54 SURFACE SIMPLIFICATION AND 3D GEOMETRY COMPRESSION

Jarek Rossignac

INTRODUCTION

Central to 3D modeling, graphics and animation, ***triangle meshes*** are used in Computer Aided Design, Visualization, Graphics, and video games to represent polyhedra, control meshes of subdivision surfaces, or tessellations of parametric surfaces or level sets. A triangle mesh that accurately approximates the surface of a complex 3D shape may contain millions of triangles. This chapter discusses techniques for reducing the delays in transmitting it over the Internet. The ***connectivity***, which typically dominates the storage cost of uncompressed meshes, may be compressed down to about one bit per triangle by compactly encoding the parameters of a triangle-graph construction process and by transmitting the vertices in the order in which they are used by this process. Vertex coordinates, i.e., the ***geometry***, may often be compressed to less than 5 bits each through quantization, prediction, and entropy coding. Thus, ***compression*** reduces storage of triangle meshes to about a byte per triangle. When necessary, file size may be further reduced through ***simplification***, which collapses edges or merges clusters of neighboring vertices to decrease the total triangle count. The application may select the appropriate level-of-detail; trading fidelity for transmission speed. In applications where preserving the exact geometry and connectivity of the mesh is not essential, the triangulated surface may be ***re-sampled*** to produce a mesh with a more regular connectivity and with vertices that are constrained to, each, lie on a specific curve, and thus may be fully specified by a single parameter. Re-sampling may improve compression significantly, without introducing noticeable distortions. Furthermore, when the accuracy of a simplified or re-sampled model received by a client is insufficient, compressed upgrades may be downloaded as needed to ***refine*** the model in a ***progressive*** fashion.

Due to space limitations, we focus primarily on triangle meshes that are homeomorphic to triangulation of a sphere. Strategies for extending the compression, simplification, and refinement techniques to more general meshes, which include polygonal meshes, manifold meshes with handles and boundaries, or nonmanifold models; to tetrahedral, higher dimensional, or animated meshes; and to models with texture or property maps, are discussed elsewhere.

GLOSSARY

Mesh: A set of triangles homeomorphic to the triangulation of a sphere.

Geometry (of a mesh): The positions of the vertices (possibly described by 3 coordinates each).

Incidence: The definition of the triangles of the mesh, each as 3 vertex Ids.

Connectivity: Incidence, combined with adjacency and order relations, which may be derived form the incidence.

Connectivity compression: Encoding the incidence, while ignoring the geometry.

Geometry compression: Encoding of the vertex locations, while assuming that the relevant connectivity information will be available to the decoder.

Simplification: The process of merging the vertices of a mesh to generate a new mesh that approximates the original one, but comprises fewer triangles.

Level-of-Detail (LOD): One of a series of increasingly simplified models that approximate an original shape by trading fidelity for a decrease in triangle count.

Re-sampling: The approximation of an original mesh by one with a new connectivity and a new set of vertices, selected to minimize error and maximize compression.

Progressive transmission: A protocol in which the client receives the lowest LOD, possibly followed by upgrades, if and when needed.

Single-rate compression: A compressed representation that does not support progressive transmission.

54.1 SIMPLE TRIANGLE MESHES

In this section, we introduce the terminology, properties, data structures, and operators used in this chapter. We assume that the mesh is simple, i.e., homeomorphic to the triangulation of a sphere.

GLOSSARY

Vertex and triangle count: A mesh with v vertices and t triangles satisfies $t = 2 - 4$.

Triangle: A node of the connectivity graph representing a closed point-set that is the convex hull of three noncollinear vertices.

Surface (of a mesh): The union of the point-sets of its triangles.

Edge: A link in the connectivity graph representing a relatively open line segment joining two vertices of a mesh, but not containing them.

Face: The relative interior of a triangle, i.e., the triangle minus the union of the point-sets of its edges and vertices.

Corner: A corner c associates a vertex c.v with one of its incident triangles c.t.

Cascading corner operators: The notation c.n.v stands for (c.n).v.

Next and previous corners: The two corners in c.t other than c are denoted c.n and c.p.

Incidence table: An array V of $3t$ entries, where V[c] is denoted c.v and contains a vertex Id Entries for c.p, c, and c.n are consecutive in V. Therefore, c.t is the integer division c.t div 3; c.n is c-2, when c mod 3 is 2, and c+1 otherwise; and c.p is c.n.n.

Opposite corner: Two corners c and b are opposite when b.n.v=c.p.v and b.p.v=c.n.v. The opposite corner of c, denoted c.o, is a corner Id stored as the entry O[c] in the O table.

Orientation: The orientation of a triangle c.t is defined by the choice of c.n amongst the other two corners of c.t. The mesh is globally oriented so that c.n.v=c.o.p.v for all corners c.

Corner Table: The two arrays, V and O.

Left and right neighbors of a corner: For convenience, we define c.l to be c.n.o and c.r to be c.p.o.

Vertex-Spanning Tree (VST): A subset of edges whose union of their point-sets with all the vertices forms a tree.

Cut-Edges: The edges contained in a given VST.

Cut: The union of the cut-edges with all of the vertices.

Hinge-Edges: Edges that are not cut edges.

Web: Union of all the faces and of all the hinge-edges.

Triangle-Spanning-Tree (TST): Rooted graph with hinge-edge links and triangle nodes.

54.1.1 TERMINOLOGY AND PROPERTIES

Consider a mesh with v vertices, e edges, and t triangles.

GEOMETRY AND INCIDENCE

A mesh is usually defined in terms of a set of ***vertices*** and its triangle-vertex ***incidence***. The vertex description comprises ***geometry*** (three coordinates per vertex) and optionally ***photometry*** (surface normals, vertex colors, or texture coordinates), not discussed in this chapter. ***Incidence*** defines each triangle by three integer references identifying its vertices. Simple and regular data structures, such as the ***Corner Table*** described below, and formats, such as the ***indexed face set*** of VRML [VRML97] may be used for representing geometry and incidence.

An ***uncompressed representation*** uses $12v$ bytes for geometry and $12t$ bytes for incidence. Given that $t \approx 2v$, the total cost is $144t$ bits, of which two thirds are used for incidence.

CUT, WEB, AND SPANNING TREES

The vertices may be imagined to be placed on the plane so that the edges and vertices are mutually disjoint. The union of these edges and vertices partition the rest of the plane into ***cells*** that are each bounded by 3 edges and 3 vertices. One of the cells is unbounded. This mapping forms a ***planar graph*** of the mesh.

A ***Vertex-Spanning Tree*** (VST) of a triangle mesh is a subset of its edges, selected so that their union with all of the vertices forms a tree (connected cycle-free graph). We assume henceforth that a particular VST has been chosen for the mesh. The edges that it includes are called the ***cut-edges***. The union of the cut-edges with all the vertices is called a ***cut***. Because the VST is a tree, there are $v - 1$ cut-edges.

We use the term ***web*** to denote the difference between the surface and the point-set of its cut. Edges that are not cut-edges are called ***hinge- edges***. The web is composed of all of the faces and of all of the hinge-edges. Removing the loopless cut from the surface will leave a web that is a simply connected (relatively open), triangulated 2D point-set in $\mathbb{R}^3$. The web may be represented by an acyclic graph, whose nodes correspond to faces and whose links correspond to hinge edges. Thus there are $t-1$ hinge edges.

Note that by selecting a leaf of this graph as root and orienting the links, we can always turn it into a binary tree, which we call the ***Triangle-Spanning-Tree*** (TST), a a spanning tree of the dual of the planar graph.

EULER EQUATION

Because an edge is either hinge or cut, the total number of edges, e, is $v-1+t-1$. Each triangle uses 3 edges and each edge is used by 2 triangles. Thus the number e of edges is also equal to $3t/2$. Combining these two equations yields $t = 2v-4$, a special case of Euler's formula $f - e + v = 2$.

54.1.2 CORNER TABLE REPRESENTATION

We present a simple data structure for meshes. We call it the Corner Table [RSS02] and use it to explain the details of connectivity compression.

DATA STRUCTURE AND OPERATORS

The corner table is composed of three arrays: G, V, and O.

The geometry is stored in the coordinate table, G, where G[v] contains the triplet of the coordinates of vertex number v, and will be denoted v.g. The order in which the vertices are listed in G is arbitrary, but defines the Id (integer) associated with each vertex.

Triangle-vertex incidence defines each triangle by the three Ids. of its vertices, which are stored as consecutive entries in the ***V-table***. Note that each one of the $3t$ entries in V represents a ***corner***(association of a triangle with one of its vertices). Let c be such a corner. Let c.t denote its triangle and c.v its vertex. Recall that c.v and c.t are integers in $[0, v-1]$ and $[0, t-1]$ respectively. Let c.p and c.n refer to the ***previous*** and ***next*** corner in the cyclic order of vertices around c.t.

Although G and V suffice to completely specify the triangles and thus the surface they represent, they do not offer direct access to a neighboring triangle or vertex. We chose to use the ***opposite*** corner Id, c.o, cached in the ***O table*** to accelerate mesh traversal from one triangle to its neighbors. For convenience, we

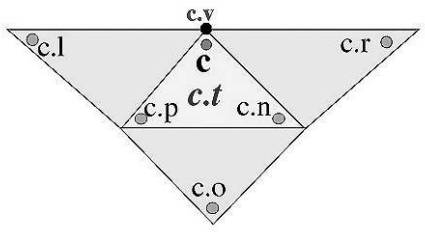


FIGURE 54.1.1
Corner operators for traversing and processing a corner table representation of a triangle mesh.

introduce the operators c.l and c.r, which return the ***left*** and ***right neighbors*** of c (Figure 54.1.1).

Given a corner c, the choice of which of the other two vertices of c.t is c.n defines one of two possible ***orientations*** for c.t. We assume that all triangles have been consistently ***oriented***, so that c.n.v=c.o.p.v for all corners c.

MESH TRAVERSAL ON A CORNER TABLE

Assume that the Boolean c.t.m is set to TRUE when the triangle c.t has been visited. The procedure: visit(c) { if !c.t.m then c.t.m = TRUE; visit(c.r); visit(c.l) } will visit all the triangles in a depth-first order of a TST.

THE COMPUTATION OF THE O-TABLE FROM V

Given the V-table, the entries in O may be computed by a double loop over corners, identifying the opposite corners. A faster approach sorts the triplets { min(c.n.v,c.p.v), max(c.n.v,c.p.v), c } into bins. All entries in a bin have the same first record: min(c.n.v,c.p.v), an integer in $[0, v-1]$. There are rarely more than 20 entries in a bin. Then, we sort the entries in each bin by the second record: max(c.n.v,c.p.v). Now, pairs with identical first two records are consecutive and correspond to opposite corners, identified by the third record in each triplet. Thus, if a sorted bin contains consecutive entries (a,b,c) and (a,b,d), we set c.o:=d and d.o:= c.

54.1.3 REDUCTIONS OF THE TRANSMISSION COST

Because it can be easily recreated, the O-table needs not be transmitted. Furthermore, the $31 - \log_2 v$ leading zeros of each entry in the V table need not be transmitted. Thus, a compact, but uncompressed representation of a triangle mesh requires $48t$ bits for the coordinates and $3t \log_2 t - 3t$ bits for the V-table.

