Computational Aesthetics in Graphics, Visualization, and Imaging (2010)
O. Deussen and P. Jepp (Editors)

Subdivision Beyond Smoothness

Michael Hansmeyer

Department for CAAD - Institute for Technology in Architecture
Swiss Federal Institute of Technology (ETH)
Schafmattstr. 32, CH-8093, Zurich, Switzerland
hansmeyer@arch.ethz.ch

ABSTRACT

This paper examines the role that subdivision processes can have in the production of form. It explores how subdivision processes can generate complex geometries from very simple input meshes. In a first step, this paper presents modifications to the established subdivision processes' weighting schemes. In a second step, this paper considers formalized and extended methods for applying these schemes. Finally the paper presents forms generated with these modified processes.

I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations

1. Introduction

Subdivision processes have traditionally been used in computer graphics to generate smooth surfaces from a coarser polygonal mesh. They are currently used extensively in 3D modeling and in character animation. This paper examines these processes and explores to what extent they can be used specifically as generators of complex geometrical form.

As a generative mechanism, subdivision processes have the appeal that they are not additive processes. While additive processes frequently rearrange (and scale, translate, rotate) pre-defined components in a combinatorial manner, subdivision processes represent a purely operations-based approach. Rather than studying the possibilities in combining numerous primitives, they allow the exploration of form inherent in a single primitive (represented as the input mesh) given changes in the process parameters.

This paper presents modifications to two established subdivision algorithms. In a first step, the weighting schemes are adapted to incorporate additional parameters. In a second step, the manner and context in which these schemes can be applied is formalized and extended. Forms generated with these schemes are presented in the appendix.

1.1 Subdivision Explained

Subdivision processes take as an input a polygonal mesh. They recursively apply a subdivision scheme to this mesh to produce a denser, generally smoother, output mesh. Subdivision schemes have two parts: topological rules and weighting rules. The topological rules specify how to obtain the graph of the refined mesh from the graph of the input mesh by generating new vertices, edges and faces. The weighting rules specify how to calculate the positions of these new vertices based on interpolation between vertices of the input mesh.

Two of the most established subdivision schemes are the Catmull-Clark process [Cat74][CC78] and the Doo-Sabin process [Doo78][DS78]. Both of these processes generate smooth rounded forms when using their standard weighting rules.

2. Expanding the Subdivision Schemes

By altering the Catmull-Clark and Doo-Sabin processes, non-rounded forms with highly diverse attributes can be produced. Using a simple hexahedron as an input mesh can yield forms with features as diverse as concavity and convexity, the appearance of branching, porosity, and fractalization – just to name a few.

The proposed modification of the schemes consists entirely of additions to the weighting rules, while the topological rules remain unchanged. The weighting rules are amended by introducing parameters that allow a variable positioning of new vertices. These parameters are henceforth referred to as weights.

Traditionally both schemes' weighting rules calculate the position of new vertices strictly as an interpolation of previous-generation vertices. These rules are amended to allow the extrusion of vertices along face, edge and vertex normals. The two modified schemes are described in detail below.

DOI: 10.2312/COMPAESTH/COMPAESTH10/075-081

2.1 Extended Catmull-Clark Scheme

The Catmull Clark process generates k new quadrilateral faces for each face in the input mesh, where k is the number of vertices of the face. The following is a step-by-step explanation of the process using the example of a hexahedron as an input mesh:

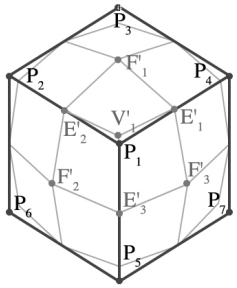

Figure 1: *Subdivided Catmull-Clark hexahedron: input mesh and first iteration mesh*

1. For each face, generate a new face point that is an average of the face's vertices. This point can be extruded along the face's normal vector n_f using weight w_{10}. The normal vectors can be scaled by the face's perimeter.

$$F_1' = (P_1 + P_2 + P_3 + P_4)/4 + \vec{n}_f \cdot w_{10} \quad (1)$$

2. For each edge, generate an edge midpoint that is a weighted interpolation of the new face points adjacent to the edge with the edge's endpoints. This edge midpoint can be extruded along the edge's normal vector n_e (defined as an average of the normal vectors of the attached faces) using weight w_{11}.

$$E_1' = ((F_1 + F_2)(1 + w_1) + (P_1 + P_4)(1 - w_1))/4 + \vec{n}_e \cdot w_{11} \quad (2)$$

3. For each initial vertex of the mesh, generate a new vertex point that is a weighted interpolation of the average F of all i face points touching the vertex with the average E of all edge midpoints for edges touching the vertex. The original vertex P factors into the equation when i exceeds 3. This new vertex can be extruded along the previous vertex's normal vector n_p using weight w_{12}.

$$V_1' = (F(1 + w_2) + 2E(1 - w_2/2) + (i - 3)P_1)/i + \vec{n}_p \cdot w_{12}. \quad (3)$$

4. Each new face point is connected to the new edge points of edges that made up the original face. Each new vertex point is connected to the new edge points of the original edges incident on the original vertex. The weights introduced above can be integrated into weighting stencils for the new face, edge, and vertex points.

