

# The Discretized Polyhedra Simplification (DPS): a Framework for Polyhedra Simplification Based on Decomposition Schemes

Carlos Andújar   Dolors Ayala   Pere Brunet

February 4, 1999

## Abstract

Automatic simplification of polyhedral objects is a major topic in many computer graphics applications. This work discusses simplification algorithms for the generation of a multiresolution family of solid representations from an initial polyhedral solid. We introduce the Discretized Polyhedra Simplification (DPS), a framework for polyhedra simplification using space decomposition models. The DPS is based on a new error measurement and provides a sound scheme for error-bounded, geometry and topology simplification while preserving the validity of the model. A method following this framework, Direct DPS, is presented and discussed. Direct DPS uses an octree for topology simplification and error control, and generates valid solid representations. Our method is also able to generate approximations which do not interpenetrate the original model, either being completely contained in the input solid or bounding it. Unlike most of the current methods, restricted to triangle meshes, our algorithm can deal and also produces faces with arbitrary complexity. An extension of the Direct method for appearance preservation, called Hybrid DPS, is also discussed.

## 1 Introduction

Many computer graphics applications, including CAD and virtual reality systems, require modeling, handling and visualization of very large and geometrically complex systems. Geometry simplification deals with generation of 3D models that resemble the input model but involve less faces, edges and vertices. A concept closely related is the approximation error —a quantification of the difference between the original model and the simplification. *Level of Detail* is concerned to the possibility of using different representations of a geometric object having different levels of accuracy and complexity. Multi-resolution models provide several level-of-detail representations of a geometric model and have become a powerful tool in many computer graphics applications, including CAD, virtual reality and scientific visualization, as they can accelerate the handling of complex models by omitting unessential computation and reducing storage

space in visualization [Cla76], [Cro82], [RB93], [MS95], [RH94], [FS93], transmission over networks [ANM97], [DLW94], [aa95], [CPD<sup>+</sup>96], [Hop96], query acceleration [Vel92], collision detection, visibility analysis and acoustic modeling [ea98]. Simplification is also used for reducing the verbosity of 3D models, adjusting the accuracy to the application’s requirements and multi-resolution interactive modeling.

The need for multi-resolution representations was already stated in 1976 in the context of real-time visualization. In [Cla76] J. Clark pointed out that objects which cover a small area in the screen could be rendered with a simplified version and he proposed a hierarchical model supporting several representations of an object.

Although multiresolution representations are sometimes obtained interactively [Cro82], [HG94], extensive research is being performed in developing algorithms for the automatic generation of these multiresolution representations.

## 1.1 Solid Simplification vs. Surface Simplification

Most of the simplification methods published so far are concerned with a sub-problem of geometry simplification which will be called *surface simplification*. In surface simplification the approximation error is measured by some distance defined on the points *on the surface*, regardless of the enclosed volume, if any. We introduce a new approach for the geometry simplification problem, the *solid simplification* (Table 1). Solid simplification deals with the approximation of solid mathematical models, i.e. point-set models, whereas surface simplification deals with entities inherently 2D (see Figure 1 a, b).

The behavior of surface simplification methods is independent on the volume enclosed by the surface, and thus many surface simplification methods can deal with non-manifold, self-intersecting and even open surface patches. Since the disparity between the original surface and the simplified one should be measured by some error metric based on the points *on the surface*, the goal is to calculate a less complex surface which approximates the original one within a tolerance bound around the surface.

Unlike surface simplification, solid simplification methods take into account the volume enclosed by a two-manifold surface. Solid simplification measures the error using the points *inside the solid*, and hence it has more freedom for modifying its geometry and its topology (genus and shells).

Solid simplification applications make us somewhat less concerned with preserving surface appearance attributes, such as curvature and color, than some visualization-oriented surface simplification methods are.

Surface simplification methods can be used for polyhedra simplification, and good results could be expected when simplifying objects with a simple topology and without  $\varepsilon$ -zones (thin zones of the space crossed by two or more sheets or the surface). Beyond these limited cases, the results obtained by surface simplification methods are very poor. Figure 2 shows two sample objects which cannot be successfully simplified with a surface simplification method.

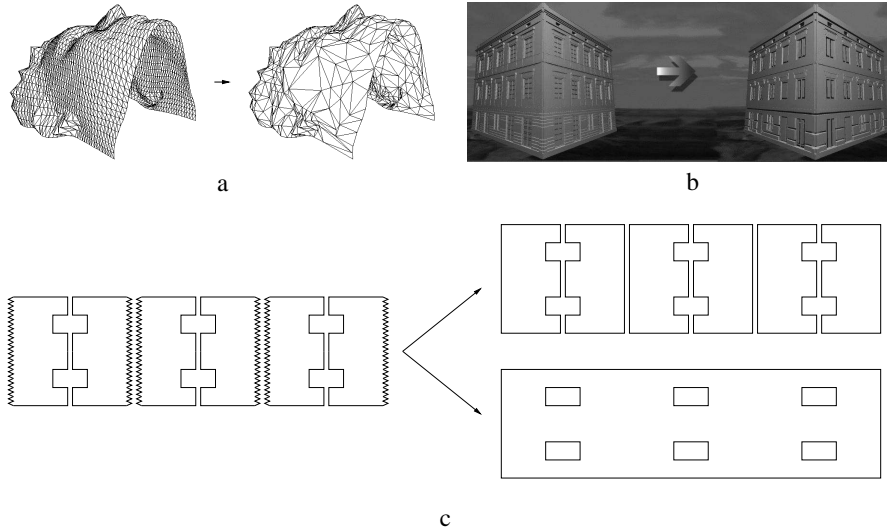


Figure 1: Surface vs. solid simplification. (a) The surface simplification problem. (b) The solid simplification problem. (c) A 2D solid being simplified by a surface simplification method (top) and a solid simplification method (bottom). The slots of the assembly must be preserved in the former case since otherwise a great error is introduced, whereas the solid simplification is able to change the topology in order to achieve a higher geometry reduction.

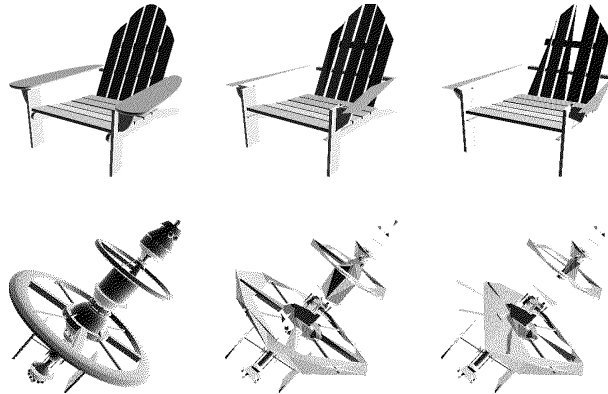





Figure 2: Top: A topology-preserving surface simplification method is unable to simplify the back of the chair into a single face, even with a high tolerance. Bottom: poor result obtained with a surface simplification algorithm over an object with many shells. Note that many components appear disconnected in the simplification.

## 1.2 Background and Previous Work

Although surface simplification is a well known problem and a great amount of algorithms have been proposed, from terrain simplification up to triangular

Table 1: Surface simplification vs. Solid Simplification

	Surface Simplification	Solid Simplification
Underlying entity	surface (2D)	point-set model (3D)
Basis for error control	points on the surface	points inside the volume
Topology simplification	very limited	high degree of freedom
Previous work	intense research	papers are still rare
Applications	visualization-oriented	more general
		

mesh simplification, polyhedra solid simplification methods are still very rare [aa96], [AAB<sup>+</sup>96].

Related issues with simplification methods are topology simplification and error bounds. Topology simplification deals with the ability of the algorithm to reduce the topology of the model, i.e., its genus and number of shells. This is an important requirement in practical applications in order to get high simplification rates in objects with complex topologies and in assemblies [aa96]. The existence of error bounds or tolerances for the approximations is also a basic user requirement which is useful, in LOD-based applications, to choose the optimal approximate representation for a distant object.

### 1.2.1 Previous work on surface simplification

Most of the current surface simplification methods are devoted to triangular meshes (mesh simplification). Some of them are 3D extensions of 2D polygonal simplification methods while others are direct 3D approaches. Although some bottom-up methods exist [Vel93], most methods follow a top-down strategy, performing a face reduction directly on the mesh by the iterative application of reduction operators (incremental methods). Different operators have been proposed: vertex removal [SZL92], [ea96], [SL96], edge collapse [RR96], [Gue96], [AS96], [Hop96], [Hop97], face removal [Ham94], superfaces merging [KT93], [KT96] and edge flipping<sup>1</sup> [CCMS97], [BBCS96]. These operators are reviewed in Section 1.2.2. Other mesh simplification methods are based on re-tiling techniques [Tur92] and multi-resolution analysis [DLW94], [aa95], [CPD<sup>+</sup>96], [GSG96]. See [Ros96], [Eri96], [ABJN97] for a survey on such methods. With a few exceptions, mesh simplification methods are unable to handle properly objects with complex topologies since topology preservation limits geometry simplification.

<sup>1</sup>Edge-flipping does not reduce the face count but may improve the fitting.

Other approaches are based on space-partitioning and clustering techniques in 3D space, like [RB93], [RR96], [Red96], [GH97]. These methods allow topological changes and yield high compression ratios but the resulting simplified representations are not valid B-Rep (they contain dangling parts or non-manifold boundaries), so they cannot be used for solid simplification.

### 1.2.2 Operators over triangle meshes

In this section we compare local mesh operators, which are a key ingredient of the most populated family of surface simplification methods and is also one of the key ingredients of our approach.

Incremental methods are based on a set of local operators which either reduce the mesh complexity (reduction operators) or improve the fitting of the approximated mesh (fitting operators).

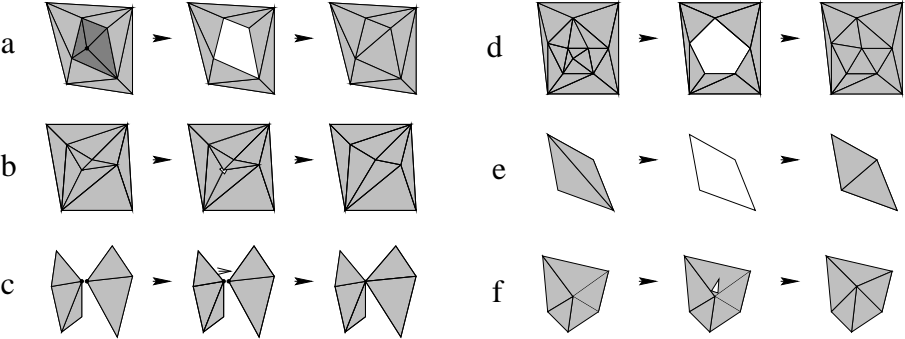


Figure 3: Reduction and fitting operators: a) vertex-removal, b) edge-collapse, c) vertex-clustering, d) face-removal, e) edge-flip and f) vertex-displacement.

The following reduction operators have been proposed in the geometry simplification literature (see Figure 3):

- *vertex-removal*

The vertex-removal operator take as parameter the vertex to be removed. The vertex and its  $t$  incident triangles are removed. The resulting hole is triangulated with  $t - 2$  triangles. The only computed parameter is the new incidence graph of the hole's triangulation. The operator reduces the number of faces by two, the number of edges by three and the number of vertices by one. Except in degenerate cases which can be easily identified by inspection of the incidence graph, the vertex-removal operator preserves the topology of the mesh. To avoid self-intersections, additional geometric tests are required. The selection of the vertex to be removed is commonly based on a curvature estimation at the vertex. Vertices with low curvature values are removed first. The vertex removal operator is used in [SZL92], [ea96], [SL96] and in most terrain simplification methods.

- *edge-collapse*

The edge-collapse operator takes as parameter the edge to be collapsed, or its equivalent, a pair of vertices sharing an edge. The two vertices are collapsed in one vertex. As a result of this collapse, the triangles sharing the edge degenerate in a segment and are removed. The only computed parameter is the new vertex position, which usually is that of one of the two old vertices, or a weighted average. The operator reduces the number of faces by two, the number of edges by three and the number of vertices by one. Except in degenerate cases which can be easily identified by inspection of the incidence graph, the edge-collapse operator preserves the topology of the mesh. To avoid self-intersections, additional geometric tests are required. The selection of the edge to be removed is commonly based on an estimation of the error in the Hausdorff distance [Grü67] sense. Usually, feasible edge-collapse operators are computed in a previous step, and stored in priority queue ordered by error. The edge-collapse operator is used in most of the state-of-the-art methods: [RR96], [Gue96], [AS96], [Hop96], [Hop97]. Due to its simplicity, the edge-collapse operator has been used successfully both in geometry simplification (lossy) and geometry compression (loss-less).

- *vertex-clustering*

The vertex-clustering of  $v$  vertices is conceptually equivalent to  $v - 1$  vertex-clustering operation involving just two vertices, so we will review only this latter form. The vertex-clustering operator takes as parameter the two vertices to be collapsed. When these vertices are connected along an edge, this operation is an edge-collapse, but disconnected vertices are also allowed to be collapsed. If the latter case, only the number of vertices is reduced; otherwise, the triangles sharing the edge degenerate in a segment and are removed. The only computed parameter is the new vertex position, which usually is that of one of the two old vertices, or a weighted average. If the vertices are connected by an edge, the operator reduces the number of faces by two, the number of edges by three and the number of vertices by one; otherwise only reduces the number of vertices. Unlike previous reduction operators, the vertex-clustering does not preserve the topology of the mesh, and creates non-manifold meshes. To avoid self-intersections, additional geometric tests are required. In the simpler methods, the selection of the vertices to be collapsed is based on geometric proximity. Usually, several vertices are clustered at a time. The vertex-clustering operator is used in [RB93], [RR96], [Red96], [GH97].

- *face-removal*

The face-removal take as parameter the triangle  $T$  to be removed. This triangle and all its neighbors sharing one vertex with  $T$  are removed. The resulting hole is triangulated with the help of a new vertex. The computed parameters are the coordinates of the new vertex, and the incidence graph of the triangulation. The operator reduces the number of faces by four, the number of edges by six and the number of vertices by two. Except in degenerate cases which can be identified [Ham94], the face-removal operator preserves the topology of the mesh. Due to the high number of triangles involved, incremental methods using face-removal rapidly arrive

to a mesh which cannot be further simplified while maintaining its validity, whereas other simpler operators such as edge-collapse could be applied. The face removal operator is used in a few methods [Ham94].

