Jarek Rossignac

Shape complexity

J. Rossignac
College of Computing, GVU Center, and
IRIS cluster, Georgia Institute of
Technology, Atlanta, Georgia, USA
http://www.gvu.gatech.edu/~jarek/

Abstract The complexity of 3D shapes that are represented in digital form and processed in CAD/CAM/CAE, entertainment, biomedical, and other applications has increased considerably. Much research was focused on coping with or on reducing shape complexity. However, what exactly is shape complexity? We discuss several complexity measures and the corresponding complexity reduction techniques. Algebraic complexity measures the degree of polynomials needed to represent the shape exactly in its implicit or parametric form. Topological complexity measures the number of handles and components or the existence of non-manifold singularities, non-regularized components, holes or self-intersections. Morphological complexity measures smoothness and feature size. Combinatorial complexity measures the vertex count in polygonal meshes. Representational complexity measures the footprint and ease-of-use of a data structure, or the storage size of a compressed model. The latter three vary as a function of accuracy.

Keywords Compression · Simplification · Morphology · Geometry

1 Introduction

Digital models of 3D shapes are common in numerous applications. They are used to represents solids, which correspond to physical objects, surfaces, which may—but need not—bound a solid, or more general geometric complexes, which may capture the interplay between solids and lower dimensional elements. So far, the accuracy of models that could be represented in 3D modeling systems has been limited by design or acquisition costs, by storage and transmission costs, by computational and visualization costs, and also by theoretical and implementation challenges. These costs are directly related to the complexity of a shape or of the representation used to store it in digital form. A large body of research has been focused on reducing these costs. Computational g,Geometry techniques [15] play an important role in reducing the asymptotic time or space cost of algorithms that construct or process polygonal representations of 3D shapes. They will not be discussed explicitly here. Instead, we investigate the following complementary aspects. In Sect. 2, we briefly discuss the complexity stemming from the use of higher order polynomials to represent the surfaces that bound a shape. In Sect. 3, we review the broad topological domain of shapes and structures that can be represented as a selective geometric complex and discuss how to map this domain to solids or even manifolds, and how to reduce their genus. Complexity may be measured in terms of sharp or narrow features, hence, in Sect. 4, we discuss the smoothness and regularity of shapes, and techniques for increasing them. In Sect. 5, we discuss lossy simplification and retiling techniques for shapes represented as triangle meshes. In Sect. 6, we discuss simple and compact data structures for CSG and BRep and provide examples of redundancy reduction and compression schemes. Due to space limitations, we do not discuss design complexity, or visual complexity.

2 Algebraic complexity

Digital representations of solid objects [29, 32] are often defined by arranging simpler shapes and by combining them through Boolean operations. In particular, a solid may be specified in constructive solid geometry (CSG) as a Boolean combination [28] of primitive solids whose dimensions, position, and orientation may be conveniently controlled by the user [11]. A primitive solid may be easily defined in its nominal position and orientation as the set of points, whose coordinates satisfy one or several polynomial inequalities. For example, a ball of radius r is the set of points (x, y, z) satisfying $x^2 + y^2 + z^2 \leq r^2$. The complexity of such a semi-algebraic representation of a solid may be measured by the number of primitives (discussed in Sect. 6) or by the degree of the polynomials used to represent them. Early CSG modelers were limited to linear polynomials [6], approximating curved primitives by facetted models that could be represented as intersections of linear inequalities or as extrusions of polygonal contours. In the 70s and 80s, commercial products provided support for a subset of quadric primitives [7]. More recently, techniques for rendering Boolean combinations of more general surfaces have been demonstrated [5] and more complex analytic formulations were proposed to extend the Boolean operations [33].

Semi-algebraic formulations are implicit and make it difficult to answer simple geometric or topological queries. For example, a solid represented by a CSG model with an arbitrarily large number of primitives could actually be empty. Furthermore, popular rendering and analysis systems are geared towards boundary representations (BReps), where each face of the solid is represented as a node in a global incidence graph. A face is a two-dimensional portion of a surface and hence is also called a trimmed surface [57]. Surfaces of CSG primitives are defined in terms of polynomials. For example, the surface may be defined by an implicit polynomial equation, such as the equation $x^2 + y^2 + z^2 = r^2$ of a sphere. The face may be specified as the intersection of the surface with a trimming volume. Such a trimming volume may be represented in semi-algebraic (CSG) form, which may be readily extracted from the original CSG formulation of the solid [47]. However, such an implicit representation of the trimming volume still suffers from the same limitations as the CSG representation: It does not tell us explicitly whether the face exists, whether it is connected, or whether it has holes.

To capture this topological information, an explicit boundary representation of a trimmed surface is usually computed through boundary evaluation [36]. The explicit representation of a face consists of the implicit (polynomial) equation of the supporting surface and of a representation of the bounding edges that delimit the face. These edges are usually portions of curves, where two implicit surfaces intersect. A parametric representation $C(s)$ of each intersection curve is preferred. It may be derived in closed form for various combinations of simple quadric surfaces [30]. The vertices, where bounding edges meet may be computed by substituting the coordinate functions of $C(s)$ in the implicit equation of a third surface and solving for s. This approach has been principally limited to quadric surfaces for which intersection vertices may be computed by finding the roots of low degree polynomials. More complex (i.e. higher order) primitives yield systems of simultaneous polynomial equations of higher order, raising the complexity of a robust root-finder and the computational cost of the boundary evaluation [26]. Hence, the term CSG usually refers to systems that support Boolean combinations of primitive solids bounded by linear or specific quadric polynomials.

Although any solid may be approximated by a CSG model, a close approximation of a sculptured (free-form) shape [24] may require a large number of quadric primitives. The alternative is to use a parametric formulation, where each face is still represented as a trimmed surface, but the supporting surface is a mapping, M, of a unit square $(u, v) \in [0, 1]^2$ in parameter space onto a set of 3D points $M(u, v)$ and where the bounding edges are represented as parametric trimming curves $(u(s), v(s))$ in the two-dimensional parameter space. A parametric curve is the set of points $(u(s), v(s))$, where u and v are piecewise polynomial or rational polynomial functions in s [12]. Typically, these are polynomials of degree 3 or less, although aesthetic or functional constraints may require higher order polynomials. In this parametric trimmed surface representation, each trimming edge is represented twice (once for each incident face). When the edge is computed as the intersection between two faces during a Boolean operation, the two representations may not match, creating a gap or self-intersection in the boundary. Such inconsistencies may undermine the integrity of subsequent algorithmic computations. This inconsistency problem, as well as the problem of detecting and tracing intersections of parametric surfaces [20], is exacerbated if higher order parametric surfaces or trimming curves are used. The immense software complexity of several commercial CAD systems [19] is in a large part due to the difficulty of performing and processing reliably the intersections of parametric surfaces. Hence some 3D modelers require that the user designs the boundary of a solid as a patchwork of abutting untrimmed parametric patches, thus significantly increasing the complexity of the design process.