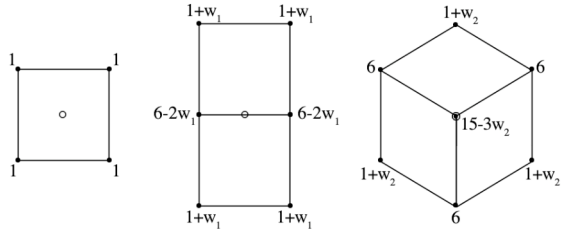

Figure 2: *Catmull-Clark stencils for face points, edge points, and vertex points with the introduction of weights in the latter two stencils. (Possible extrusion not shown)*

After one iteration of this subdivision algorithm, vertices produced can be distinguished as deriving from face midpoints, edge midpoints, or previous vertices. Thus additional weights can be added to equation (1) to control the placement of subsequent midpoint:

$$F_1'' = ((V_1'(1 + w_3) + F_1'(1 - w_3))(1 + w_4) + (E_1' + E_2')(1 - w_4))/4 + \vec{n}_f \cdot w_{10} \quad (4)$$

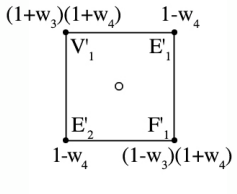

Figure 3: *Catmull-Clark face point stencil for use after first iteration*

2.2 Extended Doo-Sabin Scheme

The standard Doo-Sabin algorithm is modified in a similar manner to the Catmull-Clark algorithm. Weighting rules are amended by introducing parameters to control the position of vertices for each type of generated face. New vertices can be extruded along face and vertex normals. The process is as follows:

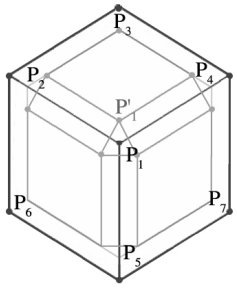

Figure 4: *Subdivided Doo-Sabin hexahedron – input mesh and first iteration mesh*

1. For each face, generate a new vertex point for each vertex of the face. This vertex's position is an interpolation of the face's midpoint and the original vertex position.

Formulas (5) and (6) are for quads and triangles respectively.

$$P_1' = (P_1 \cdot (2.25 + 2w_1) + (P_2 + P_4) \cdot (.75 - w_1) + .25P_3)/4 + \vec{n}_f \cdot w_{10} \quad (5)$$

$$P_1' = \frac{2}{3}P_1(1 + w_1/2) + \frac{1}{6}(P_2 + P_3)(1 - w_1) + \vec{n}_f \cdot w_{10} \quad (6)$$

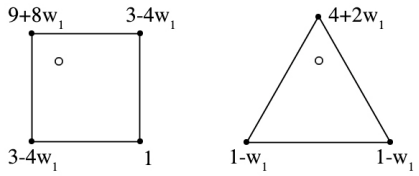

Figure 5: *Modified Doo-Sabin stencils for quads and triangles using variable weights.*

2. New faces are then formed in three steps. First, the new vertices within each face are connected to create a smaller, offset version of the face. Second, for each original vertex, the new vertices that have been generated in the faces incident to the vertex are connected. Finally, for each original edge, new vertices that have been generated in the faces adjacent to the edge are connected.

After one iteration of the algorithm, each new face can be distinguished as deriving from an original face, from connected edges, or from connected vertices. Distinct weights analogous to w_1 and w_2 can thus be introduced for these three groups of faces in subsequent generations.

The Doo-Sabin scheme can further be expanded to use additional weights for cases in which the valence of a vertex is not equal to the number of faces incident to it. (See the four rightmost configurations in figure 8 for examples of these cases.) In addition, conditional weights can be introduced to distinguish between internal and external edges.

2.3 Vertex Representation

Depending on the form of the input mesh and the weighting values, it is possible that two or more vertices are generated at identical positions. One has the option of treating these as either separate vertices, or of combining them into one vertex.

