Eurographics Symposium on Geometry Processing (2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

Stellar Subdivision Grammars

Luiz Velho

IMPA – Instituto de Matematica Pura e Aplicada

Abstract

In this paper we develop a new description for subdivision surfaces based on a graph grammar formalism. Subdivision schemes are specified by a context sensitive grammar in which production rules represent topological and geometrical transformations to the surface's control mesh. This methodology can be used for all known subdivision surface schemes. Moreover, it gives an effective representation that allows simple implementation and is suitable for adaptive computations.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations

1. Introduction

A fundamental problem in geometric modeling is the specification of curves and surfaces through a compact description that provides intuitive control to the user. Additionally, this representation should encompass a class of surfaces with good computational properties. such as smoothness and adaptation.

Subdivision surfaces generalize the traditional spline surfaces. They make possible to construct a surface from a control mesh of arbitrary topology, avoiding the regular connectivity limitations of spline surfaces. Moreover, subdivision surfaces possess a natural multiresolution structure that is computationally very powerful in practical applications.

A subdivision surface is defined as the limit surface, resulting from the application of a *subdivision scheme* to a *control polyhedron*. In this process, the polygonal base mesh is recursively subdivided and the mesh geometry is progressively modified according to subdivision rules.

A subdivision scheme is characterized by a refinement operator, that changes connectivity by subdividing the mesh, and by a smoothing operator, that modifies the geometry. These two operators are usually specified in terms of iterated matrix multiplication. Each multiplication step transforms a local neighborhood of a coarse mesh into a corresponding refined neighborhood of the subdivided mesh. This matrix representation implies an indexing scheme of the vertices in a local neighborhood that depends on the mesh connectivity, but is somewhat arbitrary.

Although the matrix notation is well suited for the continuity analysis of subdivision schemes, it is not a good description for implementation purposes. In this paper we propose the use of parametric context-sensitive L-systems with affine mapping interpretation as a way to describe subdivision schemes. This representation has the advantages of being simple, precise and directly usable for algorithmic implementation.

2. Previous Work

An overview of the theory of subdivision surfaces and their applications can be found in (Zorin and Schröder, 2000)[35], and (Warren and Weimer, 2002)[32].

The classical subdivision schemes were introduced in the late 1970's. The surfaces generated by these schemes are based on tensor product biquadratic [6], bicubic B-splines [4] and three-directional box splines [18].

The above schemes produce surfaces that approximate the control polyhedron. Some schemes are interpolating, two examples are the Butterfly scheme [7] and Kobbelt's quadrilateral scheme [13].

Subsequently, other approximating schemes have been proposed. They include the simplest scheme [24], $\sqrt{3}$ subdivision [15] and $\sqrt{2}$ subdivision [31].

© The Eurographics Association 2003.

During the 1990's, the focus was on the continuity analysis of subdivision schemes [2, 26, 33].

More recently, the main trend has been the investigation of a generic framework for subdivision. The idea is to study basic operators that serve as building blocks for a class subdivision schemes [11, 10, 22, 20].

Continuing in the direction of a general description for subdivision schemes, (Prusinkiewicz et al.)[25] proposed to use L-systems as way to specify subdivision curves.

In this paper, we extend the work of Prusinkiewicz to surfaces. The major difficulty to pass from the one-dimensional case to two dimensions is a topological one. Results from Stellar subdivision theory [16]. allow us to define basic topological operators for subdivision surfaces. We show that stellar subdivision grammars can describe any subdivision scheme. Furthermore, our framework has several advantages in terms of flexibility and implementation.

3. L-Systems

In this section we give a brief introduction to L-systems [17] and graph grammars [8].

L-systems were developed by Aristid Lindenmayer in 1968 to model the development of multicellular organisms. L-systems constitute a string rewriting mechanism, which is described by a formal grammar.

We define the L-system grammar G of a language L as:

$$G = \{\Sigma, \Pi, \alpha\}$$

where Σ is the alphabet of the language, Π is a set of rewriting rules, and $\alpha \in \Sigma^*$, is the starting string (or axiom) of L.

Each rewriting rule $\pi : \Sigma \to \Sigma^*$ defines a unique mapping of a symbol of Σ into a string $s \in \Sigma^*$.

In contrast with most other grammars, where rewriting rules are applied sequentially, in an L-system all rewriting rules are applied in parallel to the current string at each stage of the rewriting process.

We can view the set of rewriting rules Π as an *operator*. Applying the operator Π to a string means to execute all the substitutions implied by the rewriting rules. Thus, using this operator, we write the language L produced by a grammar G as

$$\begin{aligned} L &= \{\alpha, \Pi(\alpha), \Pi(\Pi(\alpha)), \ldots\} \\ &= \Pi^\infty(\alpha) \end{aligned}$$

The definition above suggests that it is possible to make a connection between L-systems and subdivision schemes.

Since their introduction, L-systems have been extended considerably, mainly by Przemyslaw Prusinkiewicz. These extensions augmented the power and applicability of L-systems, and include: context sensitivity, probabilistic rewriting, rule ranges, parametric and parametrized L-systems.

The work of Prusinkiewicz and coauthors [25] on L-System descriptions of subdivision curves makes use of context sensitive parametrized L-systems.

In a context sensitive grammar, the rewriting rules depend on the neighbors of a symbol in the string. The general form of a context sensitive rule is as follows:

$$L < P > R \to S,$$

where $P \in \Sigma$ and $L, R, S \in \Sigma^*$. Here, L and R form the left and right context of P, respectively.

Parametric L-systems operate on strings of *modules*. A module is a symbol of the alphabet with optional parameters. For example, the rewriting rule

$$A(x) \to A(2 * x + 1),$$

when applied to the parametric symbol $A(3)$ leads to $A(7)$.

The combination of these two mechanisms is sufficient to describe subdivision schemes for curves. As an example, we describe below the specification of Chaikin's subdivision scheme for closed curves.

$$P(v_l) < P(v) > P(v_r) \to P(\frac{1}{4}v_l + \frac{3}{4}v)P(\frac{3}{4}v + \frac{1}{4}v_r) \quad (1)$$

Here, v is a vertex of a closed polygonal curve, v_l and v_r are v's left and right neighbors. The parametric symbol $P(v)$ on the left side of the rewriting rule is called the *predecessor* of the two parametric symbols on the right side of the rule, which are called *successors* of $P(v)$.

