CAD/Graphics'2001
August 22-24, Kunming
International Academic Publishers

TEXTURE MAPPING ON SUBDIVISION SURFACES USING MULTIPLE IMAGES

Neo Chee Seng and Huang Zhiyong
School of Computing, National University of Singapore, Singapore 117543
E-mail: {neochees | huangzy@comp.nus.edu.sg}

ABSTRACT

Subdivision surfaces are important in geometric modeling. However, not much work has been done in texture mapping on subdivision surfaces. In this paper, we propose a texture mapping method on subdivision surfaces using multiple images. We have implemented our method on Doo-Sabin subdivision surfaces. Our solution is based on the further splitting. In a subdivision process, a new face of control meshes may fall into a region of multiple texture maps. To correctly texture map on such face, we need to further split it into multiple parts so that each part falls into a region of only one texture map.

SUBJECT DESCRIPTION: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Surfaces Representation

KEYWORDS: Texture mapping, Doo-Sabin subdivision surfaces, Catmull-Clark subdivision surfaces, Multiple images

1. INTRODUCTION

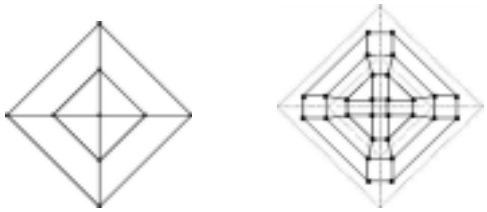
Subdivision surfaces are important in geometric modeling [1]. Modeling with subdivision surfaces, complex models can be modeled with the efficiency of polygons and the smoothness of NURBS and other spline surfaces without trimming. Texture mapping is an important technique in computer graphics [5]. A major texture mapping method on subdivision surfaces was proposed by DeRose et al. [2] and implemented on Catmull-Clark subdivision surfaces using only one image as the texture map. In this paper, we extend the algorithm to texture mapping on subdivision surfaces using multiple images. We implemented our method on Doo-Sabin subdivision surfaces because their new faces of control meshes may fall into a region of multiple texture maps. To correctly texture map on such face, we need to further split the new faces into multiple parts so that each part falls into a region of only one texture map. We have addressed the problem of maintaining smoothness of texture between any adjacent faces because of the use of multiple images. A recent work [8] also addressed the similar problem but it took a different approach.

The remaining of the paper is organized as follows: Section 2 describes the background. Section 3 presents the major techniques of our work. Section 4 shows the results. Section 5 concludes the paper.

2. BACKGROUND

Subdivision surfaces have been studied for about 20 years for representing complex surfaces. The first two schemes were proposed by Catmull and Clark [1] and Doo and Sabin [3]. Different subdivision surfaces proposed later include Loop [7] and Butterfly scheme [4]. Recently, these techniques have received more attention in computer graphics [6] because of the many benefits of subdivision.

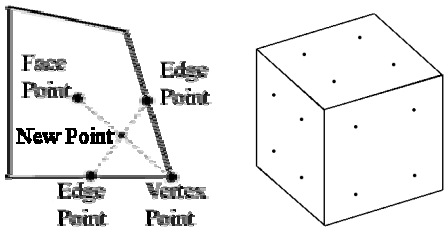
We briefly describe the Doo-Sabin subdivision surfaces because it will introduce the problem for the texture mapping using more than one image: the new faces can fall into a region of multiple texture maps. Doo-Sabin scheme is presented as a generalization of a recursive bi-quadratic B-splines patch subdivision algorithm. For non-rectangular meshes, it generates surfaces that reduce to a standard B-spline surface except at a small number of points, called extraordinary points. The limit surface of the scheme is C^1 continuous except at extraordinary points. The scheme works on all meshes regardless of their topology (Figure 2.1).



(a) (b)

Figure 2.1: One refinement using Doo-Sabin scheme.

We use a cube for initial mesh (Figure 2.2). For each subdivision, the new vertex can be derived from (1) the average of four particular points taken in a polygon - the vertex for which the new point is being defined, (2) the two edge points (the midpoints of the edges that are adjacent to this vertex in the polygon), or (3) the face point (the average of all the points in the polygon).

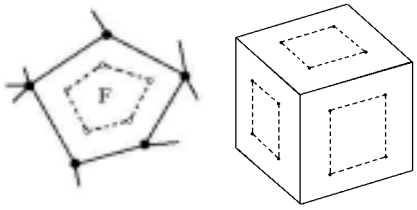


(a) (b)

Figure 2.2: (a) New point is the average of 2 edge points, 1 face point, and 1 vertex point. (b) New points calculated on the cube.