To avoid dealing with higher-order parametric formulations, one may choose to use only first degree polynomials, for which the CSG-to-boundary conversion may be performed reliably [4]. With such a choice, all surfaces are planes and all edges are line segments. Such polygonal models are popular in graphics and animation, because the software for computing and representing them
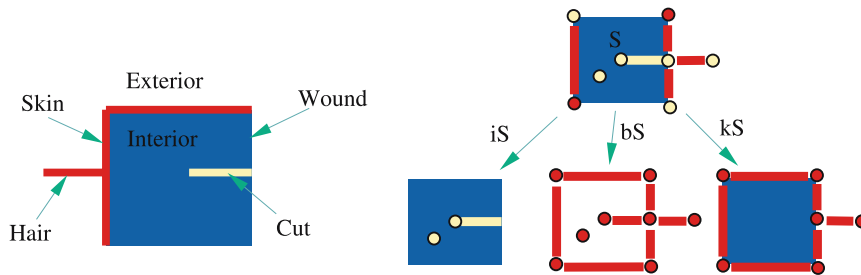
Fig. 1. The interior, exterior, and the four boundary types are shown (*left*). The effects of the interior, boundary, and closure operators are shown on a non-regularized set (*right*)

is less complex and because hardware accelerators have been optimized for rendering them. However, a close approximation of a curved shape may require a large number of planar facets, increasing its combinatorial complexity, storage size, transmission time, processing costs, and rendering delays.

Finally, note that BRep modelers are not restricted to solids and can also be used to model surfaces that do not bound a solid and more general geometric complexes, as discussed in the next section.

In summary, the designer of a 3D modeler that supports Boolean operations must chose between a CSG (implicit) or a BRep (parametric) scheme and decide on the degree of polynomial surfaces supported. The choice will impact the complexity of the software and also the growth of the combinatorial complexity as a function of the accuracy with which a particular shape is approximated.

3 Topological complexity

Although the representation of solids [35] has always been a major focus of CAD and animation systems, it is often desired or even necessary to represent 3D geometries that are not solids, or even boundaries of solids. Such geometries may arise in reverse engineering, where surface samples are erroneously interpolated by a surface that does not constitute the boundary of a solid; in analysis, where it may be important to represent the contacts between solids; or during the design of mixed-dimensional structures combining inhomogeneous solids with lower-dimensional elements.

The concept of a selective geometric complex [40] was developed to provide a common theoretical framework for representing solids, lower-dimensional elements (points, curves, surfaces) and their interactions. To discuss its topological complexity and the corresponding simplification operations, we consider that a pointset S decomposes space into 6 subsets [44]: iS, eS, sS, wS, cS, and hS. To define them, let $\sim S$ denote the complement of S. Let $B_r(p)$ be an open ball of radius r around a point p. Let $b(s)$, denoted bS for simplicity, be the boundary of S. It is defined as the set of points p for which $B_r(p)$ intersects both S and $\sim S$ for all strictly positive r. The

interior iS is $S - bS$. The exterior eS is $(\sim S) - bS$. The skin sS is the portion $(biS) \cap S$ of bS that is included in S and is separating iS from eS. The wound wS is the portion $(beS) \cap (\sim S)$ of bS that is not included in S and separates iS from eS. The hair hS is the set $(beS) \cap S$ of dangling faces, edges, and isolated vertices of bS included in S that are not bounding iS. The cut cS is the union of the cracks $(biS) \cap (\sim S)$ that are not part of S, but are not bounding eS. A set S is open if $S = iS$, i.e. it has no skin or hair. A set is closed if it has no wound or cut. The closure, kS, of a set S is $(iS) \cup (bS)$. A set S is regularized, when it has no wound, no hair, and no cut. To convert an arbitrary set S into a regularized ("clean") set, we need to "cut its hair, mend its cut, and grow back the skin over its wound". The regularized version rS of S may be formally defined by $(iS) \cup (cS) \cup (sS) \cup (wS)$ or more compactly as kiS. Hence, S is regularized if $S = kiS$. The effect of these various topological operators is shown in Fig. 1.

The selective geometric complex (SGC) representation [40, 48] was designed as an extension of the boundary representation (BRep) to the broader domain of non-regularized pointsets. In an SGC, each cell (volume, face, edge) knows the extent (the whole space, an implicit algebraic surface, a parametric or implicit curve) in which it is embedded, the list of its bounding elements, and its membership (a bit indicating whether it is part of S or of $\sim S$). Hence, marking a face as part of $\sim S$ may create a wound (if it is adjacent to both iS and eS) or a cut (if it is surrounded by iS). Furthermore, with the reference to each $(k-1)$-dimensional bounding element e of a k-dimensional set, f is associated a two-bit neighborhood indicating whether e is bounding f on its left, on its right, or on both sides. The terms left and right are defined with respect to an agreed-upon orientation of e and f. This neighborhood information permits us, for example, to specify a particular half-sphere as a subset of a spherical extent bounded by a single circle. Given the orientation of the sphere and of the circle, the neighborhood indicates which half of the sphere is of interest.

The broad topological domain of SGCs may be appropriate for annotating models with internal structures, for representing shapes with cracks, for modeling boundary conditions, or for representing a regularized model during intermediate stages of its construction.

Unfortunately, the support of non-regularized shapes adds complexity to a modeling software. For instance, to decide whether a point p not on bS lies inside a solid S, one would usually cast an arbitrary ray from p, making sure that the ray does not hit any edge or vertex of bS and does not have a tangential intersection with bS. Then, one would compute the parity of the number of times the ray crosses bS [64]. An even number of intersections indicates that p is out of S, because a point infinitely far along the ray is out of S (assuming that S has finite size). If S is not regularized, the test must be modified to discard intersections with hair and cut faces. Hence, a BRep of S must explicitly distinguish between the skin, wound, hair, and cut portions of bS. Although such a distinction is provided by the SGC representation scheme [40], early solid modelers were limited to solids (regularized sets). Because Boolean intersection and difference operations between two regularized sets with partly overlapping boundaries may produce non-regularized sets [41], these modelers automatically performed a regularization of the resulting sets (eliminating cut, hair, and wounds), ensuring that all nodes in the CSG tree represent solids.

This regularization, i.e. the extraction of the regularized version kiS of a non-regularized model S, may be viewed as a reduction in topological complexity. In an SGC representation, this regularization operation amounts to setting the membership bit to IN for all cut and wound cells and to OUT for all hair cells. The removal of the redundancy in the resulting SGC representation is discussed in Sect. 6.

Assume that we are given a set B, that approximates the valid boundary bS of a solid S, but contains inconsistencies (gaps between adjacent faces, self intersections), which may have resulted from an inaccurate reverse engineering process or from inconsistencies between the two representations of each trimming edge in a parametric model, as discussed in the previous section. The extraction of bS from B is sometimes called healing or repairing. It is a difficult process, since B does not unambiguously define S. When B is a set of faces defined as trimmed portion of oriented implicit surfaces, their extents partition the space into an arrangement of three-dimensional cells. S may be estimated [31] by a voting scheme, in which the larger cells of the arrangement that are bounded by large portions of faces of B are given a higher probability of being in S (if they lie on the interior side of the oriented face) or in $\sim S$ (otherwise).

Some scanned models are incomplete. The missing parts of their boundary are called holes. The process of filling the holes in such models [23] is different from the healing discussed above, because the holes are typically larger than the gaps in an inconsistent BRep. These incomplete models may be produced by the early stages of a reverse engineering process, which starts with a set P of discrete points sampled on bS. The process of recovering bS from P is sometimes called point cloud interpolation. Several approaches have recently been proposed for this difficult task [60]. Healing, hole filling, and point cloud interpolation may be viewed as a reduction of the topological complexity of the representation of a regularized shape S, because they simplify further processing, although one could argue that P is topologically simpler than S or bS. In fact, several researchers are exploring the use of point clouds for modeling and rendering 3D shapes [34].

