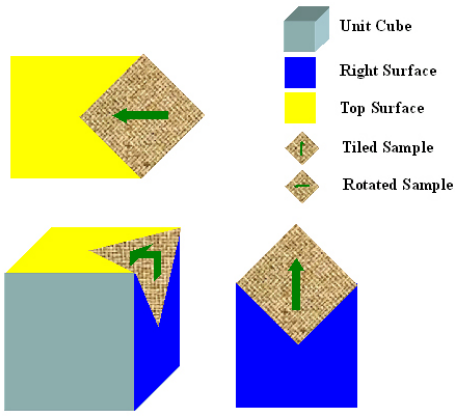


Fig. 3. Unit cube with a tiled sample.

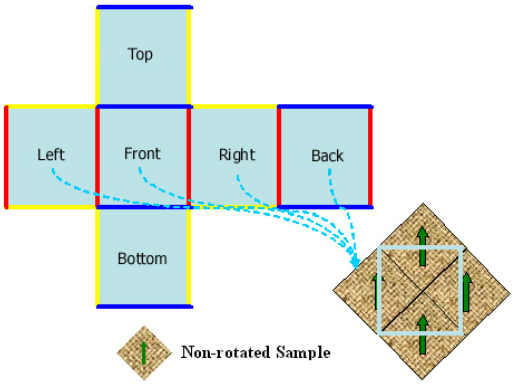


Fig. 4. Expansion of the unit cube.

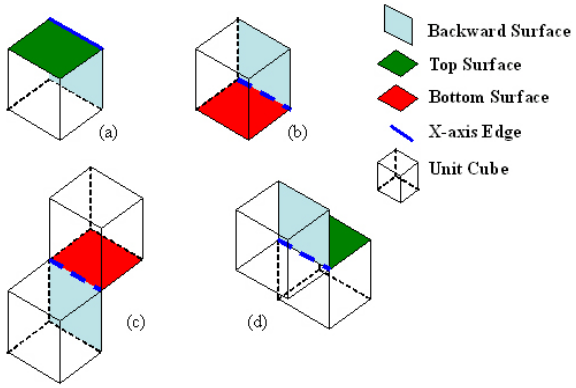
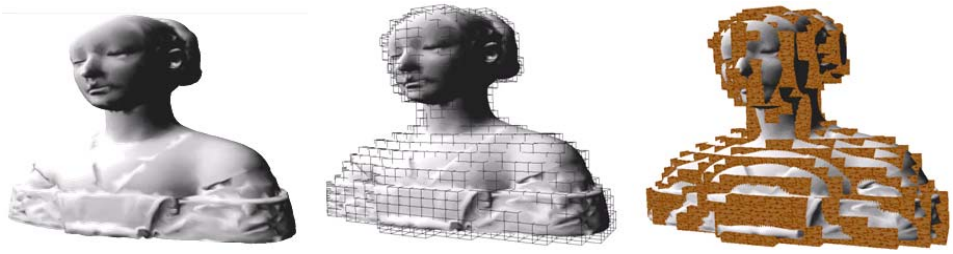


Fig. 5. (a), (b), (c), and (d) are portions of the polycube. Four structures that need to rotate a sample 180 degrees when tiling on the top or bottom surfaces.



(a) (b) (c)

Fig. 6. (a) Laruana model, (b) polycube of the model and (c) polycube with tiled textures and the model intersecting each other.

4. GENERATION OF TEXTURE MAPPING

4.1 Transforming Polycubes to Polycells

To reduce mapping distortion, cell mapping based on polycube-maps [26] is used. A polycell is composed of axis-aligned unit cells which are attached face to face. The polycube is transformed to the polycell before mapping textures onto the model. External cells are cells which intersect the portions of the tiles on the surfaces; internal cells are cells which do not. The proposed algorithm defines mapping directions according to the configuration inside a cell. The configurations inside external cells are then created. The intersection between external cells and the tiles on the polycube is determined. Each tile is subdivided into four slices (parts). An example is shown in Fig. 7. There are 63 different configurations of the slices inside a cell. If rotational and reflectional similarities are removed, these configurations can be further reduced to six basic configurations. Finally, the polycell and the configurations inside each cell are constructed.