54.2 GEOMETRY COMPRESSION

The compression of vertex coordinates usually combines three steps: quantization, prediction, and statistical coding. The combined three stages yield a 7-to-1 compression of geometry.

GLOSSARY

Normalization: Linear transformation of coordinates so that their range spans $[0, 2^B - 1]$.

Quantization: Rounding of each normalized vertex coordinate to the nearest integer.

Prediction: The prediction of the quantized location of a new vertex from its neighbors.

Parallelogram prediction: Using c.n.v.g + c.p.v.g - c.v.g as a prediction for

c.o.v.g.

Residues: The differences between the actual quantized coordinates and their prediction.

Statistical coding: A bit-efficient encoding of the residues, exploiting the bound on their magnitude and the statistics of the nonuniform distribution of their frequencies.

54.2.1 QUANTIZATION

The common use of *Floats* for vertex coordinates has three major drawbacks. First, the range of values that can be represented may significantly exceed the actual range covered by the vertex coordinates, and thus bit-combinations are "wasted on" impossible coordinates. Second, the distance between two consecutive coordinate values, i.e., the ***round-off error***, is not uniformly distributed, but depends on the distance to the origin, thus providing more accuracy for some portion of the model than for others. Third, the ***resolution*** of the representation may significantly exceed what is required by the application.

Quantization addresses these three drawbacks. It truncates the vertex coordinates to a fixed accuracy, thus, by itself, reducing storage size while preserving an acceptable geometric accuracy. It starts with a ***normalization*** process, which computes a tight, axis-aligned bounding box. Then the longest dimension of the box is divided into 2^B segments or equal size, s. The other dimensions are also divided into cells of size s, possibly enlarging the box to have uniform cells. Thus, the normalization process divides the (enlarged) bounding box into cubic cells of size s. The vertices that fall inside a given cells are snapped to the cell center. Thus, the corresponding error is bounded by half of the diagonal of the cell. The number of bits required to encode each coordinate is less than B. Choosing $B = 12$ ensures a sufficient geometric fidelity for most applications and most models. Thus, this lossy quantization step, by itself, reduces the storage cost of geometry from $96v$ bits to $36v$ bits.

54.2.2 PREDICTION

The next, and most crucial geometry compression step involves ***vertex prediction***. Both the encoder and the decoder use the same predictor. Thus, only the ***residues*** between the predicted and the correct coordinates need to be transmitted. The ***coherence*** between neighboring vertices in meshes of finely-sampled smooth surfaces limits the magnitude of the residues. For example, if the magnitude of the largest residue is less than 63 (quantized units), then the total cost for geometry drops to $21v$ bits (a sign bit and a 6-bit magnitude per coordinate). We describe below several predictors.

Because most edges are short with respect to the size of the model, adjacent vertices are in general close to each other and the differences between their coordinates are small. Thus a new vertex may be predicted by a previously transmitted ***neighbor*** [Deer95]. Instead of using a single neighbor, when vertices are transmitted in VST top-down order, a linear combination of the four ancestors in the VST may be used [TaRo98]. The four coefficients of this combination are computed to

minimize the magnitude of the residues over the entire mesh and transmitted as part of the compressed stream.

The most popular predictor for ***single-rate compression*** is based on the ***parallelogram*** construction [ToGo89]. Assume that the vertices of c.t have been decoded. We predict c.o.v.g using c.n.v.g + c.p.v.g - c.v.g. The parallelogram prediction may sometimes be improved by predicting the angle between c.t and c.o.t from the angles of previously encountered triangles.

54.2.3 STATISTICAL CODING

Some of the residues may be large. Thus, good prediction, by itself may not lead to compression. For example, if the coordinates have been quantized to B-bit integers, some of the coordinates of the corrective vector, c.o.v.g - c.n.v.g - c.p.v.g + c.v.g may require $B + 2$ bits of storage. Thus, parallelogram prediction may expand storage rather than compress it. However, the distribution of the residues is usually biased toward zero, which makes them suitable for statistical compression [Salo00].

For instance, assume that all residues are integers in $[-63, +63]$. Furthermore suppose that in 99% of the cases, the most significant bit of the magnitude of the residue is 0. The entropy of this bit is $-0.99 \log_2(0.99) - 0.01 log_2(0.01)$, which amounts to 0.05 bit per coordinate. Arithmetic coding compression may be used to reduce the total storage size of these most significant bits close to its theoretical ***entropy***. Furthermore, if in 95% of the cases the second most-significant bit of the residue magnitude is 0, its cost per coordinate may be reduced to 0.15 bits. It the third and fourth bits have respective probabilities of 90% and 80% of being zero, their respective costs are 0.50 and 0.72 bits per coordinates. Even if we assume no statistical compression of the sign and of the two least significant bits, the total cost per coordinate will be 4.42 bits per coordinate, or equivalently $13.3v$ bits.

54.3 CONNECTIVITY COMPRESSION

As discussed above, typically, geometry may be encoded with $7t$ bits, provided that connectivity information is available during geometry decompression to locate previously decoded neighbors of each vertex along the surface. This section presents techniques for compressing the connectivity information from $3t(2v - 4) \log_2 v$ bits to bt bits, where b is guaranteed never to exceed 1.80 and in practice is usually close to 1.0. As a result, many meshes may be encoded with a total of less than $8t$ bits ($7t$ for geometry, $1t$ for connectivity) with a small error bound for quantization.

Two observations could lead to misjudgement of the importance of incidence compression. (1) Some applications [ABK98] produce unorganized clouds of 3D point samples for which the incidence is derived algorithmically, and thus needs not be transmitted. (2) Recently developed graphic techniques for producing images of 3D surfaces directly from clouds of unstructured 3D samples [LeWh85] eliminate altogether the need for computing and transmitting the incidence information. Thus, one may conclude that it is not only unnecessary to transmit the incidence, but also unadvisable, considering that uncompressed, it is more expensive than geometry.

However, capturing and transmitting the incidence information has several important benefits. (1) Its reconstruction is a computationally expensive and delicate

process, thus, it is usually faster to transmit and decompress the incidence than to recompute it. (2) To correctly render a cloud of unstructured samples as a continuous surface, the samples must be uniformly distributed over the surface and the density of their distribution must ensure that the distance between neighboring samples along the surface is smaller than the size of the smallest feature. Incidence information permits significant reduction in the density of samples in low-curvature portions of the surface. Thus, the samples in nearly flat regions need not be transmitted, since their approximation will be reproduced automatically by the graphics rasterization at rendering time. (3) The most effective ***geometry compression*** techniques are based on the prediction of the location of the next vertex from the locations of its previously decompressed neighbors. Transmitting information describing who the previously decoded neighbors of each vertex are and how they are organized around a new vertex is equivalent to transmitting the incidence graph. (4) The incidence may be compressed to about a bit per triangle and thus the overhead of transmitting it is negligible when compared to the savings in geometry storage to which it leads.

GLOSSARY

Border edge: An edge of the recovered portion of the triangle mesh during decompression that has, so far, only one incident triangle. The natural orientation of a border edge is consistent with the orientation of the incident triangle.

Hole: A maximally connected component of the relative interior of the not-yet-recovered portion of the mesh.

Loop: A chain of border edges forming a closed manifold boundary of a hole.

Gate: The border edge where the next new triangle will be attached during decompression.

Active loop: The loop containing the gate.

Tip corner: The corner c of the new triangle, where c.n.v and c.p.v bound the gate.

Tip vertex: The vertex, c.v, associated with the tip corner, c.

Right edge: The edge joining c.v and c.n.v, where c is the tip corner.

Left edge: The edge joining c.p.v and c.v, where c is the tip corner.

Offset: The number of vertices in the active loop from the gate to the tip vertex of an S-triangle.

clers string: Connectivity encoding as a sequence of labels in C,L,E,R,S.

Valence (or degree): The number of triangles incident upon a given vertex.

54.3.1 EDGEBREAKER

Instead of retracing the chronological evolution of research results in the field of single-rate incidence compression, we first describe in detail Edgebreaker [Ross99], one of the simplest and most effective single-rate compression approaches. The source code for Edgebreaker is publicly available[1]

[1] `www.gvu.gatech.edu/ jarek/edgebreaker/eb`.

Then, we briefly review several variants and other approaches, using Edgebreaker's terminology to characterize their differences and respective advantages.

The Edgebreaker compression visits the triangles in a spiraling (depth- first) TST order and generates the ***clers string*** of labels, one label per triangle, which indicate to the decompression how the connectivity of the mesh can be rebuilt by attaching new triangles to previously reconstructed ones. We first describe the Edgebreaker decompression process.

EDGEBREAKER DECOMPRESSION

During decompression, the reconstructed portion of the mesh is a surface with one or more holes, bounded by ***loops*** of oriented ***border edges.*** Each decompression step attaches a new triangle to a border edge, called the ***gate***, in the active loop (Figure 54.3.1).

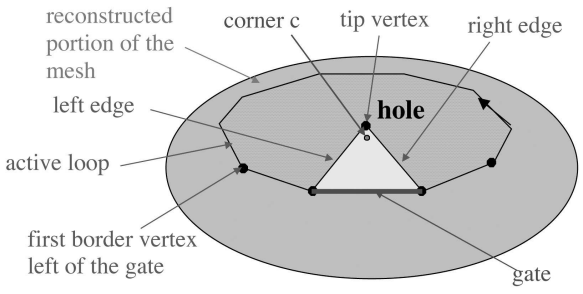
The labels in the *clers* string define where the tip of the new triangle is. Edgebreaker uses only five labels: C, L, E, R, and S. Label C indicates that the new triangle will have as tip a new vertex. We say that this triangle is of type C. Note that the three vertices of the triangle incident upon the gate have been previously decoded and may be used in a parallelogram prediction of the new vertex. The numbering of the vertices and hence their order in the G-table of the reconstructed mesh reflects the order in which the vertices are instantiated as tips of C-triangles.

L indicates that the tip vertex is the first border vertex on the left of the gate in the current loop (Figure 54.3.1). R indicates that the tip is the first border vertex on the right of the gate in the current loop. E indicates that the new triangle will close a hole bounded by the current loop, which therefore must have only three vertices. S indicates that the tip of the new triangle is elsewhere in the current loop. An S-triangle splits the current loop in two loops, as well as splitting the hole bounded by that loop into two holes. The one bounded by the right edge of the new triangle is the ***right hole***. The other one is the ***left hole***(Figure 54.3.2).

After the new triangle is attached, the gate is moved to the right edge of the new triangle for cases C and L. It is moved to the left edge for case R. When an S-triangle is attached, the gate is first moved to the right edge and the right hole is filled through a recursive call to decompression. Then the gate is moved to the left edge and the process resumes as if the S-triangle had been an R-triangle.