Through the remainder of this paper, we will assume that S is a solid (finite regularized set). Note that a solid may be manifold or not. A point p of bS is said to be manifold (with respect to bS) [1] if there exists a positive r such that the intersection of bS with the open ball $B_r(p)$ is homeomorphic to an open disk. Otherwise, p is said to be non-manifold. S is manifold if all points of bS are manifold. Consider a non-manifold solid S. Let N denote the set of its non-manifold points. In turn, N may be decomposed [40] into a set of relatively open curves, which we call the non-manifold edges (NME), and a set of isolated points, which we call non-manifold vertices (NMV). Support of non-manifold situations complicates algorithms for processing S or for performing Boolean operations between sets. Yet, non-manifold singularities cannot be easily avoided, since they are produced by Boolean operations on solids in tangential contact. When a NME has k pairs of incident faces, it may be replaced by k coincident manifold edge-instances, each having one pair of incident faces. Several choices of pairing the incident faces are possible. Amongst those, one should match faces that have a compatible orientation so that there exists a way of bending each one of the k manifold edge-instances into a different curved edge so that the resulting BRep is free from self-intersection [39]. At the end of this process all edges are manifold, although clusters of edge-instances may be geometrically coincident. The model may still be non-manifold, because it may have non-manifold vertices. These are vertices with two or more cycles (fans) of incident faces. A NMV with k cycles is replaced by k coincident vertex-instances. The resulting model has a manifold topology that is inconsistent with its imbedding in space. Hence, it can be represented by a data structure designed for manifold solids, although the algorithms that process it may no longer assume that all pairs of edges and of vertices are mutually disjoint. Techniques for converting non-manifold models to such pseudo-manifolds in a manner that minimizes the number of vertex-instances have been proposed [39]. We assume hereafter on that the solids are either manifold or pseudo-manifolds.

A solid may have more than one connected component. Furthermore, each component may be bounded by one or more shells (connected portions of the boundary). Finally, each shell may have one or more handles. The number of handles is called the genus of the shell. The genus affects the programming complexity of various algorithms and the efficiency (representational complexity)

of some encodings [49]. Hence, it may be desired to compute approximations of a solid S that have lower genus, or that are simply connected (bounded by a single, genus-one shell). Genus reduction may be achieved [59] by starting with a simply connected boundary of a solid that contains S and shrinking the boundary while preventing changes of genus and intersections with S. A distance field may be used to control the shrinking speed. By identifying and allowing a few initial changes of topology, a desired number of the larger handles may be created, removing the smaller ones. Similarly, one may wish to remove small shells. These are easy to identify, since one may quickly compute an axis aligned minimum bounding box containing each component. Hence, one may construct decreasingly complex approximations of S, where the number of handles and/or components is decreasing. Such an approach may be viewed as a filter that eliminates topological noise, possibly produced through a point-cloud interpolation of an insufficiently dense set of points produced by a 3D scan of a real shape.

Consider now that B is an isosurface separating all points on a regular 3D grid where some measure (say pressure) is higher than a threshold from those where the measure is lower. The topology of B is not fully defined. Local topology choices and simple optimizations [2] may be used to reduce the number of handles or shells, while reducing the representational complexity (vertex count) of the isosurface.

In summary, although it is possible and sometimes essential to support non-regularized shapes, many applications are not designed to cope with such topological complexity. Hence, such representations must often be converted to solids through regularization. Furthermore, non-manifold edges and vertices may be replaced by multiple instances of their manifold counterparts. Finally, the boundary of a solid may be replaced by an approximating surface that exhibits fewer handles and shells. The designer of a modeling system must decide which level of topological complexity will be supported and may provide simplification techniques that reduce the complexity of a particular model.

4 Morphological complexity

Shape having more numerous or smaller features may be viewed as more complex. To evaluate and reduce this morphological complexity, one needs to define what a feature is, measure its size, and provide algorithms for computing an approximation of a shape, where the identified features are larger or fewer.

We may distinguish [70] two measures of morphological complexity: smoothness and regularity. If a point p of bS is non-manifold or if the maximum curvature of bS at p is not well-defined, we say that the smoothness of p

(with respect to S) is zero, or equivalently that p is zero-smooth. Otherwise, we define the smoothness of p as the inverse, $1/c(p)$, of the maximum curvature [62] of bS at p. The smoothness of S is the minimum of the smoothness of points on bS. For instance, if S has a sharp edge, its smoothness is 0. We say that solid S is r-smooth if its smoothness is equal to or larger than r.

At each point p of bS, one may also compute the maximum radius $r(p)$ of an open ball that can touch p from both sides of bS without intersecting bS. The regularity of S is the minimum of $r(p)$ over all points p of bS. A shape is r-regular if its regularity is equal to or larger than r. Note that if S is r-regular, then it is r-smooth. However, a shape may be r-smooth, but may contain small constrictions or cracks that are not r-regular. The regularity of a point of bS is its local feature size, which is its distance to the medial axis of bS. The medial axis is the set of points having more than one closest point on bS. Hence, the regularity of bS is its least feature size. The definition proposed in [70] extends the notion of regularity to the whole space: the regularity of a point p in iS or eS is the radius of the maximum open ball $B_r(p)$ that contains p and is disjoint from bS.

To reduce the morphological complexity of a shape, one may increase its smoothness or its regularity. When bS is a triangle mesh, its smoothness may be increased through subdivision [10, 25, 68] or through smoothing [63]. Subdivision replaces a triangle mesh with a finer mesh. Iterating the process converges to a smooth surface [71]. Subdivision splits each triangle into smaller triangles by introducing new vertices and possibly adjusting the old ones. The limit surface tends to be smoother everywhere, unless it contains non-manifold singularities or self-intersections produced during subdivision. Smoothing alternates moving each vertex towards the average of its neighbors and moving it away from this average. Moving towards the neighbors smoothens the shape. Moving away prevents shrinkage [63]. Since smoothing does not increase the number of vertices, the resulting mesh has zero smoothness at all non-flat edges and vertices and hence is zero-smooth and zero-regular. However, smoothing reduces discrete measures of curvature estimated at the vertices [62], at least away from thin tubes (see Fig. 2). Shapes may also be smoothened through curvature flow, which may be implemented using level sets [55].