4.2 Rectangular Cell Construction

All external cells intersect with the model surface. Each external cell is processed to

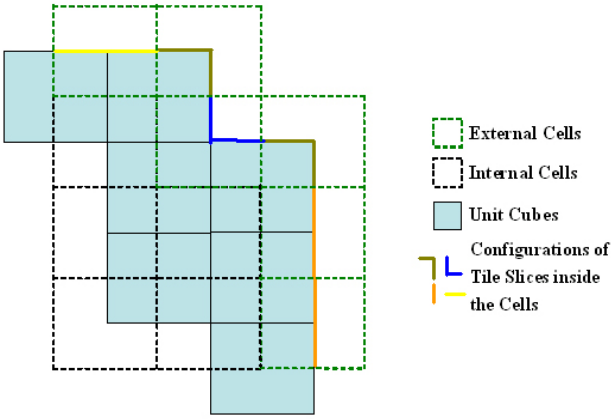
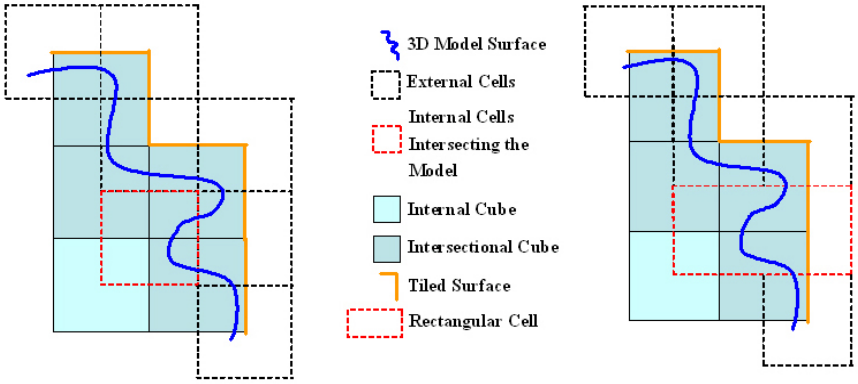


Fig. 7. 2D analogue: relation between the polycube and the polycell.



(a) (b)

Fig. 8. (a) 2D analogue: an internal cell intersects the model but is empty; (b) Analogue: combination of internal and external cells.

map textures on the model. However, although a few internal cells intersect the model, the configurations inside these cells are empty. The reason is that the polycube cannot determine the curvature of the model surface precisely. An example is shown in Fig. 8 (a). These internal cells, grouped in set *K*, cannot map any textures and consequently cause gaps on the model.

In order to solve this problem, each internal cell of *K* is combined with an external cell which is adjacent to it. Adjacent external cells for each internal cell are numbered and arranged from small to large. This algorithm stops internal cells merging with external cells.

Then, the six surfaces of the internal cell which intersect the model are examined by using the triangle-cube intersection algorithm [8]. If only one surface intersects the model, two cells, including this surface, are merged. Fig. 8 (b) shows the modification of Fig. 8 (a). Mapping directions of the basic configurations which are based on polycube-maps [24] are defined, as shown in Fig. 9. If more than one surface intersects the model, an

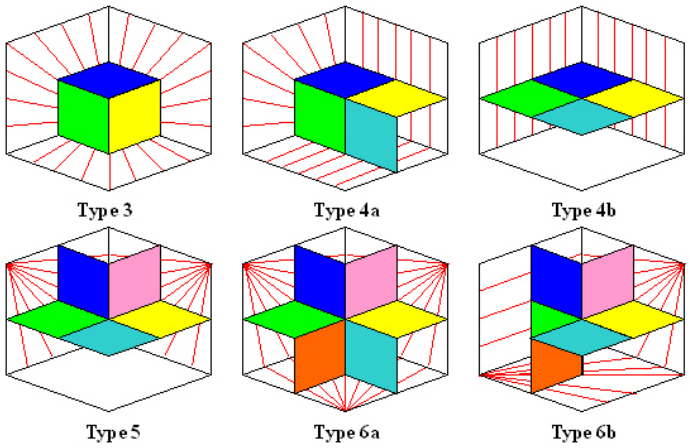


Fig. 9. Mapping directions of six basic configurations.

