Star-Flower Subdivision

Cheng-Chih Fan-Chiang
Washington State University
2710 University Drive
Richland, WA 99352
patrickf@mail.wsu.edu

Robert R. Lewis
Washington State University
2710 University Drive
Richland, WA 99352
bobl@tricity.wsu.edu

ABSTRACT

This paper presents a new subdivision scheme called "star-flower subdivision". Like other well-known subdivision schemes, such as Catmull-Clark (based on quadrilaterals) and $\sqrt{3}$-subdivision (based on triangular meshes), it also generates smooth objects after a few iterations.

This scheme has advantages over previous schemes, however, due to a slower increase in polygon complexity. It subdivides quadrilateral meshes by a factor of 3, not 4, at each iteration. Having the number of facets increase more gradually allows designers to have greater control over the resolution of a refined mesh.

We call star-flower subdivision "semi-stationary" because different subdivision rules occur on alternating levels of refinement.

We also present a strategy to handle boundaries when applying the odd iterations of the scheme.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.6 [**Computer Graphics**]: Methodology and Techniques

General Terms

star-flower subdivision

Keywords

subdivision, geometric design, surface generation

1. INTRODUCTION

Subdivision is a popular technique for generating smooth curves and surfaces for geometric modeling and computer-aided design. The first subdivision scheme published by Catmull and Clark[2] in 1978 generates B-spline surfaces which guarantee C^2 continuity on regular quadrilateral meshes. Later that year, Doo and Sabin[4] analyzed the behaviour of such surfaces near extraordinary vertices (vertices whose valence does not equal 4).

Lately, the rôle of subdivision schemes has become more and more important in 3D modeling due to its use in animation[3]. Several famous schemes such as Loop[7] and Catmull-Clark[2] are now provided to modelers and geometric designers by commercial software such as Alias|Wavefront Maya^{TM}, SoftImage|XSI^{TM}, or NewTek LightWave 3D^{TM}, because subdivision possesses several attractive features:

Locality New vertices are created from nearby neighbors with certain weights, i. e. computing averages of neighboring vertices. The old vertices change their position by using simple smooth operations which also deal with their surrounding vertices. Hence, no global functions need to be solved.

Arbitrary Topology Subdivision generates piecewise patches from arbitrary initial meshes by a sequence of refinement. There is no need for trimming or maintaining smoothness or continuity between surfaces patches.

Efficiency and Simplicity Another great advantage of subdivision algorithms is that they only adopt a small number of rules, usually in the form of linear operations, to refine a set of vertices.

The goal of this paper is to present a new subdivision scheme based on quadrilateral meshes. While it shares features with older subdivision schemes, which we discuss in Section 2, it offers the advantage of slower, more controllable refinement. In Section 3, we describe all rules for implementing the semi-stationary scheme, which we call "star-flower subdivision". Section 4 describes the problem we encounter near boundaries and proposes a natural and simple strategy to deal with it. Section 5 compares the results for different subdivision schemes. Finally, in Section 6 we summarize the new scheme and mention some interesting directions for future work, some of which is ongoing.

2. RELATED WORK

There are two types of subdivision methods for arbitrary topology control meshes: approximating and interpolating. The easiest way to distinguish one from the other is to check

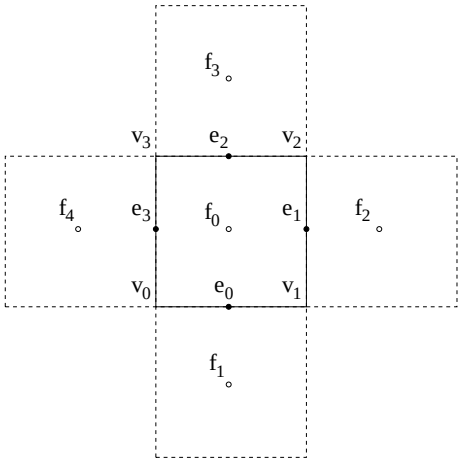

Figure 1: Notation for the Catmull-Clark scheme.

whether the position of the original vertices of a control mesh are changed or not on every iteration[11]. If the old vertices are updated by averaging their neighbors, the scheme is approximating. Otherwise, it is interpolating[12, 11].