Because the subdivision and smoothing techniques discussed above do not provide a direct control over the desired degree of smoothness or regularity, morphological filters may be preferred when more control on smoothness and regularization is desired. The r-rounding $R_r(S)$ of S is the union of all balls of radius r contained in S. The r-filleting $F_r(S)$ is the pointset that cannot be reached by a ball of radius r disjoint from S. Note that $F_r(R_r(S))$ and $R_r(F_r(S))$ [45] leave the r-regular portions of bS unchanged and tend to produce shapes that are almost everywhere r-smooth and r-regular. More precisely, they
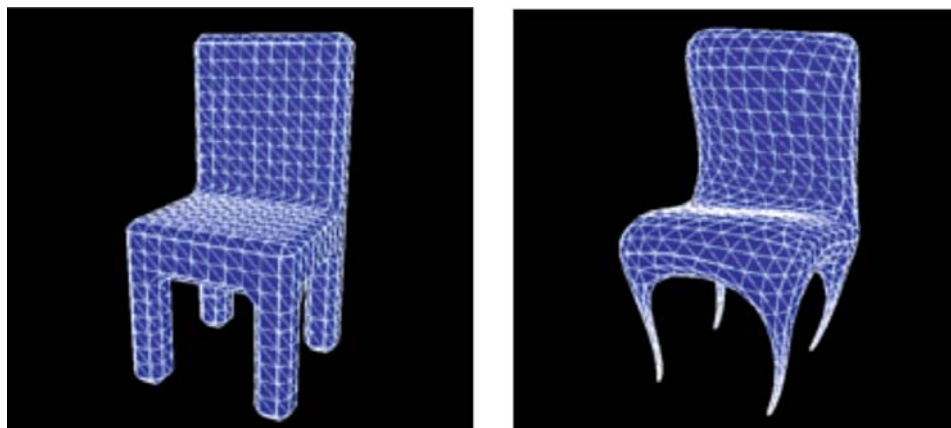
Fig. 2. The original solid bounded by a triangle mesh (*left*) has been smoothened (*right*). Note that the resulting shapes need no longer be regularized, because smoothing has squeezed thin parts into lower-dimensional dangling edges or produce self-intersections
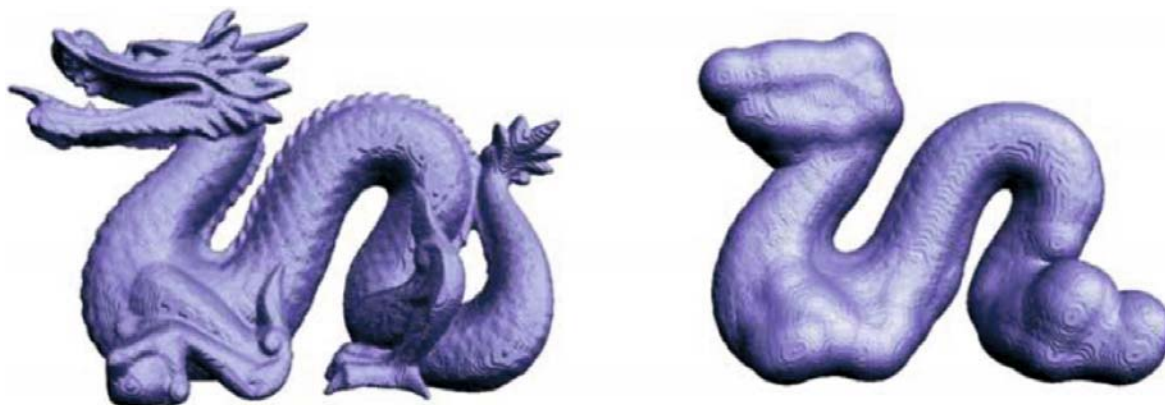

Fig. 3. The morphological complexity of the original solid (*left*) has been reduced (*right*) by using the Mason filter

modify bS only in the r-mortar $M_r(S)$, defined as $F_r(S) - R_r(S)$ [70]. However, $F_r(R_r(S))$ tends to shrink the solid and $R_r(F_r(S))$ tends to grow it. To alleviate this bias, the Mason filter [70] identifies the connected components of $M_r(S)$ and includes in the result only the components that have more than 50% of their volume in S (Fig. 3). Instead of this binary decision, the tightening [69] filter tightens bS inside the r-mortar. The tightening process may change the topology of bS and guarantees that the boundary of the r-tightening $T_r(S)$ of S is r-smooth everywhere. Although there is no guarantee that $T_r(S)$ is r-regular, the irregular singularities are rare and tend to be isolated. Finally, the boundary of $T_r(S)$ is obtained by minimizing surface area.

The geometric error (discrepancy) between S and its morphological simplification S', which may be $F_r(R_r(S))$, $R_r(F_r(S))$, r-Mason(S), or $T_r(S)$, is bounded in the following way. Every point of bS' lies within a tolerance zone $Z_r(bS)$ of all points within distance r from bS. However, the reverse is not true and there usually are points of bS that lie further than r from bS'. For instance, whole shells

of bS may be removed through morphological simplification. Nevertheless, note that these simplifications confine their effect to the r-mortar $M_r(S)$, which is usually a very small subset of $Z_r(bS)$.

In summary, one may measure the morphological complexity of a solid S as the largest value of r for which S is r-regular or simply r-smooth. Complexity measured in terms of smoothness may be reduced indirectly through subdivision or smoothing. When a more precise control is needed, morphological filters may be used. In particular, the r-tightening guarantees to produce a shape that is r-smooth and has minimal area boundary. In general, the resulting shape is also r-regular almost everywhere.

5 Combinatorial complexity

In many applications, piecewise planar boundaries are preferred to higher degree polynomial surfaces. The bound-

ary bS of a solid S is usually represented by a set of surface samples (called vertices) and by a connectivity graph defining a triangle mesh that interpolates the vertices and approximates the desired surface. (Although many modelers support more general polygonal faces, these may always be each represented as a set of contiguous triangles. If desired, one bit per edge may be stored to distinguish between the original edges and those introduced through this artificial triangulation. This conversion to triangle meshes leads to significantly simpler algorithms and to more compact and more regular data structures.)

The complexity of a triangle mesh may be simply formulated as its vertex-count. Because vertex-count impacts storage size, algorithmic complexity, and rendering performance, various simplification techniques have been proposed for reducing the vertex-count.

When an edge of the triangle mesh is collapsed, its two bounding vertices are clustered into one and its two incident triangles may be removed. Hence, each edge-collapse operation [17] reduces the vertex count by 1 and the triangle count by 2. Most triangle mesh simplifications operate by performing a series of edge-collapses. For each edge collapse, two vertex-clusters are merged, the cluster-representative vertex position is computed (maybe as the one that minimizes some error estimate), and the resulting error is recorded. A priority queue may be maintained to accelerate the selection of the edge-collapse that will result in the lowest error. At each stage, the edge with the smallest error is collapsed and the vertex representatives and cost of collapsing neighboring edges updated. The process stops when the desired reduction of vertex-count has been achieved or when the error estimate exceeds a prescribed tolerance. The various approaches differ in their strategy for selecting which edge to collapse next and where to position the new cluster-representative vertex. The error between the original surface S and the surface S' produced by one or more edge-collapses may be measured in different ways. Computing the Hausdorff error $H(S, S')$, which is defined as the minimum r such

that $bS' \subset Z_r(bS)$ and $bS \subset Z_r(bS')$ is difficult and expensive [9, 44]. Hence, simpler bounds or estimates of $H(S, S')$ are usually used. Ronfard and Rossignac [42] have used, as the vertex-representative of a cluster, the point that minimizes the sum of the squares of the distances to the extents (supporting planes) of each triangle incident upon a vertex of the cluster. To estimate the error, they used the maximum of the squares of the distances to the extents. This solution required merging lists of extents as the clusters were merged, but provided a conservative bound on the Hausdorff error. Garland and Heckbert [13] have replaced the maximum by the sum of the squares of the distances to the extents. This is called the quadratic error. As a result, they needed only to keep track of the 10 coefficients of a quadratic polynomial in the coordinates of the vertex representative. When two clusters are merged, the coefficients of the quadratic error of the new cluster may be obtained by simply adding their counterparts in the quadric error formulation of each cluster. This simplification of the simplification process has led to a very efficient and effective combinatorial complexity reduction technique [14], at the cost of replacing the maximum error by the L^2 error measure.