In addition to reduction operators, which modify the count of the geometric entities of the mesh, surface simplification methods often rely on fitting operators:

- *edge-flip*

The edge-flip operator take as parameter two adjacent triangles (or their shared edge). The non-planar quadrilateral resulting of merging together both triangles is triangulated using the opposite diagonal. There are no computed parameters. The edge-flip operator preserves the topology of the mesh but not the incidence graph. To avoid self-intersections, additional geometric tests are required. The edge-flip is used for two purposes: a) to improve the fitting of the simplification to the original surface, in non-flat regions, and b) to create well-shaped triangles, in flat-regions. The edge-flip operator is used in [Hop96], [CCMS97], [BBCS96], to minimize the error produced by another previous operator, such as edge-collapse.

- *vertex-displacement*

The vertex-displacement operator take as parameter the vertex to be moved. The only computed parameters are the new coordinates, usually given as an offset vector. The vertex-displacement operator preserves the topology of the mesh and the incidence graph. To avoid self-intersections, additional geometric tests are required. The vertex-displacement is used to locally improve the fitting of the simplification to the original surface. Its relevance is due to the fact that the vertex displacement is closed in the domain of valid triangle meshes (i.e. it preserves the validity); this does not hold for arbitrary polyhedra, since vertex-displacement would produce non-planar faces.

For many authors, edge-collapse is the most suitable operator for face reduction. There are two reasons for that: its simplicity (only affecting two triangles at a time), allowing further simplification when other operators cannot be applied, and its generality, in the sense that all topology-preserving reduction operators can be derived from a series of edge-collapses and edge-flips:

- One vertex-removal can be replaced by an edge-collapse and several edge-flips.
- The multiple vertex-clustering of  $n$  vertices can be replaced by  $n - 1$  vertex-clustering of two vertices, which in turn is a generalization of edge-collapse to arbitrary pairs of vertices. Since we are interested in keeping the two-manifold property of the surface, only edge-collapses are suitable.
- One face-removal can be replaced by two vertex removal, and hence by two edge-collapses and several edge-flips. Note that, in the face-removal we are considering, the triangulation is performed introducing a new vertex. Otherwise, three edge-collapses instead of two would be necessary.

### 1.2.3 Previous work on simplification using decomposition models

Decomposition approaches [And98b] automatically guarantee both the topology simplification and the error bounds. These approaches use an intermediate space decomposition representation (usually a voxel or octree representation). Unlike other approaches, topology is simplified in a discretization process, instead of being simplified by means of operations over the brep.

A simplification method based on a decomposition model was introduced in [AAB95], where an algorithm for the special case of orthogonal polyhedra is presented. The algorithm used a special kind of octree, the Maximal Division Classical Octree (MDCO) [BJN<sup>+</sup>88] as the intermediate model. This representation scheme is discussed in Section 3.1. The input of the method [AAB95] is a closed polyhedron, which is discretized into an MDCO. For each terminal grey node of the MDCO, the color of the eight vertices of the corresponding octant is computed, resulting in black (inside the polyhedron) and white (outside) points. There are  $2^8 = 256$  possible configurations of the vertex colors of a terminal node [LC87], [Sri81]. This configurations can be grouped by means of symmetries into 14 equivalence classes (see Figure 4). Terminal grey nodes can be non-regular (all vertices black or all white) or regular. Regular nodes can be ambiguous or non-ambiguous.

After the vertex color computation, a reconstruction algorithm similar to Marching Cubes (MC) [LC87] is applied in two stages: the first one computes the geometric information of each vertex of the result by means of a look-up table indexed by the configuration of the eight vertices of the terminal node; in a second stage the incidence graph is computed by grouping coplanar vertices into loops and faces, following the algorithm presented in [Nav86]. Unlike MC, the resulting polyhedron is orthogonal and the faces are not limited to triangles. Furthermore, the ambiguity problem [GW94] can be easily solved by simply subdividing once more ambiguous terminal nodes and adopting a black or white proximity criterion.

In order to guarantee an error bound in the sense of the Hausdorff distance, non-regular regions must be reconstructed separately. In addition to be restricted to orthogonal simplifications, due to the non-regular regions problem the method presented in [AAB95] usually tends to increase the number of shells.

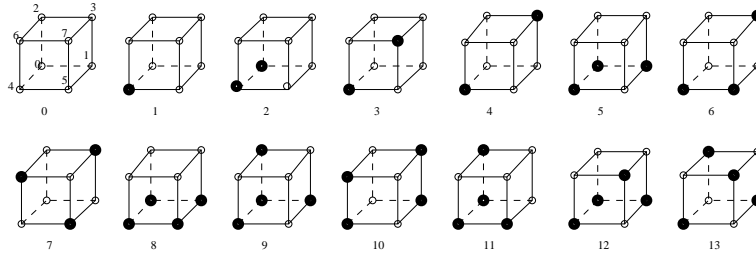


Figure 4: The 14 equivalence classes of terminal nodes. Class 0 is non-regular. Classes 1, 2, 5, 8, 9 and 11 are non-ambiguous, and classes 3, 4, 6, 7, 10, 12 and 13 are ambiguous.



A first extension of [AAB95] for the case of general polyhedra was proposed in [AAB<sup>+</sup>96]. The algorithm has three major steps: discretization of the input model, generation of a multi-resolution of octrees by simply pruning the higher octree level, and reconstruction of a polyhedron for each octree. The incidence graph reconstruction is based on a classification of terminal grey nodes by planarity criteria and an iterative refinement guided by the minimization of an energy function involving coplanarity of vertices, linear separability and stabbing of nodes by the edges. The geometry reconstruction is based on a numerical programming problem minimizing the previous goal function. Unfortunately, the method presented in [AAB<sup>+</sup>96] is quite complicated and, due to the non-regular regions problem, often increases the number of shells.

A different approach based on signal processing is presented in [HHK<sup>+</sup>95]. The algorithm has two steps. In the first one, the input polyhedron is sampled and low-pass filtered resulting on a 3D grid of scalar values. The density is computed using the following integration restricted to the interior points:

$$f(i, j, k) = \int \int \int h(x, y, z) dx dy dz \quad (1)$$

where  $h(x, y, z)$  is a low-pass filter such that the contribution of points is maximum at the voxel's center and null at some finite radius  $R$ :

$$h(p) = (R - \text{dist}(p, c)) / R. \quad (2)$$

In the second step the surface is extracted with the original Marching Cubes (MC) in [HHK<sup>+</sup>95] and an improved version in [aa96]. Unfortunately, the approximation error is not bounded, and the resulting triangle mesh has many redundant triangles due to the verbosity of MC. Furthermore, the simplification smoothes the object's surface and the user must provide an isodensity value for MC.

#### 1.2.4 Surface extraction and octree-to-boundary conversion

The surface extraction problem is closely related to the simplification problem [And98b]. On one hand, most surface simplification algorithms appeared as a tool for reducing the verbosity of surface extraction problems such as Marching Cubes, which produces high density of triangles even in regions of low curvature. On the other hand, the simplification methods based on decomposition models [AAB95], [AAB<sup>+</sup>96], [HHK<sup>+</sup>95], [aa96] must use some kind of surface extraction algorithm. The immediate consequence is that any low-verbosity conversion algorithm from a decomposition model to a boundary representation can be used as the reconstruction step of a simplification method based on decomposition schemes. Several low-verbosity conversion algorithms have appeared in the context of voxels and octrees.

In [MSS94], an improved MC is presented. The method is based on a discretization of face orientations, resulting in many coplanar triangles that are merged together in a post-process step.

In [JAS95] a surface extraction method based on face octrees [Bru90] is presented. The main contribution is that the generated faces are not restricted

to lie inside a voxel. The algorithm has three main steps: construction of a Geometrically Deformed Model [MBL<sup>+</sup>91] using the minimization of an energy function, construction of the face octree and reconstruction of the polyhedron. Unfortunately, [JAS95] is unable to reconstruct the polyhedron inside some ‘thin’ regions, and hence approximation error is not bounded.

### 1.3 Problem Statement

The solid simplification problem for general polyhedra can be stated as follows: given a general, two-manifold polyhedron  $P$  with  $n_f(P)$  faces, a multi-resolution family of two-manifold polyhedra  $P_l, P_{l-1}, \dots, P_0$  approximating the initial object  $P$  must be generated. We look for a simplification algorithm fulfilling the following requirements:

- The approximation of the individual polyhedra  $P_k$  must be monotonically decreasing from the closest approximation  $P_l$  to the coarser one  $P_0$ . More precisely, a set of tolerances  $\varepsilon_l, \varepsilon_{l-1}, \dots, \varepsilon_1$  with  $\varepsilon_k < \varepsilon_{k-1}$  must exist such that some geometric distance between  $P_k$  and  $P_{k-1}$  is bounded by  $\varepsilon_k$ . Several bound definitions will be discussed in Section 2.1.2.
- The number of faces of the polyhedra  $P_k$  should be monotonically decreasing from the best approximation  $P_l$  to  $P_0$ , that is  $n_f(P_{k-1}) < n_f(P_k)$ .
- Both the geometry and the topology –genus and number of shells– of the initial polyhedron  $P$  must be simplified. For topology simplification we mean the proper modification of the genus and the number of shells in order to achieve a high simplification ratio. Note that topology simplification may suppose both the increment or decrement of genus and shells. Usually,  $P_0$  will be a genus-0 approximation to  $P$ .
- Relevant features of  $P$  such as sharp edges must be kept as much as possible during the simplification sequence from  $P_l$  to  $P_0$ .
- Flat regions of the initial polyhedron  $P$  must be approximated by large, planar faces in  $P_k$  whenever possible.

### 1.4 Contribution

In this work we introduce the *Discretized Polyhedra Simplification (DPS)*, a framework for polyhedra simplification using space decomposition models. Several criteria for DPS methods are discussed, and two new algorithms, the Direct DPS and the Hybrid DPS are presented. These methods, which are based on an extension of the classical octree representation scheme, improve previous methods presented in [AAB95] and [AAB<sup>+</sup>96].

These new approaches generate error-bounded two-manifold approximations, and are capable of reducing the topological complexity. Furthermore, these methods are not restricted to triangular meshes, and the Direct DPS algorithm deals and produces faces of arbitrary complexity (including holes). The Direct DPS has many applications in image acceleration, occlusion analysis, query

acceleration and acoustic modeling. The Hybrid DPS is also suitable to LOD-based visualization of complex assemblies.

## 1.5 Organization of this document

In section 2, the DPS framework is presented, and its relationship with surface extraction methods and other simplification methods is discussed. In the next sections, two different DPS methods are presented and discussed. The Direct approach presented in section 3 is a complete parameterization of a DPS that does not use the original BRep information once the decomposition model has been generated, and thus it can be used as a surface extraction algorithm from volume datasets. In section 4, the Hybrid DPS approach is outlined, which is intended to exploit the advantages of classic surface simplification methods and DPS methods by combining them in a region-based classification.

## 2 The Discretized Polyhedra Simplification (DPS) framework

### 2.1 The DPS framework

The DPS framework models a family of simplification methods which have in common the use of an intermediate space decomposition scheme to generate a multi-resolution family of solid representations. The DPS framework involves five components: a decomposition scheme, an error metric, and discretization, reconstruction and face reduction processes. The DPS pattern (Figures 5 and 6) shows the integration of these components.

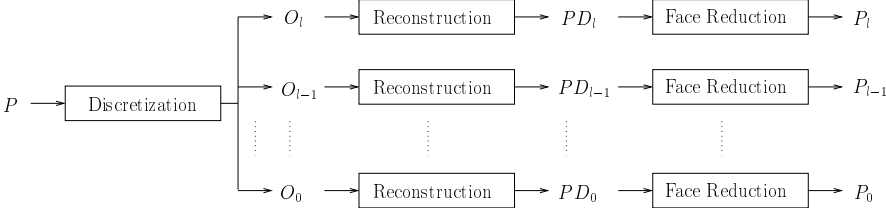


Figure 5: The DPS framework.  $O_k$  stands for a  $2^k \times 2^k \times 2^k$  division of the space

#### 2.1.1 Decomposition Scheme

A space decomposition scheme allows sound topology simplification. Although any decomposition scheme might be used in a DPS method, only those with equal-sized, regular cells are practical, since they provide a convenient mechanism to guarantee a tight and uniform error bound. The proposed methods rely either on a voxel scheme or on a Maximal Division Classical Octree (MDCO) [BJN+88].

Cell interpretation may strongly impact a DPS method. For instance, a disjoint partitioning of the space (where each point belongs to a unique cell) may yield to different results with respect to the same method using a quasi-disjoint decomposition.

Besides the closeness of cell’s boundary, coloring of such cells may vary from the classic solid modeling interpretation—in which a cell is black if it contains part of the solid, and is white otherwise—to more sophisticated coloring schemes such the volume buffers proposed in [HHK+95] and [aa96], through schemes distinguishing among black—completely inside—, white—completely outside—and grey—containing a part of the boundary—nodes.

#### 2.1.2 Error Metric

Almost all the simplification methods use a metric based on the points of the surface; such a metric is called *on-metric*. On-metrics are not appropriate for solid simplification since they limit the topology reduction. The DPS framework

is based on a new approach for error measurement, the *in-metrics*, i.e., based on the points *inside* the solid. Symmetrically, *out-metrics* are based on the points *outside* the solid. The following distances represent these three approaches: *In-Hausdorff* distance is the symmetric Hausdorff distance [Grü67] defined over the points *inside* the volume enclosed by the solids. A solid  $P'$  is said to approximate  $P$  within a bound  $\varepsilon$  in the *In-Hausdorff* distance sense iff

$$\forall p \in P \exists p' \in P' \mid \text{dist}(p, p') < \varepsilon \quad \text{and} \quad \forall p' \in P' \exists p \in P \mid \text{dist}(p, p') < \varepsilon . \quad (3)$$

The *On-Hausdorff* (resp. *Out-Hausdorff*) is the symmetric Hausdorff distance defined over the points *on the boundary* (resp. *outside* the solid): A solid  $P'$  is said to approximate  $P$  within a bound  $\varepsilon$  in the *On-Hausdorff* distance sense iff

$$\begin{aligned} \forall p \in \text{Surf}(P) \exists p' \in \text{Surf}(P') \mid \text{dist}(p, p') < \varepsilon \quad \text{and} \\ \forall p' \in \text{Surf}(P') \exists p \in \text{Surf}(P) \mid \text{dist}(p, p') < \varepsilon , \end{aligned} \quad (4)$$

where  $\text{Surf}(P)$  stands for the boundary of  $P$ .