In this way the old vertices of the control polygonal curve are mapped to new vertices of the subdivided curve. This scheme is illustrated in Figure 1.

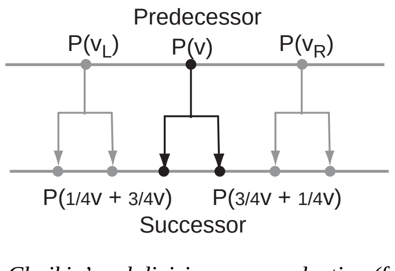


Figure 1: *Chaikin's subdivision as a production (from cf.[25]).*

The L-system notation is very suitable for a computer implementation. The programming language `L+C` [12] combines features of the C++ programming language with the L-systems notation. This language has been used in (Prusinkiewicz et. al, 2002)[25] to implement algorithms for subdivision curves.

Program 1 shows an example of the code to generate the curve in Figure 2. The start string is a square control polygon, the rewriting rule is given by Equation 1, and the L-system operator is applied 3 times. The output is a subdivided curve generated by a geometric interpretation.

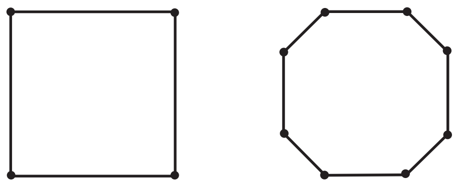
Program 1 Chaikin's scheme for a subdivision curve using the L+C language.

```
V2f v1(0,0), v2(0,1), v3(1,1), v4(1,0);
module P(V2f);

ring L−system: 1;
derivation length: 3;

axiom: P(v1) P(v2) P(v3) P(v4);
P(vl) < P(v) > P(vr) :
    { produce P(0.25*vl + 0.75*v) P(0.75*v + 0.25*vr) }

interpretation:
P(v): { produce LineTo(v) Circle(0.01) }
```


(a) α (b) $\Pi(\alpha)$

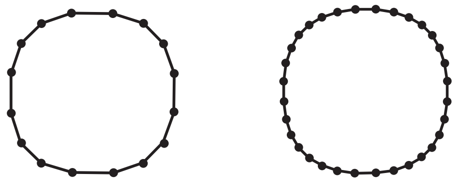

(c) $\Pi^2(\alpha)$ (d) $\Pi^3(\alpha)$

Figure 2: *Chaikin's subdivision curve generated by Program 1, (from cf.*[25]*).*

4. Subdivision Curves and Surfaces

In this section we discuss the basic concepts of subdivision for curves and surfaces.

As we mentioned before, a subdivision scheme is composed by a *refinement* operator, and a *smoothing* operator. The refinement operator changes the connectivity of the combinatorial mesh structure in order to increase the mesh resolution. The smoothing operator changes the geometry of the mesh in order to bring vertices closer to their limiting positions.

In the curve case, the refinement operator is very simple, because the connectivity is trivial. This is due to the fact that the topology of a manifold curve is inherited from the topology of the real line.

More precisely, closed curves are homeomorphic to a circle, and open curves are homeomorphic to a line interval. Therefore, there are only two kinds of neighborhoods to be considered: a two-sided interval for internal points and a one-sided interval for boundary points. Figure 3 shows these neighborhoods.

(a) internal point (b) boundary point

Figure 3: *One dimensional neighborhoods.*

There are two common types of refinement operators for curves: edge insertion and vertex insertion. Edge insertion replaces each edge by two new edges, while vertex insertion replaces each vertex by two new vertices. The Chaikin subdivision scheme discussed in the previous section uses vertex insertion.

A consequence of the above facts is that the main interest in subdivision schemes for curves lies in the study of smoothing operators and their geometric properties.

Note that the one-dimensional setting facilitates considerably the implementation of curve subdivision using L-systems, since there is a direct correspondence between the sequential structure of strings in a grammar and the topology of the real line, i.e., a polygonal curve is represented by a sequence of vertices.

In the surface case, the situation is a lot more complex. First of all, the two-dimensional topology requires a combinatorial description of the mesh connectivity. Second, some neighborhoods in a 2D mesh may not have a regular structure.

The topology of a surface is represented by a combinatorial manifold structure, a mesh $M = (F, E, V)$, where F, E, and V are respectively sets of faces $f = (v_o, v_1, v_2)$, edges $e = (v_0, v_1)$ and vertices v. The geometry of the surface is given by an embedding function that associates points $p \in \mathbb{R}^3$ to the vertices $v \in V$.

We can classify a mesh based on the type of faces $f \in F$. The most common types are triangular and quadrilateral meshes. A triangular (quadrilateral) mesh has only three (four) sided faces. These meshes are homogeneous. We can also have heterogeneous meshes composed by arbitrary *n*-sided faces.

The mesh structure may be regular or irregular. In a regular mesh, all neighborhoods have the same connectivity structure. The subdivision process only generates regular neighborhoods. However, if we start with an irregular mesh as the control polyhedron, some vertices of this base mesh will have an non-regular neighborhood (these vertices are called *extraordinary* vertices) and the refined mesh will have a *semi-regular* structure.

Similarly to the one-dimensional case, there are two types of neighborhoods for a two-dimensional manifold: a disk for interior points and a half-disk for boundary points. Figure 4 shows these two neighborhoods. In a mesh, these neighborhoods have a combinatorial description.

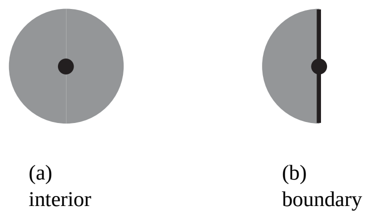


Figure 4: *Two-dimensional neighborhoods.*

If we consider the class of piecewise smooth surfaces, then we have to include other kinds of neighborhoods – e.g., crease edges and vertices [3].