In the latter case, the linear relationship between changes to subdivision weights and changes to the subdivided output is disrupted for subsequent generations. The output takes on unexpected forms at certain weighting values. This effect can also be induced by specifying a minimum distance beneath which generated vertices are fused.

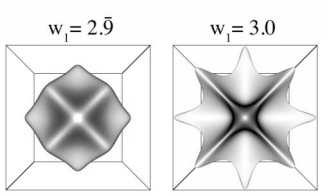

Figure 6: *Subdivision of a hexahedron, with values for w_1 in the first iteration of 2.99 and 3.0. The latter value causes the edge points to converge in the center, forming one single vertex and therefore significantly altering the shape.*

Further non-linearities can be introduced by limiting the number of edges (and thus faces) that can be attached to a vertex – particularly those vertices that have been fused. This can create porosity in the subdivided form.

3. Configuring the Subdivision Schemes

The previous section presented modifications to the subdivision schemes, primarily though the introduction of weighting parameters. This section presents ways in which these schemes can be applied to an input mesh. Specifically this section considers methods of specifying and using non-stationary and non-uniform weights to increase the scope of generatable forms.

3.1 Non-Stationary Weights

Non-Stationary weights imply that a weight can assume a distinct value at each iteration of the subdivision process. The nomenclature for these weights will be as follows: w^i_n, where n is the weight number (for either the Catmull-Clark or Doo-Sabin processes respectively), and i is the iteration starting at number 1. Either discrete values can be assigned to certain iterations, or a linear or exponential trend can be specified. For example:

$$w_n^i = a + b(i-1)^q \quad (7)$$

where *a* is an initial value and *b* represents an incremental change.

In addition to changing the weights from iteration to iteration, it is possible to change the subdivision scheme used at each generation. As the output from one scheme can serve as the input of the other, a single form can be produced by any combination of the Catmull-Clark and Doo-Sabin schemes.

3.2 Non-Uniform Weights

Non-uniform weights imply using distinct weighting values at different parts of the input mesh within one subdivision iteration. Weights are thus location-specific. Three methods of setting non-uniform weights are presented:

1. Variation based on position in the mesh's environment
2. Variation based on the mesh's topology and motives
3. Variation based on tagging vertices and faces

3.2.1 Variation Based on Position in the Mesh's Environment. This involves placing points that represent sets of subdivision weights in space, either at locations directly on a mesh or within the mesh's environment. For each face the distances to the sets are calculated to determine the sets' levels of influence. These levels of influence are themselves weighted values. The subdivision weights contained in each set are then weighted according to the sets' levels of influence at each face. This concept is shown in figure 7.

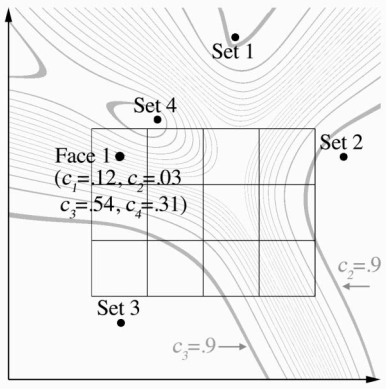

Figure 7: *2D example showing four weight sets placed in the environment of a mesh and their effect on face 1. The curved lines represent the weight sets' level of influence c, with the bold lines indicating a weighted value of 0.9.*

The level of influence weight c_a of a set a is calculated as shown in equation (8), where the *dist* function returns the distance between a set and the face, *h* is the strength of the set, n is the total number of sets, *t* is a 'tightness' control exponent greater than 1, and the maximum possible distance of two points is assumed to be 1. The subsequent calculation of a weighting value w_x based on values contained in the sets $w_{x,a}$ and the sets' level of influence weights *c* is shown in equation (9).

$$c_a = \frac{(1 - dist(pos_{face}, pos_{set.a}))^t \cdot h_a}{\sum_{i=1}^{n}(1 - dist(pos_{face}, pos_{set.i}))^t \cdot h_i} \quad (8)$$

$$w_x = \sum_{i=1}^{n_a}(w_{x,i} \cdot c_i) \quad (9)$$

In a simplified version of this scheme, once can assign two sets of weights to positions along an x, y, or z axis. Values are interpolated linearly if the face's corresponding coordinate falls between the sets, and otherwise the value of the closest set is used. In both cases, translation, scaling and rotation of the input mesh modifies the position of vertices relative to the weight sets and thus leads to changes in output.

3.2.2 Variation Based on the Mesh's Topology and Motives. Rather than placing weight sets in the mesh's environment, it is possible to vary weights according to the mesh's topology – specifically the vertex valence and the number of faces that a vertex is part of. Based on either one of these factors, or a combination of the two, one can assign attractor/deflector values to each vertex. Alternatively these values can be assigned to specific face/edge configurations and to motives commonly found in the input mesh or to those that get produced during the subdivision iterations.

