Teaching meshes, subdivision and multiresolution techniques

Stephan Bischoff, Leif Kobbelt

Computer Graphics Group, RWTH Aachen

Abstract

In recent years, geometry processing algorithms that directly operate on polygonal meshes have become an indispensable tool in computer graphics, CAD/CAM applications, numerical simulations, and medical imaging. Because the demand for people that are specialized in these techniques increases steadily the topic is finding its way into the standard curricula of related lectures on computer graphics and geometric modeling and is often the subject of seminars and presentations. In this article we suggest a toolbox to educators who are planning to set up a lecture or talk about geometry processing for a specific audience. For this we propose a set of teaching blocks, each of which covers a specific subtopic. These teaching blocks can be assembled so as to fit different occasions like lectures, courses, seminars and talks and different audiences like students and industrial practitioners. We also provide examples that can be used to deepen the subject matter and give references to the most relevant work.

1 Introduction

In the last decade the processing of polygonal meshes has emerged as an active and very productive research area. This can basically be attributed to two developments:

- Modern geometry acquisition devices, like laser scanners and MRT, easily produce raw polygonal meshes of ever growing complexity
- Downstream applications like analysis tools (medical imaging), computer aided manufacturing, or numerical simulations all require high quality polygonal meshes as input.

The need to bridge the gap between raw triangle soup data and high-quality polygon meshes has driven the research on efficient data structures and algorithms that directly operate on polygonal meshes rather than on a (most often not feasible) intermediate CAD representation.

We roughly structure the main area of geometry processing into three major sub-topics: *meshes*, *subdivision* and *multiresolution techniques*. The topic *meshes* covers the basic data structures and algorithms that are used for representing and modifying polygonal geometry. Here we find algorithms that are used to create, analyze, smooth, decimate or parameterize polygonal meshes. *Subdivision methods* provide a link between (discrete) polygonal meshes and conventional (continuous) spline surface representations. Their main application is geometric modeling and adaptive meshing for finite element computations. *Multiresolution techniques* decompose the model into a hierarchy of meshes that represent different levels of detail. Technically this hierarchy can be exploited in order to significantly speed up many algorithms. More important, however, is the semantic effect in that such hierarchies can be used for intuitive modeling metaphors or highly efficient geometry compression algorithms.

Due to the increasing demand for people that are specialized in geometry processing, the topic is finding its way into related lectures and is also the subject of many seminars and talks. There is, however, no canonical curriculum to draw from. In this article we propose to alleviate this problem by structuring the subject into a set of *teaching blocks* each of which covers a certain subtopic. Each teaching block consists of ...

- ... a number of keywords that compactly describe the essence of the block. After teaching a teaching block the students should be able to reconstruct the contents by means of the keywords.
- ... a list of references to recent publications.
- ... the actual contents of the block in form of a short description. This description is not meant to be tutorial in nature but as a rough guideline for lecturers.

The idea is that these teaching blocks can be put together in various ways such as to accommodate different audiences and occasions:

- A one-semester general course on geometry processing would comprise all of the presented blocks and combine it with more classical material on spline theory and NURBS.
- As part of an advanced course on computer graphics, geometry processing can be covered by the more practical blocks on meshes and subdivision.
- An advanced and more specialized course on geometric modeling would focus on the subdivision and multiresolution blocks, possibly including the more theoretical material on the convergence analysis.
- A seminar for industrial practitioners would primarily focus on the practical blocks and the examples.

Figure 1 gives an overview of all the blocks and their relationships.

2 Preliminaries

In this paper we will only deal with the core topics in geometry processing. However, for many of these topics some previous knowledge in mathematics and/or geometric modeling is indispensable for the students.

Concepts that should be known from calculus include continuity/differentiability of functions, sequences of functions, convergence criteria, Taylor expansion and approximation power. From linear algebra the basic concepts of vector spaces, linear maps and spectral theory should be available as well as knowledge of notions from affine geometry.

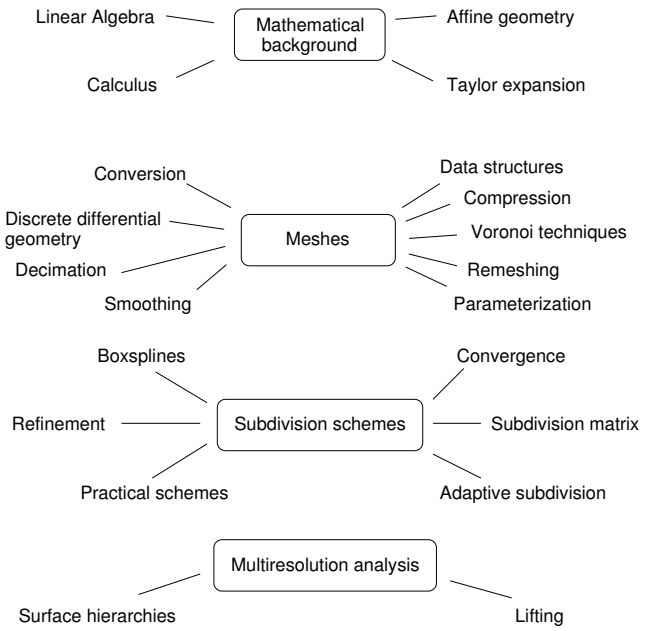


Figure 1: Teaching block overview. For the sake of completeness we have also included a "mathematics" block that is, however, not further explained in this paper.

We have also experienced that students benefit from setting the topic into the broader view of computer graphics/geometric modeling. In particular the students should be familiar with the different approaches to represent surface geometry either explicitly, parametrically or implicitly and to use points, patches or volume elements as the basic structural primitive.

3 Meshes

In the following we will describe the basic data structures and algorithms that are used to process polygonal meshes. Most of these data structures and algorithms can be understood without previous knowledge. Their implementation is often straightforward which makes them suited as practical exercises.

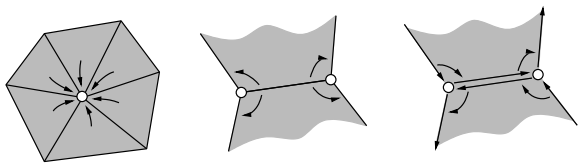
3.1 Data structures

▷ *winged-edge, halfedge*

▷ [9, 10, 49, 66, 97]

The many data structures that are available for representing polygonal meshes are designed such as to facilitate the access to local neighborhood information, e.g. enumerating the 1-ring of a vertex. Furthermore, as constant-size data structures can be stored more compactly one often restricts oneself to triangle meshes or uses edges as the topological primitive.

Example: Winged- and Halfedge The prevalent data structures for representing orientable two-manifold polygonal meshes are the *winged-edge* and the *halfedge* data structures. In contrast to a simple *shared vertex* representation, they allow one to easily access neighborhood information.



Shared vertex, winged edge and halfedge representation

The winged-edge data structure associates with each edge eight references: two vertices, two faces and four incident edges. Since edges cannot be oriented globally consistent, a case distinction is necessary during traversal.

Splitting an edge into two neighboring halfedges results in the halfedge data structure, where each halfedge stores four pointers to a vertex, its next and opposite halfedge and to a face.

Face based data structures *Face-based* data structures are especially convenient for subdivision and multiresolution hierarchies [99]. Here the basic structuring element is a face that contains pointers to its adjacent vertices and faces and for each adjacent face the index of the adjacent edge. In a *quadtree* one additionally stores pointers to the child faces.

Other examples There are a number of less commonly used data structures, including *quad edge* that simultaneously encodes a mesh and its dual and is able to represent non-orientable manifolds, *radial edge* for handling non-manifold meshes and *directed edges* which is very memory efficient but restricted to triangular meshes.

3.2 Voronoi diagram and Delaunay triangulation

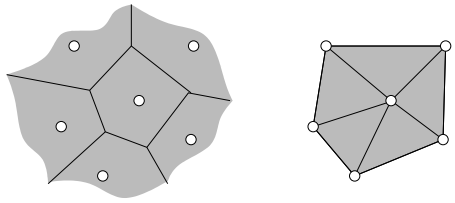
▷ *Voronoi diagram, Delaunay triangulation*

▷ [6, 19, 30, 33, 39, 84]

Given n points $\mathbf{p}_i \in \mathbb{R}^d$ the Voronoi region corresponding to $\mathbf{p}_i$ is defined as

$$V_i = \{\mathbf{p} : \text{dist}(\mathbf{p}, \mathbf{p}_i) \leq \text{dist}(\mathbf{p}, \mathbf{p}_j) \text{ for all } j \neq i\}$$

resulting in a partition of $\mathbb{R}^d$ into *Voronoi regions*.



Voronoi–Diagram and Delaunay–Triangulation

The dual of the Voronoi diagram is called the *Delaunay triangulation*. For $d = 2$ we have that a triangulation is Delaunay iff for each edge the circle circumscribing one adjacent triangle does not contain the opposite vertex. Among all possible triangulations, the Delaunay triangulation is the one that *maximizes the smallest angle*.

Example: Fortune's sweep-line algorithm In case $d = 2$ a nice visual interpretation of the Voronoi diagram can be given as follows. Embed $\mathbb{R}^2$ as the $z = 0$ plane into $\mathbb{R}^3$ and locate on each point $\mathbf{p}_i$ a cone of opening angle 45°. If one then views the configuration from $z = -\infty$, the Voronoi diagram is given by the visible parts of the cones. Fortune's algorithm exploits this observation by sweeping a slanted plane over the points thereby successively constructing the Voronoi diagram or the Delaunay triangulation.