A solid  $P'$  is said to approximate  $P$  within a bound  $\varepsilon$  in the *Out-Hausdorff* distance sense iff

$$\forall p \notin P \exists p' \notin P' \mid \text{dist}(p, p') < \varepsilon \quad \text{and} \quad \forall p' \notin P' \exists p \notin P \mid \text{dist}(p, p') < \varepsilon . \quad (5)$$

The In-Hausdorff is a good quantification of the difference between two solids and allows topology simplification (especially shell reduction).

Although On-metrics are the most used in surface simplification, In-metrics have a clear advantage over the On-metrics: they allow topology simplification (particularly shell reduction) in regions where two sheets of the polyhedron surface are near one of each other, with independence of their area (see Figures 1, 2). Since this circumstance is common in CAD models, specially in object assemblies, In-metric based approaches may yield to higher compression ratios, and thus, it is a very good error metric for solid simplification problems.

Brep-based simplification methods cannot properly handle In-metric bounds; only DPS methods offer the appropriate framework for that criterion.

### 2.1.3 Discretization

The discretization process is the conversion of the input solid  $P$  into a multi-resolution family of decomposition representations. The discretization proceeds through a space subdivision producing the more accurate model followed by iterative grouping of adjacent cells creating coarser representations. Grouping in the octree case is achieved by pruning of deepest level. Since irregular decomposition schemes are impractical for surface reconstruction purposes, from now on we will focus on regular decomposition schemes –either a voxelization or an octree. The notation  $O_k$  stands for a  $2^k \times 2^k \times 2^k$  division of the space. Every intermediate representation  $O_k$  approximates  $P$  within an error bound  $\varepsilon_k$  in some metric distance.

According to how the discretization is performed, DPS methods can be classified in *top-down* and *bottom-up* methods. In the top-down approach, the user

provides a resolution parameter which defines the maximum resolution of the decomposition. The discretization step is decomposed in two sub-steps. The space decomposition step produces only the more accurate member of the decomposition multi-resolution family. Coarser representations are obtained by iteratively grouping adjacent cells, and thus making intensive use of the coherence of the discretization of an object with different resolutions. For instance, simplification in octree space is as simple as removing the last subdivision level. The equivalent simplification in voxel space is the grouping of eight neighbor voxels into a single, greater voxel. Note that in this approach, topology simplification is distributed between the space decomposition and the grouping steps. The pseudo-code for the top-down discretization step is shown below:

```

function TopDownDiscretization( $P$ : Polyhedron,  $L$ : int)
   $O_L := \text{space\_decomposition}(P, L)$ 
  for  $k := L - 1$  to 1 step -1
     $O_k := \text{group}(O_{k+1})$ 
  end
  return  $O_{1..L}$ 
end

```

In the bottom-up approach, the discretization step is performed by a step-wise refinement of the previously calculated decomposition model. The first iteration generates a very coarse decomposition, e.g. a single cell, and the rest of the multi-resolution members are obtained by iteratively refining the previous decomposition. In the case of an octree, each discretization step subdivides each terminal node once. This process continues until a user-defined maximum depth is reached or, more likely, when the resolution increase does not allow a face reduction with respect to the original polyhedron  $P$ . The pseudo-code for the discretization step in a bottom-up DPS is shown below:

```

function BottomUpDiscretization( $P$ : Polyhedron,  $L$ : int)
   $O_1 := \text{one\_cell}()$ 
  for  $k := 2$  to  $L$ 
     $O_k := \text{refine}(O_{k-1}, P)$ 
  end
  return  $O_{1..L}$ 
end

```

Top-down DPS methods have the possibility of making a better use of the coherence between successive simplifications; more precisely, certain parts of the boundary of the last generated polyhedron,  $P_{k+1}$  can be used in the generation of the current one,  $P_k$ , and thus accelerating the algorithm and reducing the discrepancy between consecutive representations.

The main advantage of bottom-up DPS methods is that, when the refinement step is included inside the main loop, no user-defined parameters must be provided; the simplified representations are obtained from the coarsest one to the finest until the face count difference with respect to the original polyhedron is not relevant, so it never produces representations with more faces than the

original polyhedron that would be discarded.

#### 2.1.4 Reconstruction

The reconstruction process is the generation of a polyhedral representation  $PD_k$  for each member  $O_k$  of the multi-resolution family of decomposition models. Since classic decomposition schemes are approximate representations of polyhedral solids, there exist many ways to interpret the underlying object, e.g. Marching Cubes [LC87]. Surface fitting algorithms and isosurface extraction algorithms are candidates for the reconstruction step. Since the aim of the DPS method is to generate simplified models, the conciseness of the reconstruction algorithm is often the key ingredient for a good simplification ratio. The reconstruction must guarantee that the distance between  $PD_k$  and  $P$  is bounded. This can be accomplished by using an octree as the decomposition model and confining the boundary of  $PD_k$  to particular cells of the octree.

According to the input data of the reconstruction process, DPS methods can be classified into *direct* and *hybrid* methods. A DPS method is said to be direct if the only input necessary for its reconstruction step is the decomposition model of the current iteration. The method is said to be hybrid if it uses both the decomposition model and the polyhedron generated in the previous iteration (the original polyhedron in the case of the first iteration). Hybrid DPS methods require a top-down approach, since the information provided by a coarser polyhedron is not as useful for reconstruction purposes as the one provided by a more precise one. Hybrid DPS methods may exploit such pre-calculated geometry both to accelerate the execution and to increase the smoothness between successive approximations.

#### 2.1.5 Face Reduction

The face reduction is the incremental simplification of each intermediate polyhedron  $PD_k$  by topology-preserving operators. Avoiding application of topology reduction operators over the B-Rep is the key for producing manifold boundaries.

Reduction operators may range from simple coplanar facets merging up to lossy techniques such as edge-collapsing, vertex removal and vertex clustering techniques. The resulting polyhedron  $P_k$  has fewer faces than  $PD_k$  but the same topology. The octree provides a bound of the surface which is exploited in the face reduction.

Unlike other simplification methods in which the geometry reduction is performed directly in the polyhedral representation domain, in a DPS method geometry simplification is distributed in the discretization, reconstruction and face reduction steps.

Furthermore, the topology of the generated polyhedra is independent of that of the original polyhedron, and it is constructed regarding only the space decomposition model, so topology simplification is achieved in a direct and robust way in the discretization step, instead of applying topology reduction operators

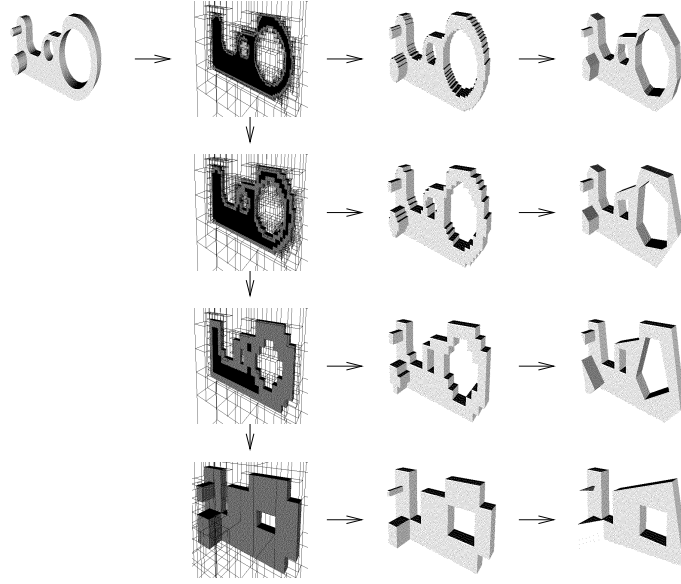


Figure 6: DPS method 3D example. From left to right: original polyhedron, multi-resolution of octrees (some front grey nodes have been culled to keep black nodes visible), reconstructed models and final multi-resolution.

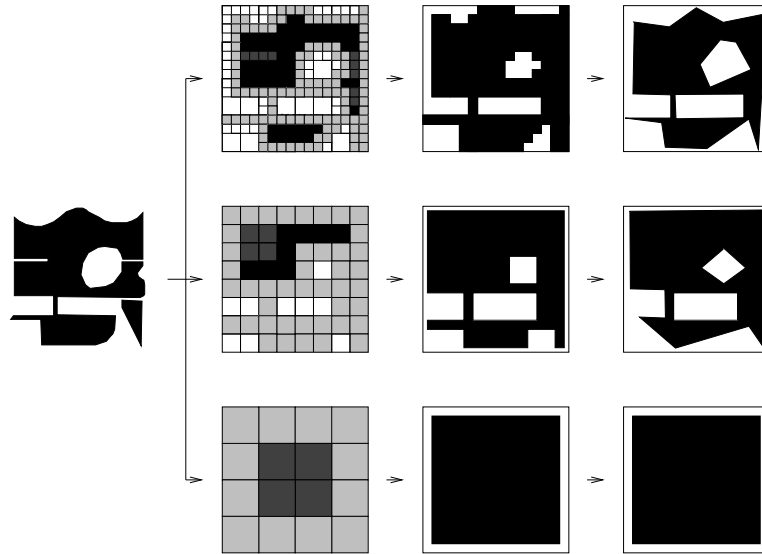


Figure 7: DPS method 2D example. From left to right: original polyhedron, multi-resolution of spatial decompositions, reconstructed models and final multi-resolution.

directly to the BRep representation, which frequently produce non-manifold boundaries [GH97].



## 2.2 Previous DPS methods

The DPS framework is both a generalization of previous simplification algorithms and an extension of isosurface extraction and surface fitting algorithms to the solid simplification field. The methods presented in [AAB95] and [AAB<sup>+</sup>96] follow the DPS pattern. In [AAB95] a method is presented which is intended for orthogonal surface simplification. The decomposition scheme is a colored MDCO. The metric used is the On-Hausdorff. The reconstruction step generates an orthogonal solid. No further face reduction is applied.

Its main contribution is the ability to create faces of arbitrary topology, even with holes. Unfortunately, the results are restricted to orthogonal solids and present the problem of the non-regular regions [And98b], i.e., regions composed by terminal grey nodes whose 8 vertices have the same color. These regions must be processed and reconstructed separately at the expense of increasing the number of shells.

In [AAB<sup>+</sup>96] an improved version is presented, producing arbitrarily oriented faces, but still has the non-regular regions problem.

The method presented in [HHK<sup>+</sup>95] is in fact a single-iteration DPS. The decomposition model is a voxel grid where the color of voxel's center (referred as volume buffer) is calculated by applying a low-pass filter over the object's volume in its neighborhood. The reconstruction is performed with a corrected MC in the first version, and with an enhanced, less verbose MC in [aa96].

### 3 The Direct DPS method

Direct DPS (Figure 6) is a solid simplification method following the DPS pattern (Figure 5). A maximal division classical octree (Section 3.1) is used as the decomposition scheme; the approximation error is measured using the In-Hausdorff distance (the extension to support the Out-Hausdorff distance is straightforward); the direct reconstruction is a region-based combination of an orthogonal reconstruction (Section 3.5) and an extended DMC (Section 3.6), and the face reduction is based on merging of adjacent faces (Section 3.8). In each iteration the octree is pruned removing the last level (Figure 6).

#### 3.1 Maximal Division Classical Octree

Some DPS methods are based on the voxel representation [HHK<sup>+</sup>95], [aa96] whereas others [AAB95], [AAB<sup>+</sup>96] use hierarchical octree representations. Octree representations can be seen as voxel representations with a hierarchy on top of them. However, octree representations are more flexible as they allow information being transmitted from the leaves to the parent nodes, as it will be discussed in next sections.

The Maximal Division Classical Octree [BJN<sup>+</sup>88], denoted as  $MDCO(P, l)$ , is an octree representation of  $P$ , containing White (W), Black (B), Grey (G) and terminal grey (TG) nodes with all TG nodes belonging to the last level  $l$ . White nodes correspond to cubic regions completely outside  $P$  and Black nodes correspond to cubic regions completely inside  $P$ . Grey nodes correspond to cubic regions containing part of the object boundary and therefore must be subdivided by bisecting each direction into eight octants. These octants are represented as the eight son nodes of the initial Grey node. Finally, TG nodes are Grey nodes at the deepest allowed level  $l$  of the tree and are not subdivided (see Figure 11 b).

The boundary of  $P$  is completely contained in the set of TG nodes. From now on we will refer to sets of nodes using calygraphic letters, i.e.  $\mathcal{B}$  is the set of B nodes, and so on.

There are several ways to define the cubic region  $\mathcal{C}$  associated to an octree node. The usual definition leads to a disjoint decomposition of the space, where a cell of size  $l$  centered in  $c_x, c_y, c_z$  includes the points defined by the three half-opened intervals given by the equation

$$\mathcal{C} = \{(x, y, z) \mid x \in [c_x - l, c_x + l), y \in [c_y - l, c_y + l), z \in [c_z - l, c_z + l)\} \quad (6)$$

where the closed endpoint of each interval is defined by convention. The border of this cube is closed in a vertex, three edges and three faces. This definition is used in octrees such as the Extended Octrees [Nav86], where entities such as vertices must belong to a unique cube. Such an MDCO will be referred as *Disjoint MDCO*.

Another way to define the cube is as follows:

$$\mathcal{C} = \{(x, y, z) \mid x \in [c_x - l, c_x + l], y \in [c_y - l, c_y + l], z \in [c_z - l, c_z + l]\}, \quad (7)$$

i.e., by means of three closed intervals. This definition leads to a non-disjoint decomposition, and we will show that it yields to a more flexible representation of the object's boundary since TG nodes with underlying connected regions are 6-connected. Such an MDCO will be referred as *quasi-disjoint MDCO*. Due to its convenient connectivity properties, the Direct MDCO DPS method is based on a this kind of MDCO.