The reconstruction of the connectivity graph from the *clers* string is trivial,

FIGURE 54.3.1
The terminology used to describe the Edgebreaker decompression.



except for the S-triangles. Indeed, a C-triangle is attached to the gate and a new vertex is used for its tip. The tips of the L, R, and E triangles are known. The Id of the tip of each S-triangle could be encoded using $\log_2(k)$ bits, where k is the number of previously decoded vertices. A more economical approach is to encode an ***offset*** o indicating the number of vertices that separate the gate from the tip in the current loop (Figure 54.3.2). Because the current loop may include a large fraction of the vertices, one may still need up to $\log_2(k)$ bits to encode the offset. Although one may prove that the total cost of encoding the offsets is linear in the number of triangles [Gumh99], the encoding of the offsets constitutes a significant fraction of the total size of the compressed connectivity. Hence, several authors strived to minimize the number of offsets [AD01b].

The author has observed [Ross99] that the ***offsets need not be transmitted*** at all, because they can be recomputed by the decompression algorithm from the *clers* string. The observation is based on the fact that the attachment of a triangle of each type changes the number of edges in the current loop by specific amounts (Figure 54.3.2). C increments the edge-count. R and L decrement it. E removes a loop of three edges and thus decreases the edge- count by 3. S splits the current loop in two parts, but if we count the edges in both parts, it increments the total edge count. Each S label starts a recursive call that fills in the hole bounded by the right loop and terminates with the corresponding E label. Thus ***S and E*** labels work as pairs of ***parentheses***. Combining all these observation, we can compute the offset by identifying the substring of the *clers* string between an S and the corresponding E, and by summing the edge-count changes for each label in that substring. To avoid the multiple traversals of the *clers* string, all offsets may be precomputed by reading the *clers* string once and using a stack for each S.

The elegant ***Spirale Reversi*** approach [IsSn99] for decompressing *clers* strings

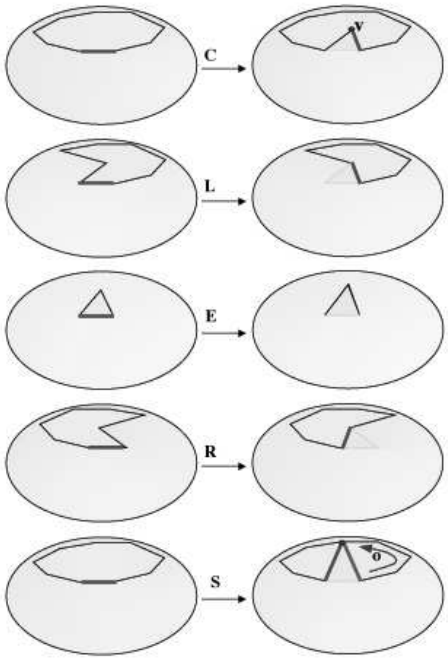


FIGURE 54.3.2
The five Edgebreaker mesh reconstruction operations attach a new triangle to the gate (indicated by a red line on the left column) in the active border loop around a hole in the partly reconstructed mesh. The gate for the next operation is indicated by a red line in the right column. The C operation (top) creates a new vertex (v). The tip of the S-triangle (bottom) may be identified by an offset o. The S operation first puts the gate on the right edge of the new triangle and proceeds to fill the right hole using a recursive call. Then it sets the gate to the left edge of the new triangles and resumes the process. Note that C and S increment the edge count, L and R decrement it, and E reduces it by 3.

that have been created by the Edgebreaker compression avoids this preprocessing by reading the *clers* string backwards and building the triangle mesh in reverse order.

A third approach, ***Wrap&Zip*** [RoSz99], also avoids the preprocessing and directly builds a Corner Table as it reads the *clers* string. It does not require maintaining a linked list of border vertices. For each symbol, as a new triangle is attached to the gate, Wrap&Zip fills-in the known entries to the V and O-tables. Specifically, it fills in c.o for the tip corner c of the new triangle and for its opposite, c.o. It assigns vertex Ids. for the tips of C-triangles as they are created, by simply incrementing a vertex Id counter. It defers assigning the Ids. of other vertices until a Zip process matches them with vertices that already have an Id. Thus, it produces a ***web***, as defined earlier. The border edges of the web must be matched into pairs. The correct matching could be specified by encoding the structure of the cut [Tura84] [TaRo98]. However, as discovered in [RoSz99], the information may be trivially extracted from the *clers* string by orienting the border edges of the web as shown in Figure 54.3.3. Note that these border-edge orientations are consistent with an upward orientation of the cut-edges toward the root of the VST.

The zipping part of Wrap&Zip matches pairs of adjacent border edges that are oriented away from their shared vertex. Only L and E triangles open new zipping opportunities. Zipping the borders of an E triangle may start a chain of zipping operations (Figure 54.3.4). The cost of the zipping process is linear, since there are as many zipping operations as edges in the VST and the number of failed zipping tests equals the number of E or L-triangles.

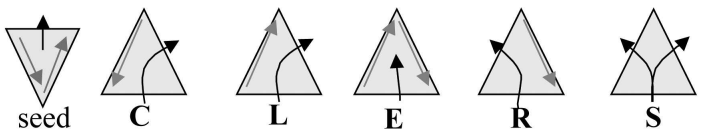
GUARANTEED ENCODING OF THE CLERS STRING

Except for the first two triangles, there is a one-to-one mapping between each C triangle and each vertex. Consequently, the number of C-triangles is $v-2$, and the number of non-C-triangles is $t-(v-2)=v-2$. Thus exactly half of the triangles are of type C, and Edgebreaker guarantees a compression of not more than 2.0 bits per triangle [Ross99] using a trivial code, where the first bit is used to distinguish C-triangles from non-C-triangles.

Given that the subsequences CE and CL are impossible, a slightly more complex code [KiRo99] may be used to guarantee that the compressed file will not exceed $1.83t$ bits.

Further constraints exist on the *clers* string. For example, CCRE is impossible, because CCR increments the length of the loop, which must have been at least 3. By exploiting such constraints to better estimate the probability of the next symbol, a more elaborate code was developed [Gumh00], which guarantees $1.778t$

FIGURE 54.3.3
The borders of the web are oriented clockwise, except for the seed and the C triangles.



bits when using a forward decoding [RoSz99], and $1.776t$ bits when using a reverse decoding scheme [IsSn99]. Hence, the Edgebreaker encoding of the connectivity of any mesh (homeomorphic to a sphere) may be compressed down to $1.78t$ bits. This brings it within 10% of the proven $1.62t$ theoretical lower bound for encoding planar triangular graphs, as established by [Tutt62], who by counting all possible planar triangulations of v vertices proved that an optimal encoding uses at least $v \log_2(256/7) \approx 3.245v$ bits, for a sufficiently large v.

STATISTICAL ENCODING

Recent developments in the guaranteed compression ratios discussed above not only have a theoretical importance, but ensure excellent and fast compression and decompression for meshes with irregular connectivity and for large collections of small meshes.

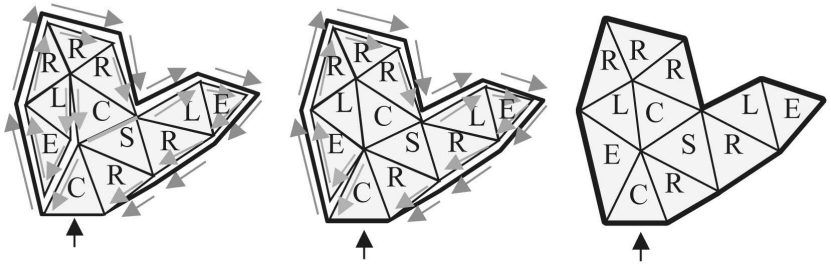
In practice however, better compression ratios may often be obtained. For example, one may encode CC, CS, and CR pairs as single symbols. Each odd C symbol will be paired with the next symbol. After an even number of C symbols, special codes lead to a guaranteed $2.0t$-bit enconding [RoSz99].

Furthermore, by arranging symbols into words that each start with a sequence of consecutive Cs and by using a ***Huffman*** code, we often reduce storage to less than $1.0t$ bits. For example, $0.85t$ bits suffice for the Huffman codes of the Stanford Bunny. Including the cost of transmitting the associated 173 word dictionary brings the total cost to $0.91t$ bits. A `gzip` compression on the resulting bit stream improves it only by 2%.

CONNECTIVITY PREDICTION

In addition to vertex location prediction, the connectivity of the next vertex may also be predicted using the same information. The ***Delphi*** connectivity prediction

FIGURE 54.3.4
We assume that the part of the mesh not shown here has already been decoded into a web with properly oriented borders (exterior arrows). Building the TST (shown by the labeled triangles) for the substring CRSRLECRRRLE produces a web whose free borders are oriented clockwise for all non-C-triangles and counterclockwise for C triangles (left). Each time Wrap&Zip finds a pair of edges oriented away from their common vertex, it matches them. The result of the first zip operation (center) enables another zip. Repeating the process zips all the borders and restores the desired connectivity (right).



scheme [CoRo02] works as follows. The triangle connectivity, and its *clers* string produced by the Edgebreaker compression, is estimated by ***snapping*** the tip-vertex to the nearest bounding vertex of the active loop, if one lies sufficiently close. If no bounding vertex lies nearby, the next *clers* symbol is estimated to be a C. If the guess is correct, a single confirmation bit is sufficient. Otherwise, an entropy-based code is received and used to select the correct CLERS symbol from the other four possible ones (or the correct tip of an S-triangle). Experiments indicate that, depending on the model, up to 97% of Dephi's guesses are correct, compressing the connectivity down to $0.19t$ bits. When the probability of a wrong guess exceeds 40% the Delphi encoding ceases to be advantageous.

54.3.2 OTHER CONNECTIVITY COMPRESSION APPROACHES

CUT-BORDER MACHINE

Although invented independently, the ***cut-border machine*** [GuSt98] has strong similarities with Edgebreaker. Because it requires encoding the offset of S-triangles and because it was designed to support manifold meshes with boundaries, cut-border is slightly less effective than Edgebreaker. Reported connectivity compression results range from $1.7t$ to $2.5t$ bits. A context-based arithmetic coder further improves them to $0.95t$ bits [Gumh99]. Gumhold proposes [Gumh00] a custom variable length scheme that guarantees a total of less than $0.94t$ bits for encoding the offsets, thus proving that the cut-border machine has linear complexity.

TOPOLOGICAL SURGERY AND MPEG-4

Turan [Tura84] noted that the connectivity of a planar triangle graph can be recovered from the structure of its VST and TST, which he proposed to encode using a total of roughly $12v$ bits. There is a simple way to reduce this total cost to $6v$ bits by combining two observations: (1) The binary TST may be encoded with $2t$ bits, using two bits to indicate the presence or absence of left and right children. (2) The corresponding (dual) VST may be encoded with $1t$ bits, one bit per vertex indicating whether the node is a leaf and one bit indicating whether it is the last child of its parent. (Recall that $2v = t + 2$.) This scheme does not impose any restriction on the TST. Note that for less than the $2t$ bits budget needed for encoding the TST alone, Edgebreaker encodes the *clers* string, which not only describes how to reconstruct the TST, but also how to orient the borders of the resulting web, so as to define the VST, and hence the complete incidence. This economy comes from the fact that it uses a ***spiraling*** TST.