The refinement operators can be classified in primal and dual operators. There are operators for triangular and quadrilateral meshes. Primal operators are based on face split, and dual operators on vertex split. Also, they can perform quadrisection, trisection and bisection of faces.

Figures 5 and 6 show, respectively, the primal and dual quadrisection refinement operators for triangular and quadrilateral meshes.

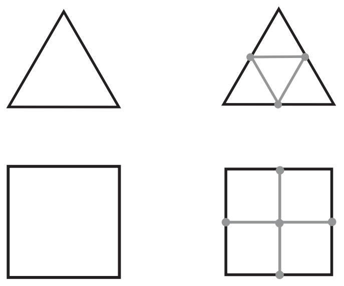


Figure 5: *Primal quadrisection refinement operators.*

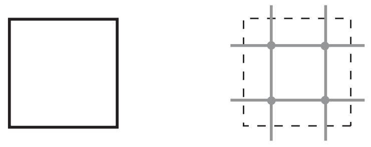


Figure 6: *Dual quadrisection refinement operator.*

Figure 7 shows the trisection $\sqrt{3}$ refinement operator, while Figure 8 shows the bisection $\sqrt{2}$ refinement operator.

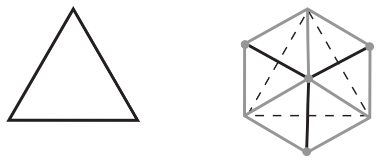


Figure 7: *Trisection* $\sqrt{3}$ *refinement.*

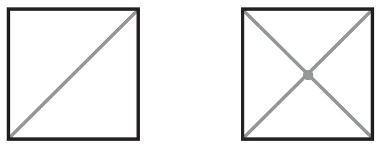


Figure 8: *Bisection* $\sqrt{2}$ *refinement.*

It is apparent from the description of two-dimensional refinement operators that the implementation of subdivision schemes for surfaces using L-systems is not a straightforward task.

What we need in order to solve this problem is to find the right building blocks that allow us to define refinement operators for subdivision using a graph grammar, such as L-system. We will see in the next section that stellar subdivision theory provides such building blocks.

5. Stellar Theory

This section reviews the theory of Stellar subdivision and discusses its relation with refinement operators in subdivision schemes.

Stellar theory combines the abstract and piecewise approaches to combinatorial topology [16]. Its main focus is the study of equivalences between simplicial complexes. Therefore, Stellar theory applies to triangular meshes. This does not impose any loss of generality, since every manifold surface can be triangulated.

© The Eurographics Association 2003.

Before we introduce the Stellar theory, we need to establish some notation related to topology of simplicial complexes.

Two simplices σ_1, σ_2 are *independent* if $\sigma_1 \cap \sigma_2 = \emptyset$. The *join* $\sigma_1 \star \sigma_2$ of independent simplices σ_1, σ_2 is the set $\sigma_1 \cup \sigma_2$. The join of complexes K and L, written $K \star L$, is $\{\sigma \star \tau : \sigma \in K, \tau \in L\}$. The *boundary* of a complex K is written as ∂K. The *link* of simplex $\sigma \in K$, denoted $\mathrm{link}(\sigma, K)$, is defined by $\mathrm{link}(\sigma, K) = \{\tau \in K : \sigma \star \tau \in K\}$. And finally, the *star* of σ in K, $\mathrm{star}(\sigma, K)$, is the join $\sigma \star \mathrm{link}(\sigma, K)$.

Note that in the mesh representation of a manifold surface, the star of every vertex $v \in V$ is a combinatorial disk, or half-disk. These neighborhoods are the discrete equivalents of the surface neighborhoods in Figure 4.

The star and link operators provide a combinatorial description of the neighborhood of a simplex. They can be used to define atomic changes in a simplicial complex that do not alter the topology of the underlying surface.

The stellar exchange operators are such modifications. They make local changes to the neighborhood of an r-simplex in a n-dimensional complex K, $r \leq n$, while maintaining the combinatorial integrity of K.

Definition 1 Let K be an n-dimensional simplicial complex, take a non-empty r-simplex $A \in K$, such that $\mathrm{link}(A, K) = \partial B \star L$ for some non-empty simplex $B \notin K$, and some (possibly empty) complex L.
The operation that changes K into K' by removing $A \star \partial B \star L$ from K and inserting $\partial A \star B \star L$ is called a *stellar exchange* and is written $K \overset{\kappa(A,B)}{\longmapsto} K'$.

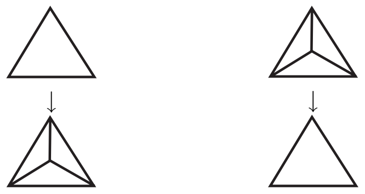
The stellar exchange operation unifies the notions of bistellar move and stellar subdivision. Informally, a *bistellar move* replaces an r-simplex by a simplex of dimension $(n-r)$ in a n-dimensional complex , while a *stellar subdivision* inserts a vertex into an r-simplex, $\sigma \in K$, of a n-dimensional complex K.

When K is a two-dimensional simplicial complex we have the following stellar operators: *face split* and its inverse *face weld*; *edge split* and its inverse *edge weld*; and *edge flip*; Note that the edge flip is its own inverse. These operations are illustrated in Figure 9(a) (face split/weld), Figure 9(b) (edge split/weld), and Figure 10 (edge flip).

The stellar moves are face split, face weld and edge flip. The stellar subdivision operators are face split, face weld, edge split and edge weld. Note that face split/weld are at the same time stellar moves and subdivision operators.

The fundamental result of Stellar theory is given by theorem 2:

Theorem 2 (Newman [21], Pachner [23]) Two connected combinatorial n-dimensional manifolds are piecewise linearly homeomorphic if and only if they are related by a sequence of elementary stellar exchanges.


(a) *face split* and *face weld*

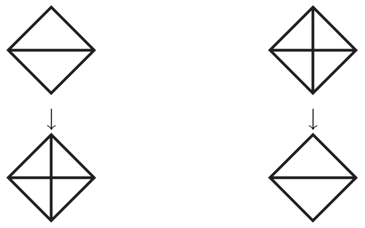

(b) *edge split* and *edge weld*

Figure 9: *Stellar subdivision operators and inverses.*

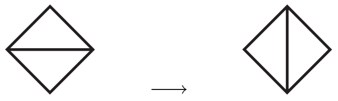


Figure 10: *Bistellar move on edges: the* flip *operator.*

The above theorem says that stellar exchanges are the operators to transform between any two triangulations of a combinatorial manifold. In particular, stellar exchanges can be used as the building block for mesh refinement operators. This is made clear by further results of Stellar subdivision theory.