One of the interesting properties of the MDCO scheme is presented in the following theorem:

**Theorem 3.1** *In a MDCO, a B node is never 26-adjacent to a W node.*

*Proof:* In a MDCO, the intersection of two 26-adjacent nodes is not empty (they share at least a vertex, edge or face). If a B node were 26-adjacent to a W node, this non-empty shared region would be, at the same time, completely in and out the polyhedron, so this circumstance can never occur. ■

### 3.2 Discretization: Border and Interior TG nodes

The discretization consists of the construction of the MDCO from a boundary representation. This is a well known problem based on a simultaneous space subdivision and clipping of the boundary of the polyhedron; the implementation details are given in Section 3.9.3. Depending on the selected error criterion, TG nodes can be further classified into border and interior nodes, as follows (see Figure 11 b, e, h):

**Definition** *A TG node is a border TG node ( $BTG^W$ ) if at least one of its 26-neighbor nodes is a W node.*

**Definition** *A TG node is an interior TG node ( $ITG^W$ ) if none of its 26-neighbor nodes is a W node.*

The corresponding definitions for  $BTG^B$  and  $ITG^B$  are obtained by simply exchanging the role of B and W nodes:

**Definition** *A TG node is a  $BTG^B$  node if at least one of its 26-neighbor nodes is a B node.*

**Definition** *A TG node is a  $ITG^B$  node if none of its 26-neighbor nodes is a B node.*

An important property that will be relevant for the reconstruction algorithm is that, given a polyhedron  $P$ , only  $BTG^W$  (resp.  $BTG^B$ ) nodes are relevant for the generation of an approximating polyhedron  $P_k$  fulfilling the In (resp. Out) Hausdorff distances with respect to  $P$ , respectively. In other words, by definition of the MDCO, the boundary of  $P$  spreads into border and interior TG nodes, but the boundary of  $P_k$  only needs to traverse the  $BTG$  nodes.

The TG nodes of a MDCO have the following connectivity property:

**Theorem 3.2** *Given a one-shell polyhedron, let  $BTG$  be the set of  $BTG^W$  nodes and  $ITG$  the set of  $ITG^W$  nodes of its MDCO representation. Then,  $BTG$  has a unique 6-connected component.*

*Proof:* The MDCO can be viewed as a (6,26) 3D digital picture  $\mathcal{P}$  whose black points are defined by the center of the B and TG nodes of the MDCO. The border points of the (6,26) digital picture are the black points that are 26-adjacent to one or more white points, so the border points of  $\mathcal{P}$  correspond exactly to the  $BTG^W$  nodes (the black points derived from B and  $ITG^W$  nodes are not 26-adjacent to any other white point). Since the border of a black component with respect to a white component in a (6,26) digital picture is 6-connected [KR89], the set  $BTG^W$  nodes is 6-connected. ■

Note that previous theorem is not valid for  $BTG^B$  nodes, and that the MDCO yields to a quasi-disjoint decomposition.

**Corollary 3.2.1** *Each maximally 26-connected subset of the set  $B \cup ITG^W$  is enclosed inside a 6-connected subset of  $BTG^W$  nodes.*

*Proof:* The 26-neighbors of B and  $ITG^W$  nodes cannot be W by Theorem 3.1 and definition of  $ITG^W$  nodes. ■

### 3.3 Error control

The three error metrics for general DPS methods are suitable for the Direct DPS. In fact, as we will show in Section 3.5.2, the associated algorithms have a full symmetric structure, and the implementation of the reconstruction and face reduction steps for one of such criteria can be trivially extended to generate polyhedra fulfilling the other two criteria. From now on the discussion is centered on the In-Hausdorff distance, which is a good solid simplification error metric, and the symmetric definitions of the algorithm steps for the other criteria are specified when required.

In order to generate a polyhedron  $PD$  from an MDCO  $O$  we define a set of conditions of  $PD$  with respect to  $O$  in order to guarantee some distance bound. This set of conditions is called a **feasibility**,  $\mathcal{F}$ .

Here we define several feasibilities, each of them leading to different solids:

**Definition** Feasibility is said to be homogeneous if it includes at least the following two rules: (a) cubic regions associated to the B nodes of  $O$  are completely inside  $PD$  and (b) cubic regions associated to W nodes are completely outside  $PD$ .

The following three feasibilities are useful as In, Out and On-metrics criteria, respectively:

**Definition** *The Solid Overflow Feasibility  $\mathcal{SOF}$  is an homogeneous feasibility that also includes these two rules: (a) cubic regions associated to the  $BTG^W$  nodes contain a part of the boundary of  $P$  and (b) cubic regions associated to the  $ITG^W$  nodes contain a part of the solid.*

It is called ‘Solid Overflow’ because, in some way, TG nodes are treated as B nodes and thus all TG nodes contain at least a part of the solid.

**Definition** *The Solid Underflow Feasibility  $\mathcal{SUF}$  is an homogeneous feasibility that includes these two rules: (a) cubic regions associated to the  $BTG^B$  nodes*

contain a part of the boundary of  $P$  and (b) cubic regions associated to the  $ITG^B$  nodes contain a part of the ‘background’.

It is called ‘Solid Underflow’ because, in some way, TG nodes are treated as W nodes and thus all TG nodes contain at least a part of the background.

Note that both the Overflow and Underflow feasibilities are weak in a sense that  $ITG$  nodes are not required to be completely inside/outside the solid though they can be. The immediate advantage of this definition against the strong version (requiring  $ITG^W$  nodes to lie completely inside the solid) is that the original polyhedron  $P$  satisfies the proposed feasibilities. This situation is illustrated in Figure 8.

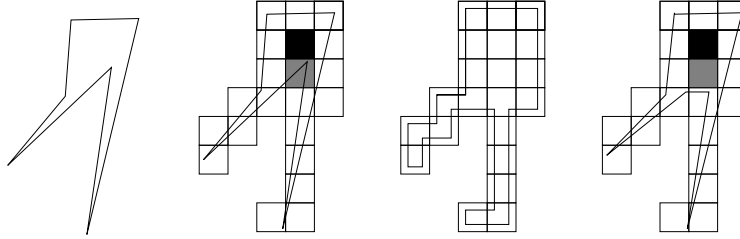


Figure 8: The three reconstructions on the right all satisfy the OOS feasibility, whereas only the two on the right would satisfy a strong OOS feasibility. In this case, the optimal reconstruction is the original polyhedron, and a strong feasibility enforcement will increase the number of faces.

**Definition** *The Solid Balance Feasibility  $SBF$  is an homogeneous feasibility that includes this rule: cubic regions associated to the TG nodes contain a part of the boundary, thus all TG nodes contain a part of the solid and a part of the background.*

**Definition** *Given a MDCO  $O$  and a feasibility  $\mathcal{F}$ , a polyhedral object  $P$  is said to be  $\mathcal{F}$  – feasible with respect to  $O$  iff  $P$  fulfills all the conditions of  $\mathcal{F}$ .*

A polyhedron  $P$  is said to be *In-Feasible* (resp. *Out-Feasible*) with respect to a MDCO iff  $P$  fulfills all the conditions of the SOF (resp. SUF) feasibility.

**Definition** *A feasibility  $\mathcal{F}$  is said to be compatible iff for any valid MDCO  $O$ , there exists a polyhedral object  $P$  so that it is  $\mathcal{F}$  – feasible with respect to  $O$ , i.e.,  $\forall P \exists P' \mid P$  is  $\mathcal{F}$ -feasible with respect to MDCO( $P$ ).*

Now, we show what feasibility should be used to guarantee each of the error bounds proposed in Section 2.1.2.

**Lemma 3.2.1** *If  $P_i$  is  $\mathcal{I} \setminus$  – feasible with respect to MDCO( $P, l$ ) then  $P_i$  satisfies the In-Hausdorff distance with respect to  $P$ , where  $\varepsilon$  is  $\frac{\sqrt{3}}{2l}$ , i.e. the length of the main diagonal of a terminal node of MDCO( $P, l$ ).*

*Proof:* In order to satisfy the In-Hausdorff bound with respect to  $P$ ,  $P_i$  must verify these requirements: (a) for every point  $Q$  in  $P$  there exists a point  $R$  in  $P_i$  such that  $dist(Q, R) < \varepsilon$ , and (b) for every point  $Q$  in  $P_i$  there exists

a point  $R$  in  $P$  such that  $dist(Q, R) < \varepsilon$ . Since  $P_i$  is  $\mathcal{I} \setminus$ -feasible,  $B$  nodes of  $MDCO(P, l)$  are inside  $P_i$ ;  $W$  nodes of  $MDCO(P, l)$  are outside  $P_i$ ,  $BTG^W$  nodes contain both in and out points of  $P_i$ , and  $ITG^W$  nodes contain in points of  $P_i$ . Every point of  $P$  belongs to a  $B$ ,  $BTG^W$  or  $ITG^W$  node of  $MDCO(P, l)$ , and thus there exists a point of  $P_i$  such that its distance is at most the main diagonal of a terminal node of  $MDCO(P, l)$ . Symmetrically, every point of  $P_i$  belongs to a  $B$ ,  $BTG^W$  or  $ITG^W$  node, and thus there exists a point of  $P$  such that its distance is at most  $\varepsilon$ , so  $P_i$  fulfills both conditions. ■

Symmetrically:

**Lemma 3.2.2** *If  $P_i$  is  $\mathcal{O} \sqcap \sqcup$ -feasible with respect to  $MDCO(P, l)$  then  $P_i$  satisfies the Out-Hausdorff bound with respect to  $P$ , where  $\varepsilon$  is defined as in Lemma 3.2.1.*

The proof is symmetric to that of the Lemma 3.2.1.

**Lemma 3.2.3** *If  $P_i$  is  $\mathcal{SBF}$ -feasible with respect to  $MDCO(P, l)$  then  $P_i$  satisfies the On-Hausdorff bound with respect to  $P$ , where  $\varepsilon$  is defined as in Lemma 3.2.1.*

*Proof:*  $P_i$ 's boundary traverses exactly the same set of nodes traversed by  $P$  (the TG nodes), and thus the Hausdorff distance is at most  $\varepsilon$ . ■

### 3.4 Parameterization of Orthogonal Reconstructions

In this section a parameterized discrete interpolation between the white and the black boundaries is discussed. The parameterization is based on the concept of discrete offsets.

The *White Surface*  $WS(O)$  is the cuberille surface that separates  $W$  nodes from the rest of nodes. Symmetrically, the *black surface*  $BS(O)$  separates  $B$  nodes from the rest. The following theorem shows the symmetry in the generation of both boundaries:

**Theorem 3.3**  $WS(O) = BS(\overline{O})$

*Proof:*  $WS(O) =$  faces shared by  $BTG^W$  and  $W$  nodes of  $O =$  faces shared by  $BTG^B$  and  $B$  nodes of  $\overline{O} = BS(\overline{O})$ . The middle equality is consequence of the black and white role exchange produced by the complement operation. ■

**Definition** *et*  $\delta$  be a fraction of the unit.  $\delta_0$ -offset(WS) is defined as WS; for  $i \geq 1$   $\delta_i$ -offset(WS) is the result of covering  $\delta_{i-1}$ -offset(WS) with sub-nodes of size  $\delta L$  that lie inside a TG node and are 26-adjacent to  $\delta_{i-1}$ -offset(WS).

Symmetrically, we can define the  $\delta$ -offsets of BS. The  $\delta$  parameter represents the thickness of the offset relative to the length  $L$  of TG nodes e.g.  $\delta = 1/4$  means that the thickness of the offset is a quarter of the length of TG nodes. Discrete offsets can be described using the Minkowski addition operator: the volume enclosed by  $\delta_i$ -offset(WS) is  $\mathcal{B} \cup (\mathcal{TG} - (\mathcal{W} \oplus \mathcal{O}_{\delta_i}))$  and the volume enclosed

by  $\delta_i$ -offset(BS) is  $\mathcal{B} \cup (\mathcal{TG} \cap (\mathcal{B} \oplus \mathcal{O}_{\delta_i}))$ , where ‘ $\oplus$ ’ is the Minkowski addition operator [Mat95] and  $\mathcal{O}_n$  is a cube of length  $2Ln$  centered at the origin,  $L$  being the length of TG nodes.

Note that for some integer  $m$ ,  $\delta_m$ -offset(WS)=BS and symmetrically, for some integer  $n$ ,  $\delta_n$ -offset(BS)=WS. The  $\delta$ -offsets from the WS and the BS are shown in Figure 9.

We can equally weight the contribution of the WS and the BS to obtain a more symmetric parameterization, introducing the constrained offsets:

**Definition** et  $\delta$  be a fraction of the unit.  $\delta_0$ -constrained offset(WS) is defined as WS; for  $i \geq 1$   $\delta_i$ -constrained offset(WS) is the result of covering  $\delta_{i-1}$ -offset(WS) with sub-nodes of size  $\delta L$  that lie inside a TG node, are 26-adjacent to  $\delta_{i-1}$ -offset(WS) and lie outside  $\delta_i$ -offset(BS).

The symmetrically we define the  $i_\delta$ -constrained offset of the BS.

The  $i_\delta$ -constrained offset(WS) and  $i_\delta$ -constrained offset(BS) are shown in Figure 9. Note that there exists some  $j$  such as beyond  $j$  the constrained offset is not modified, i.e.  $i_\delta$ -constrained offset(WS)= $j_\delta$ -constrained offset(WS) for  $i \geq j$ . Similarly we can define a limit value for the constrained offsets of the BS.

The constrained offsets define a sequence of parameterized reconstructions starting at WS and ending at BS, each of them with  $\delta$  increments: WS= $0_\delta$ -coffset(WS),  $1_\delta$ -coffset(WS), ...,  $j_\delta$ -coffset(WS),  $o_\delta$ -coffset(BS),  $(o-1)_\delta$ -coffset(BS), ...,  $0_\delta$ -coffset(BS)=BS.

This sequence is show in Figure 9.