In this section, we briefly mention the two approximating subdivision schemes which motivated us to develop our scheme.

2.1 Catmull-Clark Subdivision

One of the first subdivision schemes introduced by Catmull and Clark[2] is based on using quadrilaterals to generate B-spline surfaces on regular meshes. On a quadrilateral mesh, vertices whose valence is four are referred to as "ordinary" and other vertices are referred to as "extraordinary".

Several researchers have enhanced the original scheme with special features[5, 3] and analyzing its behavior around extraordinary vertices[4, 9, 1].

Briefly, the scheme is as follows:

1. Create face vertices from the mean of all old vertices v_i bounding each face as Figure 1 shown:

$$f_0 = \frac{v_0 + v_1 + v_2 + v_3}{4}$$

2. Create new edge vertices from the mean of two old vertices that define the edge and two new face vertices of the surfaces sharing the edge.

$$e_i = \frac{v_i + v_j + f_0 + f_{i+1}}{4}$$

where $j = (i + 1) \bmod 4$

3. Create new mesh vertices from the weighted mean of old vertices, midpoints of old edges, and new face vertices by certain weights.

$$v_0' = \frac{v_0 + 2E + (n-3)F}{n}$$

where n is the valence of v_0, E is the mean of all midpoints (e_i) of edges incident to v_0, and F is the mean of all new face vertices (f_i) of faces sharing v_0.

So each face is split into four subfaces after each iteration. The traditional subdivision methods are usually based on a 1-to-4 split operation, called a "dyadic split": after k iterations, each algorithm creates a mesh M_k from the original control meshes M_0 and the number of polygons in M_k will be 4^k times of that in M_0.

2.2 $\sqrt{3}$ Subdivision

A new subdivision scheme presented by Kobbelt[6] has a more gradual refinement at each iteration. This scheme, called $\sqrt{3}$ subdivision, is based on triangular meshes and splits every original triangle at each refinement into three subtriangles.

Because it generates nine sub-surfaces (a "triadic" split) after a double application, a single iteration is considered as the "root" of a triadic operation, hence the name. The rules for this scheme are:

1. Create new face vertices from the mean of all old vertices which bound a face:

$$f_0' = \frac{v_0 + v_1 + v_2}{3}$$

2. Create new vertices from the weighted mean of the old vertex and its nearest ("1 ring") neighbors:

$$v_i' = (1 - \alpha(n_i))\, v_i + \alpha(n_i) \frac{1}{n} \sum_{u \in N(v_i)} u$$

where

$$\alpha_n = \frac{4 - 2\cos\left(\frac{2\pi}{n}\right)}{9}$$

$N(v_i)$ is the set of all vertices immediately adjoining v_i (on the border of its 1-ring) and n_i is $\dim(N(v_i))$, the size of $N(v_i)$.

3. After inserting a new vertex for each face and connecting it to three (old) vertices which define the face, flip the original edges such that there are connections between each new face vertex and the new face vertices adjacent to it.

While it has much in common with other well-known stationary subdivision schemes, such as simplicity, locality, and efficiency, the advantage of this scheme is the slower increase of the mesh complexity.

3. THE STAR-FLOWER SUBDIVISION SCHEME

We here propose another triadic subdivision method that can generate smooth surfaces from meshes of arbitrary topology. After initialization, the rules of this scheme alternate: one set for odd and one set for even iterations.