Taubin and Rossignac have noticed that a spiraling VST, formed by linking concentric loops into a tree, has relatively few branches. Furthermore, the corresponding dual TST, which happens to be identical to the TST produced by Edgebreaker, has also few branches (Figure 54.3.5). They have exploited this regularity by ***Run Length Encoding*** the ***TST*** and the ***VST***. The resulting ***Topological Surgery*** compression technique [TaRo98] encodes the length of each run, the structure of the trees of runs, and a marching pattern, which encodes each triangle run as a generalized ***triangle strip*** [ESV96], using one bit per triangle to indicate whether the next triangle of the run is attached to the right or left.

An IBM implementation of the Topological Surgery compression has been developed for the VRML standard [THLR98] for the transmission of 3D models across the Internet, thus providing a compressed binary alternative to the original VRML ASCII format [VRML97], resulting in a 50-to-1 compression ratio. Subsequently, the Topological Surgery approach has been selected as the core of Three Dimensional Mesh Coding (3DMC) algorithm in ***MPEG-4***, the ISO/IEC multimedia standard developed by the Moving Picture Experts Group for digital television, interactive graphics, and interactive multimedia applications.

Instead of linking the concentric rings of triangles into a single TST, the ***layered structure*** shown in Figure 54.3.5 (left) may be preserved [BPZ99]. The incidence is represented by the total number of vertex layers, and by the triangulation of each layer. When the triangle layer is a simpler closed strip, its triangulation may be encoded as a triangle strip, using one marching bit per triangle. However, in practice, a significant number of overhead bits are needed to encode the connectivity of more complex layers.

HARDWARE DECOMPRESSION IN JAVA 3D

Focusing on hardware decompression, Deering [Deer95] encodes generalized triangle strips using a buffer of 16 vertices. One bit identifies whether the next triangle is attached to the left or right border edge of the previous triangle. Another bit indicates whether the tip of the new triangle is a new vertex, whose location must be encoded in the stream, or a previously processed vertex that is still in the buffer and can hence be identified with 4 bits. Additional bits are used to manage the buffer and to indicate when a new triangle strips must be started. This compressed format is supported by the Java 3D's Compressed Object node.

Chow [Chow97] has provided an algorithm for compressing a mesh into Deering's format by extending the border of the previously visited part of the mesh by a fan of not-yet-visited triangles around a border vertex. When the tip of the new triangle is a previously decoded vertex no longer in the cache, its coordinates, or an absolute or relative reference to them, must be included in the vertex stream, significantly increasing the overall transmission cost. Therefore, the optimal encoding traverses a TST differently from the spiraling TST of Edgebreaker, so as to reduce cache misses.

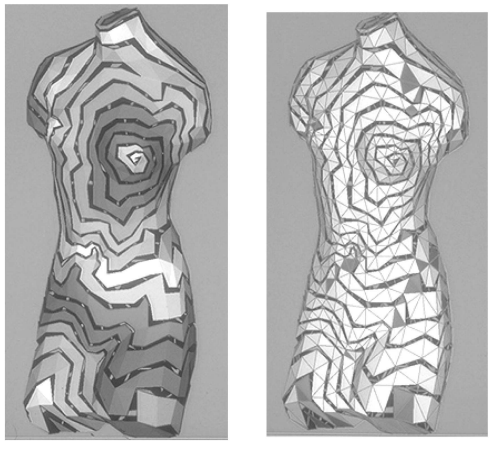


FIGURE 54.3.5
The Topological Surgery approach merges concentric circles of triangles into a single TST (left). That TST and its dual VST have relatively few runs (right).

VALENCE-BASED INCIDENCE COMPRESSION

A consequence of Euler's theoremis that the average vertex valence is 6. In fact, in most models, the valence distribution is highly concentrated around 6. For example, in a subdivision mesh, all vertices that do not correspond to vertices of the original mesh have valence 6. To exploit these statistics, Touma and Gotsman [ToGo89] have developed a valance-based encoding, which visits the triangles in the same order as Edgebreaker. As in They encode the distinction between the C- and the S-triangles as in Edgebreaker, but instead of encoding the symbols for L, R, and E, they encode the valence of each vertex and the offset for each S-triangle. When the number of incident triangles around a vertex is one less than its valence, the missing L, R, or E triangle may be completed automatically. For this scheme to work, the offset must not only encode the number of vertices separating the gate from the tip of the new triangle along the border, but also the number of incident triangles on the tip that are part of the right hole.

Only one bit is needed to distinguish a C from an S. Given that 50% of the triangles are of type C and usually about 5% of the triangles are of type S, the amortized entropy cost of that bit is around $0.22t$ bits. Therefore, about 80% of the encoding cost lies in the valence, which has a low entropy for regular and finely tessellated meshes and in the encoding of the offsets. To fix ideas, consider the example where 80% of the vertices have valence 6. A bit used to distinguish them from the other vertices may be encoded using $0.36t$ bits. The amortized cost of encoding the valence of the other 20% vertices with 2 bits each is $0.40t$ bits. Thus, the valence of reasonably regular meshes may be encoded with $0.76t$ bits. If 5% of the triangles are of type S and each offset is encoded with an average of 5 bits, the amortized cost of the offsets reaches $0.25t$ bits. Thus, offsets add about 25% to the cost of encoding the C/S bits and the valence.

Alliez and Desbrun [AD01a] managed to significantly reduce the number of S-triangles, and thus the cost of encoding the offsets, by using a heuristic that selects a gate with a low probability of producing an S-triangle. They also show that if one could eliminate the S-triangles completely, the valence-based approach would guarantee to compress the mesh with less than $1.62t$ bits, which happens to be Tutte's lower bound [Tutt62].

An improved Edgebreaker compression approach was proposed [SKR00b] for sufficiently large and regular meshes. It is based on a specially designed context-based coding of the *clers* string and uses the Spirale Reversi decompression. For a sufficiently large ratio of degree-6 vertices and sufficiently large t, this approach guarantees a worst-case storage of $0.81t$ bits.

54.4 SURFACE SIMPLIFICATION

Most of the details are usually far from the viewer. Their perspective projections on the screen appear small and thus need not be displayed at full resolution. Therefore, to avoid transmission delays, only crude approximations (called ***Levels-of-Detail*** (***LODs***)) will be transmitted initially. They are produced by single-resolution ***simplification*** processes discussed in this section. The use of LODs as a technique for accelerating graphics is supported in graphics libraries, such as OpenGL. We do not discuss here the image-based techniques that replace such distant models with

imposters [DSSD99] made by pasting, as textures, low-resolution images of them onto simple polygons. Instead, we focus on techniques that reduce the triangle count while striving to minimize the error introduced by the simplification. The connectivity and geometry of these simplified models may be compressed using the single-rate compression techniques discussed above. Refinements that upgrade their fidelity may be transmitted later, if at all needed.

We can simplify the mesh by moving one or more vertices to ***collapse*** one or more triangles, which may then be identified and discarded. Removing a collapsed triangle from the new mesh will not affect the surface, but may create a hole in the connectivity graph. For example, removing a single triangle that has been collapsed by displacing one of its vertices to the mid-point of the opposite edge will create a topological hole sometimes called a "T-junction." Therefore, we will only remove triangles that have 2 or 3 coincident vertices. The two main simplification approaches, ***vertex clustering*** and ***edge collapse***, are discussed below.

GLOSSARY

Uniform LOD: A simplified model constructed by striving to maintain a uniform distribution of error between the original and the simplified model.

View-dependent LOD: A simplified model created by adjusting the tolerance to a particular set of view parameters, e.g., increasing the accuracy of the approximation close to the viewer and to the visible silhouettes of the model.

Multi-Resolution Model (MRM): A representation from which view-dependent LODs may be efficiently extracted as the viewpoint is displaced.

Vertex-Clustering Simplification: A mesh simplification process that collapses clusters of vertices with identical quantized coordinates to one representative vertex and removes collapsed triangles when redundant.

Edge-collapse: A simplification step that collapses pairs of edge-connected vertices along nearly straight edges or nearly flat regions. Each edge-collapse also collapses two triangles, which may be removed.

Silhouette: The union of the edges of a mesh that are bounded by a front and a back-facing triangles.

Hausdorff distance: The Hausdorff deviation $H(A, B)$, between two sets, A and B, is the largest distance between a point in one of the sets and the other set.

Quadric error: The sum of the squares of the distances between a new position of a vertex v and planes that contain the triangles incident upon v in the original mesh.

54.4.1 VERTEX CLUSTERING

Vertex clustering [RoBo93], among the simplest simplification techniques, is based on a crude vertex ***quantization***, obtained by imposing a uniform, axis-aligned grid (Figure 54.4.1) and clustering all vertices that fall in the same grid cell. The simplest implementation of vertex clustering associates each vertex, v, with a cell number, v.cell computed by concatenating its three quantized coordiates. The quantized version, x' of the x coordinate is the integer part of

$s_x(x - x_{min})/(x_{max} - x_{min})$, where s_x is the number of cells in the x direction. Similar expressions are used for quantizing the y and z coordinates. If one wishes to guarantee a uniform error, $\{s_x, s_y, s_z\}$ are chosen so that the $\{x, y, z\}$ dimensions of each cell are nearly identical. The computational cost of his quantization pass is linear and does not require accessing any connectivity or incidence information. One may think of v.cell as the ***cluster name*** for the vertices in it.

A second pass selects a ***representative vertex*** for each cluster. It is often preferable to use one of the vertices of the cluster, rather than to create a new location for the representative vertex. Picking the vertex closest to the average of the cluster vertices has a tendency of shrinking the objects. Rossignac and Borrel [RoBo93] found that the vertex furthest from the center of the model's bounding box is a better choice (Figure 54.4.1). Even better choices may be obtained by using more expensive schemes, which weigh vertices based on the local curvature or on the probability that they would be on a ***silhouette*** of the object, when seen from a random direction, and then select the closest vertex to the weighted average in each cluster.

Rendering all of the triangles of the original mesh with each vertex replaced by its cluster representative will produce an image of the simplified model; see Figure 54.4.1. To reduce the redundancy of this representation, one may chose to identify and eliminate all degenerate triangles, which may be done in a single pass over all triangles. Note, however, that groups of triangles that model thin branches or small shells may collapse to dangling edges or to isolated points. It would be simple to remove them by not drawing zero-area triangles. However, this option would result in a loss of information about the presence of the object along these thin branches or in the small isolated components. Therefore, these dangling edges and vertices are usually identified and preserved. Consequently, a third step of vertex clustering removes all degenerate triangles that have 2 or 3 vertices in the same cluster, but also creates a list of dangling edges and isolated vertices.

To identify dangling edges and vertices, one can construct a list of six triplets, one per triangle, using the thtree cluster names of its vertices in all possible permutations. This list may be sorted efficiently by using hashing on the first cluster name in each triplet. All entries a,b,c that start with a,b are consecutive. If the third element, c, in all of them is either a or b, then (a,b) is a dangling edge. Similarly, if all entries that start with a are of the form a,a,a, then a is an isolated vertex.