There are three types of faces formed from subdivision: F-faces, E-faces, and V-faces. They are directly related to our texture mapping.

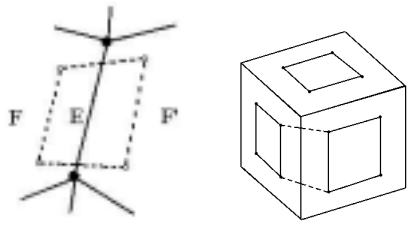
F-faces (Figure 2.3): For each n-sided face F in the original polyhedron, linking the new vertices of F forms a new n-sided face. This face is called type F-face (corresponding to a face).



(a) (b)

Figure 2.3: (a) Forming a F-face on a pentagon face. (b) Forming F-faces on the cube.

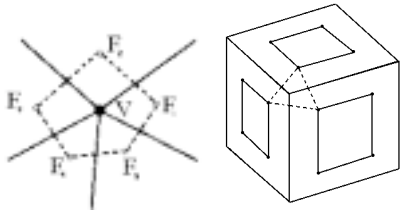
E-faces (Figure 2.4): For each edge E common to two faces F and F', a new 4-sided face is made by linking the images of the end vertices of E on the faces F and F'. This type of new face is termed a type E-face (corresponding to an edge).



(a) (b)

Figure 2.4: (a) Forming an E-face along an edge. (b) Forming an E-face on the cube.

V-faces (Figure 2.5): For each n-spoked (n>2) vertex V, where n faces meet, a new n-sided face is formed by linking the new vertices formed by V on the faces meeting at V. This type of new face is termed a type V-face (corresponding to a vertex).



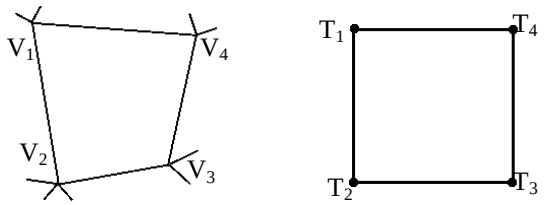
(a) (b)

Figure 2.5: (a) Forming a V-face from a vertex. (b) Forming a V-face on the cube

Texture mapping on subdivision surfaces include [2] and [8]. We discuss the method of DeRose et al. [2] on which our method is based. The goal is the construction of smooth texture coordinates for Catmull-Clark surfaces. Beginning with user supplied texture coordinates at some coarse level, the method subdivides these parameter assignments to the finest subdivision level using the same subdivision operator for texture coordinates as for the geometric coordinates of the vertices.

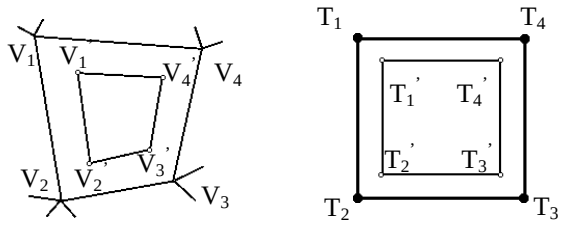
In [2], DeRose et al. have prove that smoothly varying texture coordinates result if the texture coordinates(s, t) assigned to the control vertices are subdivided using the same subdivision rules as used for the geometric coordinates (x, y, z). In other words, control point positions and subdivision can be thought of as taking place in a 5-space consisting of (x, y, z, s, t) coordinates.

For example, in Figure 2.6, a vertex V_i has the corresponding texture coordinate T_i. During the first refinement, all V_i' will be calculated from V_i and all T_i' will be calculated from T_i using the same subdivision rules (Figure 2.7).



(a) Geometric Space (b) Texture space

Figure 2.6: Vertex V_i has the corresponding Texture coordinates T_i.



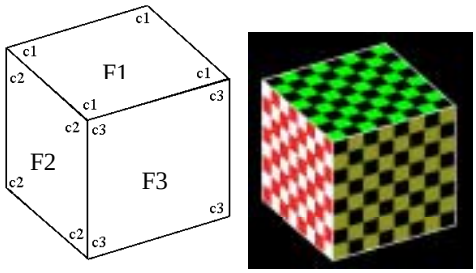
(a) Geometric Space (b) Texture space

Figure 2.7: After one refinement, Vertex V_i' has the corresponding texture coordinate T_i' calculated using the same subdivision rule.