Stellar subdivision is an important part of the Stellar theory [1]. It provides the basic operators to refine and simplify a simplicial complex.

Proposition 3 Any bistellar move is the composition of a stellar subdivision and a weld.

It is easy to verify the above result. For example, an edge flip can be expressed as an edge split followed by an edge weld.

This means that stellar subdivision operators (and their inverses) are sufficient to transform a simplicial complex. Nonetheless, we should remark that it is convenient to use also bistellar moves for refinement and simplification rules, because they make possible to generate mesh sequences that are monotonic in terms of resolution.

6. Decomposition of Subdivision Schemes

In this section we show how Stellar theory can be applied in the context of subdivision surfaces. We demonstrate that all known refinement methods for subdivision can be decomposed into a sequence of primitive stellar operations. In addition, we suggest that, in general, it is possible to factorize the smoothing methods for subdivision under the stellar decompositions, with considerable gains in simplicity of implementation.

Theorem 2 guarantees that stellar exchanges can transform between any two triangulations of a manifold. This alone would be a sufficient indication that refinement methods can be decomposed in terms of stellar operations. However, we would like to impose one extra condition: the sequence of meshes generated by the decomposed process should be *monotonic* in resolution. That is, $|M_i| \geq |M_j|$ for $i > j$, in a sequence of refined meshes, $(M_0, M_1, \ldots M_n)$, where $|M|$ denotes the size of the mesh M.

We will see, through examples, that the above requirement can be satisfied for all known subdivision schemes. This is a consequence of the fact that, in a sense, stellar operations are topological modifications to a mesh with a "fine" granularity.

Another important property of stellar operations, is that they are atomic operations which maintain the integrity of the combinatorial manifold structure. Thus, meshes are always valid under stellar operations.

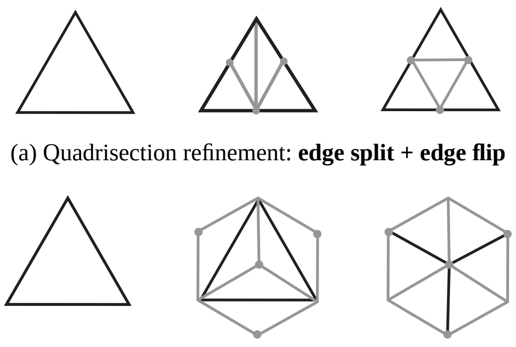
Before we start discussing the stellar decomposition of refinement methods, we need to introduce the concept of refinement levels. We associate a level $l \in \mathbb{N}$ to every topological element of a mesh. Intuitively, the level indicates at which stage of subdivision, that element was created. All vertices of the base mesh have level 0. During refinement, new vertices are assigned levels based on their parent simplex. A new vertex v is assigned level $l(v) = \max(l(w)) + 1$, $w \in \sigma$, were σ is the subdivided simplex that originated v. The level of a face is computed from the level of its vertices, i.e., for a face $f = (v_0, v_1, v_2)$, $l(f) = \{\max(l(v_i)) + 1 \mid i = \{1,2,3\}\}$. The level of an edge depends of the stellar operation that created the edge. For an edge flip operation the level of the flipped edge is the level of the parent edge plus one. For a face split operation the three new edges are assigned the level of their common vertex. For an edge split operation there are two cases: the bisection edges resulting from splitting the edge, and the transversal edges connecting the split vertex to each opposite vertex. The bisection edges are assigned the level of the parent edge plus two. The transversal edges are assigned the level of the opposite vertex plus one.

The level values of vertices, edges and faces will help guiding the decomposition process of subdivision rules.

We now demonstrate how to decompose the refinement methods described in Section 4, in terms of stellar operators.

We start with refinement of triangle meshes. The triangle quadrisection refinement can be decomposed into two steps. In the first step, we apply an edge split operation to all edges of the current mesh. In the second step, we apply an edge flip operation only to edges whose level is equal to the current level minus one. Level ranking guarantees that appropriate edges are flipped. Figure 11(a) illustrates this decomposition.

Note that the $\sqrt{3}$ refinement is already defined using stellar operators. It consists of applying first a face split operation to all triangles of the current mesh, then an edge flip is applied only to edges whose level is less than the current level. This decomposition is shown in Figure 11(b).


(a) Quadrisection refinement: **edge split + edge flip**

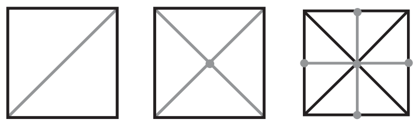
(b) $\sqrt{3}$ refinement: **face split + edge flip**

Figure 11: *Stellar decomposition of triangle mesh refinement.*

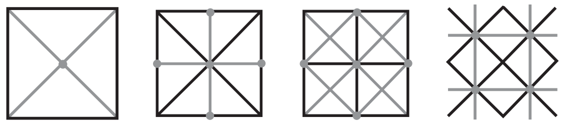
In order to present the decomposition of quadrilateral refinement methods, we have to discuss mesh structures. Stellar operators are only defined for simplicial complexes (i.e., triangle meshes). For that reason, at first sight it would seem that we could not use stellar operators to decompose quadrilateral refinement methods. However, this is not the case, because a quadrilateral mesh can always be trivially triangulated by dividing each quadrilateral into two triangles. Therefore, we can work with a triangle mesh with a *quadrilateral structure*, which we call a *tri-quad* mesh. Such a mesh is composed of *basic blocks*, each consisting of a pair of triangles. By keeping track of basic blocks we can always recover the quadrilateral mesh back from a tri-quad mesh. Note that this is a simple matter, since we can associate a basic block with the internal edge shared by the two triangles of the block.