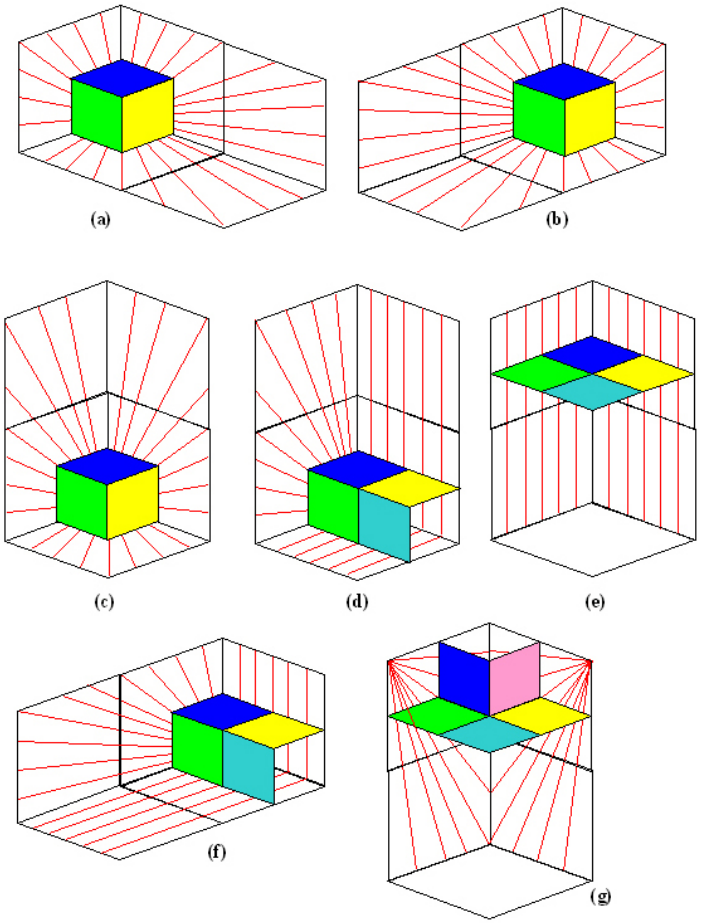


Fig. 10. Rectangular mapping directions of four configurations; (a), (b) and (c) are Type 3, (d) and (f) are Type 4a, (e) is Type 4b, and (g) is Type 5.

external cell, whose configuration map has a lower distortion, is chosen. Type 4b has the lowest distortion of all configurations. The selection from high to low is Type 4b, Type 4a, Type 3 and Type 5. Type 6a and Type 6b cannot be adjacent to the internal cell in *K* because all surfaces of the cells are connecting the slices. Therefore, these two configurations in rectangular cells are not applied.

The problem of *K*'s empty internal cells is solved by using rectangular cells. However, some of rectangular cells may create seams on the model. There are two reasons for this: first, the size of the unit cell is too large; second, the curvature of the model is too big. The size of the unit cube is reduced to decrease the appearance of the seams but that does not avoid this problem completely.

4.3 Cells Mapping

Four rectangular configurations are increased to process the internal cells in *K*. Fig. 10 shows all mapping directions of rectangular configurations. For the mapping functions of other configurations, refer to [26].

Triangles of the model are processed separately to map textures. First, the cells which intersect the bounding box of a triangle are searched. Then, the intersections between these cells and the triangle are considered. If the triangle is completely within one cell, the mapping function of the cell is used to map textures onto it. If not, the intersectional points are acquired by detecting a triangle with the cells. There are two kinds of points between a triangle and a cell:

- Three edges of a triangle and six surfaces of a cell.
- Twelve edges of a cell and the interior of a triangle.

Some intersectional points are increased in the triangle. A triangle is subdivided into several slices by cells and each slice is included in a cell. An example is shown in Fig. 11. Then, each slice is processed separately to map textures onto it according to the mapping function of the cell which includes it. Finally, textures of the polycube are mapped onto the slices of all triangles of the model.