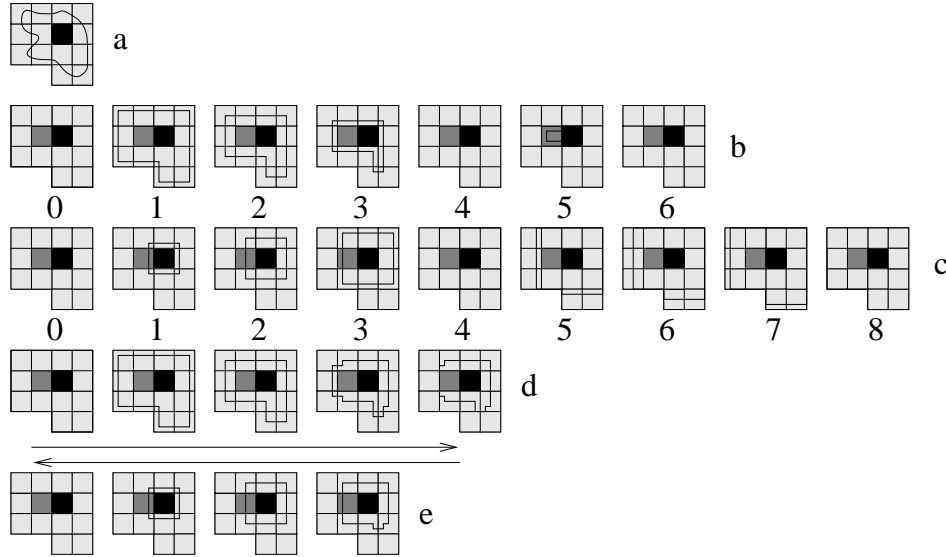


Figure 9: Parameterized reconstruction based on discrete offsets: a) Original polyhedron and its MDCO; b) WS and  $i_{1/4}$ -offsets of WS; c) BS and  $i_{1/4}$ -offsets of BS; d) WS and  $i_{1/4}$ -offsets of WS; e) BS and  $i_{1/4}$ -offsets of BS.

Symmetrically, negative offsets can be defined from the WS, providing the necessary framework for creating representations completely bounding the input solid.

### 3.5 The Orthogonal Reconstruction

Here we propose a Direct reconstruction which is based on node neighborhood information and which creates a two-manifold, orthogonal solid whose faces have arbitrary complexity and may have holes. This method is an improvement of the orthogonal reconstruction method presented in [AAB95]. Particularly, our proposal does not have the non-regular regions problem discussed in Section 1.2.3, which leads to disconnected regions even when the corresponding regions were connected in the original polyhedron, and which must be treated separately; our reconstruction method using the In-Hausdorff bound never increases the number of connected components of the solid.

The main theoretical concepts involved in the reconstruction step are presented below (see Figure 11 c, i):

#### 3.5.1 Definition of the orthogonal solids

**Definition** *iven a MDCO  $O$ , the associated Orthogonal Overflow Solid  $OOS(O)$  is defined as the solid enclosed by  $1/3_1$ -offset(WS).*

**Definition** *iven a MDCO  $O$ , the associated Orthogonal Underflow Solid  $OUS(O)$  is defined as the solid enclosed by  $1/3_1$ -offset(BS).*

#### 3.5.2 Construction of the orthogonal solids

**Definition** *The octant associated to a TG node can be partitioned, by means of 3 pairs of orthogonal planes, in 27 equal-sized sub-cubes, called small cubes.*

See Figure 10.

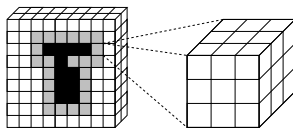


Figure 10: A BTG node with its 27 small cubes.

**Definition** *A small cube is said to be black if it is not 26-adjacent to a W node, i.e., its distance to the white boundary is greater than  $l/3$ , being  $l$  the length of TG nodes.*

**Definition** *A small cube is said to be white if it is 26-adjacent to a W node, i.e., its distance to the black boundary is less than  $l/3$ , being  $l$  the length of TG nodes.*



Now we define the OOS and OUS in terms of small cubes and other nodes of the MDCO:

**Definition** Given a MDCO  $O$ , the associated Orthogonal Overflow Solid  $OOS(O)$  can be computed as the closed union of the cubic regions associated to (a)  $B$  nodes, (b)  $ITG^W$  nodes and (c) the black small cubes of  $BTG^W$  nodes.

**Definition** Given a MDCO  $O$ , the associated Orthogonal Underflow Solid  $OUS(O)$  can be computed as the closed union of the cubic regions associated to (a)  $B$  nodes and (b) the white small cubes of  $BTG^B$  nodes.

Although On-metrics are not as useful as In-metrics, in order to be exhaustive we include the Orthogonal Balanced Solid in our discussion: the  $OBS(O)$  is the solid generated by the algorithm presented in [AAB95] using the white-proximity criterion.

A simple 2D example illustrating these concepts is shown in Figure 11.

### 3.5.3 Properties of OOS and OUS solids

Following we list some properties related to the OOS and OUS:

#### Symmetry of OOS and OUS

The following theorem justifies the parallel treatment of the In- and Out- volume versions in our discussion, since the implementation of the Direct MDCO for any of them can be trivially adapted to generate polyhedra for the other criterion, by simply inverting the B and W nodes of the MDCO:

**Theorem 3.4**  $OOS(MDCO(P, l)) = \overline{OUS(\overline{MDCO(P, l)})}$

*Proof:* The proof is similar to that of the symmetry of WS with respect to BS. ■

**Corollary 3.4.1**  $OUS(MDCO(P, l)) = \overline{OOS(\overline{MDCO(P, l)})}$

#### Containing relationships

The following theorem establishes the containing relationship between the volume of each orthogonal solid:

**Theorem 3.5**  $B \subset OUS(MDCO(P, l)) \subseteq OBS(MDCO(P, l)) \subseteq OOS(MDCO(P, l)) \subset B \cup TG$

Obviously,  $B \subseteq P \subseteq B \cup TG$ .

#### Geometric properties

The boundary of each type of orthogonal solid is confined to different sets of TG nodes.  $Bound(X)$  denotes the boundary of solid  $X$ .

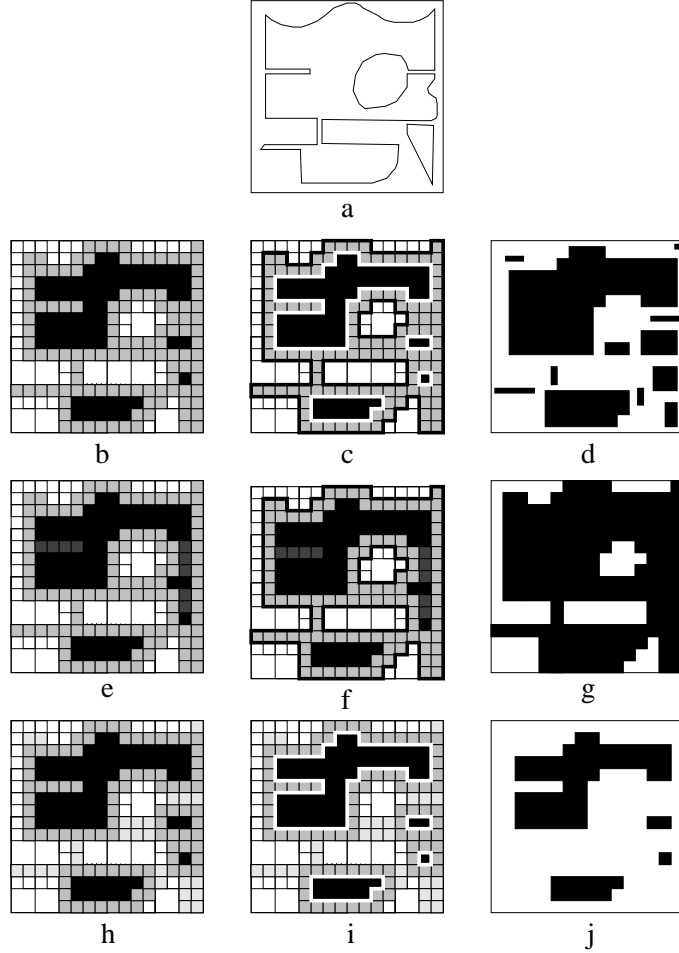


Figure 11: (a) An initial 2D polyhedron with two shells; (b) MDCO with TG nodes shaded; (c) WS and BS; (d) the OBS (e) octree with B, W,  $BTG^W$  nodes (light) and  $ITG^W$  nodes (dark); (f) idem with  $Bound^W$  highlighted; (g) the OOS (h) octree with B, W,  $BTG^B$  nodes (dark) and  $ITG^B$  nodes (light); (i) idem with  $Bound^B$  highlighted; (j) the OUS.

**Theorem 3.6**  $Nodes(Bound(OOS(O))) = BTG^W(O)$

**Theorem 3.7**  $Nodes(Bound(OUS(O))) = BTG^B(O)$

**Theorem 3.8**  $Nodes(Bound(OBS(O))) = TG(O)$

*Proof:* It comes directly by definition of OOS, OUS and OBS. ■

**Corollary 3.8.1** *The area of  $Bound(OOS(O))$  and  $Bound(OUS(O))$  are lower than that of  $Bound(OBS(O))$ .*

The following theorem is fundamental because it bounds the discrepancies between proposed solids:

**Corollary 3.8.2**  $Nodes(Bound(OOS(O) \text{ xor } OUS(O))) = ITG^W \cup ITG^B \subset \mathcal{TG}$

That is, the nodes containing the symmetric difference are restricted to a precise set of  $ITG$  nodes. In fact, the non-regular nodes referenced in [AAB95] are a subset of  $ITG^W \cup ITG^B$ .

Since we are interested in solid simplification, the Direct MDCO uses a OOS reconstruction. The OOS and OUS have a very interesting property which is the basis of the further face reduction:

**Theorem 3.9** *All OOS vertices have degree three or six.*

Figure 12 shows the only five possible vertex configurations of an OOS, which are resumed in the table below:

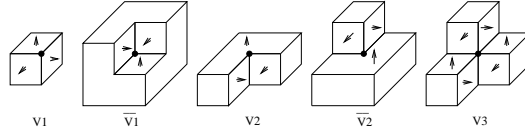


Figure 12: The five vertex configurations of an OOS. From left to right:  $V1$ ,  $\overline{V1}$ ,  $V2$ ,  $\overline{V2}$  and  $V3$ .

Vertex Type	Degree	Edge Configuration (X=convex, V=concave)
$V1$	3	X, X, X
$\overline{V1}$	3	V, V, V
$V2$	3	X, V, X
$\overline{V2}$	3	V, X, V
$V3$	6	X, V, X, V, X, V

Note that the first two configurations are the complement one of each other, and the same applies for the second couple of vertex configurations.

### Minimum Degree Transformation

As noted above, all OOS vertices have degree three except the  $V3$  vertex, which has degree six. However, the  $V3$  vertex can be easily splitted into two vertices, resulting in a  $V2$  and a  $\overline{V2}$  pair, as shown in Figure 13, where the new  $\overline{V2}$  vertex has been moved along one of the concave edges of  $V3$ .

**Definition** minimum degree solid is a polyhedral solid whose vertices have degree three (i.e. three incident faces).

If all the  $V3$  vertices of the OOS are splitted, the resulting solid is a minimum degree solid (see Figure 14). There are three basic properties of minimum degree solids which will be key ingredients in the face reduction step (see Figure 14):

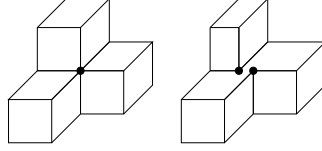


Figure 13: Splitting of a  $V3$  vertex into a  $V2$  and a  $\overline{V2}$ .

**Theorem 3.10** *Given a minimum degree solid  $P$ , if two faces  $f_1$  and  $f_2$  of  $P$  are incident to the same vertex, then  $f_1$  and  $f_2$  must be adjacent, i.e. they share a common edge.*

**Theorem 3.11** *Given a two-manifold minimum degree solid  $P$ , if two faces  $f_1$  and  $f_2$  of  $P$  are adjacent to a common edge  $(v_1, v_2)$ , then there exists exactly two faces  $f_{v_1}$  and  $f_{v_2}$  in  $P$  such that both are adjacent to both  $f_1$  and  $f_2$ ,  $f_{v_1}$  incides to  $v_1$  and  $f_{v_2}$  incides to  $v_2$ .*

**Theorem 3.12** *The dual graph of a minimum degree solid is the graph of a triangulation.*

### Properties of the boundary of OOS

The following series of properties are concerned on distance bounds between different parts of the boundary of the OOS, and provide the basis for simplified self-intersection tests which are required after reduction operations (see Figure 15 a, b):

**Theorem 3.13** *Given two 6-adjacent  $BTG^W$  nodes sharing a node face  $f$ , the boundary of the OOS stabs both nodes, and the interior of the OOS stabs  $f$ .*

*Proof:* The boundary of the OOS stabs all  $BTG^W$  nodes by definition. The shared face  $f$  does not belong to the  $Bound^W$ , and hence at least two small cubes on both sides of  $f$  must be black. ■

As a result of the previous theorem, several orthogonal structures cannot appear in an OOS (see Figure 15 c).

**Theorem 3.14** *Given a face  $f$  of the OOS, the nearest face in the direction of the normal vector of  $f$  is at least at  $\frac{5}{3}l$  and the nearest face in the opposite direction is at least at  $\frac{1}{3}l$ ,  $l$  being the edge's length of  $BTG$  nodes.*

**Corollary 3.14.1** *Given a face  $f$  of the OOS, for all concave sequence  $\{e_i\}$  of edges of  $f$ ,  $dist(e_i, e_j) \geq \frac{5}{3}l$  provided that with  $|i - j| > 1$ . For convex sequences,  $dist(e_i, e_j) \geq \frac{1}{3}l$  for  $|i - j| > 1$ .*

See Figure 15 d-h for examples of valid and invalid OOS.