3. TEXTURE MAPPING ON DOO-SABIN SUBDIVISION SURFACES

In this section, we describe our work of texture mapping on Doo-Sabin subdivision surfaces using multiple images. The major idea of our texture mapping method is the individual treatment of the F-faces, E-faces and V-faces. The problems or difficulties mainly lie on the use of multiple images as the texture maps for the faces of the initial mesh.

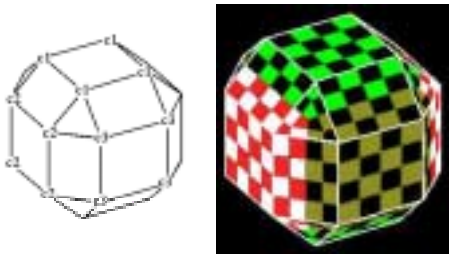
Without the loss of generality, a cube is used as the initial shape throughout all subsections. Each face is mapped with six chessboard images in different colors respectively (Figure 3.1). For example, in the initial polygon mesh, the faces F1, F2, and F3 are mapped with texture maps c1, c2, and c3 respectively.



(a) (b)

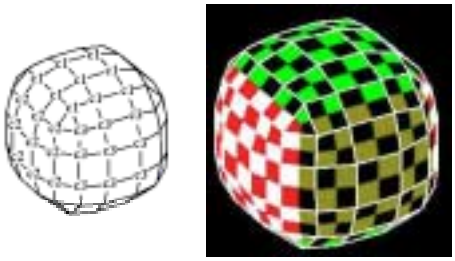
Figure 3.1: Faces of the control mesh are mapped with different texture maps.

After subdivision of the control mesh, the texture coordinates of the new vertices can be derived from their parents. The results after the first and second subdivisions are shown in Figures 3.2 and 3.3 respectively.



(a) (b)

Figure 3.2: Assignment of color ids after first refinement.



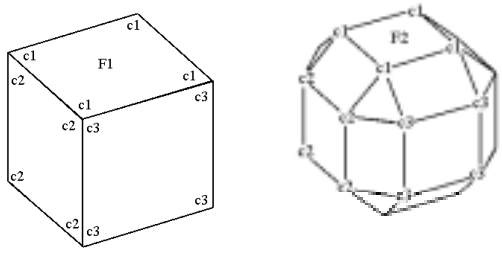
(a) (b)

Figure 3.3: Assignment of color ids after second refinement.

Now we can describe our method for texture mapping on Doo-Sabin subdivision surfaces. It is based on the different treatments to F-face, E-face and V-face, the only three types of the faces after each subdivision. When a new face falls into a region with multiple texture maps, this face will be split so that each part can be mapped with one texture map. The details are described from 3.1 to 3.3.

3.1 Texture mapping on F-faces

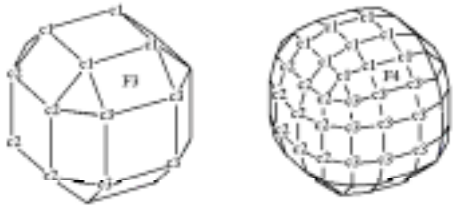
After subdivision, the new vertex of a F-face can fall into a region of single, two, or multiple texture maps depending on the previous control mesh. The easiest case is the new face falls into a region of one texture map completely. The texture coordinates of the new vertices are derived from the texture coordinates of the old vertex coordinates using the same subdivision rule because they use the same texture map, e.g., the texture map c1 as illustrated in Figure 3.4.



(a) (b)

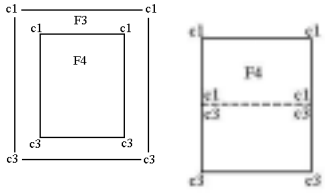
Figure 3.4: After one refinement, F1 will form F2, which is mapped with the same image c1.

The case of a new F-face falling into a region of two texture maps is illustrated in Figures 3.5 and 3.6. To derive the texture coordinates of the vertices of the new F-face, we need to split it, e.g., F4 in the illustration, into two parts so that each part falls into a region of only one texture map (Figure 3.6 (b)). Then, the texture coordinates of the new vertices are derived from the texture coordinates of the old vertex coordinates using the same subdivision rule separately with two texture maps c1 and c3.



(a) (b)

Figure 3.5: After one refinement, F3 will form F4.



(a) (b)

Figure 3.6: (a) F4 formed by face F3. (b) Splitting of new E-face into two equal parts.

We describe how a face is split now. In Figure 3.7, each vertex is represented as (geometric coordinates, texture coordinate) pair. Coordinates u0 to u3 are defined in the texture map c1 and v0 to v3 are defined in c3.