3.1 Initialization

If the given mesh is not already a quadrilateral mesh, it needs to be converted to one by the following procedure, described in [10] and illustrated in Figure 2:

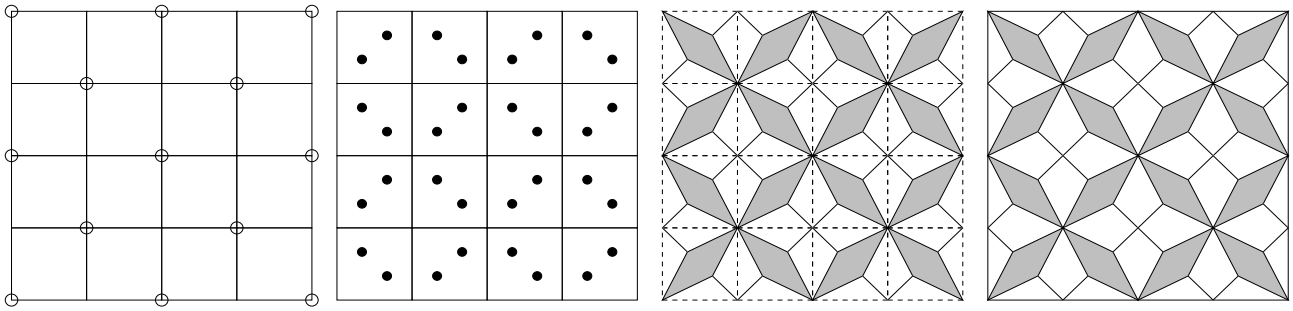

Figure 3: Star-Flower Subdivision – Odd Iterations. Flower vertices are circled in the leftmost diagram.

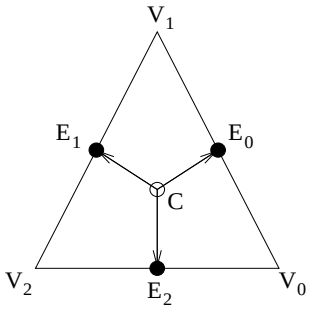

Figure 2: Converting an N-gonal Mesh to a Quadrilateral Mesh. The example shows the conversion of a triangle to three quadrilaterals, but the algorithm is suitable for arbitrary N.

1. Compute a midpoint (C) of each face by taking the mean of its surrounding vertices.
2. Compute the midpoint (E_i) of each edge surrounding that face.
3. Join each original vertex with vertices defined by the face midpoint and its adjoining edge midpoints.

Once we have a quadrilateral mesh, we designate half the vertices as "star" vertices and the other half as "flower" vertices so that each star vertex has only flower vertices as nearest neighbors and *vice versa*. If there were no extraordinary vertices and a regular geometry, these would be arranged like red and black squares on a checkerboard.

3.2 Odd Iterations

On this iteration, each face will create two new vertices we refer to as "odd". Each new vertex is connected topologically to one star vertex and two flower vertices as shown in Figure 4.

To create the odd vertices, we start with the mean of the surrounding four vertices. We assign this a weight of $\frac{2}{3}$ and combine it with a weight of $\frac{1}{3}$ for each star vertex. Hence,

$$\begin{aligned} O_0 &= \frac{1}{6}F_0 + \frac{1}{6}F_1 + \frac{1}{2}S_0 + \frac{1}{6}S_1 \qquad (1) \\ O_1 &= \frac{1}{6}F_0 + \frac{1}{6}F_1 + \frac{1}{6}S_0 + \frac{1}{2}S_1 \end{aligned}$$

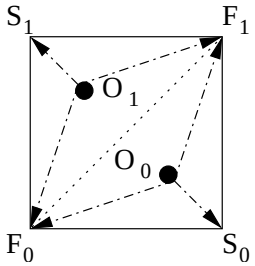

Figure 4: New Face Vertices Inserted During an Odd Iteration. Inserted vertices are shown as solid dots.

After the generation of the odd vertices, we replace each old (star or flower) vertex by a weighted average of its old value and the adjacent odd vertices:

$$\begin{aligned} F_i' &= (1-\alpha_{n_i^{OF}})F_i + \alpha_{n_i^{OF}} \frac{1}{n_i^{OF}} \sum_{u \in N^O(F_i)} u \qquad (2) \\ S_i' &= (1-\alpha_{n_i^{OS}})S_i + \alpha_{n_i^{OS}} \frac{1}{n_i^{OS}} \sum_{u \in N^O(S_i)} u \end{aligned}$$

where $i \in \{0,1\}$, $N^O(F_i)$ consists of O_i vertices in faces adjoining F_i, $n_i^{OF} = \dim(N^O(F_i))$, $n_i^{OS} = \dim(N^O(S_i))$, and we have intuitively chosen α_n to be $\frac{2}{n}$.