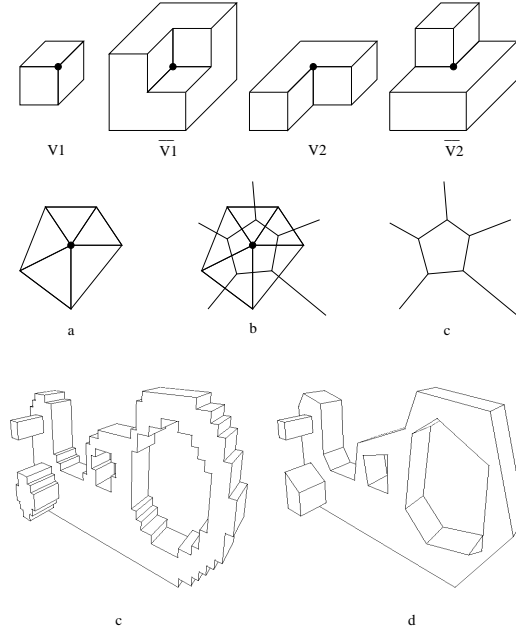


Figure 14: Minimum degree solids. Top: The four vertex configurations of minimum degree solids (faces do not need to be orthogonal). Middle: a) A vertex of degree greater than three, which do not satisfy Theorems 3.10– 3.12; b) The same vertex with its dual model; c) The dual model of a non-minimum degree solid is not a triangle mesh. Bottom: d) An orthogonal example of minimum degree solid; e) A non-orthogonal example of minimum degree solid.

### Ensuring error bounds

**Theorem 3.15**  $OOS(O)$  is In-feasible with respect to  $O$ .

**Theorem 3.16**  $OUS(O)$  is Out-feasible with respect to  $O$  iff  $O$  contains at least a  $B$  node.

**Theorem 3.17**  $OBS(O)$  is On-feasible with respect to  $O$ .

**Corollary 3.17.1**  $OOF$ ,  $OUF$  and  $OBF$  are compatible.

Note: The OUS of some MDCO may be a null, i.e. volume 0, solid when  $card(\mathcal{B}) = 0$ .

### 3.6 Discretized Unambiguous Marching Cubes

Some simplification methods based on MC [LC87] use vertex colors to extract a polyhedral surface from a 3D picture. Our approach uses the 26-neighborhood of TG nodes instead of vertex colors. Meanwhile, vertex color information can

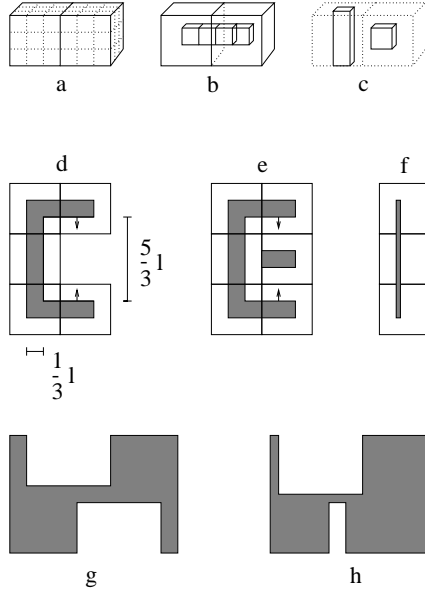


Figure 15: Geometric properties of OOS: a) 6-adjacent *BTG* nodes, with small cubes in dotted lines; b) minimum number of black small cubes in two 6-adjacent *BTG* nodes; c) impossible structure in an OOS; d) the OOS surface in the normal direction of a face is at least at  $\frac{5}{3}l$  distance, and in the opposite direction, at least at  $\frac{1}{3}l$ ; e) and f) impossible OOS; g) another example of valid face for an OOS; h) another example of invalid face for an OOS.

be derived from the MDCO such that the reconstruction of the resulting 3D digital picture is topologically equivalent to an OOS:

**Theorem 3.18** *Given a MDCO  $O$ , there exists a 3D digital picture  $\mathcal{P}$  such that the extraction of an isosurface from  $\mathcal{P}$  is non-ambiguous and generates a surface  $S$  topologically equivalent to  $OOS(O)$  enclosing an In-feasible volume, provided that the extraction guarantees that all stabbed edges, and only them, are intersected by  $S$ .*

*Proof:* The proof is based on a constructive method to obtain the 3D digital picture  $\mathcal{P}$  from the MDCO  $O$ , called *induced grid*. The grid points of  $\mathcal{P}$  are defined as the corners of the octants resulting by subdividing once the terminal nodes of  $O$ . A grid point  $v$  of  $\mathcal{P}$  is white if  $v$  has contact with a W node of  $O$ ; otherwise it is black. If neighbor grid points of  $\mathcal{P}$  are arranged into voxels, as usual in isosurface extraction, the only possible configurations are the planar, non-ambiguous classes [And98a], so extraction methods such as MC generate a closed surface. It is straightforward to see that this surface  $S$  is topologically equivalent to  $OOS(O)$ , and that limits an In-feasible solid.

The DMC [MSS94] is an extension of MC based on a discretization of the planes in 13 different orientations, so coplanar triangles can be merged forming large faces. Given a MDCO  $O$ , the Unambiguous Discretized Marching Cubes solid,

UDMC, is defined as the polyhedron generated by the DMC algorithm from the grid induced by  $O$ . The UDMC only uses the non-ambiguous, planar classes of DMC and creates In-feasible, valid solids. The only possible configurations of voxels in  $\mathcal{P}$  are those numbered 1,  $\bar{1}$ , 2,  $\bar{2}$ , 5,  $\bar{5}$ , 8 and 9 in Figure 4 (plus rotations and symmetries).

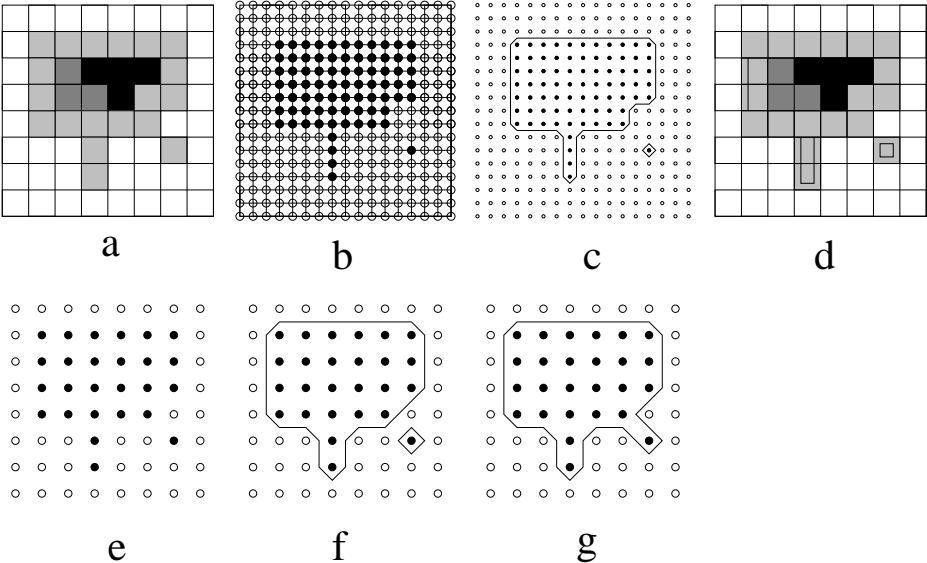


Figure 16: Orthogonal reconstruction vs. Marching Cubes reconstructions: a) MDCO with B, W, BGT and ITG nodes; b) non-ambiguous 3D digital picture obtained by subdivision; c) reconstruction obtained by MC-like from picture b; d) orthogonal reconstruction from the MDCO; e) ambiguous 3D digital picture obtained by identifying grid points with center of MDCO nodes; f) and g) two possible reconstructions from picture e.

An example of the digital picture defined above is shown in Figure 16, where several reconstructions are compared: the orthogonal reconstruction OOS obtained from the MDCO, a MC-like reconstruction from the non-ambiguous digital picture constructed by the method presented above, and the two possible interpretations obtained with a MC-like reconstruction from the ambiguous digital picture obtained by simply generating the grid points from the center of MDCO terminal nodes.

### 3.7 Direct Solid

Direct DPS uses a mixed reconstruction combining the OOS and the UDMC in a shell-basis producing the *Direct Solid*. Reconstruction selections takes into account the final number of vertices, which can be computed by inspection of the otree and its induced digital picture, respectively. The Direct Solid keeps the main properties of OOS and UDMC: it is two-manifold and In-feasible. Regions coming from the OOS have the minimum degree property.

## 3.8 Face reduction

The direct solid obtained after the reconstruction step has a simpler topology but many planar regions have an staircase-like reconstruction, so further simplification should be done. The basis of the face reduction step is based on the low degree of the OOS vertices and the application of a very simple reduction operator which always reduces the face count and never increases the degree of the vertices.

### 3.8.1 Mesh Simplification in the Dual Model

As stated above, one of the properties of the OOS is that it can be easily converted to a minimum degree solid by splitting all its  $V3$  vertices. The interest for a 3-degree solid comes from Theorem 3.12: the dual model [Man88] of a minimum degree solid is a triangular mesh. Thus we can take profit of the extense literature on mesh simplification, specially the different operators over triangle meshes, in order to find a suitable operator for the face reduction of minimum degree solids.

Most of the surface simplification methods deal with triangulated surfaces because of the simplicity of its incidence relations [And98b]. The first outcome of this fact is that reduction operators become simpler. For instance, we can modify the coordinates of a vertex in a triangle mesh without producing a non-planar face, fact that does not hold on non-triangular models. The dual equivalent of vertex displacements is the modification of face orientations. Since all the vertices of a minimum degree solid have degree three, the positions of the vertices of the modified face can be calculated by intersecting their three incident planes. Note that three non-parallel planes always intersect at a point, but the intersection of more than three non-parallel planes is usually empty.

Figure 17 shows the topology-preserving mesh operators discussed in the Section 1.2.2 and their effect in the dual model. Note that the vertex-clustering is not included because it does not preserve topology. A very important property is that, since all mesh operators are closed (the result is still a triangle mesh), their duals preserve the degree of vertices. The description of the duals of the mesh operators follows:

- *face-merge*

The face merge is the dual of the vertex-removal and the edge-collapse operators. The face merge takes as input parameters the two adjacent faces to be merged. Both faces are replaced by a single face, losing the shared edge and its shared vertices. The only computed parameter is the plane for the new face, which in fact determines the position of its vertices by intersecting its incident faces. The operator reduces the number of faces by one, the number of edges by three and the number of vertices by two. Except in degenerate cases which can be easily identified by inspection of the incidence graph, the face-merge operator preserves the topology of the mesh. To avoid self-intersections, additional geometric tests are required. The face-merge has been extensively used with coplanar faces in surface simplification literature [KCHN91], [MSS94].



- *vertex-removal (multiple face-merge)*

The multiple face-merge is the dual of the face-removal, and it can be replaced by a sequence of two face-merge. The multiple face-merge takes as input parameters the vertex to be removed (or, equivalently, the three faces to be merged). The three faces are replaced by a single face, losing their shared edges and their shared vertex. The only computed parameter is the new plane, which in fact determines the position of its vertices by intersecting its incident faces. The operator reduces the number of faces by two, the number of edges by six and the number of vertices by four. Except in degenerate cases which can be easily identified by inspection of the incidence graph, the multiple face-merge operator preserves the topology of the mesh. To avoid self-intersections, additional geometric tests are required.

- *cycle flip*

The cycle flip is the dual of the edge-flip. The cycle flip takes as input parameters the edge to be flipped. Given the cycle of four faces defined by the edge, the edge is removed and a new edge is introduced so that the pair of adjacent faces is the opposite pair (see Figure 17 d). There are no computed parameters. The operator preserves the number of faces, edges and vertices. To avoid self-intersections, additional geometric tests are required.

- *face-adjustment*

The face-adjustment is the dual of the vertex-displacement. It takes as parameter the face to be adjusted. The only computed parameter is the new plane equation. As a result of the modification of the support plane, the vertices of the face must be recalculated by intersecting its incident faces. Any of the parameters of the plane ( $A, B, C, D$ ) can be changed. Note that this vertex re-calculation can be done only in a minimum-degree solid; if more than three faces incide on a vertex, their intersection will be probably null.

The question that arises now is which of these mesh operators is more suitable for reducing the face count of our non-triangular polyhedron. We are looking for a reduction operator  $\rho$  having these properties: a)  $\rho$  must preserve the topology b)  $\rho$  must be easy to implement; c) the iterative application of  $\rho$  must reduce the face count as much as possible; and d)  $\rho$  must modify a low number of faces at a time, since each re-oriented face can intersect a non-*BTG* node and invalidate the operation.

We have realized that the face-merge, which is the dual of the most popular mesh simplification operators, vertex-removal and edge-collapse [RR96], [Gue96], [AS96], [Hop96], [Hop97], fulfills the previous criteria and can be easily implemented on minimum degree solids. The face-merge operator (see Figure 18) replaces two faces sharing an edge with a single face. In the operation, the number of faces and edges are decreased by one, and the number of vertices is decreased by two. The edge-collapse operator preserves the mesh topology, and so does its dual, and it does not modify the degree of the involved vertices.

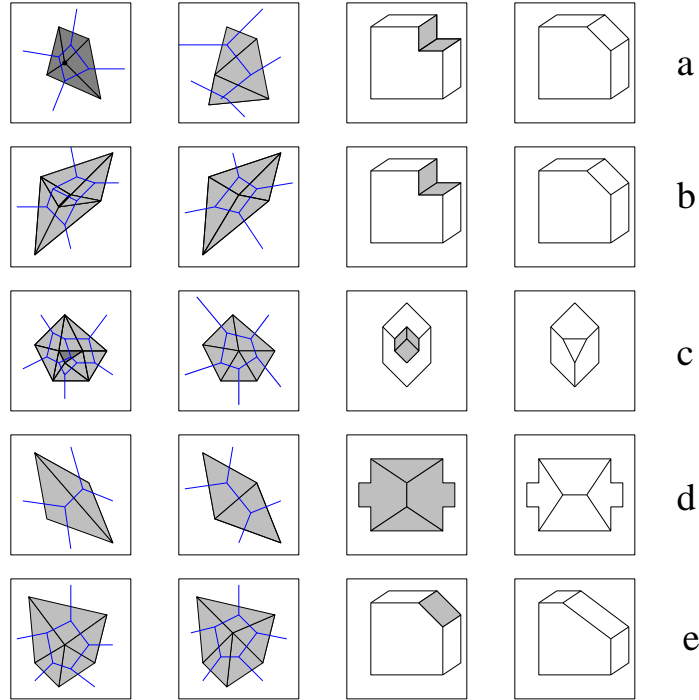


Figure 17: Dual operators. First column: triangle mesh and dual model before operator; second column: triangle mesh and dual model after operator; third and fourth column: example of the dual operator in a minimum degree solid. a) vertex removal, b) edge-collapse, c) face-removal, d) edge-flip and e) vertex-displacement.