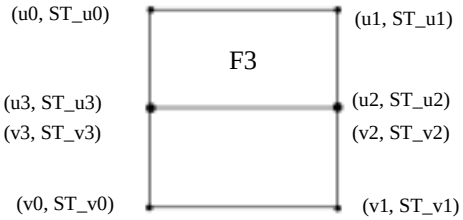
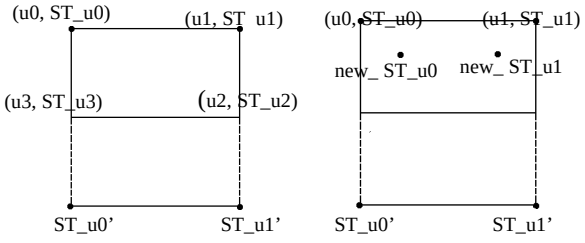


Figure 3.7: Each point is represented as (geometric coordinates, texture coordinate) pair.

Since direct calculation can be done if we have all the texture coordinates coming from a same texture map, we can calculate two extra texture coordinates so that we can perform direct calculation. The two extra texture coordinates for each texture map are calculated as follows (Figure 3.8):

ST_u0' = ST_u0 + (distance(v0, v3) / distance(u0, u3) + 1) * (ST_u3 – ST_u0),

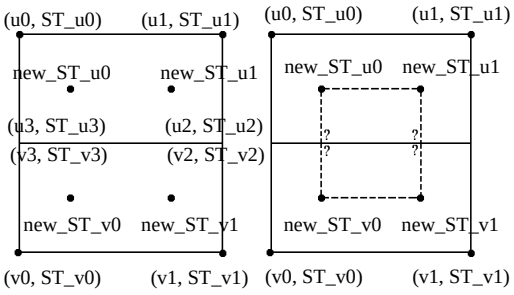
ST_u1' = ST_u1 + (distance(v1, v2) / distance(u1, u2) + 1) * (ST_u2 – ST_u1).



(a) (b)

Figure 3.8: (a) Calculation of ST_u0' and ST_u1'. (b) Calculation of new_ST_u0 and new_ST_u1.

Calculations for new_ST_v0 and new_ST_v1 are done in the same way as new_ST_u0 and new_ST_u1. Since we have four new texture coordinates, we still have to find four more texture coordinates (represented by '?" in Figure 3.9b), two for each texture map in order to map two texture maps on this face.



(a) (b)

Figure 3.9: (a) Four new texture coordinates formed. (b) Two more coordinates for each texture map are needed.

The calculation for new_ST_u3 is as follows:

$\vec{a}$ = ST_u2 – ST_u3,

$\vec{b}$ = new_ST_u0 – ST_u3,

new_ST_u3 = ST_u3 + ($|\vec{a} . \vec{b}|/|\vec{a}|^2$) $\vec{a}$.

The calculation is similar for new_ST_u2 and other two texture coordinates from another texture map (Figure 3.10).

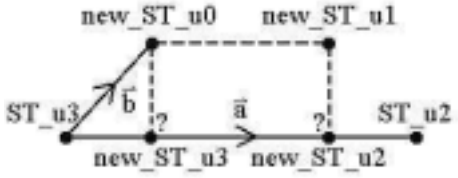
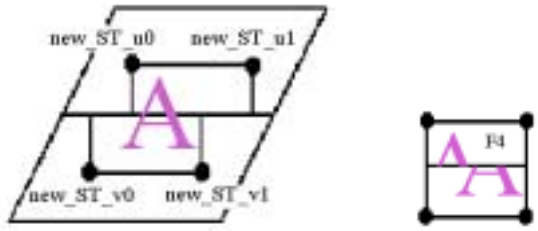


Figure 3.10: Calculation diagram for new_ST_u3 and new_ST_u2.

The above calculation is correct for regular texture mapping (that means the texture coordinates form a rectangle). The technique will fail for irregular texture mapping as shown in Figure 3.11. Hence we need to adjust the texture-coordinates to achieve smooth and continuous texture mapping on the face.



(a) (b)

Figure 3.11: (a) Discontinuity for irregular texture mapping. (b). Discontinuity in texture mapping on F4.

In order to solve the problem, we need to calculate the texture coordinates final_new_ST_u3, final_new_ST_u2, final_new_ST_v3, and final_new_ST_v2 as shown in Figure 3.12.