Example: Delaunay triangulation from convex hulls Any algorithm for computing the convex hull of an object can also be used to compute Delaunay triangulations [74]. For this one embeds the points $\mathbf{p}_i$ in $\mathbb{R}^{d+1}$ by projecting them onto the parabola $\mathcal{P} : x_{d+1} = x_1^2 + \cdots + x_d^2$. From the convex hull of $\mathcal{P}$ we remove the faces whose normals point in the $d+1$ direction. The Delaunay triangulation is then obtained by projecting back the remaining polyhedron into $\mathbb{R}^d$.

3.3 Conversion: implicit representations → meshes

▷ *signed distance field, marching cubes*

▷ [36, 48, 56, 65]

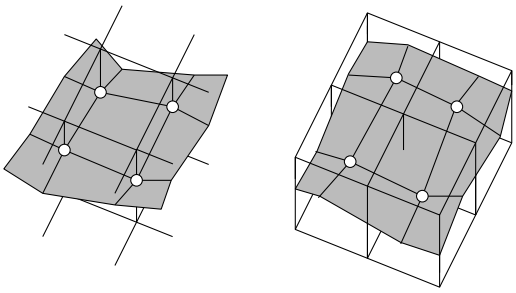
A surface $\mathcal{S}$ can be represented as the kernel of a *signed distance function* $d(x, y, z)$, i.e.

$$\mathcal{S} = \{[x, y, z] : d(x, y, z) = 0\}$$

In typical applications (e.g. medical imaging) d is sampled on a regular grid, $d_{ijk} = d(i, j, k)$, and interpolated by a piecewise tri-linear function.

Example: Marching Cubes The *marching cubes algorithm* [65] extracts a polygonal representation from the grid d_{ijk} by generating a *vertex for each edge* that intersects $\mathcal{S}$ and connects these vertices to a (triangulated) polygon. If additional (Hermite-) data is available at the grid points, one can use the *extended marching cubes* algorithm [56] in order to reconstruct sharp features.

Example: Surface Nets Dual methods like the *surface nets* algorithm [36] compute a *face for each edge* that intersects $\mathcal{S}$. This method can also be extended to reconstruct sharp features when Hermite data is available [48].



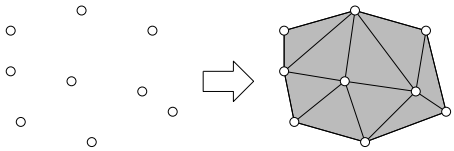
Marching cubes and Surface Nets

3.4 Conversion: point clouds → meshes

▷ *organized/unorganized point clouds,power crust,volumetric approach*

▷ [7, 17, 46]

Given a set of points $\mathbf{p}_i \in \mathbb{R}^3$ sampled from a surface S we search for a triangle mesh that interpolates or approximates these points. The various algorithms that have been proposed for this task can be classified according to whether they accept unstructured point clouds as input, whether the reconstruction is based on a signed distance function and whether they are interpolatory or approximating.



Point cloud triangulation

Example Hoppe et al. [46] estimate a normal $\mathbf{n}_i$ for each point $\mathbf{p}_i$ by fitting a (tangent) plane to the k-neighborhood of $\mathbf{p}_i$. In order to consistently orient the normals, the normal orientation is propagated along an extended Euclidean minimum spanning tree. The signed distance of an arbitrary point $\mathbf{p}$ to the object is then estimated as the distance to the tangent plane associated with the nearest $\mathbf{p}_i$. Finally a triangle mesh is extracted via the marching cubes algorithm.

Example: Volumetric approach Curless et al.'s *volumetric method* [17] takes as input a set of range images, i.e. point clouds that are organized according to a regular grid, as they are produced e.g. by laser range scanners. Each range image is scan converted to a cumulative weighted signed distance function. Time and space efficiency is achieved by resampling the range image according to the voxel ordering and by run-length encoding the volume. Finally an explicit polygonal mesh is extracted via the marching cubes algorithm. This algorithm is also able to automatically fill in gaps and hence produces watertight models.

Example: Voronoi/Delaunay filtering algorithms If the points $\mathbf{p}_i$ are sufficiently dense samples of a surface S then S can be reconstructed via *filtering* as a subset of the Delaunay triangulation of the $\mathbf{p}_i$ [1, 5, 30]. As an advanced example, Amenta et al.'s *power crust algorithm* [7] proceeds as follows: First the Voronoi diagram of all sample points $\mathbf{p}_i$ is computed. If the $\mathbf{p}_i$ are sufficiently dense, the Voronoi cells will be needle-like polyhedra orthogonal to the surface S. The two vertices of the Voronoi cell that are farthest away from $\mathbf{p}_i$ in positive and negative direction are called the *poles* of the cell. Let $\mathbf{a}_i$ be the set of all poles and r_i the radii of their corresponding Voronoi balls. Then the *power diagram* of all poles is defined as the Voronoi diagram with respect to the power-distance

$$d_{pow}(\mathbf{x}, \mathbf{a}_i) = ||\mathbf{x} - \mathbf{a}_i||^2 - r_i^2$$

Inside/outside information is propagated over the poles using the fact that two poles corresponding to the same sample point are on different sides of the surface and that an inner and an outer polar ball can only intersect shallowly. The output of the algorithm is the *power crust*, i.e. those cells of the power diagram that separate inside and outside poles.

3.5 Mesh decimation

▷ *vertex clustering, incremental decimation, edge collapse, quadric error metrics, progressive meshes, view-dependent refinement*

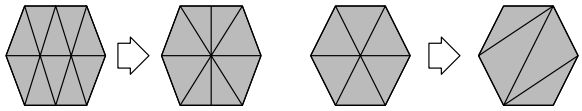
▷ [34, 37, 44, 57, 63, 81]

Mesh decimation algorithms simplify polygonal meshes by reducing their number of vertices while preserving as much of their shape and appearance as possible. One can distinguish two classes.

Vertex clustering algorithms set up a voxel grid and combine vertices that lie in the same voxel. These algorithms are typically applied in an out-of-core fashion, but provide only

limited control on the resulting mesh complexity, topology or quality.

Incremental decimation algorithms repeatedly remove the geometrically least important vertex from the mesh. This is done by either deleting a vertex together with its incident faces followed by a retriangulation of the resulting hole *(vertex removal)* or by collapsing two vertices along a common edge *(edge collapse)*.



Edge collapse and vertex removal

The decimation order is determined by an error metric like the Hausdorff distance or quadric error metrics.

Example: Quadric error metric Quadric error metrics [34] measure the squared distance of a vertex from all of its supporting planes. For this a quadric $\mathcal{Q}_i$ is associated with each vertex i. Let $\mathbf{n}_j^T\mathbf{x} = 0$ be the homogeneous Hessian normal form of the planes supported by the faces adjacent to vertex i, then Q_i is initialized as

$$Q_i = \sum_j \mathbf{n}_j\mathbf{n}_j^T$$

Whenever two vertices i and j are collapsed into a new vertex k the quadric Q_k associated with k is computed as

$$Q_k = Q_i + Q_j$$

and k's position $\mathbf{x}_k$ is determined such as to minimize the quadratic equation $\mathbf{x}^T Q_k \mathbf{x}$, i.e. by solving a linear system.

Example: Progressive Meshes and view-dependent refinement Halfedge collapses can be easily reversed *(vertex split)* resulting in a so-called *progressive mesh* representation [44]. Arranging the vertex collapses/splits in a forest allows to selectively refine a mesh based on view-frustum, screen-space error etc. [45, 51]

3.6 Mesh smoothing

▷ *Taubin's smoothing, curvature flow*

▷ [22, 53, 89]

Data that is acquired by physical measurement often exhibits noise. The removal of this noise is called *mesh smoothing*. In the following let $\mathbf{x} = [x_1, \ldots, x_n]$ be the positions of the n vertices of a triangle mesh $\mathcal{M}$. We further need a discretization of the Laplacian Δ to triangle meshes

$$\Delta x_i = \sum_{j \text{ neighbor of } i} w_{ij}(x_i - x_j)$$

where w_{ij} are some weight coefficients reflecting edge-lengths or angles. The discrete Laplacian can then be written in matrix form as $\Delta\mathbf{x}$.

Example: Signal processing approach The matrix Δ has real eigenvalues $0 \le k_1 \le \cdots \le k_n \le 2$ and the corresponding eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ can be considered as the *natural vibration modes* of the mesh. Let $\mathbf{x} = \sum \hat{x}_i\mathbf{e}_i$ be the *discrete Fourier transform* of $\mathbf{x}$ and let $f(k)$ be an arbitrary polynomial, then we have

$$f(\Delta)\mathbf{x} = \sum \hat{x}_i f(k_i)\mathbf{e}_i$$

Hence $f(k)$ can be considered as the *transfer function* of the filter $f(\Delta)$. Taubin proposes to set $f(k) = (1-\lambda k)(1-\mu k)$ where $\mu < -\lambda < 0$ in order to get a *non-shrinking* filter.