We need to preserve quadrilateral connectivity at every iteration, so we replace the old connectivity by connecting each new vertex to the flower vertices of the same face and also to the corresponding star vertex used in (2).

The whole procedure is shown in Figure 3.

3.3 Even Iterations

Due to the "distortion" of the mesh topology during odd iterations, the "even" iterations use quite similar rules to recover the desired scaled-down version of the original topology. To do this, we use the same overall rule as for the odd iteration: Insert two new vertices (which we now call "even" vertices) for each face.

Figure 5 illustrates the first phase. Two even vertices, E_0 and E_1, occur inside "flower petals", using the same weights

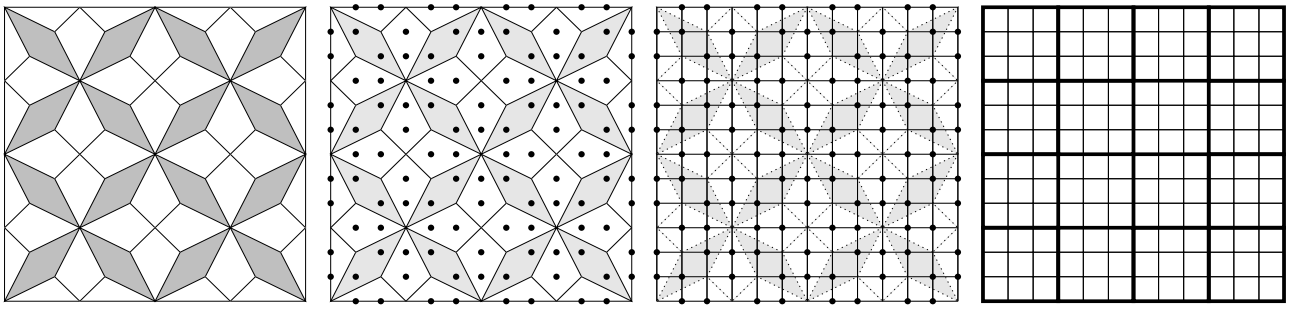

Figure 6: Star-Flower Subdivision – Even Iterations.

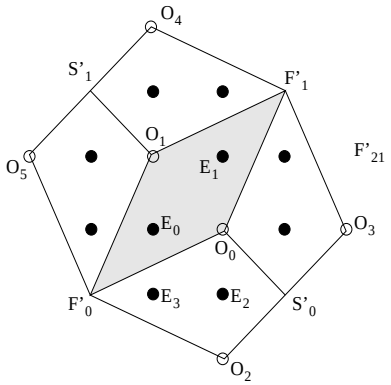

Figure 5: Inserting New Vertices and Faces During Even Iterations. Inserted vertices are shown as solid dots.

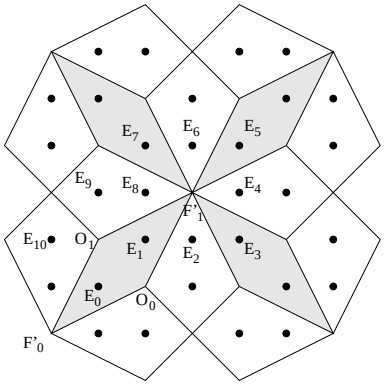

Figure 7: Replacing Old Vertices During Even Iterations.

as for the odd iteration:

$$\begin{aligned} E_0 &= \frac{1}{2}F_0' + \frac{1}{6}O_0 + \frac{1}{6}F_1' + \frac{1}{6}O_1 \\ E_1 &= \frac{1}{2}F_1' + \frac{1}{6}O_0 + \frac{1}{6}F_0' + \frac{1}{6}O_1 \end{aligned} \tag{3}$$

Two other even vertices occur inside "star points", again with the same weights as for the odd iteration:

$$\begin{aligned} E_2 &= \frac{1}{2}S_0' + \frac{1}{6}O_0 + \frac{1}{6}F_0' + \frac{1}{6}O_2 \\ E_3 &= \frac{1}{2}F_0' + \frac{1}{6}O_0 + \frac{1}{6}S_0' + \frac{1}{6}O_2 \end{aligned}$$

After getting the even vertices, we need to update each old (flower, star, and odd) vertex with the mean of the even vertices surrounding it. We separate the old vertices into two groups: odd vertices (O_i) and star/flower vertices (F_i' and S_i').