This vertex clustering approach has been used in several commercial systems as a simplification tool for accelerating the rendering of 3D models (such as IBM's 3DIX and OpenGL), not only because of its simplicity and speed, but also because it does not use adjacency information beyond the triangle/vertex incidence table, and because it may be applied to arbitrary collections of triangles, edges, and vertices, with no topological restrictions. For example, it may, in a single process, simplify 3D models of solids and 2D models of their drawings, including vector-based text and mark-up.

A second vertex clustering phase with a coarser grid may be applied to the LOD produced by a first pass. Repeating this process produces a series of LODs that all use the same initial vertex set. When, for each LOD, the cell size is reduced by two in each dimension, the vertex clusters correspond to a hierarchy of clusters, which may be stored in an octree [Same90]. Luebke [LuEr97] has used vertex clustering with such an octree to support view-dependent simplification.

Clearly, no vertex has moved by more than the length of the diagonal of a cell, and thus this length offers a bound on the maximum geometric error between

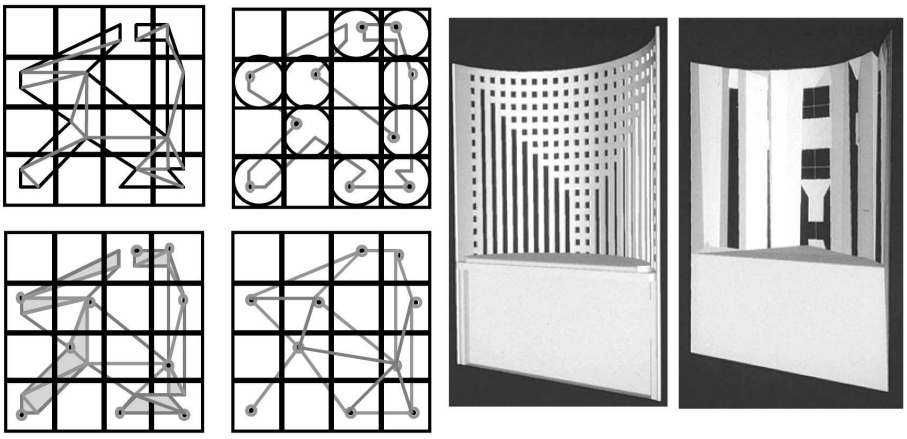
the original shape and the simplified one. This is a tight bound on the Hausdorff distance between the two shapes, which may be of considerable importance to manufacturing, robotics, and animation applications where computing clearance and detecting interferences and collisions is important.

Unfortunately, vertex clustering rarely offers the most accurate simplification for a given triangle-count reduction. One of its problems lies in the fact that the result is affected by the grid alignment, which may split nearby vertices into separate clusters, and replace them with distant representatives. This problem may be reduced by allowing the cells to float a bit [LoTa97], considerably improves the quality of some models, although at the expense of a slightly higher computational cost.

A more delicate problem may be illustrated by considering the 3D triangulated model of a scanned shape whose vertices have been sampled on a regular grid. If we use a large cell size, important features of the mesh will be simplified. If we use a small cell size, the triangulations of flat or nearly flat portions of the surface will retain an excessive triangle count, because their vertices are not allowed to slide along the surface past their cell boundaries. To overcome this constraint, we would like to have cluster cells that form ***slabs*** around the surface, with a small thickness in the normal direction that captures the tolerance, but with a wide tangential spread over flat areas. Clearly, if a manifold vertex has all its neighbors in such a slab, then moving it to one of its neighbors will not result in an error that exceeds the tolerance. By progressively adjusting the slab, several techniques identify simply connected groups of coplanar or nearly coplanar triangles and then retriangulate their surface to eliminate interior vertices [KaTa96]. Instead of local slabs, Cohen et al. [**?**] compute offset surfaces, called ***envelopes***, that bound a tolerance zone around the surface, which is used to constrain vertex merging operations.

FIGURE 54.4.1

Left: vertex clustering simplification on a 2D mesh. Top left: grouping vertices; Top right: cluster representative vertex. Bottom left: Degenerate triangles removed; Bottom right: All vertices replaced by cluster representative, dangling edges and isolated vertices added. Right: Simplified coarse approximation.



54.4.2 EDGE COLLAPSING

Collapsed triangles may be created by edge-collapse operations (Figure 54.4.2), which each merge two vertices and collapse two triangles [**?**]. These collapsed triangles may be easily removed from the connectivity graph. For example, to update a Corner Table, we identify the collapsed edge by a corner c (Figure 54.4.2) and use it to access the corners of the neighboring triangles. We mark the corners of the collapsed triangles as "unused." Then the V- and O-tables are updated in the natural way. See Figure 54.4.3 for an example.

Note that these updates assume that we have a manifold representation. Hence, most simplification techniques preclude topology-changing edge-collapses, which for example would create dangling edges or nonmanifold vertices. Many of the algorithms also assume that each vertex of the mesh has at least three incident triangles, and that no two edges have identical endpoints. Edge-collapses that violate these restrictions may be easily detected and prevented. Thus, the triangle-count reduction capacity of the restricted simplification may be reduced for objects with many holes or thin branches, such as the chair of Figure 54.4.1.

A simplified mesh may be produced by a sequence of edge-collapse opera-

FIGURE 54.4.2
Collapsing the fat edge (left) to one of its vertices collapses its two incident triangles (center). The corner table of the collapsed mesh may be updated using the corner c (right). The reverse of an edge-collapse is a vertex-split.

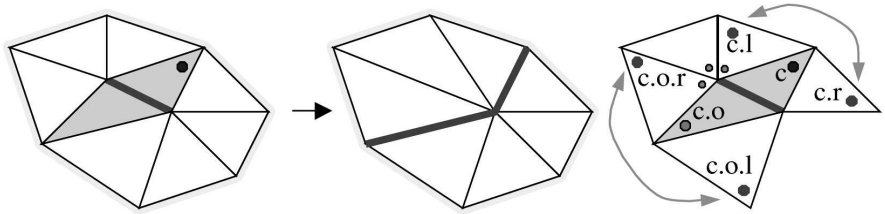
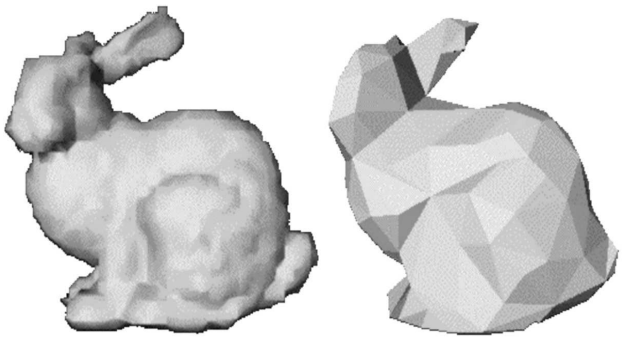


FIGURE 54.4.3
The 3D model (left) has been simplified by a sequence of edge-collapses to a model with a much lower triangle count (right).



tions [Hopp96]. Most simplification techniques strive to select at each stage the edge whose collapse will have the smallest impact on the total error between the resulting simplified mesh and the original surface. Deciding how the error should be measured is delicate and computing it precisely is time consuming. Therefore, most simplification approaches are based on a compromise where the error is estimated, as discussed below. These estimates are used to select the best candidate for the next edge-collapse. Error estimates for portions of the mesh that have been affected by prior collapses must be updated. A priority queue is used to maintain a sorted list of candidates.

54.4.3 MIXED APROACHES

It is also possible to combine vertex-clustering and edge-collapse approaches. For example, vertex-clustering may be used as a geometric and topological filter to remove small features, such as holes or bumps. Then, the result may be simplified through an edge-collapse process, which is better suited for removing redundant samples along flat portions of the surface and along nearly linear edges. The error resulting from this combination is bounded by the sum of the cell diagonal used in vertex clustering plus the estimate of the worst case error produced by edge collapsing.

In order to prevent topology changes and still exploit the speed and simplicity benefits of vertex clustering, one may split the cluster of vertices within a given cell into edge-connected sub-clusters. Collapsing each sub-cluster into a separate representative vertex will not merge disconnected components, but can still create nonmanifold singularities and dangling faces and edges. A local topology analysis may be conducted to detect and prevent these collapses. Such a combination of vertex clustering and edge-collapse was exploited for performing out-of-core simplification [Lind00] of datasets that are too large to fit in memory and thus would make the random access to the vertex data and connectivity tables, which are performed by edge-collapsing operations, too expensive.

Finally, several authors [PoHo97] [**?**] [**?**] have extended edge-collapsing simplifications by adding ***virtual edges***, which are computed using vertex clustering and make it possible to merge components and to deal with nonmanifold meshes.

54.4.4 ERROR MEASURES

The error between two shapes could be measured in ***image space***, as the discrepancy between the colors of the pixels for a particular set of views [Lu$^+$02] [Lind00]. However, such error measures rely on assumptions as to how a particular color change or displacement of the projection of a highlight or color discontinuity on the screen impact the perceived fidelity of the image. They also require a fine sampling of the higher dimensional space of view parameters, and are thus expensive to compute.

The geometric error between the projection of the two shapes on the screen can be formulated more objectively, but is also view-dependent. For example, it may be formulated as the discrepancy between the projections of the silhouettes and sharp edges of both shapes.

Thus, most simplification techniques are based on a view-independent error

formulation. The error, $E(A, B)$, between a shape A and a shape B may be formulated as the maximum of the distance, $d(p, S)$, from all points p on A or B, to S, the other shape, (B or A respectively). This formulation is equivalent to the ***Hausdorff*** distance $H(A, B)$, which may also be formulated as the smallest radius r, such that $A \subset B \uparrow r$ and $B \subset A \uparrow r$, where $S \uparrow r$ denotes the expanded set S obtained by replacing each point q of S by a ball of center q and radius r. The distance $d(p, S)$ may be computed as the minimum of distances between p and the vertices of S, the normal projections of p onto the edges of S, and the normal projections of p onto the interiors of the triangles of S.

The difficulty in computing $H(A, B)$ lies primarily in the fact that it is not sufficient to test $d(p, S)$ for all vertices p of A and B, because the maximum error may occur at points inside faces, not at vertices or edges. Consequently, the exact Hausdorff measure is often approximated by ***super-sampling*** the two surfaces and computing the Hausdorff distance between the two dense sets of samples. The popular ***Metro*** tool [CRS98] super-samples one surface only and computes the maximum of the distance between the samples and the other surface.

Another drawback of the Hausdorff distance, is that it does not require a good mapping from one shape to the other and fails to identify for example the fact that nearby and parallel portions of the two surfaces may have opposite orientation.

Thus, most simplification algorithms use a local error estimation. Consider a vertex that has moved from its initial position v to a new position v', as a result of a vertex clustering step or of a series of edge-collapses. The distance $||vv'||$, which is bounded by the cell diagonal in the vertex clustering approach, provides a conservative bound on the Hausdorff error resulting from this displacement. However, it is too conservative when the mesh was nearly planar in the vicinity of v and when the vector vv' is tangent to the surface. Clearly, we want to measure the component of the displacement of that vertex along the normal to the surface.