Example: Curvature flow approach Desbrun et al. consider mesh smoothing as a *diffusion process*

$$\frac{\partial\mathbf{x}}{\partial t} = \lambda\Delta\mathbf{x}$$

This system becomes stationary when $\frac{\partial\mathbf{x}}{\partial t} = 0$ i.e. when $\Delta\mathbf{x} = 0$. Instead of an *explicit forward Euler method* to solve this PDE one uses an *implicit scheme*

$$(I - \lambda dt\Delta)\mathbf{x}^{n+1} = \mathbf{x}^n$$

to iteratively solve the equation for each time step where λdt can be arbitrarily large.

Example: Energy minimization approach A standard measure for the global surface stretching and bending energy are the *membrane* and *thin-plate* energies resp.:

$$\int \mathbf{f}_u^2 + \mathbf{f}_v^2 \quad \text{and} \quad \int \mathbf{f}_{uu}^2 + 2\mathbf{f}_{uv}^2 + \mathbf{f}_{vv}^2$$

Applying *variational calculus* we obtain the necessary conditions

$$\Delta\mathbf{x} = 0 \quad \text{and} \quad \Delta^2\mathbf{x} = 0$$

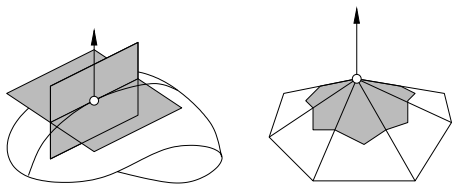
These equations can be solved iteratively by e.g. a *Gauss-Seidel* solver. The solution is identical to the stationary configuration in the curvature flow setting.

3.7 Discrete differential geometry

▷ *tangent,(Gaussian,mean,principal) curvature, principal directions*

▷ [11, 16, 26, 43, 67]

Let $\mathcal{S}$ be a smooth surface in space, $\mathbf{p}$ a point on this surface and $\mathbf{n}$ its normal vector, i.e. $\mathbf{n}$ is orthogonal to the *tangent plane* at $\mathbf{p}$. For every unit direction $\mathbf{e} = \mathbf{e}(\theta)$ in the tangent plane given by an angle θ the *normal curvature* $\kappa(\theta)$ is defined as the curvature of the intersection of $\mathcal{S}$ with the plane containing $\mathbf{n}$ and $\mathbf{e}$.



Continuous differential geometry and discrete analogue

The normal curvature takes on two extremal *principal curvatures* κ_1, κ_2 at orthogonal *principal directions* $\mathbf{e}_1, \mathbf{e}_2$. The *mean* and *Gaussian* curvatures are then defined as $\kappa_H = (\kappa_1 + \kappa_2)/2$ and $\kappa_G = \kappa_1\kappa_2$ resp. Note that κ_H can also be defined as $\kappa_H = 1/2\pi \int \kappa(\theta)d\theta$.

It is not easy to carry over these notions to non-differentiable triangle meshes. There are many ad-hoc solutions, but these are often not consistent, i.e. they do not converge to the pointwise properties of $\mathcal{S}$ when the triangle mesh is considered as an approximation of $\mathcal{S}$.

Example Meyer et al. propose the following consistent formulas: Let $\mathbf{x}_i$ be a vertex, θ_j the adjacent angles and $\mathbf{x}_j$ the adjacent vertices. Then the associated normal $\mathbf{n}_i$ and curvatures can be computed by

$$\kappa_H(\mathbf{x}_i)\mathbf{n}_i = \frac{1}{\mathcal{A}} \sum_j (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{x}_i - \mathbf{x}_j)$$

$$\kappa_G(\mathbf{x}_i) = \left(1 - 2\pi \sum \theta_j\right) / \mathcal{A}$$

Here $\mathcal{A}$ is the area around the center vertex and α_{ij}, β_{ij} are the two angles opposite the edge $\mathbf{x}_i\mathbf{x}_j$.

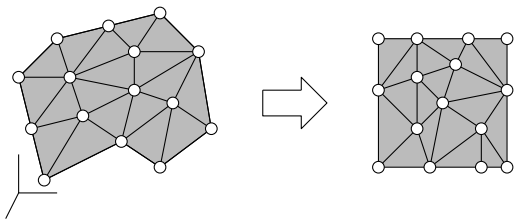
Example The usual definition of *geodesic* as a locally shortest path fails on the vertices of triangle meshes. Polthier et al. [69, 70] propose to use the notion of *straightest geodesics*, where the sum of angles on each side of the line is equal.

3.8 Parameterization

▷ *parameterization, conformal maps*

▷ [21, 32, 38, 61, 62, 75, 78, 79, 82, 83, 94]

Parameterization is the process of assigning two-dimensional coordinates $\mathbf{u}_i$ to the vertices $\mathbf{x}_i$ of a triangle mesh such that the resulting piecewise linear map becomes injective and hence invertible. Parameterizations are used e.g. for *remeshing* and *texture mapping*. Two problems have to be solved:



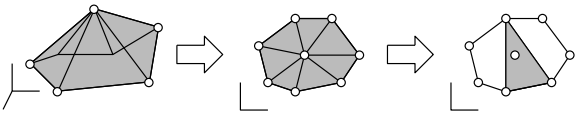
Parameterization

First, a triangle mesh can only be parameterized when it is topologically equivalent to a disk, i.e. when it has a boundary and is of genus 0. This has led to the development of various algorithms that subdivide a given mesh into *patches* that are homeomorphic to a disk [35, 38, 83].

Second, only developable surfaces can be parameterized without distortion. Therefore one tries to preserve alternative properties like *(generalized) barycentric coordinates*, *angles (conformal parameterization)*, or *area (authalic parameterization)* as good as possible or to minimize the *geometric stretch*. Note that no mapping can be conformal and authalic at the same time.

One can distinguish between non-linear methods that solve the parameterization problem iteratively[78, 79, 82] and linear methods. The latter amount to solve a linear system $A\mathbf{u} = \mathbf{b}$, where the matrix A and $\mathbf{b}$ depend on the vertex positions $\mathbf{x}_i$, and $\mathbf{u} = [u_{x,0} \dots u_{x,n}\, u_{y,0} \dots u_{y,n}]$. Whether this system leads to an admissible solution depends on the *boundary conditions*: if $A = [a_{ij}]$ is a matrix with $\sum_j a_{ij} = 0$ and non-negative *weights* $a_{ij}, i \neq j$ and if the boundary of the parameterization is convex then the solution of the linear system results in an injective mapping.

Example: Floater's weights Floater constructs a *shape-preserving* parameterization as follows [32]: The 1-ring $\mathbf{x}_j$ of each interior vertex $\mathbf{x}_i$ is mapped onto the plane via an exponential map. In a second step, the barycentric coordinates of $\mathbf{u}_i$ with respect to every triangle $\mathbf{u}_{j_0}, \mathbf{u}_{j_1}, \mathbf{u}_{j_2}$ that contains $\mathbf{u}_i$ are determined, summed up and normalized. This leads to a convex combination $\mathbf{u}_i = \sum_j \lambda_{ij}\mathbf{u}_j$ for the interior vertices. The boundary vertices are heuristically distributed on some convex shape and kept fixed, i.e. they affect only the right hand side $\mathbf{b}$. The resulting linear system is then described by the matrix $A = I - [\lambda_{ij}]$.



Floater's construction

Example: Least Squares Conformal Maps Levy et al. [62] determine a parameterization as follows: Let $\mathcal{X} : \mathbb{R}^2 \to M$ be a parameterization of a triangle mesh M and let $\mathcal{U} : M \to \mathbb{R}^2$ be its inverse (*local coordinates*). Consider a triangle $T \in M$ and represent $\mathcal{U}|_T$ with respect to a x, y-coordinate frame that lies within T, i.e.

$$\begin{array}{rccl} \mathcal{U}|_T : & T & \to & \mathbb{R}^2 \\ & (x,y) & \mapsto & (u(x,y), v(x,y)) \end{array}$$

$\mathcal{U}|_T$ is conformal, if $\frac{\partial \mathcal{U}}{\partial x} \perp \frac{\partial \mathcal{U}}{\partial y}$, and $||\frac{\partial \mathcal{U}}{\partial x}|| = ||\frac{\partial \mathcal{U}}{\partial y}||$, i.e. if

$$c(\mathcal{U}|_T) = \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} \right) = (0,0)$$

Note that $\mathcal{U}|_T$ is linear and hence $c(\mathcal{U}|_T)$ is actually a constant. The deviation of $\mathcal{U}$ resp. $\mathcal{X}$ from a conformal map can thus be measured as

$$\sum_{T \in M} \int_T ||c(\mathcal{U}|_T)||^2 dxdy = \sum_{T \in M} ||c(\mathcal{U}|_T)||^2 A_T$$

where A_T is the area of T. This is a quadratic equation that can be minimized using the conjugate gradients algorithm.

LSCM can also handle free boundaries. Furthermore, Levy et al. proposed a multigrid framework to compute the LSCM for very large meshes [75].

Other examples: Desbrun et al [21] derived a map that is equivalent to the least squares map by way of minimizing the Dirichlet energy and called it *discrete conformal map*. Sander et al. [78] minimize the *geometric stretch* of a parameterization and extend their method such that the approximation of signals that are defined on the surface is optimized [79].