The updated star and flower vertices apply the same weights as during odd iterations:

$$\begin{aligned} F_1'' &= (1-\alpha_{n_i^F})F_1' + \alpha_{n_i^F}\frac{1}{n_i^F}\sum_{u\in N(F_i')} u \\ S_1'' &= (1-\alpha_{n_i^S})S_1' + \alpha_{n_i^S}\frac{1}{n_i^S}\sum_{u\in N(S_i')} u \end{aligned} \tag{4}$$

where $N(F_i')$ is the set of new vertices immediately adjoining F_i', n_i^F is $\dim(N(F_i'))$, $N(S_i')$ is the set of new vertices immediately adjoining S_i', n_i^S is $\dim(N(S_i'))$, and α_n is the same as for the odd iteration. Figure 7 shows these new vertices schematically.

Because the odd vertices have a fixed valence (3), we apply a special weight for them. We choose $\beta = \alpha_3 = \frac{2}{3}$ since O_0' has a valence of 3, but use it to weight the four even vertices that will become its neighbors:

$$O_1' = (1-\beta)O_0 + \beta\frac{1}{4}(E_0 + E_1 + E_9 + E_{10}) \tag{5}$$

Empirical results show that this leads to surfaces that become very smooth after a few iterations.

Finally, in order to regain the original connectivity, edge creation during even iterations is starting from all even vertices to their two (topological) vertical and two horizontal neighbors as shown in Figure 8. For example, in the "flower petal", E_0 connects horizontally to O_0' and E_8 and links vertically to O_1' and E_3. In the "star point", E_2 connects horizontally to S_0'' and E_3 and links vertically to O_0' and O_2'.

The whole procedure for even iterations of star-flower subdivision is illustrated in Figure 6. By performing the odd and even iterations successively, we can generate extremely smooth objects within a few iterations.

If further iterations are desired, all (new) odd and even vertices, together with (updated) star and flower vertices form a set of vertices on a quadrilateral mesh which must again

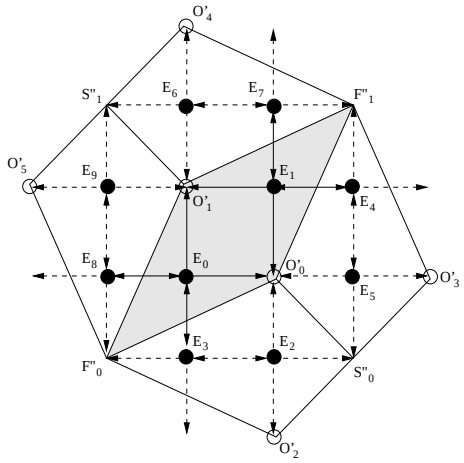

Figure 8: Reestablishing Connectivity for New Face Vertices During Even Iterations.

be partitioned into star flower vertices as described in Section 3.1. The iterations may then proceed.

3.4 Analysis

After a single pair of odd and even iterations, star-flower subdivision performs a triadic split, increasing the face count by a factor of 9. Contrast this with two iterations of the Catmull-Clark scheme, which increases the face count by a factor of 16. The smaller face factor implies a higher degree of controllability of the tradeoff between face count and smoothness.

In addition to the gradual refinement, star-flower subdivision also contains several features of other schemes:

Locality It is local: When generating new vertices or replacing old ones, the star-flower scheme only uses a small number of neighbors.

Arbitrary Topology The scheme works for vertices of any valence in closed meshes. (We will discuss meshes with boundaries in the next section.)