The error resulting from the collapse of a vertex v_1 to its neighbor v_2, can be estimated by the dot-product $v_1v_2 \cdot N_1$, where N_1 is the surface normal computed at v_1. Although simple, this formulation does not guarantee a conservative error bound.

Ronfard and Rossignac [RoRo96] used the maximum of the distance between the new position of v' and the planes that contain the original triangles incident upon v. The distance between v' and the plane containing vertices (v, a, b) is $vv' \cdot (va \times vb)/||va \times vb||$. The right term may be pre-computed and cached for each triangle in the original mesh using its vertices v, a, and b. Note that for very sharp edges and vertices, an additional plane is necessary to measure excursions of v' that would be tangential to all the faces incident upon v and yet would move away from the surface. That normal to that additional plane may be estimated using

the surface normal estimation at v. The cost of this approach lies in the fact that, as vertices are merged through series of edge collapses, one needs to keep track of the list of all the planes that contain the triangles incident to these vertices in the original model. Furthermore, for each new edge-collapse candidate, the distance between the new position v' must be computed to all these planes. If the edge collapse is executed, the lists of planes must be merged.

By trading the conservative maximum error bound of [RoRo96] for a quadratic error measure, Garland and Heckbert [**?**] have considerably reduced the computational cost of error estimation. The square of the distance $D_1(P)$ between a point P and a plane, Plane(Q_1,N_1) through a point Q_1 with unit normal N_1 is $(N_1 \cdot Q_1P)^2$. It is a quadratic polynomial in the coordinates (x, y, z) of P.

Based on this observation, in a preprocessing stage, we pre-compute the 10 coefficients of $D_1(P)$ for each corner c_k, considering Plane$(c_k.v.g, N_k)$, where the normal N_k is computed as $(c_k.p.v.g - c_k.v.g) \times (c_k.p.v.g - c_k.v.g)$. Note that N_k is not a unit vector. Its length is proportional to the area of c.t. Then for each vertex v_m, we compute the coefficients by adding the respective coefficients of its corners. They define the function D_m associated with that vertex.

During simplification, we estimate the cost of an edge collapse that would move a vertex v_1 to a vertex v_2, by $D_1(v_2)$. We perform the collapse with the lowest estimate and add to each coefficients of D_2 the corresponding coefficient of D_1.

54.4.5 OPTIMAL QUANTIZATION FOR EACH LOD

Assume that simplification produces a mesh A that approximates the original mesh O within a conservative sampling error estimate e_S. The vertex quantization performed during the compression of A will produce a mesh C with a quantization error e_Q with respect to A. Thus, the conservative bound on the total error is $e_S + e_Q$. How should we choose the optimal combination of a triangle count t for A and of the number B of bits used by the quantization? Assume for instance that we have a fixed bit budget. If we use a LOD with too many triangles, we will have very few bits per coordinate, and thus will need to quantize them drastically. Given the magnitude of the resulting quantization error, we may decide that the model is over-sampled, and that we are better off by increasing B and using a lower LOD. Setting $e_S = e_Q$ yields a solution that is in general significantly suboptimal. An approximate solution to this optimization problem has been derived [KRS99] by formulating e_S as $K(t)/t$, and by approximating the shape complexity function $K(t)$ by a constant K, which may be estimated from a curvature integral over the model. In fact, for a sphere, $K(t)$ is constant.

54.5 RE-SAMPLING

The simplification approaches described above reduce the sampling of the mesh, while decreasing its accuracy. They strive to minimize the estimated error for a given vertex count reduction. In contrast, the re-sampling techniques described in this section strive to reduce storage cost while maintaining an acceptable accuracy. They are based on two strategies: (1) increase the regularity of the connectivity of the mesh, so as to increase connectivity compression; (2) constrain each new vertex to lie on a specific curve, so as to reduce to one the number of coordinates that must be specified per vertex.

GLOSSARY

Regular subdivision: A sequence of mesh refinement steps, each of which inserts new vertices in the middle of the edges and splits triangles into four. The positions of the new and old vertices are adjusted by a vector computed from the neighboring vertices.

Wavelet compression: A hierarchical encoding of regularly spaced data as a

series of residues to values predicted by interpolation from a coarser model.

Normal meshes: The interleaving of mesh refinement steps with displacement steps, which adjust the inserted vertices along estimated surface normal. Statistical properties of the adjustments make them suitable for wavelet compression and progressive transmission.

54.5.1 REPULSION-BASED RETILING

An early simplification technique based on re-sampling [Turk92] first places ***samples*** on the surface of the mesh and distributes them evenly through an iterative process using ***repulsive forces*** computed from estimates of the geodesic distances [PoSc99] between samples. The repulsive forces may be adjusted taking local curvature into account so as to increase sample density in high curvature areas. Then it refines the mesh by inserting these samples as new vertices and hence splitting the triangles that contain them. Finally, the old vertices are removed through edge-collapse operations that preserve topology. This elegant process acts as a low pass filter and produces pleasing simplifications for smooth surfaces. Unfortunately, it may produce unnecessarily large errors near sharp features and does not reduce the cost of encoding the vertices.

54.5.2 NORMAL MESHES

Another approach [KSS00] uses the MAPS algorithm [Le+98] to compute a crude LOD, which can be compressed using any of the single-rate compression schemes discussed above. Once received and restored by the decompression module, the crude LOD is used as the coarse mesh of a subdivision process. Each subdivision stage splits each edge of the mesh into two and each triangle into four, by the insertion of one new vertex per edge. They use the ***Loop*** subdivision (Chapter 54), which splits edge (c.n.v,c.p.v) by inserting point (c.v.g+c.o.v.g+3c.n.v.g+3c.o.v.g)/8 and then moves the old vertices by a fraction toward the average of their old neighbors.

After each subdivision stage, they download a displacement field of corrective vectors and use them to adjust the vertices, to bring the current level subdivision surface closer to the desired surface. The distribution of the coefficients of the corrective vectors is concentrated around zero and their magnitude diminishes as subdivision progresses. They are encoded using a wavelet transform and compressed using a modified version of the SPIHT algorithm [SaPe96] originally developed for image compression.

Instead of encoding corrective 3D vectors, the ***Normal Mesh*** approach [GVSS00] restrict each offset vector to be parallel to the surface normal estimated at the vertex. Only one corrective displacement value needs to be encoded per vertex, instead of three coordinates. They use the ***Butterfly subdivision*** [DLG90], which preserves the old vertices, and for each pair of opposite corners c and c.o splits edge (c.n.v,c.p.v) by inserting a point computed from 8 vertices.

They encode the corrective displacement values of each new vertex using the unlifted version of Butterfly wavelet transform [DLG90] [ZSS96]. Further subdivision stages will generate a smoother mesh that interpolates these displaced vertices. The difficulty of this approach lies in the computation of a suitable crude LOD and in handling situations where no suitable displacement may be found for a new ver-

tex along the estimated surface normal, because for example the normal does not intersect the original mesh. Furthermore, that original mesh and the connectivity constraint imposed by the regular subdivision process limit the way in which the retiling can adapt to the local shape characteristics, and thus may result in less effective compression ratios. For example, regular meshes may lead to sub-optimal triangulations for surfaces with high curvature regions and saddle points, where vertices of valence different than 6 are more appropriate.

54.5.3 PIECEWISE REGULAR MESHES

The surface of the mesh may be algorithmically decomposed into ***reliefs*** [SRK02]. Each relief may be re-sampled by a regular grid of parallel rays. Triangles are formed between samples on adjacent rays and also, to ensure the proper connectivity, at the junction of adjacent reliefs.

The sampling rate (i.e., the density of the rays) may be chosen so that the resulting Piecewise Regular Mesh (PRM) has roughly the same number of vertices as the original mesh. In these situations, the PRM typically approximates the original mesh with the mean square error of less than 0.02% of the diameter of the bounding box. Because of the regularity of the sampling for each relief, the PRM may be compressed for both connectivity and geometry down to about $2t$ bits.

The PRM compression algorithm uses a modified version of the Edgebreaker compression and of the Spirale Reversi decompression to encode the global relief connectivity and the triangles which do not belong to the regular regions. First, Edgebreaker compression is used to convert the mesh to be encoded into a CLERS string. Then, the CLERS string is turned into a binary string using the context-based range, which reduces the uncertainty about the next symbol for a highly regular mesh.

The geometry of the reliefs is compressed using an iterated two-dimensional variant of the differential coding [Salo00]. The regular resampling causes the entropy of the parallelogram rule residues to decrease by about 40% when compared to the entropy for the original models, because on reliefs, two out of three coordinates of the residual vectors become zero.

Since this approach does not require global parametrization, it may be used for models with complex topologies. It is faster than the combination of the MAPS algorithm [Le$^+$98] and the wavelet mesh compression algorithm of [GVSS00] [KSS00], while offering comparable compression rates.

54.5.4 SWINGWRAPPER

The ***SwingWrapper*** approach [Atte01], shown in Figure 54.5.1, partitions the surface of an original mesh M into simply connected regions, called ***triangloids***. From these, it generate a new mesh M'. Each triangle of M' is a linear approximation of a triangloid of M. By construction, the connectivity of M' is fairly regular, and can be compressed to less than a bit per triangle using EdgeBreaker. The locations of the vertices of M' are compactly encoded with a prediction technique that uses a single correction parameter per vertex. Instead of using displacements along the surface normal or heights on a grid of parallel rays, SwingWrapper requires that all C-triangles be isosceles, with left and right edges having a prescribed length

L. SwingWrapper strives to reach a user-defined output file size rather than to guarantee a given error bound. For a variety of popular models, a rate of $0.4t$ bits yields a mean square error of about 0.01% of the bounding box diagonal.

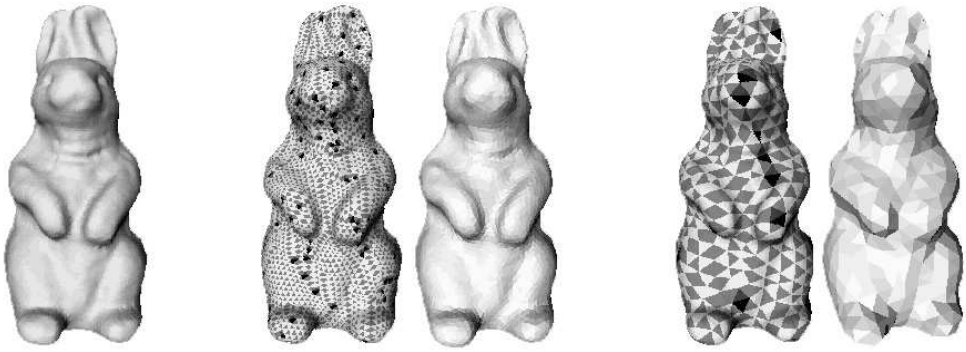
54.6 PROGRESSIVE TRANSMISSION

When the full resolution model of a mesh is unnecessary or when immediate graphic feedback is needed, a compressed crude model obtained through simplification or resampling is downloaded first. Then, later, if a higher resolution is required, the full-resolution mesh could be downloaded using a compact encoding produced by a single-rate compression. When the storage size of the compressed crude model is small compared to the storage size of the full mesh, the overhead of transmitting both, instead of the full mesh only, is small. Yet, the benefits are considerable if the full model never becomes necessary or if the user is not willing to wait for the full resolution model. However, in this binary mode scenario, when the accuracy of the crude model is no longer sufficient, the delay associated with downloading the full resolution mesh could be avoided if an intermediate resolution model could be downloaded instead and would provide sufficient accuracy. In fact, it may be desired to offer a series of intermediate LODs. Each one could be compressed independently using a single-rate compression scheme.