3.9 Mesh compression

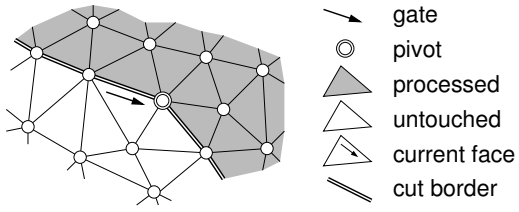
▷ *cut-border, connectivity vs. geometry compression*

▷ [3, 40, 47, 77, 92]

3.9.1 Connectivity coding

Let M be a mesh with n vertices. If M is given in shared vertex representation, one needs $\log(n)$ bits per vertex for storing the *mesh connectivity*, i.e. the indices referencing the point list with n entries. The algorithms presented in this section are based on traversal strategies that encode the mesh connectivity as a command sequence for a reconstruction automaton. These programs can then be efficiently encoded using only a constant number of bits per vertex.

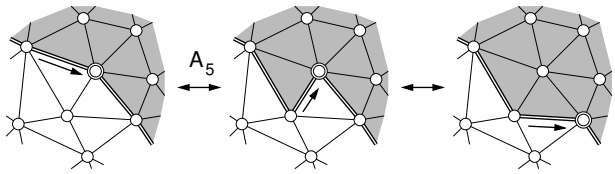
Most connectivity coding algorithms encode mesh elements and their incidence relation with respect to one or more *cut-borders* that are propagated over the mesh. The cut-borders are stored in a stack, the top element being the active cut-border.



Connectivity coding

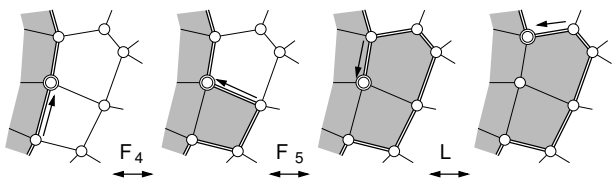
One can distinguish *growing operations* that process the current face and advance the cut-border (see examples below) and the special operations *split* and *merge*. The *split operation* S_i is performed when the current cut-border touches itself at the i-th vertex, a *merge operation* $M_{s,i}$ is performed when the current cut-border touches the s-th cut-border from the stack at vertex i (once per handle).

Example: Valence-based coding Touma and Gotsman [92] proposed a valence-based coding scheme for triangle meshes that achieves less than 2 bit/vertex on the average. To avoid the handling of special cases all holes are first closed by triangle fans around a dummy vertex. The *add operation* A_i introduces a vertex of valence i. If a vertex has no more *free edges*, its neighborhood can be completed.



Touma–Gotsman

Example: Edge-based methods *Face Fixer* [47] is an *edge-based* scheme that encodes arbitrary polygonal meshes with an average of 2–3 bits/vertex. The *face/hole* operations F_l/H_l attach a face/hole with l edges to the gate. The *glue operations* L and R identify the gate with the next/previous edge on the cut-border. The decoding proceeds in reverse order.

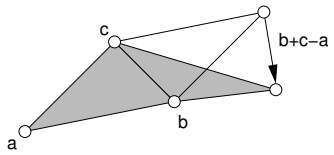


Face Fixer

Other schemes The original *cut-border machine* [40] and the *edge-breaker* algorithm [77] are examples of *face-based* methods. Alliez et al. proposed a *progressive* encoding scheme [3]. The *forest split* scheme [90] is also progressive and is used in the MPEG 4 standard.

3.9.2 Geometry compression

The mesh geometry (i.e. the vertex positions) is first quantized (usually to 10–12 bits) and then encoded losslessly using a predictive scheme, like the parallelogram rule. Huffman or arithmetic coders can then take advantage of the low entropy of the prediction errors.



Prediction error of the parallelogram rule

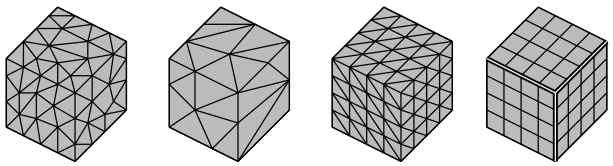
Example: Normal meshes Guskov et al. [42] propose a geometry representation called *normal meshes*. These meshes are semi-regular and hence need no connectivity information (except for the base mesh that is encoded traditionally). Vertex positions are predicted using a subdivision scheme and displaced in normal direction, i.e. the tangential components are zero and one only needs to store one scalar value per vertex for the normal component. Applying wavelet compression to normal meshes, Khodakovsky et al. [50] achieve significant geometry compression rates.

3.10 Remeshing

▷ *irregular, semi-regular, regular connectivity*

▷ [4, 29, 38, 61, 93]

Remeshing is the process of approximating a given geometry by a mesh with a special connectivity. The resulting meshes are categorized as *irregular*, *semi-regular* or *regular*.


Original, irregular, semi-regular and regular (re-)mesh

Irregular remeshing Turk's remesher [93] distributes points on the original geometry and then relaxates them via repulsion forces. When in equilibrum state, the points are connected to form a triangle mesh. Surazhsky et al. [86] describe a remesher including a post-processing step, that reduces the number of irregular vertices by propagating edge flips over the mesh. Furthermore, each mesh decimation algorithm can be considered as a special remeshing operation that produces irregular meshes.

Semi-regular remeshing A mesh is called semi-regular (or of *subdivision-connectivity*), if its connectivity can be obtained by uniformly subdividing some (coarse) base mesh. This type of connectivity is the basis for many multiresolution algorithms. Eck et al. [29] describe a remeshing algorithm that works on arbitrary input meshes. First the

mesh is partitioned into triangular patches by taking the dual of a Voronoi-like partitioning. Each patch is then parameterized in the plane and resampled to produce a semi-regular mesh. The MAPS [61] algorithm tracks vertices through a mesh decimation hierarchy to produce a parameterization over a suitable base mesh. Regular resampling of the base mesh again leads to a semi-regular mesh.

Regular remeshing Regular meshes can efficiently be stored and transmitted as the vertex positions can be arranged in matrix form and no connectivity information is needed. *Geometry images* [38] are produced by successively introducing cuts into a mesh in order to open it into a topological disk and to reduce the distortion of the subsequent parameterization.

Other examples Alliez et al. [4] create an *atlas* of the mesh and conformally parameterize each patch over the unit square. Then they use standard image processing operations on these images instead of on the mesh. Approaches that do not need a parameterization of the mesh include the *shrink-wrapping* algorithm [59] and the *Anisotropic Polygonal Remeshing* method [2].

4 Subdivision

Subdivision schemes have become increasingly popular in recent years because they provide a simple and efficient construction of smooth curves and surfaces. In contrast to plain piecewise polynomial representations like Bézier patches and NURBS, subdivision schemes can easily represent smooth surfaces of arbitrary topology.

Implementation and application of subdivision surfaces is straightforward and intuitive, hence these topics can be taught in a basic computer graphics course or to industrial practitioners who might not be interested in the mathematical background. The analysis of subdivision schemes, however, is mathematically involved and therefore better suited for in-depth courses on geometric modeling or for a seminar.

In the following sections, c_i^k generally signifies the i-th control point of a control polygon or of a control mesh on subdivision level k. We will also freely move forth and back from the curve to the surface setting, depending on which of the two is better suited for presenting the concepts.

4.1 Subdivision schemes

▷ *2-scale relation, subdivision mask, scaling function*

▷ [99, 31, 73, 96]

We start out with curves of the type $\sum_i c_i^k \phi_i(x)$ where $\phi_i(x) = \phi(x - i)$ are integer shifts of some *scaling function* $\phi(x)$, and the points c_i^k make up the *control polygon* of the curve. If the ϕ_i satisfy a *2-scale-relation*

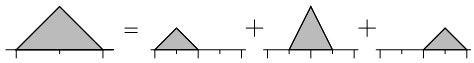
$$\phi_i(x) = \sum \alpha_j \phi_{2i+j}(2x) \quad (*)$$

it follows that $\sum c_i^k \phi_i(x) = \sum c_j^{k+1} \phi_j(2x)$ for a certain *subdivided control polygon* c_j^{k+1}, where

$$c_j^{k+1} = \sum \alpha_{j-2i} c_i^k \quad (**)$$

Examples and notations:

- The 2-scale relation $(*)$ can most easily be demonstrated for linear B-splines.

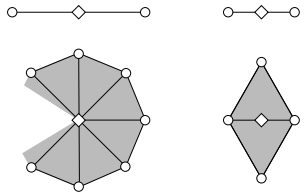


2-scale relation

- Formula $(**)$ can be split for the even and odd control points as

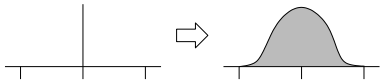
$$c_{2j}^{k+1} = \sum \alpha_{2i} c_{j-i}^k \quad \text{and} \quad c_{2j+1}^{k+1} = \sum_j \alpha_{2i-1} c_{j-i}^k$$

- A more graphical way to give the coefficients α_i is by means of *subdivision masks*,



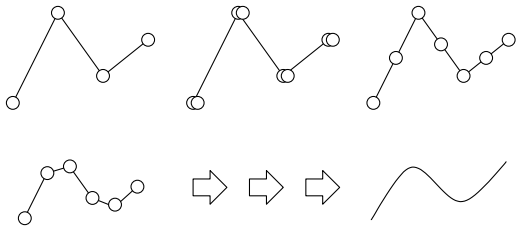
Subdivision masks

- Given subdivision coefficients α_i, it is in general not possible to find a closed form expression for the basis function $\phi(x)$. However, if the basis function exists, it can be approximated by applying the subdivision scheme to the Dirac vector.