We remark that it is also possible to transform a triangle mesh into a quadrilateral mesh. There are various algorithms for this purpose, for example see (Velho, 2000)[29]. Another option is to apply one step of Catmull-Clark refinement [4], which transforms an arbitrary mesh into a quadrilateral mesh.

The primal quadrisection refinement of quadrilateral structures is decomposed into two steps using stellar operators. We start with a basic block of two triangles. At refinement level k, the internal edge of basic blocks has level $k-1$, while the four external edges have level k. The primal refinement alternates into splitting internal and external edges of basic blocks. This is illustrated in Figure 12(a).



(a) Primal quadrisection: **edge split** ^{2}



(b) Dual quadrisection: **edge split** ^{2} **+ edge flip**

Figure 12: *Stellar decomposition of quadrilateral mesh refinement.*

For the dual quadrisection refinement of quadrilateral structures, we define a dual basic block, that consists of a group of four triangles with one common vertex. Here, at refinement level k, the four external edges of a dual block have level $k-1$, while the four internal edges have level k (note that this is consistent with the primal scheme). The dual refinement is decomposed into three steps. First, we split the external edges, then the internal edges and finally we flip the edges connecting the center of two new dual blocks. This is illustrated in Figure 12(b).

Note that, similarly to $\sqrt{3}$ refinement, the $\sqrt{2}$ bisection refinement is also defined in terms of stellar operators. It simply consists of the application of an edge split operator. Thus, it does not need a decomposition. Although this scheme operates on triangle meshes, it requires a quadrilateral structure – more precisely, it is based on a four-directional mesh [5]. Essentially, it always splits the internal edge of basic blocks, which are interleaved in subsequent refinement levels. Figure 8 shows the $\sqrt{2}$ refinement.

The other issue in the implementation of subdivision schemes is related to smoothing methods. In principle, we could decompose only the refinement operators and compute the smoothing rules in the same way that it is done for the non-decomposed schemes. Nonetheless, the decomposition of the refinement operators allows us to factorize also the smoothing operators. This is a known fact in the CAGD literature [32]. In our case, it is quite natural and more convenient for implementation.

Smoothing operators are defined in such a way to compute the position of mesh vertices as a linear combination of their neighbors‘ positions. Usually, we distinguish between *new* and *old* vertices. New vertices are created at the current subdivision step, and old vertices are inherited from previous subdivision steps.

Note that when we decompose a refinement method, intermediate subdivision levels are introduced. In order to factorize smoothing rules of subdivision operators, we define rules that are associated with new vertices created by the stellar operators and rules associated with old vertices at intermediate refinement levels. The combined effect of these new factored rules should be equivalent to the original non-factored smoothing rules.

The factorization of smoothing rules received a great deal of attention in recent years [22, 34, 28].

In (Velho, 2001) [30], it has been shown how to factorize the Doo-Sabin and Catmull-Clark subdivision schemes in the context of bisection refinement. In Appendix A, we derive a factorization of Loop subdivision scheme based on stellar refinement. We believe, that it should be possible to factorize most existing subdivision schemes for surfaces.

7. Subdivision Grammars

In this section we introduce a description of subdivision surfaces using L-systems. We exploit the results of previous sections to define a topological graph grammar for subdivision schemes. This representation will be based on stellar refinement operators and factorization smoothing rules.

As we mentioned before, a mesh is a combinatorial manifold structure, $M = (F,E,V)$, consisting of sets of faces, edges and vertices.

To represent a mesh we adopt a half-edge based topological data structure [19]. This structure is augmented with a field, indicating the subdivision level of edges and vertices.

The subdivision process creates a sequence of meshes $(M_0, M_1, \ldots, M_k)$, by successively applying a subdivision scheme $\mathscr{S}$ to the current mesh M_j, i.e., $M_{j+1} = \mathscr{S}(M_j)$. The process starts with the base mesh M_0.

In the context of a grammar description of subdivision, the simplicial structure elements constitute the alphabet Σ, the base mesh is equivalent to the starting string (or axiom), $\alpha \in \Sigma^*$, and the subdivision scheme is defined through the rewriting rules of a stellar grammar Π, i.e., $M_0 \equiv \alpha$ and $\mathscr{S} \equiv \Pi$.

We now present the general mechanisms for describing subdivision surfaces using L-systems.

The input is a base mesh M_0, which has to be initialized with the appropriate level values. For arbitrary triangle meshes, both vertices and edges are assigned level zero. Quadrilateral meshes are first converted to tri-quad meshes. In this conversion, all vertices are assigned level zero, the internal edge of all basic blocks is assigned level zero and external edges of basic blocks are assigned level one [†].

The rewriting rules of the subdivision grammar will be defined using the following notation:

$$P(p\,;\,\mathrm{cond})\mid Q \;\;\rightarrow\;\; V(v = \mathrm{assgn}) \parallel S$$

where P and V are modules, (i.e., parametric symbols). P is the predecessor being replaced by successor V, Q is the context of P (i.e., its combinatorial neighborhood) and S is the context of V (i.e., the neighborhood that replaces Q).

The parameters p and v are topological elements of the mesh. The clause *cond* is a conditional logical expression specifying a subset of module representatives $p \in P$, such that $cond(p) =$ ***true***. The clause *assgn*, is a geometric assignment function that computes the value of v.

Note that the changes to the connectivity structure of the mesh resulting from applying a rewriting rule will be handled automatically by the topological operator. Also, the level of the elements involved are automatically updated. Note also, that either one of the two clauses, *cond* and *assgn* can be omitted.

The rewriting rules comprise stellar operators and vertex update rules. Below we give the specific form of these rules.

- Face split:

$$F \rightarrow V \parallel E^3F^3$$

- Edge split (interior edge):

$$E|F^2 \rightarrow V \parallel E^4F^4$$

- Edge split (boundary edge):

$$E|F \rightarrow V \parallel E^3F^2$$

- Edge flip:

$$E|F^2 \rightarrow E \parallel F^2$$

- Vertex:

$$V \rightarrow V$$

We note that it is not valid to flip boundary edges. We also remark that all the rules, except edge flip produce a new vertex. This is because edge flip is a stellar move, while the other operators are stellar subdivisions. The edge flip just rearranges the connectivity of the edge neighborhood.