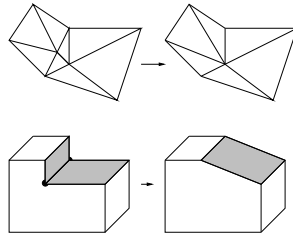


Figure 18: Edge-collapse and its dual, face merge.

The face-removal is equivalent to two face-merge operators so it does not add reduction power; however the face-removal involves three faces at a time, and hence it decreases the chances for a valid replacement, so we do not use directly this operator in our face reduction step.

The dual of the edge-flip does not reduce the face count, but can improve the fitting of the solid being simplified with respect to the OOS. However, in the context of our DPS method, a closer fitting to the OOS does not imply a closer fitting to the original polyhedron, and using the cycle-flip to improve the fitting

with respect to the original solid is both complicated and inefficient. Hence, the cycle-flip operator is not considered in our face reduction. However, the other incidence-preserving operator, the face-adjustment, can be used in conjunction with the face-merge to find out a new plane generating a feasible polyhedron.

### 3.8.2 Face reduction based on face-merge operator

The face reduction step consists in the iterative application of the face-merge operator, starting from the direct solid and preserving both the minimum degree and the In-feasibility of the polyhedron being simplified. This operation is repeated until no more faces can be merged without violating the In-feasibility. Thus the resulting polyhedron, which has a 2-manifold boundary topologically equivalent to the direct solid but with fewer faces, satisfies the In-Hausdorff bound within an  $\varepsilon$  equal to the length of the main diagonal of the TG nodes of the MDCO.

The face reduction algorithm is outlined below:

```

procedure FaceReduction( $P$ : Polyhedron)
  while there faces to be merged do
     $(f_1, f_2) := \text{select\_two\_faces}(P)$ 
     $(a, b, c, d) := \text{compute\_new\_plane}(P, f_1, f_2)$ 
    if feasible then merge_faces( $P, f_1, f_2, a, b, c, d$ )
  end

```

The face-merge operator, which is graphically shown in Figure 18, take as parameter the pair of faces being merged. From now on the following notation will be used (see Figure 19). The faces being merged are referred as  $f_1$  and  $f_2$ . Let  $e$  be the common edge, and  $v_1$  and  $v_2$  its extreme vertices. Let  $V$  be the set of vertices of  $f_1$  and  $f_2$ , excluding  $v_1$  and  $v_2$ . Let  $f_3$  be the third face containing  $v_1$  (it is unique since all vertices have degree 3) and  $f_4$  is the third face containing  $v_2$ . Let  $F$  be the set of faces inciding in some vertex of  $V$ , excluding  $f_1$  and  $f_2$ .

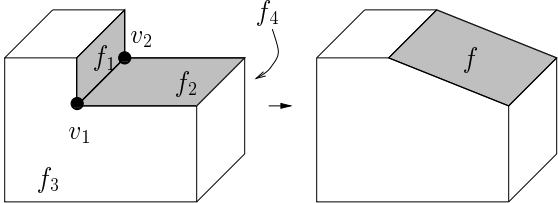


Figure 19: Face merge notation.

#### First step: selection of faces

This function selects a pair of neighbor faces to be merged. At this stage, there are a few quick tests that can prune some face pairs which would not create a feasible solid. Given a pair of faces  $f_1, f_2$ , the following conditions are checked:

$$adj(f_1, f_2) \tag{8}$$

$$nvert(f_1) + nvert(f_2) \geq 7 \tag{9}$$

$$nvert(f_3) \geq 4 \text{ and}$$

$$nvert(f_4) \geq 4 \tag{10}$$

$$L3 \not\subseteq Nodes(f_1) \cup Nodes(f_2) \tag{11}$$

$$O2 \not\subseteq Nodes(f_1) \cup Nodes(f_2) \tag{12}$$

Conditions 8–10 are inherent incidence constraints to the face merge operator in order to preserve validity. Equation 8 says that  $f_1$  and  $f_2$  must share exactly one edge. Equation 9 guarantees that the new face will have at least three vertices (seven minus two times the two vertices that are removed). Equation 10 guarantees that faces  $f_3$  and  $f_4$  will have at least three vertices.

Conditions 11 and 12 are necessary but not sufficient for a successful face merge which are explained below. A subset  $S$  of TG nodes is linearly separable iff there exists a plane intersecting the interior of all nodes in  $S$ . If such a plane does not exist, it is impossible to create a face stabbing all nodes in  $S$ , and hence the faces inside  $S$  cannot be merged together maintaining the feasibility of the solid.

The  $L3$  and  $O2$  are two common linearly non-separable configurations (see Figure 20). If a set of TG nodes contains one of such configurations, then it is linearly non-separable. Since both  $Nodes(f_1)$  and  $Nodes(f_2)$  must be linearly separable, any linearly non-separable subset of  $Nodes(f_1) \cup Nodes(f_2)$  must contain at least one node in  $Nodes(f_1) \cap Nodes(f_2)$ , which is a superset of  $Nodes(e)$ . Hence, the presence of  $L3$  and  $O2$  configurations in  $Nodes(f_1) \cup Nodes(f_2)$  can be determined walking along  $Nodes(f_1) \cap Nodes(f_2)$  nodes (Equations 11 and 12).

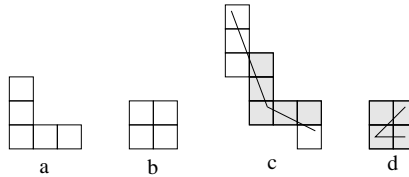


Figure 20: Linearly non-separable regions: a)  $L3$ ; b)  $O2$ ; c) linearly non-separable set containing an  $L3$  subset; d) linearly non-separable set containing an  $O2$  subset.

### Second step: computing the new plane

The new plane of  $f$  will determine not only the orientation and position of the new face, but also the coordinates of all vertices in  $V$ . There are several strategies to compute the new plane, which are shown in Figure 21. Each strategy is based in one of these point sets:

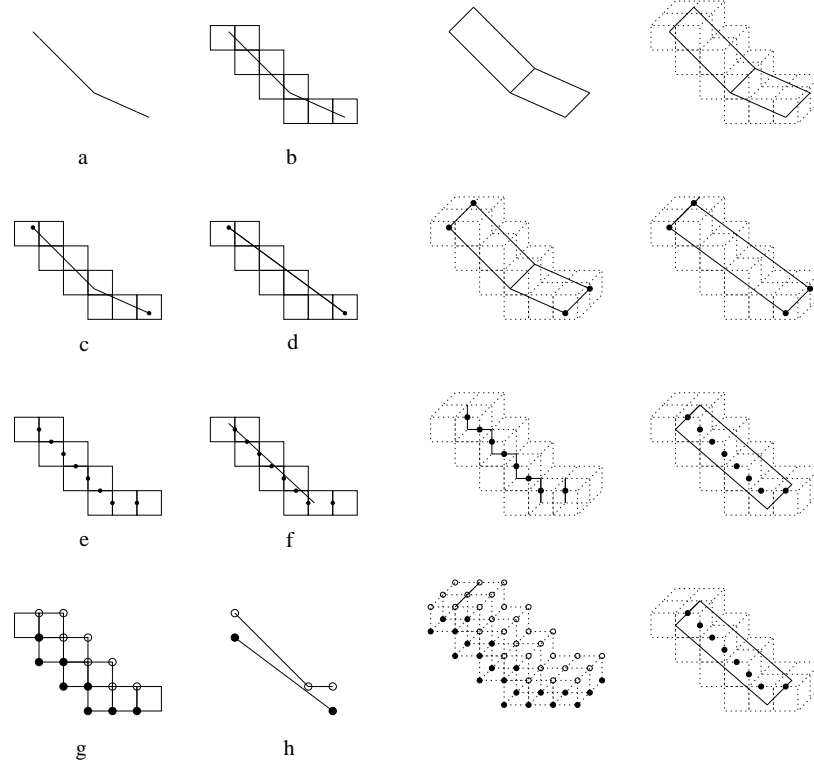


Figure 21: Computing new plane for a face-merge. On the left: 2D examples; on the right, the same examples in 3D. a) faces to be merged; b)  $Nodes(f_1) \cup Nodes(f_2)$ ; c) points in  $V$ ; d) plane calculated by regression of  $V$  points; e) middle point of stabbed edges; f) plane calculated by regression of middle point of stabbed edges; g) positive and negative points; h) maximal points of convex hull of negative points and minimal points of convex hull of positive points; since both convex hulls do not intersect, positive and negative points are linearly separable.

- *Polyhedron vertices in  $V$*

As defined above,  $V$  is the set of vertices of  $f_1$  and  $f_2$ , excluding  $v_1$  and  $v_2$ , and hence they provide a good starting point for the estimation of the new plane.

- *Middle point of stabbed edges*

Stabbed edges are edges of nodes in  $Nodes(f_1) \cup Nodes(f_2)$  such that their four incident nodes belong to  $Nodes(f_1) \cup Nodes(f_2)$ , so they must be stabbed by the new plane and hence they can be used to compute the new plane.

- *Positive and Negative points*

Positive points are corners of the stabbed edges in  $Nodes(f_1) \cup Nodes(f_2)$  such that they are outside the polyhedron being simplified. Symmetrically,

negative points are the corners inside the polyhedron. In order to be feasible, to plane of the new face must separate the positive from the negative points. That is the well know problem of separability of two sets of points, in this case on a grid. One well-known result in this field is shown in the following theorem [SW70].

**Theorem 3.19** *Two sets of points are linearly separable iff their convex hulls do not intersect.*

Both the set of vertices in  $V$  and the middle points of stabbed edges admit an average plane or a regression approximation. The last strategy, based on separability of positive and negative points, uses a linear programming method. The separability approach is the only one that always returns a separating plane if such a plane exists. The computation of the separability plane is a linear programming problem involving four unknowns (those of the equation of the plane) and at most  $n$  linear constraints,  $n$  being the number of positive and negative points. The number of constraints can be greatly reduced by pre-computing and storing the convex hulls of the positive and negative points. This separating plane can be computed in optimal  $O(n)$  time using the linear programming technique presented in [Meg83].

None of the previous strategies for the new plane computation, including the separability approach, guarantees a feasible replacement, since although the plane may stab all nodes, the nodes intersected by the actual face after computing its edges might be different to  $Nodes(f_1) \cup Nodes(f_2)$ . However, it is possible to represent these additional conditions as a set of linear constraints. There are four unknowns corresponding to the four coefficients of the plane,  $(A, B, C, D)$ . To guarantee that the plane separates the positive and negative points, there must be a restriction with the form

$$x_i A + y_i B + z_i C + D \geq 0 \quad (13)$$

for each positive point  $(x_i, y_i, z_i)$  and a restriction

$$x_i A + y_i B + z_i C + D \leq 0 \quad (14)$$

for each negative point. To guarantee that the edges of  $f$  are contained in edge nodes, there must be permutations of:

$$(-B' y_0 - C' z_0 - D') A + (A' y_0) B + (A' z_0) C + A' D \geq 0 \quad (15)$$

where  $(A', B', C', D')$  is the equation of the plane of an adjacent face  $f'$  and  $(x_0, y_0, z_0)$  is the origin of a node in  $Nodes(f) \cap Nodes(f')$ . To guarantee that the vertices of  $f$  lie in the same nodes than the vertices of  $f_1$  and  $f_2$ , there must be permutations of:

$$(x_0 v_x - x_0 - x_{min} v_x) A + (x_0 v_y - x_0 - x_{min} v_y) B + (x_0 v_z - x_0 - x_{min} v_z) C - D \geq 0 \quad (16)$$

where for each edge containing a vertex in  $V$ ,  $(x_0, y_0, z_0)$  is a point of the support line of the edge,  $(v_x, v_y, v_z)$  is a vector in the direction of that line and  $x_{min}$  is the x-coordinate of the origin of the node containing the vertex at the intersection

of  $f$  with that edge. Similarly, we can define restrictions for  $x_{max}$ ,  $y_{min}$ ,  $y_{max}$ ,  $z_{min}$ ,  $z_{max}$ . This linear programming problem can be solved in optimum  $O(n)$  time [PM85] using method proposed in [Meg83]. Unfortunately, the number of constraints is  $O(n)$  where  $n$  is the number of nodes in  $Nodes(f)$ .

We have found that computing the normal vector of the plane as the average of points in  $V$  and computing  $D$  with a linear programming method is quite fast and generates very good results, and hence this is the approach adopted in our implementation.

### Third step: checking feasibility

Before committing the operation, several simple tests should be performed in order to guarantee the OOS-feasibility preservation.

The edges of the new face are generated by merging the cyclically ordered vertices in  $V$ . The coordinates of vertices in  $V$  are calculated by intersecting its three incident faces. If these planes do not intersect in a point, or if the intersection point does not belong to a TG node, the face merge is not feasible.

In order to guarantee that the new solid is In-feasible, the face merge must fulfill these two conditions: a)  $Nodes(f_1) \cup Nodes(f_2) = Nodes(f)$ , i.e. the new face is contained and intersects exactly the same set of nodes than the merged faces, and b) the resulting solid has a non-selfintersecting boundary, i.e., the faces in  $F \cup \{f\}$  –those whose vertices have changed– do not intersect other faces in  $Nodes(f)$ .

The former condition can be easily checked by a rasterization of  $f$ , but there is no need to evaluate it when any of the following conditions is true: a) the plane of  $f$  has been computed by the linear programming approach described in the previous section; b) the distance of the new plane to the vertices  $v_1$  and  $v_2$  is greater than  $\varepsilon$ .

The latter condition can be checked looking for self-intersections of  $f$  with the others faces. Note that, although more faces have also changed their geometry (those in  $F$ ), their vertices still lie in the original plane, so any self-intersection of a face in  $F$  must be caused by their modified edges, which also belong to  $f$ , so testing self-intersection for  $f$  is sufficient. Only the faces intersecting  $Nodes(f)$  can intersect  $f$ , so the self-intersection test simply looks for intersections between edges in  $Nodes(f)$  and  $f$ .

### Fourth step: merging faces

The incidence changes of a face merge in the BRep model are described below:

- The faces  $f_1$  and  $f_2$  are removed.
- The edge  $e$  is removed. Since the boundary is two-manifold, each edge is shared by exactly two faces.
- The vertices  $v_1$  and  $v_2$  are deleted, since they have degree three before operator's application.