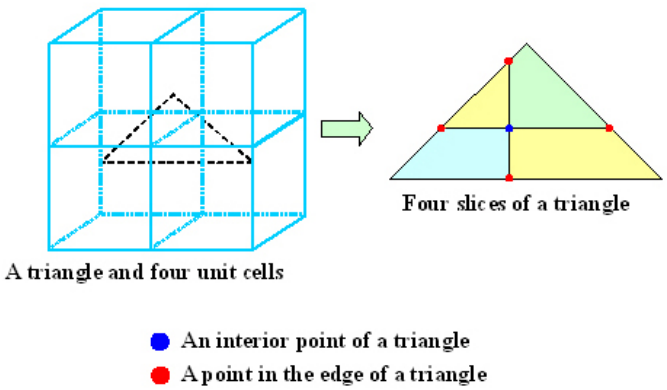


Fig. 11. A triangle is subdivided by cells into four slices.

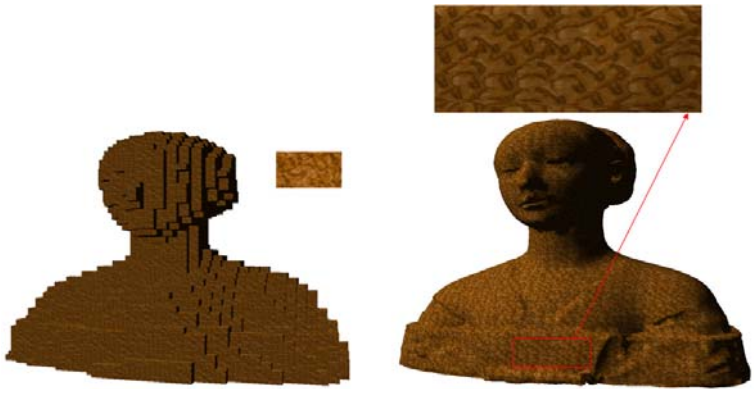
5. RESULTS

To evaluate the effectiveness of the proposed approach, several experiments were done. The proposed algorithm was implemented in C# language on VC.Net with a Pentium 4 3.4GHz PC with 2 GB memory.

Table 1 shows the related information of a Laruana model, a teapot model and a Daba model during texture mapping. For the first part, three stochastic textures were tested. First, the inputs were a 183 × 100 texture T_1 and the Laruana model. The size of a tile was set as 32 × 32. The number of unit cubes was 2555. The number of surfaces on the polycube was 4264. The number of different tiles was 804. The time for constructing the polycube with tiles was about 5 minutes and 3 seconds. The time for mapping from the polycube to the model was 33 seconds. Fig. 12 shows (a) the input texture and the polycube of the Laruana model with tiled textures and (b) the mapping result with a zoomed-in view. Next, the inputs were a 183 × 100 texture T_2 and the teapot model. The size of a tile was 32 × 32. The number of unit cubes was 1425. The number of surfaces on the polycube was 2568. The number of different tiles was 648. It took about 3 minutes and 56 seconds to construct the polycube with tiles. It took about 15 seconds to map textures from the polycube to the model. Fig. 13 (a) shows the input texture and the polycube of the teapot model with tiled textures. The mapping results with a zoomed-in view are shown in Fig. 13 (b). Third, the inputs were a 433 × 640 texture T_3 and the Daba model. The size of a tile was 64 × 64. The number of unit cubes was 1798. The number of surfaces on the polycube was 2730. The number of different tiles was 541. The time for constructing the polycube with tiles was about 6 minutes and 15 seconds. The time for mapping from the polycube to the model was 27 seconds. Fig. 14 shows (a) the input texture and the polycube of the Daba model with tiled textures and (b) the final result with a zoomed-in view.