Efficiency and Simplicity Its rules are simple and symmetrical: Insert two new vertices and reconnect the topology on each iteration.

An additional feature, common with other quadrilateral mesh-based subdivision schemes, is the ability to take advantage of vertex sharing (i.e., triangle stripping) to draw each face as two triangles with four vertices without resorting to strip optimization algorithms.

4. BOUNDARIES

As we have shown, star-flower subdivision works quite well on closed meshes. We can refine such objects to any degree we like. In the case of objects with boundaries (or holes) however, there is a special situation near borders. Figure 9 illustrates it.

We start with a boundary face as indicated in the left diagram. As illustrated in the middle diagram, after an odd

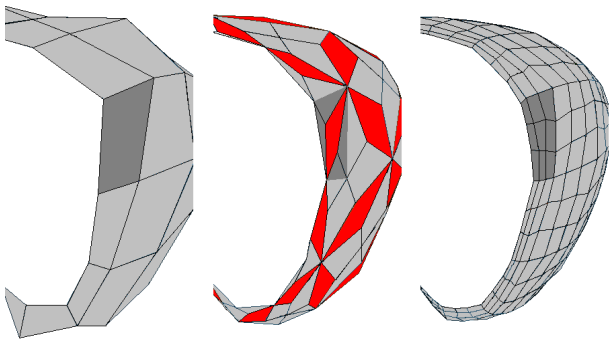

Figure 9: Star-Flower Subdivision Near Boundaries.

iteration, each old boundary face generates one new face (shown in red) and, depending on how many boundaries the original quadrilateral contains, one to three triangles, each of which has one edge on the boundary.

There is no way to find face-mates which share the boundaries, as they do not exist. Consequently, we can't generate any new face points so that the new boundary face has four vertices after odd iterations. The natural solution for this problem is to do nothing for new boundary surfaces during those iterations, i. e., preserve triangular faces and mark them as boundaries.

We can still recover a triadic split operation on even iterations by computing two new face points. Due to the missing face vertices that the odd iteration did not generate, the weights of new face points in the boundary faces are slightly revised:

$$\begin{aligned} E_0 &= \frac{5}{9}F_0' + \frac{2}{9}S_0' + \frac{2}{9}O_i \\ E_1 &= \frac{2}{9}F_0' + \frac{5}{9}S_0' + \frac{2}{9}O_i \end{aligned} \tag{6}$$

where F_0' and S_0' are boundary vertices and E_0 and E_1 are new face points, which are close to F_0' and S_0', respectively.

The right diagram of Figure 9 shows the result.

5. RESULTS

To demonstrate our scheme, we have applied it to several different objects. Figure 10 shows its application to a collection of simple meshes. To make a clearer comparison of star-flower and Catmull-Clark schemes, Figure 11 shows successive refinements of a wireframe cube (with hidden surfaces removed). Note the smaller polygon count for the star-flower scheme. Figure 12 compares star-flower with $\sqrt{3}$ subdivision applied to the canonical Stanford bunny.

6. CONCLUSIONS AND ONGOING WORK

We have proposed star-flower subdivision, a new semi-stationary subdivision scheme which generates smooth surfaces based on quadrilateral meshes, with a slower increase of polygon count upon refinement than previous work. At the same time, star-flower subdivision maintains the same advantages of locality, simplicity, efficiency, and arbitrary topology as previous schemes. We modeled several different smooth objects using this new scheme.

Although all of our experimental results show the qualitative smoothness of objects, analysis[8, 6] to prove the scheme generates at least C^2-continuous surfaces for regular vertices (those whose valence is equal to 4) is a necessary of future work, and is now under way. In addition, we will also attempt to prove C^1-continuity near extraordinary vertices, probably by using Fourier analysis[4, 1]. Additional requirements for practical application, such as sharp creases[3], surface texturing, and the modifications of weights for boundaries and old vertices are also part of our anticipated future work.

7. REFERENCES

[1] A. A. Ball and D. J. T. Storry. Conditions for tangent plane continuity over recursively generated b-spline surfaces. *ACM Transactions on Graphics*, 7(2):83–102, 1988.