The vertex rule does not cause any changes to the connectivity, it only affects the geometry of the mesh.

The conditional clause usually depends on the level of the element relative to the current level, for example:

$$(l(v) < c)$$

where $l(v)$ is the level of v and c is the current level.

The assignment clause computes the geometry of a mesh vertex as a linear combination of its neighbors. For most schemes, the subdivision smoothing rule is based on the one-ring neighborhood $N_1(v)$ of the vertex v. Thus, the rule assumes the form:

$$v = g(v) = \sum_{v_i \in N_1(v)} \beta_i v_i$$

where β_i are the coefficients of the subdivision mask.

The topological subdivision grammar Π is specified through a sequence of rewriting rules with the form described above. In these rules, the symbols denote topological type, i.e., F, E, V, for faces, edges and vertices, respectively. The rewriting rules are executed sequentially in the order they are specified in the grammar. Each rule is applied in parallel to all elements matching the topological type, context and condition.

We remark that in this rewriting process, the mesh structure is not copied, as it is usually done in L-systems. Instead, the topological operators insert new elements and modify the connectivity structure of the mesh. The integrity of the combinatorial structure is guaranteed to always remain valid by the action of stellar operators. The geometry values of the vertices is handled in the following way. Each vertex has a double-buffer that contains the current and next geometric value. This buffer is automatically swapped after each subdivision step.

We remind that the subdivision level of topological elements is automatically handled by the stellar operators, as discussed in the previous section. Also, when a rule is applied to the mesh, it increments the current subdivision level, except for the vertex rule that does not change the level of a vertex, nor the current subdivision level.

8. Examples

In this section we give examples of the specification of some popular surface subdivision schemes using our topological graph grammar.

We describe the subdivision rules for, $\sqrt{3}$, Loop, Catmull-Clark, Doo-Sabin, Peters, and 4-8 subdivision.

[†] Meshes with *n*-sided faces can be either triangulated or converted to a quadrilateral mesh in a pre-process.

sqrt3 Subdivision

The specification of the $\sqrt{3}$ subdivision scheme is straightforward because it comes directly from the original definition [15]. We apply a face split and then an edge swap.

$$\begin{aligned}
&F(f) \to V(v = \text{avrg}(f)) \parallel E^3F^3 \\
&V(v; l(v) < c) \to V(v = \text{kob}(v)) \\
&E(e; l(e) < c)|F^2 \to E \parallel F^2
\end{aligned}$$

The smoothing rule for new vertices is applied together with the face split operation. It is an average of the vertices of the triangle.

$$\text{avrg}(V) := \frac{1}{|V|} \sum_{v_i \in V} v_i$$

The smoothing rule for old vertices is defined as follows:

$$\text{kob}(v) := (1-\alpha_n)v + \alpha_n \frac{1}{n} \sum_{v_i \in N_1(v) \wedge (l(v_i) < c)} v_i$$

where c is the current level, n is the one-half of the valence of v, and $\alpha_n = \frac{4-2\cos(\frac{2\pi}{n})}{9}$.

Loop

The specification of Loop scheme is a bit more involved, because the original definition needs to be factorized.

The subdivision process consists of two steps of subdivision where triangles are appropriately bisected to achieve a triangle quadrisection. Then, the geometry of new and old vertices are updated.

$$\begin{aligned}
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(e)) \parallel E^4F^4 \\
&E(e; l(e) < c)|F^2 \to E \parallel F^2 \\
&V(v; l(v) = c) \to V(v = \text{loop}_1(v)) \\
&V(v; l(v) < c) \to V(v = \text{loop}_2(v))
\end{aligned}$$

The smoothing rules *loop*$_1$ and *loop*$_2$ are as follows:

$$\text{loop}_1(v) := \frac{1}{2}v + \frac{1}{8} \sum_{v_i \in N_1(v) \wedge (l(v_i) = c)} v_i$$

and

$$\text{loop}_2(v) := (1-2k\beta)v + \beta \sum_{v_i \in N_1(v)} v_i$$

where $\beta = \frac{1}{k}(5/8 - (3/8 + 1/4\cos(\frac{2\pi}{k}))^2)$, and $k = \deg(v)$.

In the Appendix A we derive the factorization of these smoothing rules for Loop subdivision.

Catmull-Clark

The Catmull-Clark scheme employs a tri-quad mesh. This scheme is very suitable for decomposition. It consists simply as the alternating splitting of internal and external edges of basic blocks.

The factorization of smoothing rules has been know for quite some time [9]. A more recent demonstration of this factorization can be found in [30].

$$\begin{aligned}
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4 \\
&V(v; l(v) < c) \to V(v = \text{catm}(v)) \\
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4
\end{aligned}$$

The smoothing rule *catm* is as follows:

$$\text{catm}(v) := \frac{1}{2}v + \frac{1}{16} \sum_{v_i \in N_1(v)} v_i$$

Doo-Sabin

The Doo-Sabin scheme is a dual scheme for quadrilateral meshes. As such, it needs the conversion to a tri-quad mesh, and an initialization step, where all primal basic blocks are subdivided to obtain dual quad blocks. This is accomplished with the following grammar rule, which is applied only at initialization.

$$E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4$$

After initialization, the three steps of stellar dual quadrilateral refinement are applied.

$$\begin{aligned}
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4 \\
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4 \\
&E(e; l(e) = c)|F^2 \to E \parallel F^2
\end{aligned}$$

Note that the smoothing rules are computed together with the first two refinement steps.

We should remark that these rules apply only for meshes with regular quadrilateral structure. For irregular meshes the rules are more complicated, but can be derived from (Velho, 2001)[30].

Peters

The simplest subdivision scheme of Peters [24], has a very similar decomposition to the Doo-Sabin scheme. The only difference is in the factorization of smoothing rules. Instead of averaging all vertices in the star of a new vertex, only the edge endpoints are averaged (i.e., we replace $N_1(v)$ by e in the above rules).