Limit function

Example: Lane-Riesenfeld scheme Lane and Riesenfeld [60] give an algorithmic formulation for uniform B-spline subdivision. A single subdivision step is performed by first doubling all control points and then taking n times the average of each two consecutive control points.



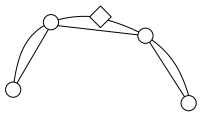
Lane-Riesenfeld scheme

From this view subdivision can more generally be considered as a topological splitting step, followed by a smoothing (averaging) step.

Example: 4-point scheme The subdivision mask of the 4-point scheme [27] is given by

$$[\alpha_i] = [-1, 0, 9, 16, 9, 0, -1]\,/16$$

and can easily be constructed using cubic interpolation.



4-point scheme

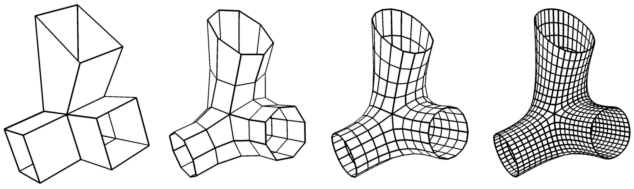
The 4-point scheme makes a good example of an *interpolatory* scheme and is also suited to demonstrate the convergence analysis (cubic precision by definition).

Example: Bivariate schemes The most widespread examples of subdivision schemes are the ones by Catmull-Clark, Doo-Sabin, Loop, Kobbelt and the Butterfly scheme. They can easily be used to demonstrate the different classes of subdivision schemes

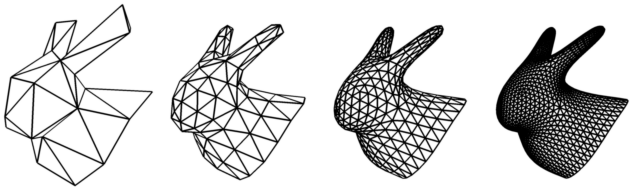
- approximating / interpolating
- quadrangle based / triangle based
- primal / dual

The table below gives a brief overview of the basic properties of these subdivision schemes (here C^k means C^k *almost everywhere*)

Doo-Sabin [23]	approx.	C^1	quad.	dual
Catmull-Clark [12]	approx.	C^2	quad.	primal
Kobbelt [52]	interpol.	C^1	quad.	primal
Butterfly (mod.) [28, 100]	interpol.	C^1	tri.	primal
Loop [64]	approx.	C^2	tri.	primal
$\sqrt{3}$ [54]	approx.	C^2	tri.	dual



Catmull–Clark subdivision scheme



Loop subdivision scheme

4.2 Uniform B-splines and box splines

▷ *piecewise polynomials, uniform B-splines, box splines*

▷ [20, 31, 72, 73]

The uniform B-splines $N^n(x)$ of degree n over the knot vector $\mathbb{Z}$ are defined by iterative convolution

$$\begin{aligned} N^0(x) &= \begin{cases} 1, & x \in [0,1) \\ 0, & \text{otherwise} \end{cases} \\ N^n(x) &= \int_0^1 N^{n-1}(x-t)dt \end{aligned}$$

From this recurrence it can easily be seen that

$$N^n(x) \in S^n_{\mathbb{Z}} = \{\text{piecewise polynomials over } \mathbb{Z}\}$$

Because $S^n_{\mathbb{Z}} \in S^n_{\mathbb{Z}/2}$, the uniform B-splines satisfy a 2-scale relation

$$N^n_i(x) = \sum \alpha^n_j N^n_{2i+j}(2x)$$

where the coefficients α^n_j can be computed by $\alpha^n_j = \binom{n+1}{j}/2^n$ These coefficients can be computed by repeatedly convolving the Dirac vector with $[1,1]$ (averaging), i.e.

$$\alpha^n = \frac{1}{2^n}\left([1] * [1,1] * \cdots * [1,1]\right)$$

Box splines are the generalization of univariate, uniform B-splines to higher dimensions and can also be defined using a convolution formula. Given directions $\mathbf{v}_0, \ldots, \mathbf{v}_m \in \mathbb{Z}^2$, the box splines are defined as

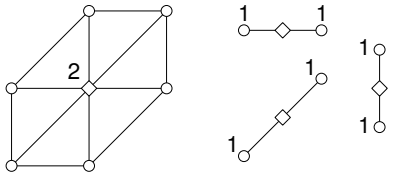
$$\begin{aligned} B(\mathbf{x}|\mathbf{v}_0\,\mathbf{v}_1) &= \begin{cases} 1, & \mathbf{x} \in [\mathbf{v}_0, \mathbf{v}_1]\,[0,1)^2 \\ 0, & \text{otherwise} \end{cases} \\ B(\mathbf{x}|\mathbf{v}_0 \ldots \mathbf{v}_m) &= \int_0^1 B(\mathbf{x} - t\mathbf{v}_m|\mathbf{v}_0 \ldots \mathbf{v}_{m-1})dt \end{aligned}$$

Analogously to the univariate case, box splines satisfy a 2-scale relation and the corresponding subdivision masks can be produced by convolving the Dirac function with the mask $[1,1]$ in the directions $\mathbf{v}_i$.

Example Let $\mathbf{v}_0 = [0,1]$, $\mathbf{v}_1 = [1,0]$ and $\mathbf{v}_2 = [1,1]$. Convolving the Dirac impulse in these directions results in

$$\begin{bmatrix}1\end{bmatrix} \xrightarrow{\otimes \begin{bmatrix}0\\1\end{bmatrix}} \begin{bmatrix}1\\1\end{bmatrix} \xrightarrow{\otimes \begin{bmatrix}1\\0\end{bmatrix}} \begin{bmatrix}1&1\\1&1\end{bmatrix} \xrightarrow{\otimes \begin{bmatrix}1\\1\end{bmatrix}} \begin{bmatrix}0&1&1\\1&2&1\\1&1&0\end{bmatrix}$$

From this mask the subdivision rules can be read off by taking the even and odd row and column entries resp.



Boxspline subdivision masks

4.3 Convergence analysis (regular case)

▷ [13, 24, 25, 28, 91]

4.3.1 Calculus basics

▷ *uniform convergence, Cauchy sequence*

The control polygons c^k_i can be interpreted as a sequence of piecewise linear functions over the knot vectors $2^{-k}\mathbb{Z}$. A sequence f^k of functions converges uniformly to a limit function f if

$$||f^k - f||_\infty \xrightarrow{k\to\infty} 0$$

where $||f||_\infty = \max|f(x)|$ is the maximum norm. If f^k converges uniformly to f and if all f^k are continuous then f is also continuous. If furthermore the derivatives $(f^k)'$ exist and converge uniformly to a function g then $f' = g$.

In the setting of subdivision analysis the limit function f is often not known, hence one needs other criteria to prove

the convergence of a sequence like e.g. the Cauchy criterion. For example, if

$$||f^{k+1} - f^k|| < \beta\alpha^k$$

for $\beta > 0$ and $0 < \alpha < 1$ then f^k is a Cauchy sequence and hence uniformly convergent.

4.3.2 Generating function formalism

▷ *generating function, convolution ↔ multiplication*

Generating functions are a convenient tool to describe and analyze subdivision schemes. The idea is to replace the control polygons c_i^k as well as the subdivision coefficients α_i by their generating functions

$$c_i^k \mapsto c^k(z) = \sum c_i^k z^i \quad \text{and} \quad \alpha_i \mapsto \alpha(z) = \sum \alpha_i z^i$$

Applying a subdivision step to the control polygon c_i^k can then easily be described by

$$c^{k+1}(z) = \alpha(z)\, c^k(z^2)$$

i.e. the convolution with the subdivision mask becomes a simple polynomial multiplication.

4.3.3 Convergence criteria

▷ *(divided) difference scheme, polynomial reproduction*

The convergence of a subdivision scheme is closely related to the existence and convergence of its (divided) difference schemes. The m-th difference of a control polygon $c^k(z)$ is given by

$$(1-z)^m c^k(z)$$

If $\alpha(z)$ reproduces polynomials of degree m, the $(m+1)$st difference scheme

$$\alpha_{m+1}(z) = \frac{\alpha(x)}{(1+z)^{m+1}}$$

exists and relates the $(m+1)$st differences of $c^{k+1}(z)$ to the $(m+1)$st differences of $c^k(z)$ by

$$(1-z)^{m+1} c^{k+1}(z) = \alpha_{m+1}(z)(1-z^2)^{m+1} c^k(z^2)$$

If furthermore the difference scheme

$$2^m \alpha_{m+1}(z)$$

of the mth *divided difference scheme* $2^m\alpha_m(z)$ is contractive, the control polygons $c^k(z)$ converge to a m-times continuously differentiable curve. Let β_i be the coefficients of $2^m\alpha_{m+1}(z)$, then the contraction property follows if

$$\max\left\{\sum |\beta_{2i}|, \sum |\beta_{2i+1}|\right\} = q < 1$$

Note that one often needs to combine multiple subdivision steps in order to be able to prove the contraction property.