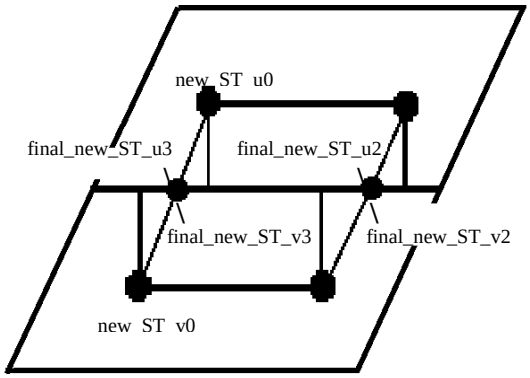


Figure 3.12: Position of new texture coordinates to maintain continuity.

The calculation for final_new_ST_u3 is as follows (Figure 3.13):

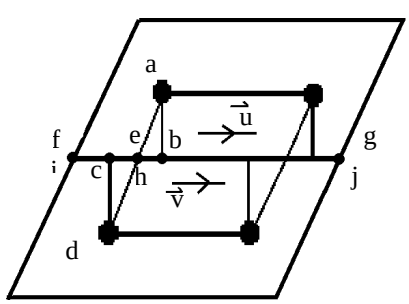
u = (g – f) / distance(f, g),
v = (j – i) / distance(i, j),

norm_dist(a, b) = distance(a, b) / distance(f, g),
norm_dist(c, d) = distance(c, d) / distance(i, j),
norm_dist(b, c) = distance(c, f) / distance(i, j) – distance(b, f) / distance(f, g),

norm_dist(b, h) = (norm_dist(a, b) * norm_dist(b, c)) / (norm_dist(a, b) + norm_dist(c, d),

e = b + u * norm_dist(b, h) * distance(f, g),
h = c – v *(norm_dist(b, c) - norm_dist(b, h)) * distance(i, j).

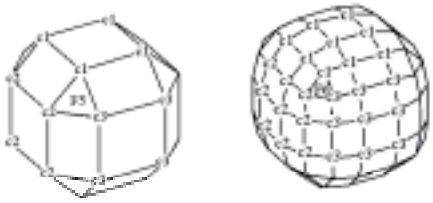
The calculations are similar for final_new_ST_u2, final_new_ST_v2, and final_new_ST_v3.



a: new_ST_u0
b: new_ST_u3
c: new_ST_v3
d: new_ST_v0
e: final_new_ST_u3
f: ST_u3
g: ST_u2
h: final_new_ST_v3
i: ST_v3
j: ST_v2

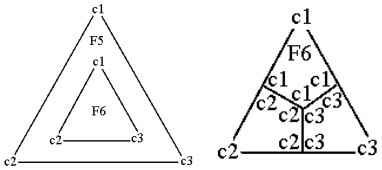
Figure 3.13: Calculation diagram for final_new_ST_u3.

Finally, we discuss the case of a F-face falling into a region of multiple texture maps (Figure 3.14). To derive the texture coordinates for the new vertices, we split the new F-face into three parts as shown in Figure 3.15b.



(a) (b)

Figure 3.14: After one refinement, F5 will form F6.

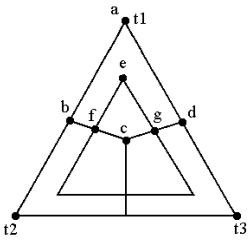


(a) (b)

Figure 3.15: (a) F6 formed by F5. (b) Splitting of new V-face into three parts.

The computation of the texture coordinates is illustrated in Figure 3.16. ST_u1 and ST_u3 can be computed on the E-Faces. The texture coordinates for this faces is calculated as follows:

new_ST_u0 = 0.25*(ST_u0 + ST_u1 + ST_u2 + ST_u3),
new_ST_u2 = ST_u2.

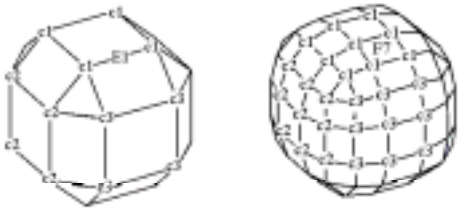


a: ST_u0
b: ST_u1
c: ST_u2, new_ST_u2
d: ST_u3
e: new_ST_u0
f: new_ST_u1
g: new_ST_u3

Figure 3.16: Calculation diagram for final_ST_u2.