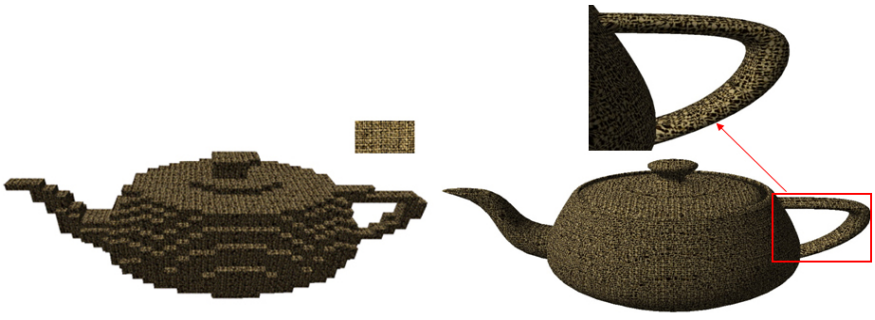
Table 1. Information of the Laruana model, teapot model and Daba model during texture mapping.

Model	Laruana		Teapot		Daba
Input texture	T_1	T_4	T_2	T_5	T_3
Size of the input texture	183 × 100	400 × 400	183 × 100	128 × 128	433 × 640
Size of a tile	32 × 32	64 × 64	32 × 32	64 × 64	64 × 64
Number of unit cubes	2555	3657	1425	1425	1798
Number of surfaces on the polycube	4264	6030	2568	2568	2730
Number of different tiles	804	940	648	679	541
Time of constructing the polycube with tiles	5 min. 03 sec.	10 min. 40 sec.	3 min. 56 sec.	8 min. 35 sec.	6 min. 15 sec.
Time of mapping textures from the polycube to the model	33 sec.	47 sec.	15 sec.	14 sec.	27 sec.



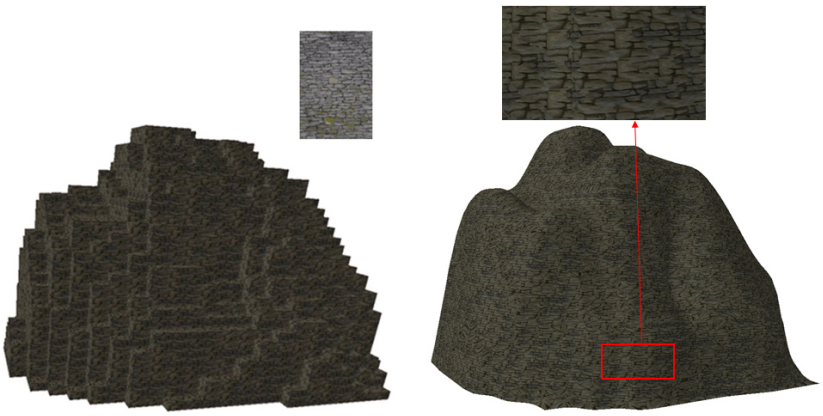
(a) (b)

Fig. 12. (a) Input texture T_1 and the polycube of the Laruana model with the tiled textures and (b) mapping result with a zoomed-in view.



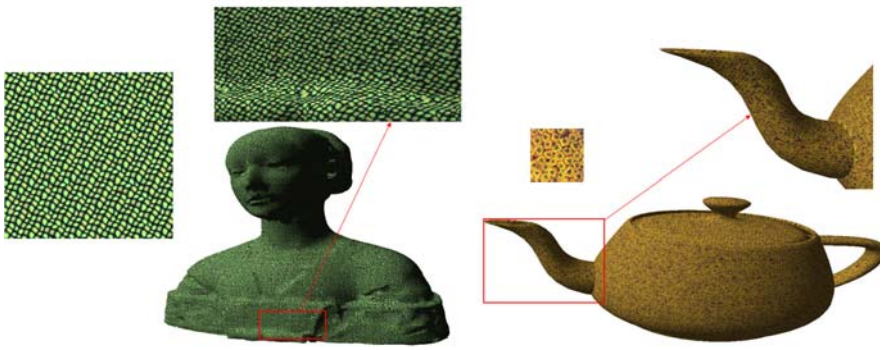
(a) (b)

Fig. 13. (a) Input texture T_2 and the polycube of the teapot model with tiled textures and (b) mapping result with a zoomed-in view.



(a) (b)

Fig. 14. (a) Input texture T_3 and the polycube of the Daba model with tiled textures and (b) mapping result of the teapot model with a zoomed-in view.



(a) (b)

Fig. 15. (a) input texture T_4 and mapping result of Laruana model with a zoomed-in view, and (b) input texture T_5 and mapping result of Laruana model with a zoomed-in view.