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

[3] T. DeRose, M. Kass, and T. Truong. Subdivision surface in character animation. In E. Fiume, editor, *Proceedings of SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, pages 85–94. ACM, ACM Press / ACM SIGGRAPH, 1998.

[4] D. Doo and M. Sabin. Behavior of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):177–181, 1978.

[5] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of the 20th annual conference on Computer graphics*, Computer Graphics Proceedings, Annual Conference Series, pages 35–44. ACM, ACM Press, 1993.

[6] L. Kobbelt. $\sqrt{3}$ subdivision. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 103–112. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5.

[7] C. T. Loop. Smooth subdivision surfaces based on triangles. Master thesis, Department of Mathematics, University of Utah, 1987.

[8] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12:153–174, 1995.

[9] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 395–404, Orlando, Florida, July 1998. ACM SIGGRAPH / Addison Wesley. ISBN 0-89791-999-8.

[10] L. Velho and D. Zorin. 4-8 subdivision. *Computer Aided Geometric Design*, 18(5):397–427, June 2001. ISSN 0167-8396.

[11] D. Zorin and P. Schroder. Subdivision for modeling and animation. In *SIGGRAPH Course Notes*, 2000.

[12] D. Zorin, P. Schroder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of the 23rd annual conference on Computer graphics*, Computer Graphics Proceedings, Annual Conference Series, pages 189–192. ACM, ACM Press, 1996.

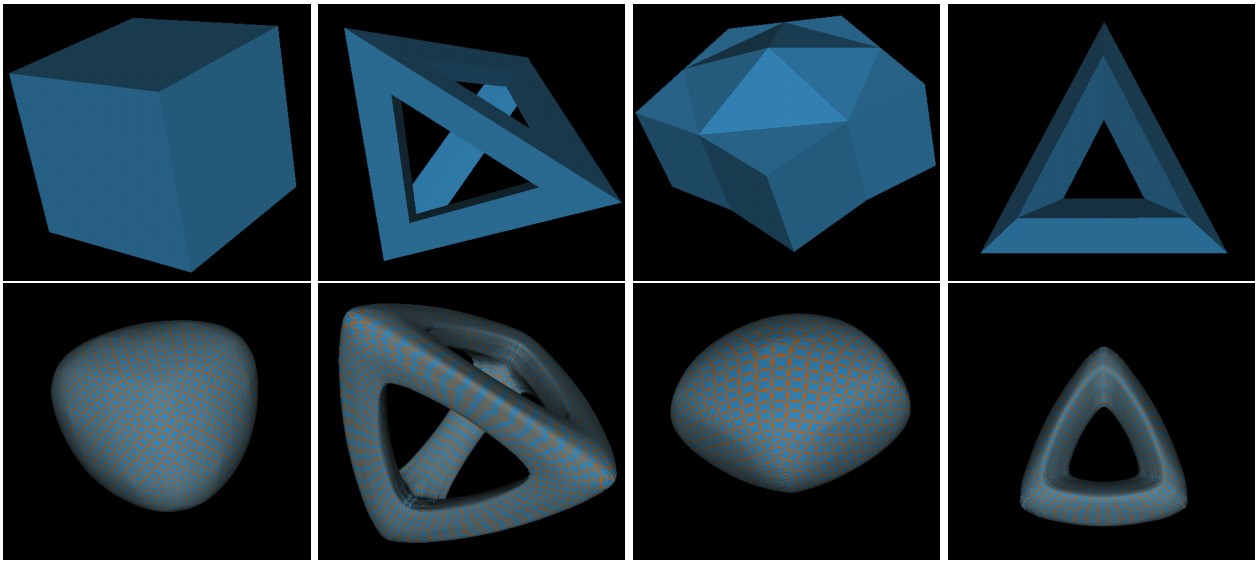

Figure 10: Application of Star-Flower Subdivision to Some Typical Meshes. The original meshes are on above and their star-flower subdivided results are below. The number of iterations from left to right are 7, 5, 5, and 5 respectively