The shortcoming of this "independent" approach lies in the fact that the transmission does not take advantage of the information already available at the client side. For example, if the accuracy requirements increase progressively, and the client ends up downloading 10 different LODs, each having twice more vertices than the previous one, the total cost for a mesh will be about $(1+2+4+8+\ldots+1024)t/1024$

FIGURE 54.5.1

The original mesh (left) containing 134,074 triangles requires 4,100,000 bytes, when stored in the WRL format. A dense partitioning of its surface into triangloids (2nd) yields a retiled mesh (3rd) which deviates from the original by less than 0.6% of the radius of the smallest ball bounding the model. Its 13642 triangles are encoded with $3.5t$ *bits. The resulting total of 6042 bytes represents a 678-to-1 compression ratio. A coarser partitioning (4th) decomposes the original surface into 1505 triangloids. The corresponding retiled triangle mesh (last) approximating the original model within 4% is encoded with 980 bytes: A 4000-to-1 compression.*



bytes, which is $2t$ bytes, or twice the cost of transmitting the full-resolution mesh. In fact the total cost will be somewhat higher, because the geometry and connectivity of crude models is less regular and will in general not compress to a byte per triangle.

This shortcoming may be addressed by using a ***progressive transmission*** approach where the connectivity and geometry of previously decoded LODs is exploited to reduce the transmission cost of the next LOD. Most progressive approaches compress the upgrade, which contains the description of where new vertices should be inserted, i.e., their location and the associated connectivity changes.

GLOSSARY

Vertex split: The inverse of an edge-collapse operation. It is specified by selecting a vertex v and two of its incident edges.

Upgrade: The information permitting to refine an LOD to a higher accuracy LOD.

Compressed Progressive Meshes: A representation combining a single-rate compression of the lowest LOD with compressed encodings of the successive upgrades.

54.6.1 PROGRESSIVE MESHES

The progressive transmission of compressed meshes started with Hoppe's ***Progressive Meshes (PM)*** [Hopp96]. It encoded the coarsest LOD and a series of vertex-split operations, which when applied to the coarse mesh reverses the sequence of simplifying edge-collapse operations that were used to create it. The advantage of PM is its ability to stop transmission at any desired accuracy, thus offering the finest granularity of upgrades: each one being a vertex-split. The compression effectiveness of PM is limited by its need to encode the absolute index of the vertex at which the vertex-split occurs. That index requires $\log_2(n)$ bits, where n is the number of vertices decoded so far.

54.6.2 COMPRESSED PROGRESSIVE MESHES

By grouping the vertex splits into batches, the ***Compressed Progressive Mesh (CPM)*** [PaRo00] eliminates the $\log_2(v)$ cost replacing the vertex indexing by a one-bit-per-vertex mask. When combined with a modified Butterfly geometry predictor estimating the location of each new vertex from its neighbors, it achieves about $11t$ bits for a combined transmission code of the complete geometry ($7.5t$ bits) and connectivity ($3.5t$ bits), while offering between 7 and 12 intermediate LODs for common meshes.

The approach is illustrated in Figure 54.6.1.

54.6.3 EDGE-MASKS

The Wavemesh of Valette and Prost uses a batch strategy similar to CPM, but

uses the mask to mark the edges that must be split [**?**]. They use a new simplification algorithm, which removes vertices in an order that permits recreation of the original connectivity with a small number of additional bits (above the cost of the mask). Experimental results suggest that the average cost for encoding the complete connectivity of the progressive mesh ranges from $1.0t$ bits to $2.5t$ bits for commonly used meshes.

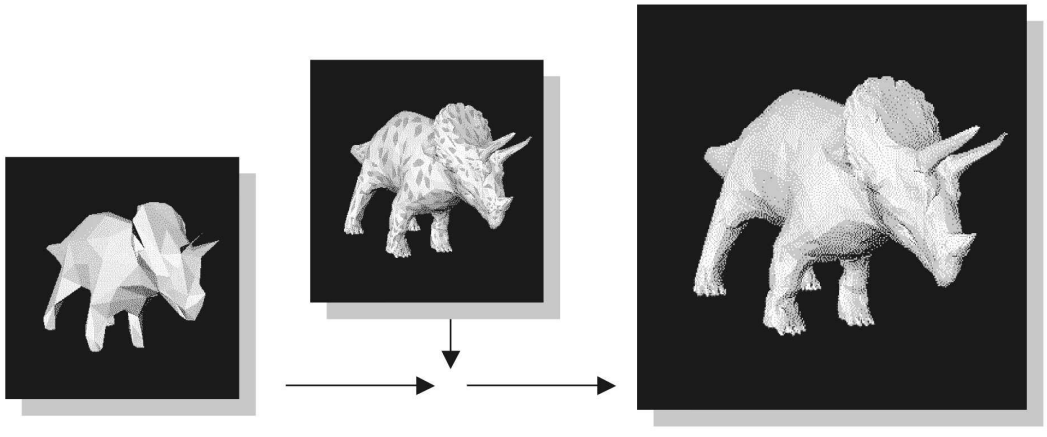
54.6.4 KD-TREES

Gandoin and Devillers [GaDe02] use a hierarchy of vertex clustering [RoBo93]. However, instead of an octree, as in [**?**] they use a k-d tree. At an x-split node they split the parent's area in two equal parts by a plane orthogonal to the x axis and store the number of vertices in the left child. This adaptive organization of the vertices leads to the factorization of the most significant bits of the x-coordinate. The alternating y-split, and z- split nodes perform a similar spit, but with a different orientation of the splitting plane.

Thus, the coordinates of the centers of the leaves of the k-d tree are defined implicitly by their position in the tree. The cost of storing them lies in the cost of encoding the numbers of vertices in the left child of each node. When only one vertex lies in a node, its least significant bits that have not been factored in the structure of the tree must be encoded.

When moving down the tree, each parent vertex is split into two vertices, which may, but need not be connected. Thus, the connectivity may encoded as a series of possibly nonmanifold vertex-splits, as proposed in [**?**]. The total cost of transmitting a progressive mesh compressed with this approach comprises $1.8t$ bits for the connectivity and about $7.8t$ bits for the geometry, when the original vertex coordinates have been quantified to 12 bits each.

FIGURE 54.6.1
The coarse mesh (left) produced by an edge-collapsing simplification is refined by a CPM upgrade which inserts new vertices and new triangles (center). The higher LOD is shown (right).



54.6.5 VALENCE DRIVEN CONQUEST

To exploit the regularity of distribution of vertex degrees (or valences), which are concentrated around 6, Alliez and Desbrun [AD01a] use a series of passes to produce decreasing LODs. Each pass reduces the number of triangles by 3 by combining three steps that each traverses the mesh in a breadth-first order. The first one removes the tip vertices of triangles that have a valence of less than 7, leaving polygonal holes in the mesh. It also tags the remaining vertices. The second phase uses these tags to retriangulate the holes, striving to create vertices with valence 3 or 9. The third one removes valence-3 vertices. Because the decimation follows a systematic traversal, it can be reversed by decompression. The upgrade information for each pass contains the degree of the vertices removed during the first step (which must be 3, 4, 5, or 6) and an encoding of the traversal irregularities. These connectivity upgrades may be compressed to an average of $1.9t$ bits, which corresponds to a 40% improvement over CPM [PaRo00]. However, the selection of which vertices are removed at each phase is only dictated by the connectivity and cannot take into account the geometry, and thus cannot favor vertices whose removal will have the lowest impact on the geometric error. Consequently, this approach will produce intermediate models which for the same triangle count will be less accurate than those produced by CPM.

54.6.6 PROGRESSIVE NORMAL MESHES

By exploiting the hierarchical nature of the wavelet formulation, the normal meshes provide an effective progressive transmission scheme, in which the bits of the coefficients are transmitted in the order of their importance [SaPe96] [Shap93]. For the same transmission cost, this approach yields fourfold better accuracy than CPM [PaRo00]. In fact, the total cost of transmitting the highest accuracy mesh is often lower than one offered by the best single-rate compression schemes [ToGo89]. However, this approach is not capable of restoring the original mesh for two reasons: (1) it has the semi-regular connectivity of a subdivision mesh and (2) the constraints on vertex locations make it impossible to restore the original surface.

54.7 SOURCES AND RELATED MATERIAL

SURVEYS

Simplification techniques have been surveyed in [HeGa97] [PuSc97] and more recently in [Lu+02]. Compression and progressive transmission techniques have been surveyed in [Ross99b] [**?**].

Topological extensions of simplification and compression not covered here are discussed in [GBTS99] [RoCa99] [PoHo97] for nonmanifold models, in [Ross99] [Lo+02] for handles, in [ToGo89] for holes, in [KRS99] [IsSn00] for quadrilateral and polygonal meshes [KRS99] [IsSn00]. Simplification and compression of meshes with properties are discussed in [GaHe98] [BPZ99] [IsSn00]. Compression and progres-

sive transmission of tetrahedral meshes have been addressed in [PRS99] [SzRo00]. Error correction strategies for progressive transmission are addressed in [AAR02].

RELATED CHAPTERS

Chapter 25: Triangulations and mesh generation
Chapter 26: Polygons
Chapter 49: Computer graphics
Chapter 53: Splines and geometric modeling

REFERENCES

[AAR02] G. Al-Regib, Y. Altunbasak and J. Rossignac. An Unequal Error Protection Method for Progressively Compressed 3-D Meshes. International Conf. on Acoustics, Speech and Signal Processing ICASSP'02. Multimedia Communications and Networking II Session. May 2002.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A New Voronoi-based Surface Reconstruction. In SIGGRAPH 98, pages 415– 421, 1998.

[AD01a] P. Alliez and M. Desbrun, Progressive encoding for lossless transmission of 3d meshes. in ACM SIGGRAPH Conference Proc., 2001.

[AD01b] P. Alliez and M. Desbrun, Valence-driven connectivity encoding for 3d meshes. EUROGRAPHICS, volume 20, no. 3, 2001.

[Atte01] M. Attene, B. Falcidieno, M. Spagnuolo, J. Rossignac. SwingWrapper: Retiling Triangle Meshes for Better EdgeBreaker Compression. Genova CNR-IMA Tech. Rep. No. 14/2001. To appear in the ACM Trans. Graph..

[BPZ99] C. L. Bajaj, V. Pascucci, and G. Zhuang, Single resolution compression of arbitrary triangular meshes with properties. Comput. Geom. Theory Appl., volume 14, pages 167–186, 1999.