Triangle mesh simplifications based on edge-collapses do not alter the topology of the shape, unless they unintentionally produce self-intersecting boundaries. Although topology preservation may be required in some applications, it considerably limits the level of reduction in the vertex-count that may be achieved for a given tolerance (error). Furthermore, edge-collapsing is usually implemented for manifold boundaries of regularized solids. In contrast, the vertex-clustering method [37] works well on non-regularized shapes, which may combine non-manifold solids with lower dimensional construction lines, 2D drawings, and even annotations. It tends to merge small components, remove small holes, and reduce thin sheets or tubes to their lower-dimensional approximations of the medial axes or skeletons (Fig. 4). Vertex-clustering makes clusters of vertices that fall within the same cell
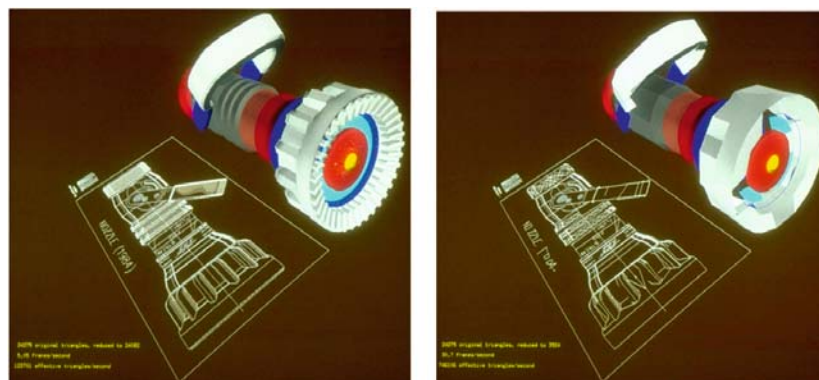

Fig. 4. The combinatorial complexity of an assembly with its drawing (*left*) has been reduced (*right*) through vertex clustering

of a three-dimensional regular grid. Consequently, it is extremely fast and simple to implement. The Hausdorff error $H(S, S')$ and the length of collapsed edges are bounded by the length of the diagonal of a cell. Hence, vertex-clustering is less effective at reducing the vertex count than edge-collapsing in over-tessellated flat regions. Therefore, several authors have proposed to combine both approaches [27].

In summary, the reduction of combinatorial complexity is a lossy process, trading accuracy for simplicity. Most of the research in this area has been focused on reducing the vertex-count of triangle meshes [27]. Several efficient simplification techniques exist and may be combined to simplify smooth manifold surfaces effectively, to allow topological changes that further reduce vertex-count, and to operate on non-regularized models.

6 Representational complexity

The reduction of combinatorial complexity, as discussed above, is accompanied by a loss of accuracy. In contrast, the techniques for reducing the representational complexity discussed in this section are loss-less. We discuss techniques for removing the redundancy in SGC representations, in CSG representations, and in triangle mesh representations. We also discuss compact loss-less encodings (compression) of the connectivity of triangle meshes and how they are affected by a lossy re-sampling preprocessing step.

As mentioned in Sect. 3, the topological complexity of the SGC representation of a non-regularized model may have been reduced by changing the membership bits of specific cells, for instance to eliminate the wound, the cut, or the hair of a model during a closure, interior, or regularization operation. As a result, some of the lower-dimensional cells may be eliminated or merged with incident higher-dimensional cells in order to reduce the representational complexity of the model stored in SGC form. For example, a dangling edge or face of the hair marked OUT may be dropped, a face marked IN and its two incident three-dimensional cells of iS may be joined, or an edge originally part of cS, but marked IN, may be merged with its only incident face of iS. The preconditions and algorithms for these drop, join, and merge operations that ensure the validity of the resulting SGC are discussed in [40].

The representational complexity of a solid stored in CSG form may sometimes be reduced by discovering that the same solid may be represented with fewer CSG primitives. A primitive may be eliminated if it is redundant [65]. Consider a primitive A in the CSG representation of a solid S. We can write S as a set-valued function $S(A)$ of A. The active zone [51] Z of A in the CSG expression of S is defined as $S(W) - S(E)$, where W is the universe and E the empty set. If A is disjoint from Z, then A is E-redundant and can be replaced by the empty set. If A contains Z, then it is W-redundant and may be replaced by W. Replacing A by E or W yields a simplification of the CSG expression, by the successive application of simple re-write rules, such as $W \cup B = W$, $W \cap B = B$, $E \cup B = B$, and $E \cap B = E$, using the fact that $B - C = B \cap (\sim C)$ and that $\sim (B \cup C) = (\sim B) \cap (\sim C)$ and $\sim (B \cap C) = (\sim B) \cup (\sim C)$. Redundancy detection, which requires testing whether $A \cap Z$ or $A - Z$ are empty may be accelerated by using s-bounds [38].

The representational complexity of a solid whose boundary is stored as a triangle mesh may be measured by storage size or by the complexity of the underlying data structure and of the associated algorithms.

Consider a triangle mesh defined by its geometry and by its triangle-vertex incidence graph. The geometry is stored as an array of vertex-entries (three coordinates per vertex). Incidence (sometimes called "topology", referencing the connectivity graph) defines each triangle by listing the three integer indices that identify its vertices.

Consider a triangle mesh with v vertices, e edges, and f triangular faces. To simplify some of the definitions, we assume that the faces and edges are relatively open cells, i.e. they do not contain their boundary in the supporting manifold [40]. We say that the mesh is edge-connected if all pairs of triangles form the two ends of some strip of triangles in the mesh. A strip is an ordered list of triangles such that two consecutive elements of the strip are incident upon a common edge.

A vertex-spanning tree (VST) of a triangle mesh is a subset of the edges of the mesh, selected so that their union with all the vertices forms a tree (connected cycle-free graph). Consider that a given VST has been selected. The edges it contains are called the cut-edges. The union of the cut-edges with all the vertices is called a cut (not to be confused with the topological cut defined in Sect. 4). Because the VST is a tree, there are $v - 1$ cut-edges. The difference between the surface and its cut is called the web. Edges that are not cut-edges are called hinge-edges. The web is the union of the faces and hinge-edges. Removing the cut, which has no loop, from the surface of a mesh will not disconnect it.

Assume first that the mesh forms a surface that is homeomorphic to a sphere. The removal of the cut produces a web that is a (relatively open) triangulated two-dimensional point-set in three-space. This web is simply connected and may be represented by an acyclic graph, whose nodes correspond to faces and whose links correspond to hinge edges. The web forms a planar triangulation and can be drawn on the plane so that no two faces overlap. Thus there are $f - 1$ hinge edges. Note that by picking 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). The TST defines a connected network of corridors (strips) through which one may visit all

the faces by walking across hinge-edges and never crossing a cut-edge. Because an edge is either hinge or cut, the total number of edges, e, is $v - 1 + f - 1$. Each face uses three edges and each edge is used by two faces. Thus, the number e of edges is also equal to $3f/2$. Combining these two equations yields $f = 2v - 4$, which shows that there are roughly twice as many triangles as vertices. When the mesh forms a manifold surface of genus h, this equation becomes $f = 2v + 4h - 4$ and the web is no longer simply connected, but has $2h$ handles. It can be flattened (i.e. turned into a simply connected open polygon) by removing $2h$ additional edges, which we call the jump edges [61]. More generally, the mesh may also have holes and non-manifold singularities, which further complicates the relation between f and v.

Although a large number of representation schemes have been proposed for triangle meshes [46] and more generally for polygonal meshes, we advocate the use of a corner table [52, 53] because of its simplicity. The corner table stores the geometry in the G table and the connectivity in the V and O tables of integers, each having $3f$ entries. Each entry $G[v]$ contains the triplet of the coordinates of vertex number v, and will be denoted $v.g$. Note that the order in which the vertices are listed in G is arbitrary, although once chosen, it defines the integer reference number associated with each vertex. Triangle-vertex incidence defines each triangle by the three integer references to its vertices. These references are stored as consecutive integer entries in the V table. Note that each one of the $3f$ entries in V represents a corner (association of a face with one of its vertices). Let c be such a corner. Let $c.f$ denote its face and $c.v$ its vertex. Remember that $c.v$ and $c.f$ are integers in $[0, v-1]$ and $[0, f-1]$, respectively. Let $c.p$ and $c.n$ refer to the previous and next corner in the cyclic order of vertices around $c.f$.