4-8 Subdivision

The 4-8 subdivision scheme was recently proposed by Velho and Zorin [31]. It is based on four-directional meshes. Thus, it has a natural decomposition in terms of edge split operations.

$$
\begin{aligned}
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4 \\
&V(v; l(v) < c) \to V(v = \text{velh}(v)) \\
&E(e; l(e) = c)|F^2 \to V(v = \text{avrg}(N_1(v))) \parallel E^4F^4 \\
&V(v; l(v) < c) \to V(v = \text{velh}(v))
\end{aligned}
$$

The smoothing rule *velh* is as follows:

$$
\text{velh}(v) := \frac{1}{2}v + \frac{1}{2k} \sum_{v_i \in N_1(v) \wedge l(v_i) = c} v_i
$$

where k is the number of vertices $v_i \in N_1(v)$, such that $l(v_i) = c$.

9. Discussion

In the previous sections we have discussed only subdivision rules for the interior vertices of a mesh. To be complete, we should discuss also rules for boundary vertices and also rules for piecewise smooth surfaces – these include rules for crease edges and pinch vertices.

The rules for boundary vertices and creases are very similar to the rules for subdivision curves. For this reason, they are not difficult to implement using rewriting rules. In those cases, the neighborhood matching mechanism incorporated in the subdivision grammar takes care of rule selection automatically.

We also have discussed only the case of uniform mesh subdivision. That is, the whole mesh is subdivided at each step of the topological rewriting process. However, in many situations it is desirable to subdivide the mesh locally. This capability is important in many applications, such as view dependent rendering.

An in-depth treatment of adaptive subdivision cannot be included in this paper due to space limitations. Nonetheless, we should remark that subdivision grammars based on stellar operators are extremely suitable for adaptive subdivision. The reason is that, stellar exchanges are local operations with a fine granularity. Therefore, they allow adaptive refinement while maintaining the consistency of a conforming mesh. Furthermore, the level tags on topological elements make possible to propagate recursively the refinement process in order to satisfy the constraints imposed by smoothing rules.

The above considerations lead us to take the risk of saying that stellar decomposition is the best way to do adaptive subdivision. Just as an extra remark, to substantiate our conjecture, we show that the only way to do adaptive subdivision using triangle quadrisection implies the use of stellar operations. This is hinted in Figure 13.

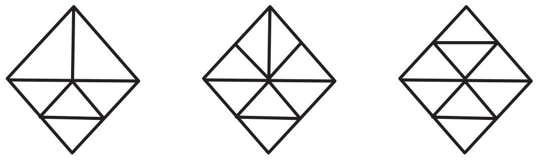


Figure 13: *Adaptive triangle quadrisection.*

Another aspect that we have not considered in this paper is the decomposition of interpolatory subdivision schemes, such as the Butterfly scheme [7] and Kobbelt's quadrilateral interpolation scheme [14]. We acknowledge that interpolatory schemes usually employ subdivision rules that depend on larger neighborhoods – typically 2-ring neighbors. All subdivision schemes discussed in Section 8 use 1-ring neighborhoods, and therefore are easier to factorize. However, recently Schaefer and Warren [27] introduced a new interpolatory subdivision scheme for quadrilateral meshes based on linear subdivision and differencing. This scheme can be easily described using a stellar subdivision grammar. We will leave as a topic for future research the investigation of grammar descriptions for other interpolatory subdivision schemes.

10. Conclusions

We presented a new description for subdivision surfaces based on a graph grammar formalism. Subdivision schemes are specified by a context sensitive grammar in which the productions rules represent topological and geometrical transformations to the surface's control mesh. Topological modifications are defined by stellar subdivision operators while geometrical modifications are specified by linear maps. Stellar subdivision grammars extend L-System descriptions of subdivision curves to surfaces.

This methodology can be used to describe all known subdivision surface schemes. The advantages of stellar subdivision grammars include: i) it provides a way to unambiguously specify a subdivision scheme; ii) it gives an effective representation that allows simple implementation; and iii) this representation is suitable for adaptive computations. Furthermore, in a system based on stellar subdivision grammars it is possible to switch subdivision schemes without re-coding. This facilitates experimentation with new subdivision schemes.

As a continuation of our work, we are planning to implement a software environment for subdivision surfaces using stellar grammars. This will give us the tools to pursue further research in the area. Another line of investigation that could lead to new insights is the connection of stellar grammars with general classes of graph grammars.

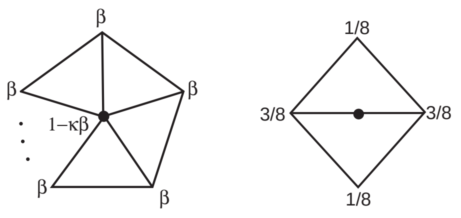
Acknowledgments

This paper was written while the author was a visiting researcher at Microsoft Research Asia. A sincere recognition goes to Baining Guo and Harry Shum for providing an stimulating research environment. Many thanks also to Przemek Prusinkiewicz for having inspired this work and to the anonymous reviewers for their insightful comments.

Appendix A: Factorization of Loop Subdivision

The Loop subdivision scheme generalizes the three-directional box spline. It produces surfaces that are C^2-continuous everywhere, except at extraordinary vertices where they are C^1-continuous.

This scheme uses primal quadrisection refinement and is composed of two masks – *new vertex* rule and *old vertex* rule. The masks for Loop subdivision rules are shown in Figure 14.



(a) old vertex (b) new vertex

Figure 14: *Loop subdivision masks.*

Under stellar decomposition, the refinement operator is decomposed in two steps: edge split and edge flip.

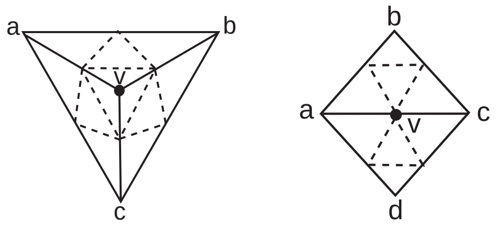
The factorization of the Loop smoothing rules comes only after one complete two-step refinement. At this point, we have the following scenario, depicted in Figures 15.