Figure 8: *Sample face/edge motives*

The assigned attractor/deflector values can be incorporated in the Catmull-Clark and Doo-Sabin subdivision equations. In the Catmull-Clark process, they can be combined with two additional weights to control the placement offace points and edge points. The two equations below are applied in the subdivision process after equations (1) and (2). U is the attractor/deflector value of a point based on its classification, and n is the number of points in a face.

$$F'_{Attracted} = F'_1 + w_6 \cdot \sum_{i=1}^{n}(P_i - F'_1)u_{P_i}. \quad (10)$$

$$E'_{Attracted} = E'_1 + w_7 \cdot ((P_1 - E'_1)u_{p_1} + (P_2 - E'_1)u_{p_2}). \quad (11)$$

3.2.3 Variation Based on Tagging Vertices and Faces. A further method to incorporate non-uniform weights is through tagging of vertices and faces in the input mesh. Each vertex can be assigned to a vertex group, and each face can be assigned to a face group. Sets of subdivision weights can then be specified per group. Edge weights are determined by averaging the weighting values of the groups to which their vertices are assigned.

The assignment of vertices and faces to groups also enables the locking of certain parts of the input mesh. The positions of vertices (and by extension edges) belonging to a certain group can be locked throughout the subdivision process, or for a number of starting iterations. This allows for the generation of hard edges and creases. A similar scheme was used in the animated film Geri's Game [DKT98].

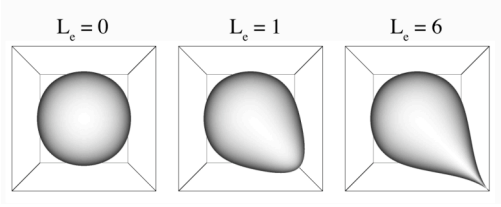

Figure 9: *7th generation subdivision of a hexahedron with the bottom right edge of its input mesh locked, where L_e specifies the number of iterations during which the edge remains locked.*

4 Conclusion

By adding degrees of freedom to both the subdivision weighting schemes and to how the schemes are applied, the scope of forms that can be generated increases greatly. The

attached illustrations always take the simplest input meshes – the platonic solids – yet generate forms that exhibit and astounding complexity.

Unlike forms generated through typical additive processes, these form are not explicable through reductionism. Rather the subdivision processes can be used to generate forms for which it is difficult to discern their source, much less the exact nature and parameters of the processes applied.

Despite this irreducibility, the processes behave largely linearly. Small changes in the parameters lead to gradual, traceable changes in the output. Forms can be modified through a subsequent tweaking of their parameters. Though they are not entirely predictable, the processes are deterministic and they are reproducible.

Rather than simply providing a mechanism to smoothen a mesh, subdivision processes can enrich these meshes with an astounding degree of complexity.

References

[Cat74] CATMULL, E.: *A Subdivision Algorithm for Computer Display of Curves Surfaces*, Dissertation, Department of Computer Science, University of Utah (December 1974)

[CC78] CATMULL, E.: *A Subdivision Algorithm for Computer Display of Curves Surfaces*, Dissertation, Department of Computer Science, University of Utah (December 1974)

[Doo78] DOO D.: *A subdivision algorithm for smoothing down irregularly shaped polyhedrons*, Proceedings on Interactive Techniques in Computer Aided Design, pp. 157 - 165, 1978

[DS78] DOO D., SABIN M.: *Behavior of recursive division surfaces near extraordinary points*, Computer-Aided Design, 10 (6) 356–360 (1978)

[DKT98] DOO D., SABIN M.: *Behavior of recursive division surfaces near extraordinary points*, Computer-Aided Design, 10 (6) 356–360 (1978)