Although G and V suffice to completely specify the faces and thus the surface they represent, they do not offer direct access to a neighboring face or vertex. We choose to use the reference to the opposite corner, $c.o$, which we cache in the O table to accelerate mesh traversal from one face to its neighbors. For convenience, we also introduce the operators $c.l$ and $c.r$, which return the left and right neighbors of corner c (Fig. 5). Note that we do not need to cache $c.f$, $c.n$, $c.p$, $c.l$, or $c.r$, because they may be quickly evaluated as follows: $c.f$ is the integer division $c.f \text{DIV} 3$; $c.n$ is $c-2$, when $c\text{MOD}3$ is 2, and $c+1$ otherwise; and $c.p$ is $c.n.n$; $c.l$ is $c.n.o$; and $c.r$ is $c.p.o$. Thus, the storage of the connectivity is reduced to the O and V arrays. We assume that all faces have been consistently oriented, so that $c.n.v = c.o.p.v$ for all corners c. For example, one may adhere to the convention that when a face $c.f$ is visible by a viewer outside of the solid (i.e. the finite set that is bounded by the tri mesh), the three vertices, $c.p.v$, $c.v$, and $c.n.v$, appear in counter-clockwise order.

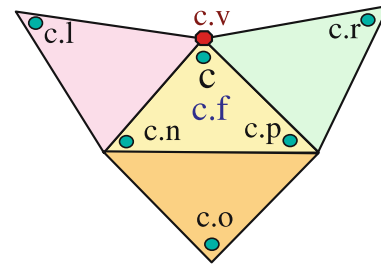The simplicity of the corner table data structure and the notation defined above simplify many of the algorithms

Fig. 5. Corner operators for traversing a corner table representation of a tri mesh

that traverse, process, or modify mesh. For example, the procedure visit(c) {if NOT $c.f.m$ then {$c.f.m := $ true; visit($c.r$); visit($c.l$)}} will visit all the faces in a depth-first order of a TST, assuming that $c.f.m$ is a Boolean set to true when the face $c.f$ has been visited.

Given the V table, the entries in O may be computed by: for $c := 0$ to $3f-2$ do for $b := c+1$ to $3f-1$ do if $(c.n.v == b.p.v)\&\&(c.p.v == b.n.v)$ then {$c.o := b$; $b.o := c$}. A faster approach is described in [43].

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, assuming that Floats are used for the coordinates, a compact, but uncompressed representation of a triangle mesh requires $48f$ bits for the coordinates and $3f \log_2 v$ bits for the V table.

The Edgebreaker compression [49] encodes the full connectivity information contained in both V and O with a linear cost of less than $2f$ bits, and hence eliminates the need for the decompression modules on the client to re-compute O from V. Edgebreaker performs a walk on a manifold surface which follows a breadth-first TST that is similar to the visit(c) procedure described above, but avoids the recursive call in most cases by using a bit-mask to mark all previously visited faces. Edgebreaker encodes one symbol out of the set $\{C, L, E, R, S\}$ per triangle in the order in which the triangles are visited. Furthermore, because half of the symbols are a C, the string of symbols (also called the clers string) may be trivially encoded using $2f$ bits plus a small overhead per handle. Further reductions of this storage cost [8, 18, 50, 56] have been proposed, achieving a guarantee of $1.80f$ bits and often an entropy of less than $1.0f$ bits.

When the connectivity of a triangle mesh is compressed as discussed above, the bulk of the storage cost is used to encode the location of the vertices (geometry). This cost may be reduced through geometry compression, which combines: a lossy quantization with loss-less prediction and statistical coding of the residues. Quantization truncates the vertex coordinates to a desired accuracy and maps them into integers that can be represented with a limited number of bits. Using 12 bit integers for each coordinate ensures a sufficient geometric fidelity for most

applications. Both the encoder and the decoder use the same predictor to guess the location of the next vertex. The most popular parallelogram predictor [66] predicting $c.o.v.g$ as $c.n.v.g + c.p.v.g - c.v.g$. The residues between the predicted and the correct coordinates are small integers, suitable for statistical compression [54]. The combination of these three steps compresses vertex location data to about $7f$ bits.

When there is no need to preserve the original connectivity or the original vertex locations, superior compression may be achieved by retiling the surface [67]. Three different retiling approaches have been designed to increase compression. The piecewise regular mesh (PRM) retiling approach [58] decomposes the surface into 3 reliefs, each one comprising triangles whose normals are closest to one of the three principal directions. Each relief is resampled along a regular grid and the global connectivity of the new triangle mesh is recomputed. When the sampling rate is chosen so that the resulting PRM has roughly the same number of vertices as the original mesh, the mean square error is less than 0.02% of the diameter of the bounding box. Because of the regularity of the sampling in each relief, the PRM may be compressed using a modified Edgebreaker down to a total about $2t$ bits, which accounts for both connectivity and geometry. SwingWrapper [3], another retiling approach, partitions the surface of an original mesh M into simply connected regions, called triangloids. From these, it generates a new mesh M'. Each triangle of M' is a linear approximation of a triangloid of M. By construction, the connectivity of M' is regular (on average, 96% of the triangles are of type C or R, 82% of the vertices have valence 6) and can be compressed to less than a bit per triangloid using Edgebreaker. The locations of the vertices of M' are encoded with about six bits per vertex of M', thanks to a prediction technique, which uses a single correction parameter per vertex. Instead of using displacements along the normal or heights on a grid of parallel rays, SwingWrapper requires that the left and right edges of all C triangles have a prescribed length L, so that the correction to the predicted location of their tip may be encoded using the angle of the hinge edge. For a variety of popular models, it is easy to create an M' with 10 times fewer triangles than M. The appropriate choice of L yields a total file size of $0.4t$ bits and a mean square error with respect to the original of about 0.01% of the bounding box diagonal. Normal meshes [16], a third retiling approach, computes a simplified model [21, 22] and uses it as the coarse mesh of a modified a butterfly subdivision [10] scheme, where each new vertex location is adjusted by a displacement along the estimated surface normal. The corrective displacements are compressed using a wavelet transform [10, 71].

In summary, SGC, CSG, and triangulated boundary representations may contain redundant information. Redundancy in SGC models may be reduced by dropping, merging, or joining cells when the result is a valid complex representing the same pointset. Redundancy in CSG may be reduced by identifying and eliminating redundant primitives. Triangle meshes may be conveniently and compactly represented using a corner table. To further reduce storage size, the connectivity of triangle meshes may be compressed using Edgebreaker to about 1 bit per triangle. The vertex location may be compressed using quantization, parallelogram prediction, and entropy coding of the residues. Considerably more aggressive compression may be achieved by retiling the original model into a more regular mesh for which vertex locations may be each encoded using a single corrective scalar.