3.2 Texture mapping on E-faces

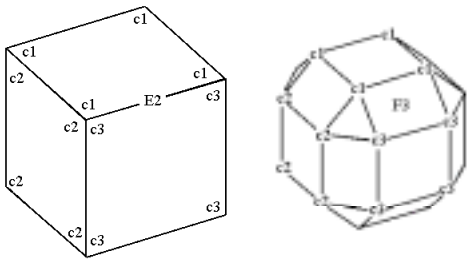
If the new E-face falls into a region of one texture map completely, the texture coordinates of the new vertices can be derived directly from the texture coordinates of the old vertex coordinates using the same subdivision rule on the texture coordinates of the old vertices in this texture map (F7 in Figure 3.17).



(a) (b)

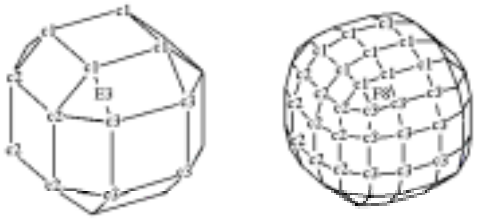
Figure 3.17: After one refinement, E1 will form F7

When the new E-face falls into a region of two different texture maps, e.g., c1 and c3 for F3 in Figure 3.18 and E3 in Figure 3.19. To compute the texture coordinates of the new vertices, we need to split the new E-face into two parts (Figure 3.20).



(a) (b)

Figure 3.18: After one refinement, E2 will form F3.



(a) (b)

Figure 3.19: After one refinement, E3 will form F8.

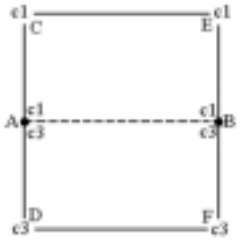


Figure 3.20: Splitting of new E-face into two parts.

As illustrated in Figure 3.21, for the new vertices a, b, c, d, e, and f, texture coordinates new_ST_u0, new_ST_u1 and new_ST_u2 can be derived directly. Then, the calculation for new_ST_u3 is as follows:

new_ST_u3 = new_ST_u0 + (distance(new_u0, new_u3) / distance(u0, u3)) * (f – e).

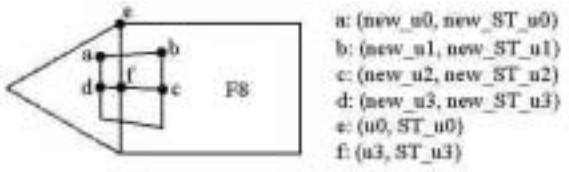
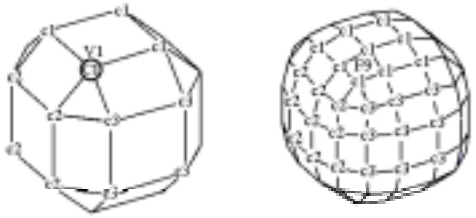


Figure 3.21: Calculation diagram for final_new_ST_u3.

3.3 Texture mapping on V-faces

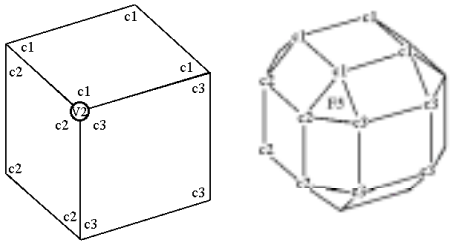
If a new V-face face falls into a region of only one texture map completely, the texture coordinates of new vertices can be derived from the texture coordinates of the old vertex coordinates using the same subdivision rule on the texture coordinates of the old vertices in the same texture map (F9 in Figure 3.22).



(a) (b)

Figure 3.22: After one refinement, V1 will form F9.

V-face will not fall into a region with two texture maps for a closed object. If it falls into a region with more than two texture maps, e.g., three texture maps c1, c2, and c3 for F5 as shown in Figure 3.23, to compute the texture coordinates of the new vertices, we need to split the new v-face into three parts as shown in Figure 3.24. The same subdivision is applied for each part using the related texture map.



(a) (b)

Figure 3.23: After one refinement, V2 will form F5.

Figure 3.24: Splitting of new V-face into three parts.

The new_ST_u2 is calculated from F-faces (Figure 3.25). new_ST_u1 and new_ST_u3 are calculated from E-faces. Hence we need to find out new_ST_u0.