For the second part, two regular or structured textures were tested. First, the inputs were a 400 × 400 texture T_4 and the Laruana model. The size of a tile was set as 64 × 64. The number of unit cubes was 3657. The number of surfaces on the polycube was 6030. The number of different tiles was 940. The time for constructing the polycube with tiles was about 10 minutes and 40 seconds. The time for mapping from the polycube to the model was 47 seconds. Second, the inputs were a 128 × 128 texture T_5 and the Teapot model. The size of a tile was set as 64 × 64. The number of unit cubes was 1425. The number of surfaces on the polycube was 2568. The number of different tiles was 679. The time for constructing the polycube with tiles was about 8 minutes and 35 seconds. The time for mapping from the polycube to the model was 14 seconds. Fig. 15 shows (a) the input texture and the mapping result of Laruana model with a zoomed-in view, and (b) the input texture and the mapping result of Laruana model with a zoomed-in view.

There were some obvious distortions in the results. These distortions were generally caused by the configurations of Type 5, Type 3 and Type 6. If the size of a unit cube decreased, the shape of the polycube would be close to the model. Then the distortions on the model were reduced. However, the size of a tile could not be too small because a tile should contain the complete structure of a texture. Also, the proposed approach was not suitable for some models which had a portion with a large curvature. In the experiments, some cubes intersected only a few regions of the model. The textures were severely squashed when mapped on these regions. To solve these problems, the size of a unit cube was reduced so that the shape of the polycube was close to that of the model. The distortion on the model was reduced simultaneously.

The proposed approach was theoretically compared with the previous method [26]. Table 2 shows the comparison of the proposed approach and the previous method. In the proposed approach, a polycube of a 3D model is constructed automatically. Users do not need to be more involved and spend more time. Also, the size of a unit cube is small and the number of unit cubes is large. In the previous method, users had to be more involved and more time was needed to construct a polycube of a 3D model. Also, the inverse warping technique is used before mapping textures on the model. Finally, the size of a unit cube is large and the number of unit cubes is small.

Table 2. Comparison of the proposed approach and the previous method.

	The proposed approach	The previous method
Polycube construction	Automatic	User intervention
Inverse warping	No	Yes
Size of a unit cube	Small	Big
Number of unit cubes	Many	Few

6. CONCLUSIONS

In this paper, an approach to automatically map textures onto 3D models has been presented. The proposed approach consists of two important processes: automatic polycube construction and cell mapping. In the first process, the triangle-cube intersection algorithm was used to construct the polycube of a model. Then textures were tiled seamlessly on the polycube. In the second process, cells of the polycell were processed separately. The textures of the cell were mapped on the model according to its mapping directions. Therefore, users can easily generate texture mapping on the model.

In the future, another method of inverse warping [26] can be used to reduce the distortion. Also, the proposed automatic polycube-based technique can be combined with geometry image approaches [13, 31] to generate a general quad-like remeshing.

REFERENCES

1. A. Watt, *3D Computer Graphics*, 3rd ed., Addison Wesley, 2000, pp. 245-247.
2. N. A. Carr and J. C. Hart, "Meshed atlases for real-time procedural solid texturing," *ACM Transactions on Graphics*, Vol. 21, 2002, pp. 106-131.
3. P. Cignoni, C. Montani, C. Rocchini, R. Scopigno, and M. Tarini, "Preserving attribute values on simplified meshes by resampling detail textures," *The Visual Computer*, Vol. 15, 1999, pp. 519-539.
4. M. F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang tiles for image and texture generation," *ACM Transactions on Graphics*, Vol. 22, 2003, pp. 287-294.
5. K. Culik II, "An aperiodic set of 13 Wang tiles," *Discrete Mathematics*, Vol. 160, 1996, pp. 245-251.
6. F. Dong, H. Lin, and G. Clapworthy, "Cutting and pasting irregularly shaped patches for texture synthesis," *Computer Graphics Forum*, Vol. 24, 2005, pp. 17-26.
7. W. Dong, N. Zhou, and J. C. Paul, "Optimized tile-based texture synthesis," in *Proceedings of Graphics Interface*, 2007, pp. 249-256.
8. V. Douglas, "Triangle-cube intersection," *Graphics Gems III*, 1992, pp. 236-239.
9. A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of ACM SIGGRAPH*, 2001, pp. 341-346.
10. C. W. Fu and M. K. Leung, "Texture tiling on arbitrary topological surfaces," in *Proceedings of Eurographics Symposium on Rendering*, 2005, pp. 99-104.
11. C. M. Grimm, "Simple manifolds for surface modeling and parameterization," in *Proceedings of Shape Modeling International*, 2002, pp. 237-244.