7 Conclusions

We have discussed several measures of complexity of 3D shapes or of their representations and have reviewed corresponding complexity reduction methods. Representational complexity may be reduced in SGC, CSG, and triangulated BRep models without altering the represented pointset through redundancy elimination, through the choice of a simpler data structure, and through compression. By choice, a system developer may limit the algebraic complexity supported by a modeling system, hence forcing designers to use a larger number of lower degree primitives to approximate the desired shape with sufficient accuracy. When the resulting combinatorial complexity exceeds the available computing or transmission capabilities, simplification techniques may be invoked to reduce the vertex count in triangulated BReps. By choice, a system developer may also limit the topological complexity supported by a modeling system to closed sets, to regularized sets, or even to manifold sets. On the other hand, a designer may reduce the topological complexity of a particular model through set-theoretic operations, through the explicit removal of handles, or by healing models whose topology was wrongly inferred from imperfect input. Finally, the morphological complexity of a shape may be reduced by tightening its boundary in the mortar to remove sharp features and thin parts. Note that combinatorial and morphological complexity measures, as well as the bit-count representational complexity measure, are not scalar values, but rather scalar functions of the error resulting from a simplification process. Hence, the answer to the question "Which of these two shapes is more complex?" may depend on the desired accuracy.

References

1. Alexandroff, P.: Elementary Concepts of Topology. Dover Publications, New York (1961)

2. Andujar, C., Brunet, P., Chica, A., Navazo, I., Rossignac, J., Vinacua, A.: Optimal iso-surfaces. CAD Conference, 2004. J Comput Aided Des Appl **1**(1–4), 503–511 (2004)

3. Attene, M., Falcidieno, B., Spagnuolo, M., Rossignac, J.: SwingWrapper: retiling triangle meshes for better EdgeBreaker compression. Genova CNR-IMA Tech. Rep. No. 14/2001. ACM Trans Graph **22**(4), 982–996 (2003)

4. Banerjee, R., Rossignac, J.: Topologically exact evaluation of polyhedra defined in CSG with loose primitives. Comput Graph Forum **15**(4), 205–217 (1996)

5. Biermann, H., Kristjansson, D., Zorin, D.: Approximate Boolean operations on free-form solids. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 185–194, August (2001)

6. Brun, J.M., Evrard, M.: EUCLID as a CAD/CAM system used in the second CAM-I benchmark. Proc. CAM-I's 3rd Geometric Modeling Seminar, P-85-MM01, Nashville, Tenn., pp. 3-81–3-124 (1985)

7. Brown, C.M., Requicha, A.A.G., Voelcker, H.B.: Geometric modelling systems for mechanical design and manufacturing. ACM/CSC-ER, 770–778 (1978)

8. Coors, V., Rossignac, J.: Delphi encoding: improving Edgebreaker by geometry based connectivity prediction. Visual Computer **20**, 1–14 (2004)

9. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Proc. Eurographics'98, vol. 17(2), pp. 167–174, June (1998)

10. Dyn, N., Levin, D., Gregory, J.A.: A butterfly subdivision scheme for surface interpolation with tension control. ACM Trans Graph **9**(2), 160–169 (1990)

11. van Emmerik, M., Rappoport, A., Rossignac, J.: Simplifying interactive design of solid models: a hypertext approach. Visual Computer **9**(5), 239–254 (1993)

12. Farin, G., Hoshek, J., Kim, M.S. (eds.): Handbook of Computer Aided Geometric Design. North Holland, Amsterdam (2002)

13. Garland, M., Heckbert, P.: Surface simplification using quadric error metrics. Proc. ACM SIGGRAPH'97, pp. 209–216 (1997)

14. Garland, M.: QSlim 2.0 [Computer Software]. University of Illinois at Urbana-Champaign, UIUC Computer Graphics Lab (1999) http://graphics.cs.uiuc.edu/~garland/software/qslim.html

15. Goodman, J.E., O'Rourke, J. (eds.): Handbook of Discrete and Computational Geometry. CRC Press, London (2004)

16. Guskov, I., Vidimce, K., Sweldens, W., Schroeder, P.: Normal meshes. In: Siggraph'2000 Conference Proceedings, pp. 95–102 (2000)

17. Hoppe, H., DeRose, T. Duchamp, T., McDonald, J., Stuetzle, W.: Mesh optimization. In: Computer Graphics: Siggraph'93 Proceedings, pp. 19–25 (1993)

18. King, D., Rossignac, J.: Guaranteed 3.67V bit encoding of planar triangle graphs. 11th Canadian Conference on Computational Geometry (CCCG99), pp. 146–149, Vancouver, CA, August 15–18 (1999)

19. Kormos, J.G.: The SDRC GEOMOD - interactive solid modeling program. Proc. CAM-I's 3rd Geometric Modeling Seminar, P-85-MM01, Nashville, Tenn., pp. 3-125–3-167, March (1985)

20. Kriezis, G.A., Patrikalakis, N.M., Wolter, F.-E.: Topological and differential equation methods for surface intersections. J Comput Aided Des **24**(1), 41–55 (1992)

21. Khodakovsky, A., Schroeder, P., Sweldens, W.: Progressive geometry compression. In: SIGGRAPH 2000, Computer Graphics Proceedings, 271–278 (2000)

22. Lee, A.W.F., Sweldens, W., Schroeder, P., Cowsar, L., Dobkin, D.: MAPS: multiresolution adaptive parametrization of surfaces. In: SIGGRAPH'98 Conference Proceedings, pp. 95–104 (1998)

23. Liepa, P.: Filling holes in meshes. Proceedings Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 200–205 (2003)

24. Llamas, I., Kim, B.M., Gargus, J., Rossignac, J., Shaw, C.D.: Twister: a space-warp operator for the two-handed editing of 3D shapes. ACM Trans Graph (TOG), Proc. ACM SIGGRAPH **22**(3), 663–668 (2000)

25. Loop, C.: Smooth spline surfaces over irregular meshes. Comput Graph **28**, 303–310 (1994)

26. Lazard, S., Peñaranda, L., Petitjean, S.: Intersecting quadrics: an efficient and exact implementation. Proceedings 20th Annual Symposium on Computational Geometry, pp. 419–428 (2004)

27. Luebke, D., Reddy, M., Cohen, J., Varshney, A., Watson, B., Hubner, R.: Levels of Detail for 3D Graphics. Morgan Kaufmann, San Francisco (2002)

28. Mantyla, M.: Boolean operations of 2-manifold through vertex neighborhood classification. ACM Trans Graph **5**(1), 1–29 (1986)

29. Mantyla, M.: An introduction to solid modeling. Computer Science Press (1988)

30. Miller, J., Goldman, R.: Geometric algorithms for detecting and calculating all conic sections in the intersection of any two natural quadric surfaces. Graph Models Image Process **57**(1), 55–66 (1995)

31. Murali, T.M., Funkhouser, T.: Consistent solid and boundary representations from arbitrary polygonal data. Symposium on Interactive 3D Graphics, pp. 155–162, April (1997)

32. Okino, N., Kakazu, Y., Kubo, H.: TIPS-1: technical information processing system for computer-aided design. In: Hatvany, J. (ed.) Drawing and Manufacturing, Computer Languages for Numerical Control, pp. 141–150. North-Holland, Amsterdam (1973)

33. Pasko, A., Adzhiev, V., Sourin, A., Savchenko, V.: Function representation in geometric modelling: concepts, implementation and applications. Visual Computer **11**(8), 429–446 (1995)

34. Pfister, H., Zwicker, M., van Baar, J., Gross, M.: Surfels: surface elements as rendering primitives. Proceedings of 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 335–342, July (2000)

35. Requicha, A.: Representation of rigid solids: theory, methods, and systems. ACM Computing Surveys **12**(4), 437–464 (1980)