Without loss of generality, consider the diagram in Figure 15(a). Note that new vertices have been computed as averages of edges endpoints. Therefore, the new vertices, w_i, in the 1-ring of the old vertex v are: $1/2(v+a)$, $1/2(v+b)$ and $1/2(v+c)$. So, if we compute the new value of v as:

$$\begin{aligned} v &= (1-2k\beta)v+\beta\sum w_i \\ &= (1-2k\beta)v+k\beta v+\beta(a+b+c) \\ &= (1-k\beta)v+\beta(a+b+c) \end{aligned}$$

we get the original Loop rule for an old vertex.

Also in the diagram for new vertex v in Figure 15(b), $v=\frac{1}{2}(a+b)$, and the other new vertices w_i, in the 1-ring of v are respectively, $1/2(a+c)$, $1/2(c+b)$, $1/2(c+d)$ and $1/2(d+a)$.



(a) old vertex (b) new vertex

Figure 15: *Loop factorization.*

So, if we compute the new value of v as

$$\begin{aligned} v &= \frac{1}{2}v+\frac{1}{8}w_i \\ &= \frac{1}{2}(\frac{1}{2}(a+b))+\frac{1}{8}(\frac{1}{2}(2(a+b+c+d))) \\ &= \frac{2}{8}(a+b)+\frac{1}{8}(a+b+c+d)) \\ &= \frac{3}{8}(a+b)+\frac{1}{8}(c+d)) \end{aligned}$$

we get the original Loop mask for a new vertex.

References

1. J. Alexander. The combinatorial theory of complexes. *Ann. Math.*, 31:294–322, 1930.
2. 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.
3. H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 113–120, July 2000.
4. E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput. Aided Design*, 10:350–365, 1978.
5. C. de Boor, D. Hollig, and S. Riemenschneider. *Box Splines*. Springer-Verlag, New York, NY, 1994.
6. D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Comput. Aided Design*, 10:356–360, 1978.
7. N. Dyn, J. Gregory, and D. Levin. A Butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–190, 1990.

8. Hartmut Ehrig, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors. *Graph-Grammars and Their Application to Computer Science, 4th International Workshop, Bremen, Germany, March 5-9, 1990, Proceedings*, volume 532 of *Lecture Notes in Computer Science*. Springer, 1991.

9. M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 35–44, Anaheim, California, August 1993. ISBN 0-201-58889-7.

10. I. Ivrissimtzis, N. Dodgson, and M. Sabin. A generative classification of mesh refinement rules with lattice transformations. Research Report UCAM-CL-TR-542, Cambridge University, Cambridge, UK, September 2002.

11. I. Ivrissimtzis and H.P. Seidel. Polyhedra operators for mesh refinement. In *Proceedings of Geometric Modeling and Processing 2002*, pages 132–137, Wako, Saitama, Japan, July 2002. IEEE.

12. R. Karwowski. The L+C modeling language, 2002. Ph.D. Thesis, Univerity of Calgary.

13. L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. *Computer Graphics Forum*, 15(3):409–420, August 1996.

14. L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13(8):743–761, 1996.

15. L. Kobbelt. $\sqrt{3}$ subdivision. In *Proceedings of SIGGRAPH*, Computer Graphics Proceedings – Annual Conference Series, pages 103–112, 2000.

16. W. B. R. Lickorish. Simplicial moves on complexes and manifolds. In *Proceedings of the Kirbyfest*, volume 2, pages 299–320, 1999.

17. A. Lindenmayer. Developmental algorithms for multicellular organisms: A survey of L-systems. *Journal of Theoretical Biology*, 54:3–22, 1975.

18. C. Loop. Smooth subdivision for surfaces based on triangles. Master's thesis, University of Utah, 1987.

19. M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, ISBN 07167-8015-1, Rockville, Maryland, 1988.

20. H. Muller and M. Rips. Another metascheme of subdivision surfaces. *Visualization and Mathematics III*, pages 203–221, 2002.

21. M. H. A. Newman. On the foundations of combinatorial analysis situs. *Proc. Royal Acad.*, 29:610–641, 1926.

22. P. Oswald and P. Schröder. Composite primal/dual $\sqrt{(3)}$-subdivision schemes. *Computer Aided Geometric Design, (accepted for publication)*, 2002. http://cm.bell-labs.com/who/poswald/sqrt3.pdf.

23. U. Pachner. PL homeomorphic manifolds are equivalent by elementary shellings. *Europ. J. Combinatorics*, 12:129–145, 1991.

24. J. Peters and U. Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics*, 16(4):420–431, 1997.

25. P. Prusinkiewicz, F. F. Samavati, C. Smith, and R. Karwowski. L-system description of subdivision curves. *Submitted*, 2002. http://pages.cpsc.ucalgary.ca/ samavati/lsdsc.htm.

26. U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

27. S. Schaefer and J. Warren. A factored interpolatory subdivision scheme for quadrilateral surfaces. In *Proceedings of the fifth Curves and Surfaces*, 2003.

28. J. Stam. On subdivision schemes generalizing uniform b-spline surfaces of arbitrary degree. *Computer Aided Geometric Design. Special Edition on Subdivision Surfaces*, 18:383–396, 2001.

29. L. Velho. Quadrilateral meshing using 4-8 clustering. In *Proceedings of CILANCE 2000 - Symposium on Mesh Generation and Self-adaptivity*, pages 61–64, December 2000.

30. L. Velho. Using semi-regular 4–8 meshes for subdivision surfaces. *Journal of Graphics Tools*, 5(3):35–47, 2001.

31. L. Velho and D. Zorin. 4-8 subdivision. *Computer-Aided Geometric Design*, 18(5):397–427, 2001. Special Issue on Subdivision Techniques.

32. J. Warren and H. Weimer. *Subdivision Methods For Geometric Design: A Constructive Approach*. Morgan-Kaufmann, 2002.

33. D. Zorin. *Stationary Subdivision and Multiresolution Surface Representations*. PhD thesis, Caltech, 1997.

34. D. Zorin and P. Schröder. A unified framework for primal/dual quadrilateral subdivision schemes. *Computer Aided Geometric Design. Special issue on Subdivision Surfaces*, 18(5):429–454, 2001.

35. D. Zorin and P. Schröder. Subdivision for modeling and animation, 2000. Course Notes - ACM-SIGGRAPH Course 23.