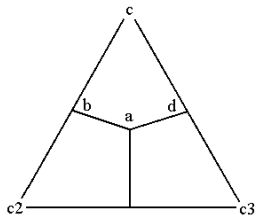
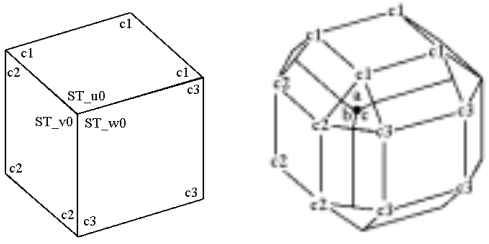


Figure 3.25: Calculation diagram for new_ST_u2, where a, b, c, and d refer to new_ST_u0, new_ST_u1, new_ST_u2, and new_ST_u3 respectively.

The number of multiple-textures faces is a consistent for any number of subdivisions, which is equal to the number of vertices in the initial starting mesh. For example, the cube has 8 vertices initially. After a subdivision refinement, the number of multiple-textured faces is 8. After the second refinement, the number of multiple-textured faces is still 8. With this property, we can store the initial texture coordinates ST_u0, ST_v0 and ST_w0 in Figure 3.26(a). These coordinates will be used as the center coordinates for multiple-texture face as shown in the figure 3.26(b). Hence we can use this value for new_ST_u0.



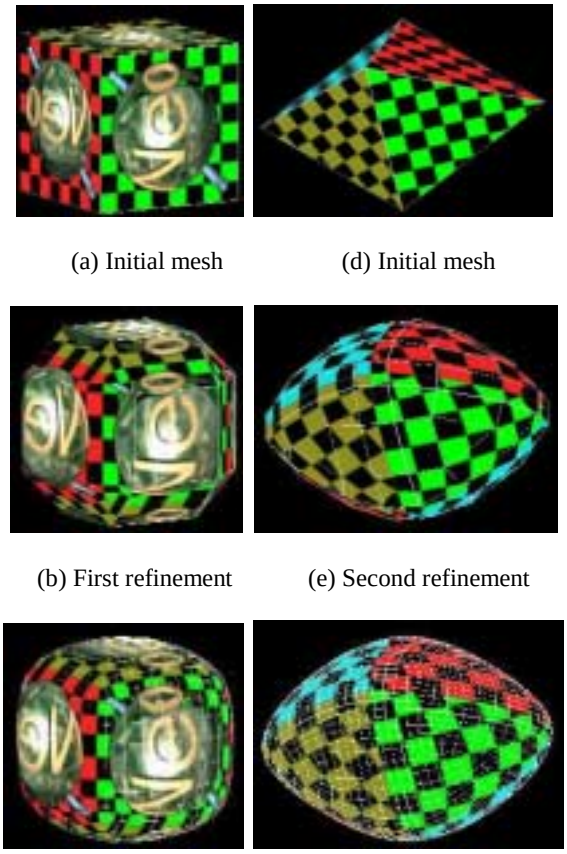
(a) (b)

Figure 3.26: (a) Initial texture coordinates for initial mesh. (b) Center texture coordinates for multiple texture mapped face, where a, b, and c refer to ST_u0, ST_v0, and ST_w0 of (a) respectively.

4. IMPLEMENTATION AND RESULTS

The coding is done on a Pentium III PC running on Windows NT using MS Visual C++ 6.0 and OpenGL libraries. We have implemented the texture-mapping algorithm for Doo-Sabin subdivision surfaces described in Section 3.

From examples shown in Figures 4.1 to 4.4, each mesh gets smoother after each refinement. We can see that the textures are continuous after each refinement.

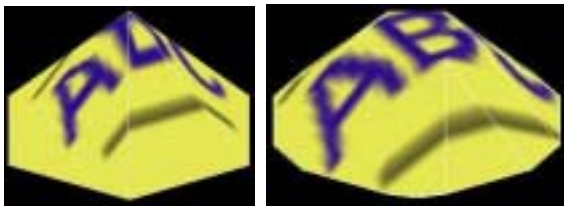


(a) Initial mesh (d) Initial mesh

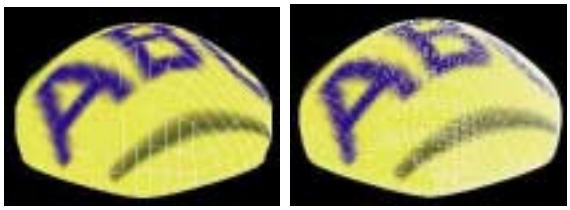
(b) First refinement (e) Second refinement

(c) Second refinement (f) Fourth refinement

Figure 4.1: Two examples (left and right columns) with six and eight texture maps respectively.