4.4 Subdivision matrix formalism

▷ *subdivision matrix*

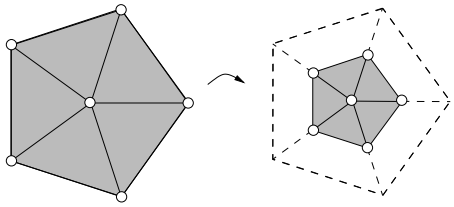
▷ [8, 71, 76, 98]

Local properties of a subdivision scheme can easily be computed using the subdivision matrix formalism. This formalism is especially useful

- for analyzing the convergence properties of surface schemes at extraordinary vertices and
- for computing explicit masks for the limit points and tangents

The basic idea is to track a finite neighborhood of a vertex p through different subdivision levels. Let $\mathbf{p}^k$ be a column vector that comprises p and a sufficiently large regular neighborhood of p at subdivision level k. Then there exists a *subdivision matrix* satisfying

$$\mathbf{p}^k = S\mathbf{p}^{k-1}$$



The subdivision matrix maps the 1-ring $\mathbf{p}^k$ to $\mathbf{p}^{k+1}$

Let $1 = \lambda_0 > \lambda_1 = \lambda_2 > \lambda_3 \ldots$ be the eigenvalues of S, let $\mathbf{x}_0, \mathbf{x}_1, \ldots$ be the corresponding (right) eigenvectors, i.e. $S\mathbf{x}_i = \lambda_i \mathbf{x}_i$ and let $\mathbf{y}_0, \mathbf{y}_1, \ldots$ be the associated left eigenvectors (with $\mathbf{x}_i^T \mathbf{y}_j = \delta_{ij}$). Then we can expand $\mathbf{p}^0$ as

$$\mathbf{p}^0 = \sum \omega_i \mathbf{x}_i$$

where we have set $\omega_i = \mathbf{y}_i^T \mathbf{p}^0$. Subdividing the mesh k times means applying S^k to $\mathbf{p}^0$, yielding

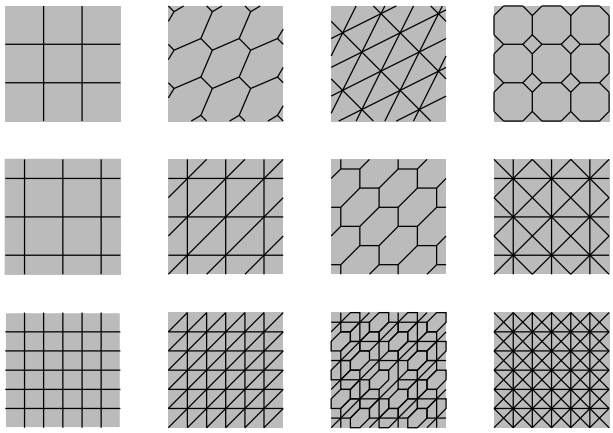
$$\mathbf{p}^k = S^k \mathbf{p}^0 = \sum \lambda_i^k \omega_i \mathbf{x}_i \quad (***)$$

Hence, as $\lambda_0 = 1 > \lambda_1$ the scheme is convergent and $\lim \mathbf{p}_0^k = \omega_0$. Further analysis of the scheme requires a reparameterization by the *characteristic map* which is defined as the limit surface associated with the planar control net $[\mathbf{x}_1, \mathbf{x}_2]$. If this map is *regular and injective* the subdivision scheme produces C^1 continuous surfaces and the eigenvector decomposition $(***)$ can be viewed as a generalized Taylor expansion of the limit surface around the extraordinary vertex. In particular, the limit tangents are given by ω_1 and ω_2 resp.

4.5 Topological refinement

▷ *uniform refinement, primal/dual graph*

Splitting operators can be constructed by combining *uniform refinement* and *duality*.



Refinement and duals

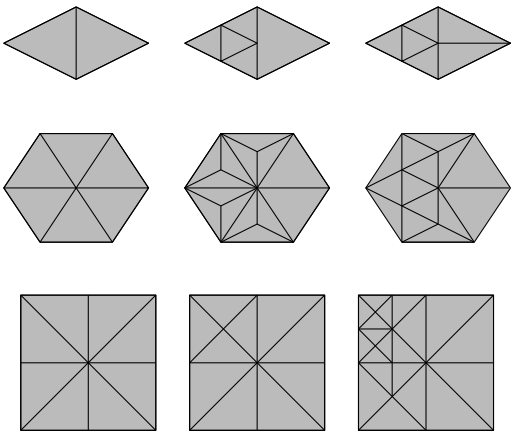
The lower row shows the uniform refinement of the lattices in the middle row. The upper row shows their dual lattices.

4.6 Adaptive subdivision

▷ *red/green triangulation,* $\sqrt{3}$ *subdivision*

▷ [99, 54, 95]

The complexity of the control meshes increases exponentially with the subdivision level k. Adaptive subdivision schemes reduce the costs by subdividing only in *critical* areas, e.g. along the object silhouette or when the normals of two adjacent faces differ to much. Regular tilings, however, cannot be adaptively subdivided without introducing gaps. This leads to ad-hoc solutions like the *red-green triangulation*. Dropping the regularity requirements leads to subdivision schemes that are better suited for adaptive subdivision.



Adaptive subdivision

5 Multiresolution techniques

When it comes to geometry processing, multiresolution techniques offer two distinct advantages: technical and semantical. First, algorithms that are able to exploit a multiresolution representation typically can achieve significant speedups. Second, multiresolution representations naturally separate the detail of a model from the base shape. These two advantages allow one to implement intuitive modeling metaphors for interactive editing of triangle meshes.

5.1 Wavelets and Multiresolution analysis

▷ *scaling function, wavelets, filter bank, (bi-)orthogonality*

▷ [15, 18, 85, 55]

Wavelets can be introduced using either a summation notation or a matrix formalism. The summation notation is better suited to demonstrate the convolution nature of the reconstruction and decomposition operators and nicely fits within the subdivision framework presented in the previous section. The matrix formalism, however, avoids the cumbersome index notation, eases the handling of boundaries and leads to compact formulas that are much easier to read. In this section we use both notations simultaneously for educational purposes.

The starting point for multiresolution analysis is a set of nested spaces

$$V^k \subset V^{k+1}$$

and corresponding complement spaces W^k satisfying

$$V^{k+1} = V^k \oplus W^k$$

We assume that V^k and W^k are spanned by scaled translates of a *scaling function* $\phi(x)$ and a *mother wavelet* $\psi(x)$ respectively, i.e.

$$\begin{aligned} V^k &= \operatorname{span}\{\phi_i^k(x) = \phi(2^k x - i)\} \\ W^k &= \operatorname{span}\{\psi_i^k(x) = \psi(2^k x - i)\} \end{aligned}$$

For the matrix formalism we also introduce the row vectors

$$\mathbf{\Phi}^k = [\phi_i^k], \quad \mathbf{\Psi}^k = [\psi_i^k]$$

The decomposition $V^k \oplus W^k = V^{k+1}$ implies a 2-scale relation on the basis functions

$$\left.\begin{aligned} \phi_i^k &= \sum \alpha_j \phi_{2i+j}^{k+1} \\ \psi_i^k &= \sum \beta_j \phi_{2i+j}^{k+1} \end{aligned}\right| \; [\,\mathbf{\Phi}^k \mid \mathbf{\Psi}^k\,] = \mathbf{\Phi}^{k+1} \left[\begin{array}{c|c} A_k & B_k \end{array}\right]$$

for certain coefficients α_j, β_j and matrices A_k, B_k resp. On the other hand each function $\in V^{k+1}$ has a representation with respect to $\mathbf{\Phi}^{k+1}$ as well as with respect to $[\,\mathbf{\Phi}^k \mid \mathbf{\Psi}^k\,]$,

$$\left.\begin{aligned} &\sum c_i^{k+1}\phi_i^{k+1} = \\ &\qquad \sum c_i^k \phi_i^k + \sum d_i^k \psi_i^k \end{aligned}\right| \; \mathbf{\Phi}^{k+1}\mathbf{c}^{k+1} = [\,\mathbf{\Phi}^k \mid \mathbf{\Psi}^k\,] \left[\frac{\mathbf{c}^k}{\mathbf{d}^k}\right]$$

where we define the column vectors $\mathbf{c}^k$ and $\mathbf{d}^k$ as

$$\mathbf{c}^k = [c_i^k], \quad \mathbf{d}^k = [d_i^k]$$

Applying the 2-scale relation on the right hand side of the equation we see that the control points on different scales are related by

$$\left.\begin{aligned} c_j^{k+1} &= \sum \alpha_{j-2i} c_i^k \\ &+ \sum \beta_{j-2i} d_i^k \end{aligned}\right| \; \mathbf{c}^{k+1} = \left[\begin{array}{c|c} A_k & B_k \end{array}\right] \left[\frac{\mathbf{c}^k}{\mathbf{d}^k}\right]$$

The above equation defines a *reconstruction operator* or a *synthesis filter*. The most general way to describe an associated *decomposition operator* or *analysis filter* is via dual bases $\tilde{\phi}(x)$ and $\tilde{\psi}(x)$ such that

$$\begin{aligned} \langle\, \phi_i^k(x), \tilde{\phi}_j^k(x) \,\rangle &= \langle\, \psi_i^k(x), \tilde{\psi}_j^k(x) \,\rangle &&= \delta_{ij} \quad \text{and} \\ \langle\, \phi_i^k(x), \tilde{\psi}_j^k(x) \,\rangle &= \langle\, \psi_i^k(x), \tilde{\phi}_j^k(x) \,\rangle &&= 0 \end{aligned}$$