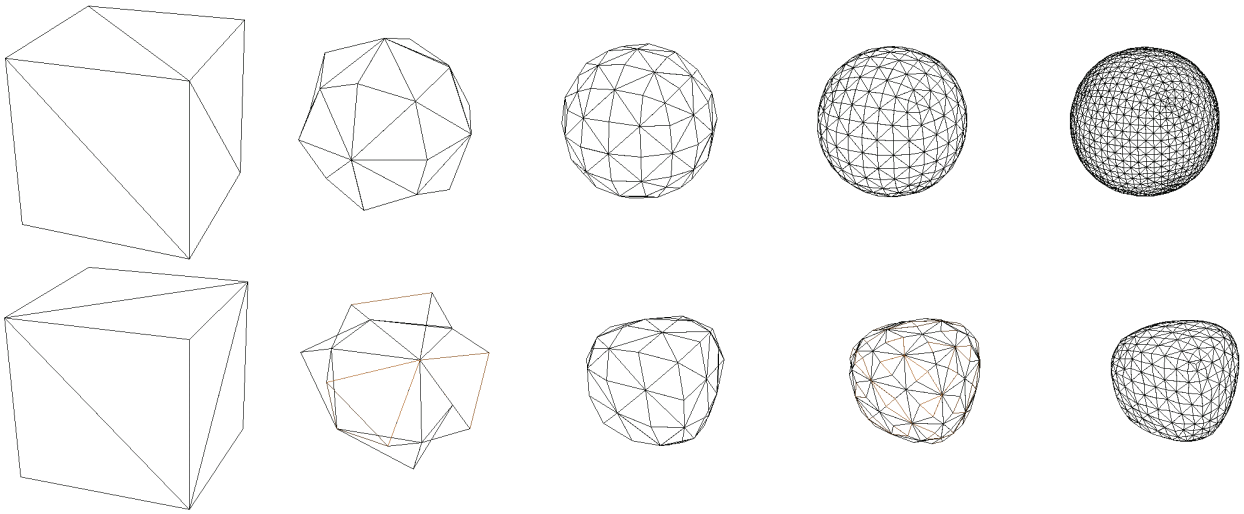

Figure 11: Comparision of the First Four Iterations of Catmull-Clark (top) and Star-Flower (bottom) Subdivision Schemes.

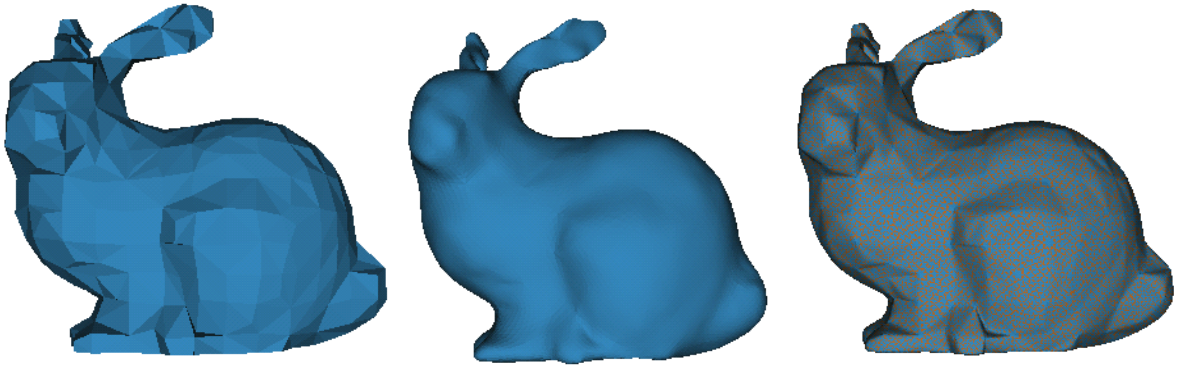

Figure 12: Refinements of the Stanford Bunny. Left: the original mesh. Middle: $\sqrt{3}$-subdivision after 3 iterations. Right: star-flower subdivision after 3 iterations.