[Chow97] M. Chow, Optimized geometry compression for real-time rendering. in Proc. Conference on Visualization '97, 1997, pages 347–354.

[Co+96] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F. and Wright, W. Simplification envelopes. In Computer Graphics Proc., Annual Conf. Series (Siggraph '96). ACM Press,

[CoRo02] V. Coors and J. Rossignac. GVU Tech. Rep.. 2002. Guess Connectivity: Delphi Encoding in Edgebreaker.

[CRS98] P. Cignoni, C. Rocchini and R. Scopigno, Metro: measuring error on simplified surfaces. Proc. Eurographics '98, volume 17(2), pp 167-174, June 1998.

[Deer95] M. Deering, Geometry compression. in Proc. 22nd Annual ACM Conference on Computer Graphics, 1995, pages 13–20.

[DLG90] N. Dyn, D. Levin, and J. A. Gregory, A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans. Graph., volume 9, no. 2, pages 160–169, 1990.

[DSSD99] X. Decoret, G. Schaufler, F. Sillion and J. Dorsey, Multi-layered impostors for accelerated rendering. Computer Graphics Forum , volume 18, No. 3, 1999, pages 61–73.

[ESV96] F. Evans, S. S. Skiena, and A. Varshney, Optimizing triangle strips for fast rendering. in IEEE Visualization '96, 1996, pages 319–326.

[GaDe02] P.M. Gandoin and O. Devillers, Progressive Lossless Compression of Arbitrary Simplicial Complexes. ACM SIGGRAPH 2002.

[GaHe98] M. Garland and P. Heckbert. Simplifying Surfaces with Color and Texture using Quadratic Error Metric. Proc. of IEEE Visualization, pages 287–295, 1998.

[GBTS99] A. Gueziec, F. Bossen, G. Taubin, and C. Silva, Efficient Compression of Non-manifold Polygonal Meshes. In IEEE Visualization, pages 73–80, 1999.]

[Gumh00] Gumhold, S., Towards optimal coding and ongoing research, 3D Geometry Compression Course Notes, SIGGRAPH 2000.

[Gumh99] S. Gumhold, Improved cut-border machine for triangle mesh compression. in Erlangen Workshop '99 on Vision, Modeling and Visualization. IEEE Signal Processing Society, Nov. 1999.

[GuSt98] S. Gumhold and W. Straer, Real time compression of triangle mesh connectivity. in Proc. 25th Annual Conference on Computer Graphics, 1998, pages 133–140.

[GVSS00] I. Guskov, K. Vidimce, W. Sweldens, and P. Schroeder, Normal meshes. in Siggraph'2000 Conference Proc., July 2000, pages 95–102.

[HeGa97] Heckbert, P. and Garland, M., Survey of surface sim- plification algorithms. In Tech. Rep.. Carnegie Mellon University-Dept. of Computer Science (to appear), 1997.

[HeGa97] P. Heckbert and M. Garland. Survey of polygonal simplification algorithms. In Multiresolution Surface Modeling Couse. ACM SIGGRAPH Course Notes, 1997.

[Hopp96] H. Hoppe, Progressive meshes. Computer Graphics, volume 30, no. Annual Conference Series, pages 99–108, 1996.

[Hopp98] H. Hoppe, Effcient implementation of progressive meshes. Computers and Graphics, volume 22, no. 1, pages 27–36, 1998.

[Hopp99] H. Hoppe, New quadric metric for simplifying meshes with appearance attributes. IEEE Visualization 1999, pages 59–66, October 1999.

[IsSn00] M. Isenburg and J. Snoeylink, Face fixer: Compressing polygon meshes with properties. in Siggraph 2000, Computer Graphics Proc., 2000, pages 263–270.

[IsSn99] M. Isenburg and J. Snoeyink, Spirale Reversi: Reverse decoding of the Edgebreaker encoding. Tech. Report TR-99-08, Computer Science, UBC, 1999.

[KaTa96] A.D. Kalvin and R.H. Taylor, Superfaces: Polygonal mesh simplification with bounded error. IEEE Comput. Graph. Appl., 16(3), pages 64–67, 1996.

[KiRo99] D. King and J. Rossignac, Guaranteed 3.67V bit encoding of planar triangle graphs. 11th Canad. Conf. Comput. Geom. (CCCG"99), pages 146–149, Vancouver, CA, August 15-18, 1999.

[KRS99] D. King, J. Rossignac, and A. Szymczak, Connectivity compression for irregular quadrilateral meshes. Tech. Rep. TR–99–36, GVU, Georgia Tech, 1999.

[KSS00] A. Khodakovsky, P. Schroeder, and W. Sweldens, Progressive geometry compression. in SIGGRAPH 2000, Computer Graphics Proc., 2000, pages 271–278.

[Le+98] A. W. F. Lee, W. Sweldens, P. Schroder, L. Cowsar, and D. Dobkin, MAPS: Multiresolution adaptive parametrization of surfaces. in SIGGRAPH'98 Conference Proc., 1998, pages 95–104.

[LeWh85] Levoy, M. and Whitted, T. The Use of Points as a Display Primitive. Tech. Rep. TR 85-022, University of North Carolina at Chapel Hill, 1985.

[Lind00] P. Lindstrom, Out-of-core simplification of Large Polygonal Models. Proc. ACM SIGGRAPH, pages 259–262, 2000.

[Lo+02] H. Lopes, J. Rossignac, A. Safanova, A. Szymczak and G. Tavares. A Simple Compression Algorithm for Surfaces with Handles. ACM Sympos. Solid Modeling, Saarbrucken. June 2002.

[LoTa97] K-L. Low and T.S.Tan, Model Simplification using vertex clustering. Proc. Sympos. Interactive 3D Graphics, ACM Press, pages 75–82, 1997.

[Lu+02] D. Luebke, M Reddy, J. Cohen, A. Varshney, B. Watson, R. Hubner, Levels of Detail for 3D Graphics. Morgan Kaufnamm, 2002.

[LuEr97] Luebke, D. and Erikson, C., View-dependent simplification of arbitrary polygonal environments. In ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97), 1997, pages 199–208.

[PaRo00] R. Pajarola and J. Rossignac, Compressed progressive meshes. IEEE Transactions on Visualization and Computer Graphics, volume 6, no. 1, pages 79–93, 2000.

[PoHo97] J. Popovic and H. Hoppe, Progressive simplicial complexes. Computer Graphics, volume 31, pages 217–224, 1997.

[PoSc99] K. Polthier, M. Schmies, Geodesic Flow on Polyhedral Surfaces. Procs. of Eurographics Workshop on Scientific Visualization, Vienna 1999.

[PRS99] Implant Sprays: Compression of Progressive Tetrahedral Mesh Connectivity. R. Pajarola, J. Rossignac, and A. Szymczak, IEEE Visualization 1999, San Francisco, October 24-29, 1999.

[PuSc97] E. Puppo and R. Scopigno, Simplification, LOD and multiresolution: Principles and applications, Tutorial at the Eurographics'97 conference, Budapest, Hungary, September 1997.

[RoBo93] J. Rossignac and P. Borrel, Multi-resolution 3D approximations for rendering complex scenes. Geometric Modeling in Computer Graphics, Springer-Verlag, Berlin, pages 445–465, 1993.

[RoCa99] J. Rossignac and D. Cardoze, Matchmaker: Manifold Breps for non-manifold r-sets. Proc. ACM Sympos. Solid Modeling, pages 31–41, June 1999.

[RoRo96] R. Ronfard and J. Rossignac, Full range approximation of triangulated polyhedra. Proc. Eurographics 96, 15(3), pages 67–76, 1996.

[Ross99] J. Rossignac, Edgebreaker: Connectivity compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics, volume 5, no. 1, pages 47–61, 1999.

[Ross99b] J. Rossignac, Compression and progressive refinement of 3D models. Shape Modeling International, Aizu, Japan, 1-4 March, 1999.

[RoSz99] J. Rossignac and A. Szymczak, Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker. Computational Geometry, Theory and Applications, 14(1/3), 119-135, November 1999.

[RSS02] J. Rossignac, A. Safonova, A. Szymczak Edgebreaker on a Corner Table: A simple technique for representing and compressing triangulated surfaces. in Hierarchical and Geometrical Methods in Scientific Visualization, Farin, G., Hagen, H. and Hamann, B., editors Springer-Verlag, Heidelberg, Germany, to appear in 2002.

[Salo00] D. Salomon, *Data Compression: The Complete Reference.* Second Edition, Springer-Verlag Berlin Heidelberg. 2000.

[Same90] H. Samet, *Applications of Spatial Data Structures.* Reading, Addison-Wesley, 1990.

[SaPe96] A. Said and W. A. Pearlman, A new, fast, and effcient image codec based on set partitioning in hierarchical trees. IEEE Trans. Circuits Syst. Video Technol., volume 6, no. 3, pages 243–250, June 1996.

[Schi98] M. Schindler, A fast renormalization for arithmetic coding. in Proc. of IEEE Data Compression Conference, 1998, p. 572.

[Shap93] J.Shapiro, Embedded image coding using zero-trees of wavelet coeffcients, IEEE Trans.Signal Process.41, 3445-3462, 1993.

[SKR00b] A. Szymczak, D. King, J. Rossignac, An Edgebreaker-based Efficient Compression Scheme for Connectivity of Regular Meshes. Journal of Comput. Geom. Theory Appl., 2000.]

[SRK02] A. Szymczak, J. Rossignac, and D. King. Piecewise Regular Meshes: Construction and Compression. To appear in Graphics Models, Special Issue on Processing of Large Polygonal Meshes, 2002.

[SzRo00] Grow&Fold: Compressing the connectivity of tetrahedral meshes. A. Szymczak and J. Rossignac. Comput. Aided Design. 32(8/9), 527-538, July/August, 2000.

DELETE THIS?

[TaRo98] G. Taubin and J. Rossignac, Geometric compression through topological surgery. ACM Trans. Graph., volume 17, no. 2, pages 84–115, 1998.

[THLR98] G. Taubin, W. Horn, F. Lazarus, and J. Rossignac, Geometry coding and VRML. Proceeding of the IEEE, volume 96, no. 6, pages 1228–1243, June 1998.

[ToGo89] C. Touma and C. Gotsman, Triangle mesh compression. in Graphics Interface, 1998, pages 26–34.

[Tura84] G. Turan, On the succinct representations of graphs. Discrete Applied Mathematics, volume 8, pages 289–294, 1984.

[Turk92] G. Turk, Re-tiling polygonal surfaces. Proc. ACM Siggraph 92, pages 55–64, July 1992.

[Tutt62] W. Tutte, A census of planar triangulations. Canadian Journal of Mathematics, pages21-38, 1962.

[VRML97] ISO/IEC 14772-1, *The Virtual Reality Modeling Language (VRML)*. 1997.

[ZSS96] D. Zorin, P. Schroeder, and W. Sweldens, Interpolating subdivision for meshes with arbitrary topology. Computer Graphics, volume 30, pages 189–192, 1996.