36. Requicha, A., Voelcker, H.: Boolean operations in solid modeling: boundary evaluation and merging algorithms. Proc. IEEE **73**(1), 30–44 (1985)

37. Rossignac, J., Borrel, P.: Multi-resolution 3D approximations for rendering complex scenes. In: Geometric Modeling in Computer Graphics. Springer, Berlin Heidelberg New York (1993)

38. Cameron, S., Rossignac, J.: Relationship between S-bounds and active zones in constructive solid geometry. Proceedings of Theory and Practice of Geometric Modeling, pp. 369–348, Blaubeuren, Germany, October (1988)

39. Rossignac, J., Cardoze, D.: Matchmaker: manifold breps for non-manifold r-sets. Proceedings of ACM Symposium on Solid Modeling, pp. 31–41, June (1999)

40. Rossignac, J., O'Connor, M.: SGC: a dimension-independent model for pointsets with internal structures and incomplete boundaries. In: Wosny, M., Turner, J., Preiss, K. (eds.) Geometric Modeling for Product Engineering, pp. 145–180. North-Holland, Amsterdam (1989)

41. Rossignac, J., Requicha, A.: Constructive non-regularized geometry. J Comput Aided Des **23**(1), 21–32 (1991)

42. Ronfard, R., Rossignac, J.: Full range approximation of triangulated polyhedra. Proc. Eurographics'96, Vol. 15(3), pp. 67–76 (1996)

43. Rossignac, J.: Surface simplification and 3D geometry compression. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry. CRC Press, London (2004)

44. Rossignac, J.: Education-driven research in CAD. Comput Aided Des (CAD) **36**(14), 1461–1469 (2004)

45. Rossignac, J.: Blending and offsetting solid models. Ph.D. thesis, University of Rochester, NY (1985)
46. Rossignac, J.: Through the cracks of the solid modeling milestone. In: Coquillart, Strasser, Stucki (eds.) From Object Modelling to Advanced Visual Communication, pp. 1–75. Springer, Berlin Heidelberg New York (1994)
47. Rossignac, J.: CSG formulations for identifying and for trimming faces of CSG models. In: Woodwark, J. (ed.) CSG'96, Set-theoretic solid modeling techniques and applications. Information Geometers, Bath, UK (1996)
48. Rossignac, J.: Structured Topological Complexes: A Feature-Based API for Non-manifold Topologies. In: Hoffman, C., Bronsvort, W. (eds.), Proceedings of the ACM Symposium on Solid Modeling, pp. 1–9. ACM Press, New York (1997)
49. Rossignac, J.: Edgebreaker: connectivity compression for triangle meshes. IEEE Trans Visual Comput Graph **5**(1), 47–61 (1999)
50. Rossignac, J., Szymczak, A.: Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker. Computational Geometry, Theory and Applications **14**(1/3), 119–135 (1999)
51. Rossignac, J., Voelcker, H.: Active zones in CSG for accelerating boundary evaluation, redundancy elimination, interference detection and shading algorithms. ACM Trans Graph **8**, 51–87 (1989)
52. Rossignac, J., Safonova, A., Szymczak, A.: 3D compression made simple: Edgebreaker on a Corner Table. Shape Modeling International Conference, pp. 278–283, Genoa, Italy, May (2001)
53. Rossignac, J., Safonova, A., Szymczak, A.: Edgebreaker on a Corner Table: a simple technique for representing and compressing triangulated surfaces. In: Farin, G., Hagen, H., Hamann, B. (eds.) Hierarchical and Geometrical Methods in Scientific Visualization, pp. 41–50. Springer, Berlin Heidelberg New York (2003)
54. Salomon, D.: Data Compression: The Complete Reference. Springer, Berlin Heidelberg New York (2000)
55. Sethian, J.: Level Set Methods and Fast Marching Methods. Cambridge University Press (1999)
56. Szymczak, A., King, D., Rossignac, J.: n Edgebreaker-based efficient compression scheme for connectivity of regular meshes. Journal of Computational Geometry: Theory and Applications (2000)
57. Schmitt, B., Pasko, G., Pasko, A., Kunii, T.L.: Modelling and representing surfaces: rendering trimmed implicit surfaces and curves. Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (2004)
58. Szymczak, A., Rossignac, J., King, D.: Piecewise regular meshes: construction and compression, graphics models. Special Issue on Processing of Large Polygonal Meshes **64**, 183–198 (2002)
59. Szymczak, A., Vanderhyde, J.: Extraction of topologically simple isosurfaces from volume datasets. Proc. IEEE Visualization, pp. 67–74 (2003)
60. Dey, T., Goswami, S.: Provable surface reconstruction from noisy samples. Proceedings of the Twentieth Annual Symposium on Computational Geometry (2004)
61. Taubin, G., Rossignac, J.: Geometric compression through topological surgery. ACM Trans Graph **17**(2), 84–115 (1998)
62. Taubin, G.: Estimating the tensor of curvature of a surface from a polyhedral approximation. Proceedings of the 5th International Conference on Computer Vision, pp. 902 (1995)
63. Taubin, G.: A signal processing approach to fair surface design. Proceedings of the 22nd Annual Conference on Computer graphics and Interactive Techniques, pp. 351–358, September (1995)
64. Tilove, R.: Set membership classification: a unified approach to geometric intersection problems. IEEE Trans. Comput **C-29**(10), 874–883, October (1980)
65. Tilove, R.:A null-object detection algorithm for constructive solid geometry. Commun ACM **27**(7), 684–694 (1984)
66. Touma, C., Gotsman, C.: Triangle mesh compression. Graphics Interface 1998, Vancouver, BC, Canada, pp. 26–34 (1998) [See http://www.informatik.uni-trier.de/~ley/db/conf/graphicsinterface/graphicsinterface1998.html]
67. Turk, G.: Retiling polygonal surfaces. Proc. ACM Siggraph'92, pp. 55–64, July (1992)
68. Warren, J., Weimer, H.: Subdivision methods for geometric design: a constructive approach. Morgan Kaufmann, San Francisco (2002)
69. Williams, J., Rossignac, J.: Tightening: curvature-limiting morphological simplification. GVU Tech Report GIT-GVU-04-27 (2004)
70. Williams, J., Rossignac, J.: Mason: Morphological simplification. Graphical Models **67**(4), 285–303 (2005)
71. Zorin, D., Schroeder, P., Sweldens, W.: Interpolating subdivision for meshes with arbitrary topology. Computer Graphics **30**, 189–192 (1996)

JAREK ROSSIGNAC is Full Professor in the College of Computing and the Chair of IRIS (Interaction with Robots, Images, and Shapes) at the Georgia Institute of Technology. His research is on the interactive design, compression, transmission, and visualization of multi-resolution representations of complex 3D shapes, biomedical structures, and animations. He holds a PhD in E.E. from the University of Rochester, a French Diplome d'Ingenieur from the E.N.S.E.M. and a Maitrise in M.E. from the University of Nancy, France. Until 1996, he was Senior Manager and Visualization Strategist at IBM's T.J. Watson Research Center.

He authored 18 patents and 100 refereed papers for which he received 10 Best Paper Awards, 5 IBM Corporate Awards, and the Senior Faculty Research Award from the College of Computing at Georgia Tech. He chaired 20 conferences, served on 7 Editorial Boards and on 63 Program committees. He Guest-Edited 8 special issues and gave numerous keynotes and distinguished lectures. He served as the Director of the GVU Center and as the Chair of the Solid Modeling Association. He is a Fellow of the Eurographics Association.