12. B. Grünbaum and G. C. Shephard, *Tilings and Patterns*, W. H. Freeman & Co., New York, 1986.
13. X. Gu, S. Gortler, and H. Hoppe, "Geometry images," in *Proceedings of ACM SIGGRAPH*, 2002, pp. 355-361.
14. A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin, "Image analogies," in *Proceedings of ACM SIGGRAPH*, 2001, pp. 327-340.
15. V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM Transactions on Graphics*, Vol. 24, 2005, pp. 795-802.
16. V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobicks, "Graphcut textures: Image and video synthesis using graph cuts," in *Proceedings of ACM Transactions on Graphics*, 2003, pp. 277-286.
17. Y. Y. Lai and W. K. Tai, "Transition texture synthesis," *Journal of Computer Science and Technology*, Vol. 23, 2008, pp. 280-289.
18. B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," in *Proceedings of ACM Transactions on Graphics*, Vol. 21, 2002, pp. 362-371.
19. L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transactions on Graphics*, Vol. 20, 2001, pp. 127-150.
20. J. Maillot, H. Yahia, and A. Verroust, "Interactive texture mapping," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, 1993, pp. 27-34.
21. D. Piponi and G. Borshukov, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, 2000, pp. 471-478.
22. P. Sander, Z. Wood, S. J. Gortler, J. Snyner, and H. Hoppe, "Multi-chart geometry images," in *Proceedings of the Symposium on Geometry Processing*, 2003, pp. 146-155.
23. A. Sheffer and J. C. Hart, "Seamster: Inconspicuous low-distortion texture seam layout," in *Proceedings of the Conference on Visualization*, 2002, pp. 291-298.
24. O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, "Bounded-distortion piecewise mesh parameterization," in *Proceedings of the Conference on Visualization*, 2002, pp. 355-362.
25. J. Stam, "Aperiodic texture mapping," Technical Report, No. R046, European Research Consortium for Informatics and Mathematics, 1997.
26. M. Tarini, K. Hormann, P. Cignoni, and C. Montani, "Polycube-maps," in *Proceedings of ACM SIGGRAPH*, Vol. 23, 2004, pp. 853-860.
27. H. Wang, "Proving theorems by pattern recognition II," *Bell Systems Technical Journal*, Vol. 40, 1961, pp. 1-42.
28. H. Wang, "Games, logic, and computers," *Scientific American*, Vol. 213, 1965, pp. 98-106.
29. H. Wang, Y. He, X. Li, X. Gu, and H. Qin, "Polycube splines," in *Proceedings of Symposium on Solid and Physical Modeling*, 2007, pp. 241-251.
30. Q. Wu and Y. Yu, "Feature matching and deformation for texture synthesis," in *Proceedings of ACM Transactions on Graphics*, Vol. 23, 2004, pp. 364-367.
31. C. Y. Yao and T. Y. Lee, "Adaptive geometry image," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 14, 2008, pp. 948-960.

Chin-Chen Chang (張勤振) received his Ph.D. degree in Computer and Information Science from National Chiao Tung University in Hsinchu, Taiwan, in 1998. He is currently an Associate Professor in the Department of Computer Science and Information Engineering, National United University in Miaoli, Taiwan. His current research interests include computer graphics, image synthesis and digital multimedia.

Chen-Yu Lin (林震雨) received his M.S. degree in Institute of Computer Science and Engineering from National Chiao Tung University in Hsinchu, Taiwan, in 2007. His current research interests include computer graphics, texture synthesis and image synthesis.