Results

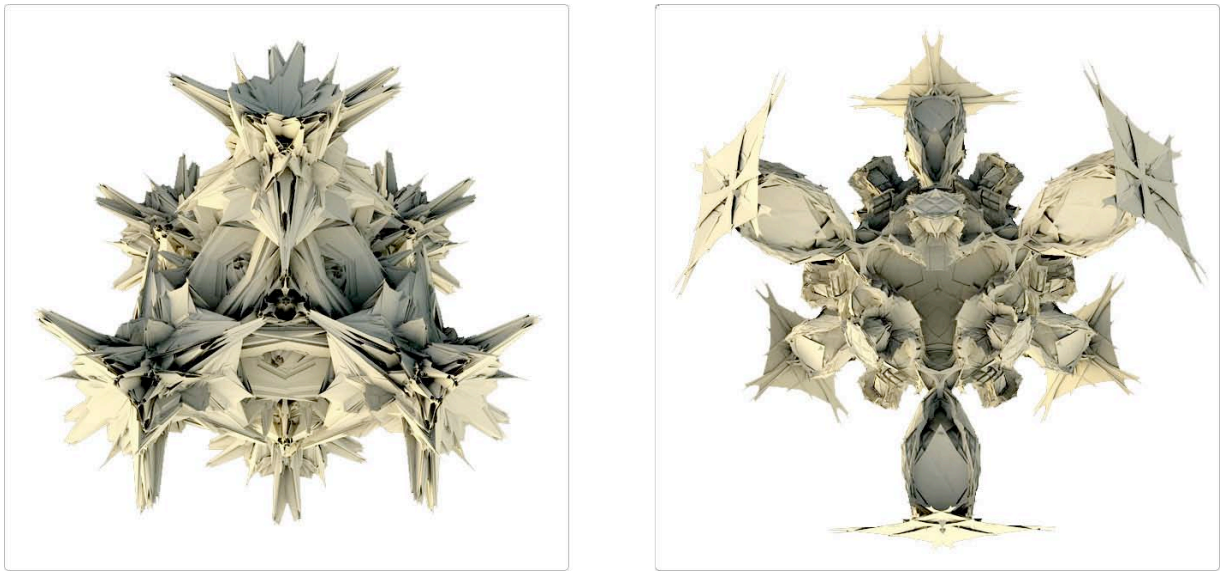

Illustration 1: *Two 8th-generation subdivisions of a hexahedron using only the Catmull-Clark process. Weights are non-stationary but uniform. The figure on the right relies heavily on the use of surface-normals extrusion for vertices.*

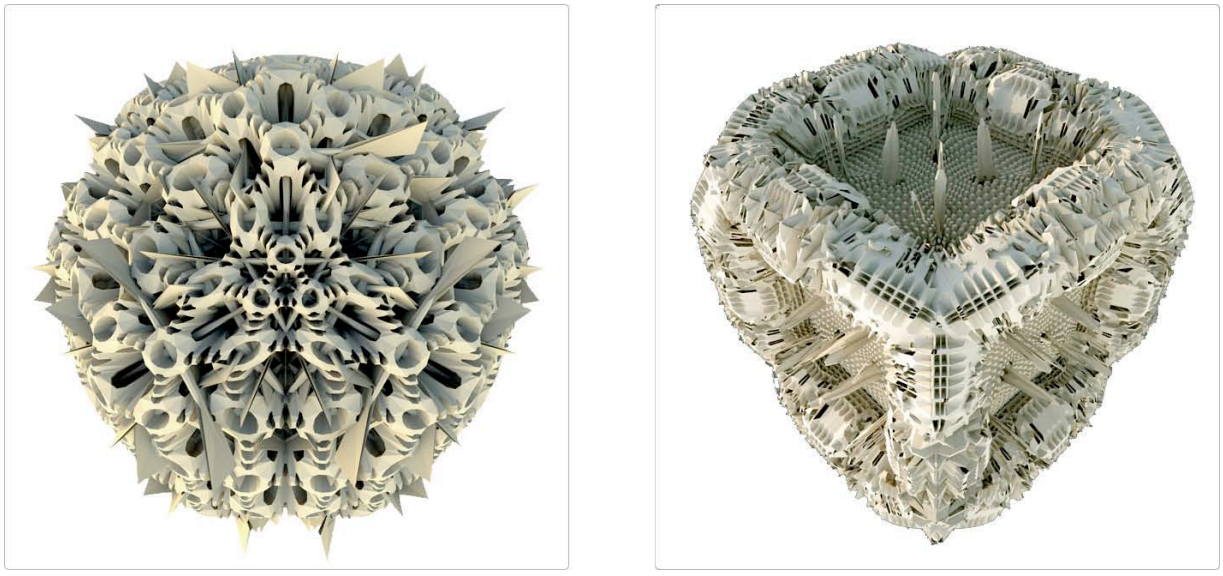

Illustration 2: *Two 10th generation subdivisions of a dodecahedron and hexahedron. Both forms use combinations of the Catmull-Clark and Doo Sabin processes. Weights are non-stationary but uniform. The figure on the right has several non-linearities in its process to due vertex fusing.*

Illustration 3: *Two subdivisions of three vertically stacked hexahedrons. Forms are in their 8^{th} and 9^{th} generation and have up to 11 million faces. Both the Catmull-Clark and Doo-Sabin processes are employed, and weights are non-stationary and non-uniform. In the right figure porosity occurs on several levels due to limited vertex valence.*