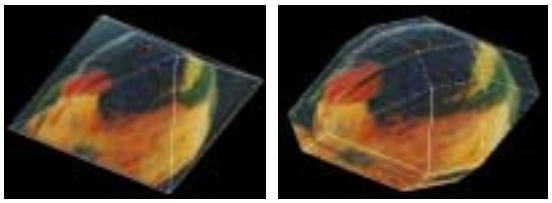


(a) Initial mesh (b) Second refinement

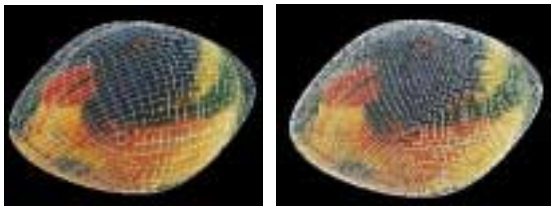


(c) Fourth refinement (d) Sixth refinement

Figure 4.2: Refinement done on a hexagon-based pyramid. Six texture maps are used.



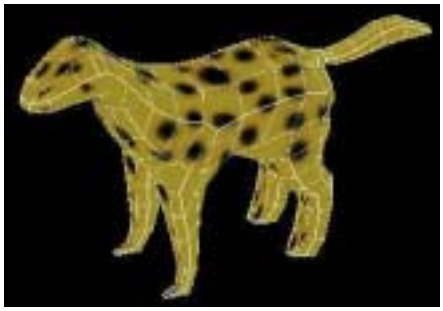
(a) Initial mesh (b) First refinement



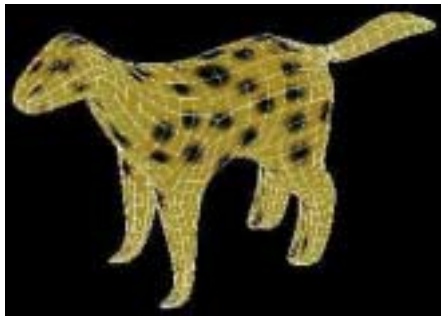
(c) Fourth refinement (d) Sixth refinement

Figure 4.3: Use a natural image but cut it into four pieces to map on the initial control mesh.

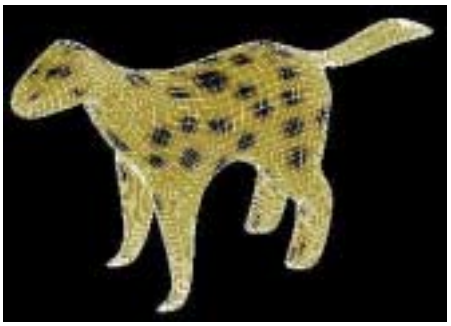
(a) Initial mesh.



(b) First refinement



(c) Second refinement



(d) Third refinement

Figure 4.4: Refinement done on an animal. Only one texture map is used.

5. CONCLUSIONS AND FUTURE WORK

We have proposed and implemented a method of texture mapping on subdivision surfaces using multiple images based on the idea of the further splitting. We have implemented our method on Doo-Sabin subdivision surfaces and shown some of the results.

In the current implementation, the algorithm takes a lot of time and memory after a few subdivision refinements. We are working to address the problem.

6. ACKNOWLEDGEMENT

This research was partly supported by NUS ARF R-252-000-051-112.

REFERENCES

[1] E Catmull and J Clark. *Recursively Generated B-spline Surfaces On Arbitrary Topological Meshes.* Computer-Aided Design, 10:350-355, 1978.

[2] T. DeRose, M. Kass, and T. Truong. *Subdivision Surfaces in Character Animation.* SIGGRAPH '98 Proceedings, 85-94, 1998.

[3] D Doo and M Sabin. *Behaviour Of Recursive Division Surfaces Near Extraordinary Points.* Computer-Aided Design, 10:356-360, 1978.

[4] N. Dyn, D. Levin, and J. A. Gregory. *A Butterfly Subdivision scheme for Surface Interpolation with Tension Control.* ACM Trans. Graphics. 9, 2 160-169, April 1990.

[5] P. S. Heckbert. *Survey of Texture Mapping.* IEEE Computer Graphics and Applications. 215-223, August 1990.

[6] L. Kobbelt. $\sqrt{3}$ *-Subdivision.* SIGGRAPH '00 Proceedings, 103-112, 2000.

[7] C Loop. *Smooth Subdivision Surfaces Based On Triangles.* Master's thesis, University of Utah, Dept. of Mathematics, 1987.

[8] D. Piponi and G. Borshukov. *Seamless Texture Mapping of Subdivision Surfaces by Model Pelting and Texture Blending.* SIGGRAPH '00 Proceedings, 471-478, 2000.