In matrix notation the above condition can be written as

$$\langle\, [\,\mathbf{\Phi}^k \mid \mathbf{\Psi}^k\,], [\,\tilde{\mathbf{\Phi}}^k \mid \tilde{\mathbf{\Psi}}^k\,] \,\rangle = I$$

If there also exists a 2-scale relation for the dual basis

$$\begin{aligned}\tilde{\phi}_i^k &= \sum \gamma_j \tilde{\phi}_{2i+j}^{k+1} \\ \tilde{\psi}_i^k &= \sum \delta_j \tilde{\phi}_{2i+j}^{k+1}\end{aligned} \quad \left| \quad [\,\tilde{\mathbf{\Phi}}^k \mid \tilde{\mathbf{\Psi}}^k\,] = \tilde{\mathbf{\Phi}}^{k+1} \left[\begin{array}{c|c} C_k & D_k \end{array} \right] \right.$$

the decomposition operator can easily be described as

$$\begin{aligned}c_i^k &= \sum \gamma_{j-2i} c_j^{k+1} \\ d_i^k &= \sum \delta_{j-2i} c_j^{k+1}\end{aligned} \quad \left| \quad \left[\frac{\mathbf{c}^k}{\mathbf{d}^k} \right] = \left[\frac{C_k^T}{D_k^T} \right] \mathbf{c}^{k+1} \right.$$

This general setting is called the *biorthogonal wavelet* setting.

In the *semi-orthogonal* setting we further require that the decomposition $V^{k+1} = V^k \oplus W^k$ is orthogonal,

$$\langle \mathbf{\Phi}^k, \mathbf{\Psi}^k \rangle = 0$$

which leads to better approximation properties of the reconstruction operator. In the fully *orthogonal wavelet* setting, we even require that

$$\langle [\,\mathbf{\Phi}^k \mid \mathbf{\Psi}^k\,], [\,\mathbf{\Phi}^k \mid \mathbf{\Psi}^k\,] \rangle = I$$

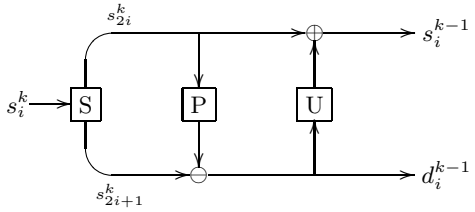
In this case the primal and dual basis are the same and the decomposition operator becomes trivial.

5.2 Lifting scheme

▷ *lifting scheme, split-predict-update-merge*

▷ [80, 87, 88]

Lifting allows us to construct filter banks entirely in the spatial domain and can hence also be taught to audiences that are not acquainted with Fourier methods. Instead of explicitly specifying scaling functions and wavelets, the decomposition process is made up of so-called splitting, prediction, update, scaling and merging steps that are arranged in a flow chart



In the simplest case the prediction operator P is a subdivision operator and the update operator U is chosen such as to preserve higher order moments. The reconstruction operator is derived from the decomposition operator by simply reversing all arrows and changing the signs. Using the lifting scheme, the wavelet decomposition/reconstruction can be performed in-place and in linear time.

Example: Haar Wavelets The *Haar transform* is a special wavelet transform. Its scaling function and mother wavelet are given by

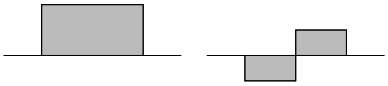
$$\phi(x) = \begin{cases} 1, & x \in [0,1) \\ 0, & \text{otherwise} \end{cases}$$

and

$$\psi(x) = \frac{1}{2}\left(\phi(2x-1) - \phi(2x)\right)$$

leading to the prediction and update steps

$$\begin{aligned} d_i^{k-1} &\leftarrow s_{2i+1}^k - s_{2i}^k \\ s_i^{k-1} &\leftarrow s_{2i}^k + \frac{1}{2} d_i^{k-1} \end{aligned}$$



Haar scaling function and wavelet

Example: B-spline Wavelets Using linear splines as scaling functions leads to the prediction step

$$d_i^{k-1} \leftarrow s_{2i+1}^k - \left(s_{2i}^{k+1} + s_{2i+2}^{k+1} \right) /2$$

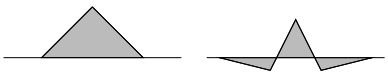
To preserve the average (0th moment)

$$2 \sum s_i^k = \sum s_i^{k+1}$$

one assumes that the update step has the form

$$s_i^k \leftarrow s_{2i}^{k+1} + \alpha \left(d_{i-1}^k + d_i^k \right)$$

and solves for $\alpha = 1/4$. Due to symmetry reasons the 1-st order moment is then also preserved.



Linear B-spline scaling function and wavelet

5.3 Surface hierarchies

▷ *semi-regular meshes, coarse-to-fine hierarchy, fine-to-coarse hierarchy*

▷ [41, 58, 101]

In order to carry over the concept of wavelet decomposition to arbitrary polygonal surfaces, one has to mimic the behavior of the reconstruction and decomposition operators. For this let $\uparrow$ and $\downarrow$ be a pair of compatible upsampling and downsampling operators, i.e. $\downarrow (\uparrow (\mathcal{M}))$ has the same mesh connectivity as $\mathcal{M}$. We can then define a hierarchy

$$\begin{array}{ccccccc} \cdots & \rightarrow & \mathcal{M}^k & \rightarrow & \mathcal{M}^{k+1} & \rightarrow & \cdots \\ & \nearrow & & \nearrow & & \nearrow & \\ \cdots & & \mathcal{D}^k & & \mathcal{D}^{k+1} & & \cdots \end{array}$$

of meshes $\mathcal{M}^k$ and associated detail $\mathcal{D}^k$ by defining a reconstruction operator as

$$S: \quad \mathcal{M}^k, \mathcal{D}^k \mapsto \mathcal{M}^{k+1} = \uparrow (\mathcal{M}^k) + \mathcal{D}^k$$

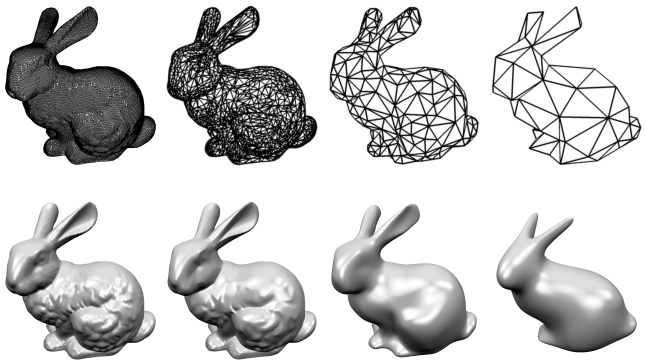
and a decomposition operator as

$$A: \quad \mathcal{M}^{k+1} \mapsto \begin{cases} \mathcal{M}^k = \downarrow (\mathcal{M}^{k+1}) \\ \mathcal{D}^k = M^{k+1} - \uparrow (\mathcal{M}^k) \end{cases}$$

Note that $\mathcal{D}^k$ and $\mathcal{M}^{k+1}$ have the same connectivity.

By definition the reconstruction and decomposition are inverse to each other. The different frequency bands are captured by the detail coefficients D^k. In general multiple detail coefficients are associated with each vertex (one for each level). This redundancy can be avoided by choosing an

interpolatory upsampling operator $\uparrow$. Furthermore, in order to achieve intuitive results, the detail coefficients should be encoded with respect to *local frames*. In the semi-regular setting *(coarse-to-fine hierarchies)* $\uparrow$ can be chosen to be a subdivision operator and lifting can be used to improve the filters. In the irregular setting *(fine-to-coarse hierarchies)* the downsampling $\downarrow$ is performed by some mesh decimation algorithm. Upsampling $\uparrow$ is then done by re-inserting the vertices, followed by a smoothing step.



Multiresolution hierarchy

6 Exercise courses

As programming a polygonal mesh data structure can be quite cumbersome it has proven to be more effective to employ one of the publicly available libraries like e.g. CGAL [14] or OpenMesh [68]. Exercises that we have assigned per student and per week are e.g. to implement

- Garland and Heckbert's error quadric mesh decimation scheme
- Taubin's $\lambda|\mu$ smoothing
- mesh parameterization using Floater's weights
- a tool to visualize mesh curvatures
- Marching cubes
- Delaunay triangulation (2D)
- Loop subdivision

References

[1] U. Adamy, J. Giesen, and M. John. New techniques for topologically correct surface reconstrucion. In *Proc. 11th IEEE Visualization Conference (VIS)*, pages 373–380, 2000.

[2] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic polygonal remeshing. In *SIGGRAPH proceedings*, pages 485–493, 2003.

[3] P. Alliez and M. Desbrun. Progressive compression for lossless transmission of triangle meshes. In *SIGGRAPH Proceedings*, 2001.

[4] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. In *SIGGRAPH proceedings*, pages 347–354, 2002.

[5] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.

[6] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *SIGGRPAH 1998 Proceedings*, pages 415–422, 1998.

[7] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *Sixth ACM Symposium on Solid Modeling and Applications*, pages 249–260, 2001.

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

[9] B. G. Baumgart. A polyhedron representation for computer vision. In *National Computer Conference*, pages 589–596, 1975.

[10] S. Campagna, L. Kobbelt, and H.-P. Seidel. Directed edges — A scalable representation for triangle meshes. *Journal of Graphics Tools: JGT*, 3(4):1–12, 1998.

[11] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice Hall College Div, 1976.

[12] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10:350–355, September 1978.

[13] A. Cavaretta, W. Dahmen, and C. Micchelli. Stationary subdivision. In *Memoirs of the AMS*, volume 453, 1991.

[14] CGAL - computational geometry algorithms library. http://www.cgal.org.

[15] C. K. Chui. *An Introduction to Wavelets (Wavelet Analysis and Its Applications, Volume 1)*. Academic Press, 1992.

[16] D. Cohen-Steiner and J.-M. Morvan. Restricted delaunay triangulations and normal cycle. In *ACM Symposium on Computational Geometry*, pages 237–246, 2003.

[17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH 96 proceedings*, pages 303–312, 1996.

[18] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

[19] M. de Berg, M. van Krefeld, M. Overmars, and O. Schwarzkopf, editors. *Computational Geometry*. Springer, 2000.

[20] C. de Boor, K. Hoellig, and S. Riemenschneider. *Box splines*. Springer, 1993.

[21] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Eurographics 2002 Proceedings*, 2002.

[22] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH 99 proceedings*, pages 317–324, 1999.

[23] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, September 1978.

[24] N. Dyn. Subdivision schemes in computer aided geometric design. In *Advances in Numerical Analysis II, Wavelets, Subdivision and Radial Functions*, pages 36–104, 1991.

[25] N. Dyn, J. A. Gregory, and D. Levin. Analysis of uniform binary subdivision schemes for curve design. *Constructive Approximation*, 7:127–147, 1991.

[26] N. Dyn, K. Hormann, S.-J. Kim, and D. Levin. Optimizing 3d triangulations using discrete curvature analysis. In *Innovations in Applied Mathematics*. Vanderbilt University Press, 2001.

[27] N. Dyn, D. Levin, and J. A. Gregory. A 4-point interpolatory subdivision scheme for curve design. *Computer Aided Geometric Design*, 4:257–268, 1987.

[28] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[29] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH 95 Proceedings*, pages 173–182, 1995.

[30] H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. *ACM Trans. Graphics*, 13:43–72, 1994.

[31] G. Farin. *Curves and Surfaces for CAGD*. Morgan Kaufmann, 2002.

[32] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Comp. Aided Geom. Design*, 14:231–250, 1997.

[33] S. Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[34] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 96 proceedings*, pages 209–216, 1996.

[35] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

[36] S. F. F. Gibson. Using distance maps for accurate surface reconstruction in sampled volumes. In *IEEE Volume Visualization Symposium*, pages 23–30, 1998.

[37] C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and compression of 3d-meshes. In A. Iske, E. Quak, and M. Floater, editors, *Tutorials on multiresolution in geometric modeling*. Springer, 2002.

[38] X. Gu, S. Gortler, and H. Hoppe. Geometry images. In *SIGGRAPH Proceedings*, pages 355–361, 2002.

[39] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *ACM Trans. Graphics*, 4:74–123, 1985.

[40] S. Gumhold and W. Straßer. Real time compression of triangle mesh connectivity. In *SIGGRAPH proceedings*, pages 133–140, 1998.

[41] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *SIGGRAPH 99 proceedings*, pages 325–334, 1999.

[42] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *SIGGRAPH 00 Proceedings*, pages 95–102, 2000.

[43] B. Hamann. Curvature approximation for triangulated surfaces. In *Geometric Modeling, Computing Supplement 8*, pages 139–153. Springer, 1993.

[44] H. Hoppe. Progressive meshes. In *SIGGRAPH 96 proceedings*, pages 99–108, 1996.

[45] H. Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH 97 proceedings*, pages 189–198, 1997.

[46] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 92 Proceedings*, pages 71–78, 1992.

[47] M. Isenburg and M. Snoeyink. Face fixer: Compressing polygon meshes with properties. In *SIGGRAPH proceedings*, pages 263–270, 2002.

[48] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Siggraph 02 Proceedings*, pages 339–346, 2002.

[49] L. Kettner. Designing a data structure for polyhedral surfaces. In *Proc. of the 14th ACM Symp. on Computational Geometry*, pages 146–154, 1998.

[50] A. Khodakovsky and I. Guskov. Compression of normal meshes. In *Geometric Modeling for Scientific Visualization*, pages 189–206, 2002.

[51] J. Kim and S. Lee. Truly selective refinement of progressive meshes. In *Graphics Interface Proceedings*, pages 101–110, 2001.

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

[53] L. Kobbelt. Discrete fairing and variational subdivision for freeform surface design. *The Visual Computer Journal*, 2000.

[54] L. Kobbelt. $\sqrt{3}$ subdivision. In *SIGGRAPH 00 Proceedings*, pages 103–112, 2000.

[55] L. Kobbelt. Multiresolution techniques. In G. Farin, J. Hoschek, and M.-S. Kim, editors, *The Handbook of Computer Aided Geometric Design*. Elsevier, 2002.

[56] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH 2001 proceedings*, pages 57–66, 2001.

[57] L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. In *Graphics Interface '98 Proceedings*, 1998.

[58] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH 98 proceedings*, pages 105–114, 1998.

[59] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. In *Eurographics 99 proceedings*, pages 119–130, 1999.

[60] J. M. Lane and R. F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Pattern Analysis Machine Intell.*, 2(1):35–46, 1980.

[61] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. In *SIGGRAPH 98 Proceedings*, pages 95–104, 1998.

[62] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Siggraph 2002 Proceedings*, pages 362–371, 2002.

[63] P. Lindstrom. Out-of-core simplification of large polygonal models. In *SIGGRAPH 00 proceedings*, pages 259–262, 2000.

[64] C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[65] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3d surface reconstruction algorithm. In *SIGGRAPH 87 proceedings*, pages 163–169, 1987.

[66] M. Mantila. *An Introduction to Solid Modeling*. Computer Science Press, Maryland, 1988.

[67] M. Meyer, M. Desbrun, P. Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, 2002.

[68] OpenMesh. http://www.openmesh.org.

[69] K. Polthier and M. Schmies. Straightest geodesics on polyhedral surfaces. In *Mathematical Visualization*, page 391, 1998.

[70] K. Polthier and M. Schmies. Geodesic flow on polyhedral surfaces. In *Proc. Joint EG - IEEE TCVG Symposium*, pages 179–188, 1999.

[71] H. Prautzsch. Smoothness of subdivision surfaces at extraordinary points. *Adv. Comp. Math.*, 9:377–390, 1998.

[72] H. Prautzsch and W. Boehm. Box splines. In G. Farin, J. Hoschek, and M.-S. Kim, editors, *The Handbook of Computer Aided Geometric Design*. Elsevier, 2002.

[73] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*. Springer, 2002.

[74] Qhull. http://www.thesa.com/software/qhull/.

[75] N. Ray and B. Levy. Hierarchical least squares conformal maps. In *Eurographics 2003 Proceedings*, 2003.

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

[77] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[78] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *SIGGRAPH 2001 proceedings*, pages 409–416, 2001.

[79] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parameterization. In *Eurographics Workshop on Rendering 2002 Proceedings*, pages 87–100, 2002.

[80] P. Schröder, W. Sweldens, M. Cohen, T. DeRose, and D. Salesin. *Wavelets in Computer Graphics*. Siggraph 96 Course Notes, 1996.

[81] W. Schroeder, J. Zarge, and W. Lorensen. Decimation of triangle meshes. In *SIGGRAPH proceedings*, pages 65–70, 1992.

[82] A. Sheffer and E. de Sturler. Surface parameterization for meshing by triangulation flattening. In *Proc. 9th International Meshing Roundtable*, pages 161–172, 2000.

[83] A. Sheffer and J. Hart. Seamster: Inconspicuous low–distortion texture seam layout. In *IEEE Visualization Proceedings*, pages 291–298, 2002.

[84] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[85] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley Cambridge, 1996.

[86] V. Surazhsky and C. Gotsman. Explicit surface remeshing. In *Symposium on Geometry Processing proceedings*, pages 20–30, 2003.

[87] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[88] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.

[89] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 95 proceedings*, pages 351–358, 1995.

[90] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus. Progressive forest split compression. In *SIGGRAPH proceedings*, pages 123–132, 1998.

[91] A. Taylor. *Advanced calculus*. Ginn and Company, 1955.

[92] C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface*, 1998.

[93] G. Turk. Re-tiling polygonal surfaces. In *SIGGRAPH proceedings*, pages 55–64, 1992.

[94] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc.*, 13:743–768, 1963.

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

[96] J. D. Warren and H. Weimer. *Subdivision Methods for Geometric Design*. Morgan Kaufman, 2001.

[97] K. Weiler. The radial edge structure: A topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications*, 1998.

[98] D. Zorin. *C^k Continuity of Subdivision Surfaces*. PhD thesis, California Institute of Technology, Department of Computer Sciences, 1996.

[99] D. Zorin et al. Subdivision for modeling and animation. In *SIGGRAPH 00 Course Notes*, 2000.

[100] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *SIGGRAPH 96 Proceedings*, pages 189–192, 1996.

[101] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH 97 Proceedings*, pages 259–268, 1997.