- A new face  $f$  is introduced, whose plane was calculated in the previous step. Its edges and vertices are computed as described in the previous sections.
- The faces  $f_3$  and  $f_4$  lose one vertex.

Note that the vertex coordinates recalculation exploits the minimum degree property of the solid. The  $V3$  split operation is performed only when needed, i.e., when one or more vertices in  $V \cup \{v_1, v_2\}$  are  $V3$ .

A more complicated face-merge is shown in Figure 22 which, depending on the underlying TG node size, would not be feasible.

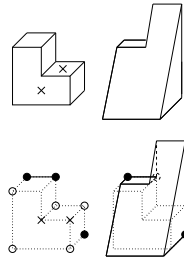


Figure 22: Top left: polyhedron before face-merge of stabbed faces; bottom left: unaffected vertices (black circle), vertices to be recalculated (white) and vertices to be removed (cross); right: polyhedron after face-merge.

The step by step simplification of an OOS including  $V1$ ,  $V2$  and  $V3$  vertices into a tetrahedron is shown in Figure 23. In Figure 24, a single face is derived from a collection of  $V3$  vertices.

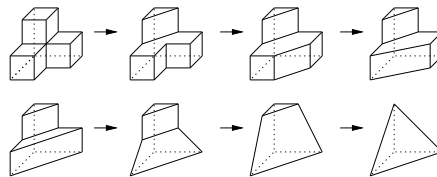


Figure 23: Face-merge Simplification of a OOS with  $V1$ ,  $V2$  and  $V3$  vertices into a tetrahedron.

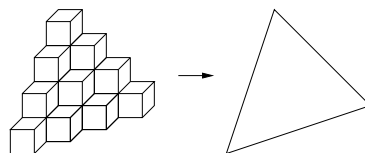


Figure 24: Simplification of a large  $V3$  region.



### 3.8.3 Extended SOF-feasibility

The Direct DPS uses the octree mainly for two purposes: in the reconstruction step, the octree provides the final topology of the simplified solid and an initial geometric description; in the face reduction step, the octree is used as a bound of the simplified solid to guarantee the error bound measured with the In-Hausdorff metric. The SOF feasibility, defined in Section 3.3, forces each new face to intersect exactly the same set of nodes intersected by the faces being merged, and guarantees that each polyhedron of the multi-resolution family fulfills the In-Hausdorff bound within  $\varepsilon = \frac{\sqrt{3}}{2^l}$ ,  $l$  being the depth of the octree. In this section we will show that this feasibility is too much restrictive, in the sense that it keeps many pairs of faces from being merged even when there exists a replacing face inside a tolerance  $\frac{\sqrt{3}}{2^l}$ . We will show that the SOF feasibility can be easily and efficiently relaxed to allow most of such face merges, and thus increasing the face reduction, at the only expense of a little increment of the tolerance.

The main idea is to adapt the shape of the octree-bound in order to approximate better the bound induced by the Hausdorff distance around the original faces. In the SOF-feasibility, the octree bound corresponding to a pair of faces  $f_1, f_2$  is the orthogonal region composed by the set of nodes intersected by the faces,  $Nodes(f_1) \cup Nodes(f_2)$ . Figure 25 a-d compares the octree bound of one and two faces, with respect to their respective Hausdorff bounds. Although the maximum error of the new face with respect to the original ones is the same within both bounds, the octree bound is clearly more restrictive: the two faces of Figure 25 c cannot be merged fulfilling the octree bound while they can be merged using the Hausdorff bound.

Unlike the Hausdorff bound, the octree bound is not symmetric with respect to the involved faces and, with the exception of orthogonal coplanar faces, it is not uniform. As stated before, a necessary condition for a successful face merge is that the convex hulls of the positive and negative points do not intersect. Let  $d_{min}$  be the minimum distance between both convex hulls. Two faces  $f_1, f_2$  are candidates to be merged iff  $d_{min}$  of their octree bound is not zero. Figure 25 e, h shows the upper and lower convex hulls of one and two faces, respectively. In the former case,  $d_{min}$  is positive, so the positive and negative points are linearly separable (e.g. by the original face), but note that this distance is far less than the maximum error. This restriction in the freedom of modifying the new face becomes crucial when the new plane separate both point sets but the actual face do not intersect exactly  $Nodes(f_1) \cup Nodes(f_2)$  due to the intersection of adjacent faces, and because  $d_{min}$  approaches very rapidly to zero as the resolution increases (Figure 25 k)—a few discrete face orientations are exceptions, such as orthogonal faces. In the latter case, Figure 25 h,  $d_{min}$  is zero so the two faces will not be merged, although there are many replacing faces inside the Hausdorff bound.

The Extended SOF feasibility, XOF, is a relaxation of the SOF-Feasibility that overcomes most of its problems at the expense of a little tolerance increment, allowing higher simplification rates. Given two faces  $f_1$  and  $f_2$  being replaced by  $f$ , the condition of feasibility  $Nodes(f_1) \cup Nodes(f_2) = Nodes(f)$  of the original feasibility is replaced in the extended feasibility by the following rules.

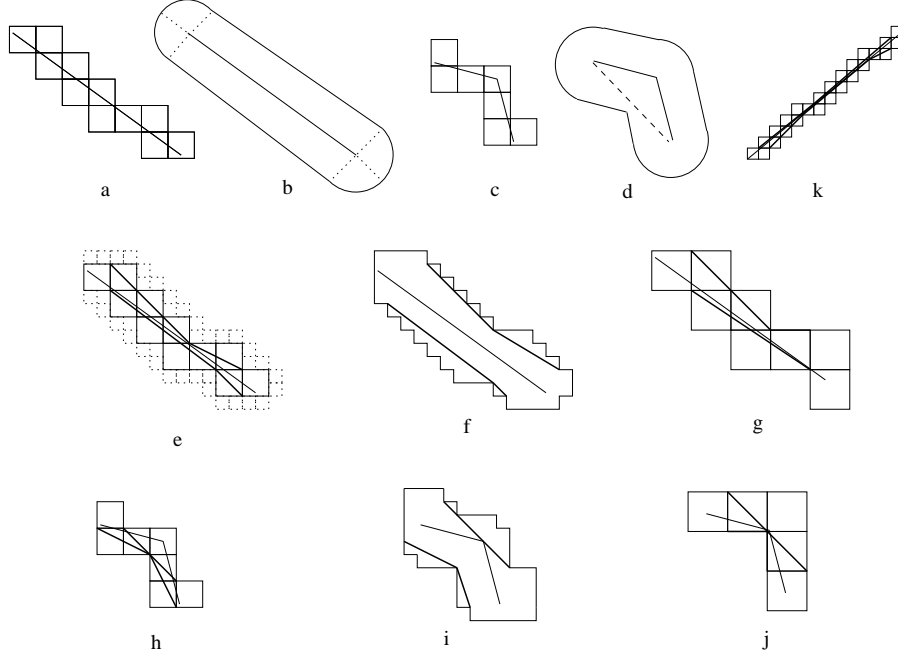


Figure 25: Extended feasibility: a) octree-bound of a face; b) Hausdorff bound of a face; c) octree-bound of two faces; d) Hausdorff bound of two faces; e) upper and lower convex hull of positive and negative points; f) extended octree bound; g) octree-bound with nodes fifty percent larger; h) octree-bound of two faces with  $d_{min} = 0$ ; i) extended octree-bound with  $d_{min} = \varepsilon$ ; j) the octree-bound with nodes fifty percent larger still has  $d_{min} = 0$ ; k)  $d_{min}$  rapidly approaches zero as resolution or face size increase.

Let  $N_{old}$  be  $Nodes(f_1) \cup Nodes(f_2)$ , and let  $N_{new}$  be  $Nodes(f)$ . If  $N_{new} = N_{old}$  then the face is feasible. If not, the following further tests are made. Let  $D_{new}$  the set difference  $N_{new} - N_{old}$ , and symmetrically, let  $D_{old}$  the set difference  $N_{old} - N_{new}$ . For each node  $n$  in  $D_{new}$ ,  $n$  must have a 6-adjacent in  $N_{old}$ ; otherwise the merge is not feasible. Furthermore,  $n$  is divided once resulting eight son nodes. For each son  $s$  intersected by  $f$ , let  $Adj_n$  be the three 6-adjacent nodes of  $s$  that lie outside  $n$ . If  $n$  is B, all nodes in  $Adj_n$  must belong to  $\mathcal{B} \cup N_{old}$ . If  $n$  is W, all nodes in  $Adj_n$  must belong to  $\mathcal{W} \cup N_{old}$ . In both cases, at least one node in  $Adj_n$  must belong to  $N_{old}$ ; otherwise the merge is not feasible. Finally, for each node in  $D_{old}$ , there must exist a 26-adjacent son  $s$  in  $D_{new}$  intersected by  $f$ .

The extended octree-bound is shown in Figure 25 f. Note that  $d_{min}$ , which was near zero in the octree bound, now is greater than  $\varepsilon$ . Now, the two faces of Figure 25 c can be merged (Figure 25 i).

The extended feasibility captures the most important advantages of the OOS-feasibility: a) it guarantees a tight bound of the error, which is  $1.5\varepsilon$  instead of  $\varepsilon$ , i.e.,  $\frac{3\sqrt{3}}{2+1}$ ; and b) each B and W node in  $D_{new}$  is used exclusively by a single sheet of the surface. This guarantees that the old test for self-intersections is still

valid in the extended feasibility approach. The extended feasibility has these extra properties not found in the OOS-feasibility: a) the shape of the extended octree-bound is closer to the Hausdorff bound than the original octree-bound (Figure 25 b, f); b) the extended octree-bound is more symmetric than the original octree-bound; c) in most cases  $d_{min}$  of the extended octree-bound is far greater than  $d_{min}$  of the original octree-bound. The main outcome of the extended feasibility is that it increases the number of face merge operations and hence allows the generation of more compact simplifications.

Although the maximum error is incremented a half, it is important to notice that the extended octree-bound is very different to the octree-bound corresponding to nodes 50% bigger (Figure 25 g, j). Note that increasing the size of the nodes always increases the maximum error but does not necessarily increase  $d_{min}$ , as it is shown in Figure 25 j, where two faces cannot be merged using the 50% bigger octree-bound but can be merged using the extended octree-bound.

The extended feasibility improves the fitting to the Hausdorff bound by subdividing some nodes once more. Subdividing more times can improve this fitting, but we prefer only one subdivision because: a) it guarantees that no self-intersections appear inside B and N nodes, since each node is used exclusively by a connected sheet of the surface; b) it is more efficient; since  $f$  intersects the B/W node, the classification of its center with respect the support plane of  $f$  can be used to know which octants are intersected by  $f$ ; c) the relative increment of  $d_{min}$  achieved with further subdivisions is very small.

### 3.9 The Direct DPS algorithm implementation

In this section, the implementation of the algorithm for the Direct MDCO DPS based on the In-volume criterion is presented. It uses a special data structure that links the boundary representation of a polyhedron with the set of TG nodes in its MDCO representation: the TGMap.

#### 3.9.1 The TGMAP Data Structure

A TGMap is a data structure involving the set of TG nodes of an MDCO together with the BRep of a polyhedron. More precisely, a TGMap is a vector  $\langle BRep, MDCO, LNodes, LFaces \rangle$  where:

*BRep* is the boundary representation of a polyhedron. It contains the relations  $F \rightarrow V$  plus the relation  $V \rightarrow F$  i.e., the incident faces to a vertex. The complexity of the polyhedron is unrestricted. It may have many shells, arbitrary genus and faces with holes of arbitrary complexity.

*MDCO* is the quasi-disjoint MDCO representation of the polyhedron.

*LNodes* and *LFaces* are mappings between the BRep and the MDCO. *LNodes* contains, for every face of the BRep, a list of (pointers to) the TG nodes traversed by that face. *LFaces* contains, for each TG node, a list of (pointers to) the faces of the BRep traversing that node.

### 3.9.2 Implementation Overview

The algorithm is graphically presented in Figure 26.

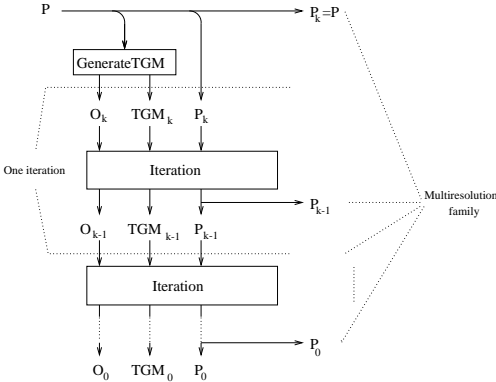


Figure 26: General simplification algorithm.  $TGM_k$  contains the relations  $LNodes$  and  $LFaces$

The algorithm of each iteration is graphically presented in Figure 27, where  $TGM$  includes the relations  $LNodes$  and  $LFaces$ .

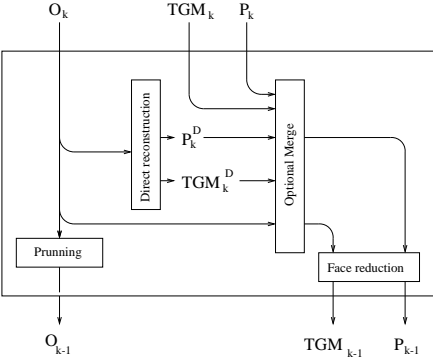


Figure 27: Loop iteration

The first polyhedron of the output multi-resolution family is the initial polyhedron  $P$ . Then the MDCO representation of  $P$ ,  $O_k$ , is generated by the GenerateTGMap function. The relations  $LNodes$  and  $LFaces$  are obtained as a sub-product of the octree construction process in the GenerateTGMap function.

After the first  $O_k$  has been obtained, the algorithm enters in a loop where in each iteration a new polyhedron  $P_{k-1}$  is constructed from the MDCO  $O_k$  and the relations  $LNodes$  and  $LFaces$ . At the end of each iteration, the resulting octree  $O_{k-1}$  is obtained by a pruning process applied to  $O_k$ , being  $k$  the depth of the octree. The last used octree  $O_1$  ( $O_0$  is created but not used) produces a very simple, genus 0 polyhedron  $P_0$ .

### 3.9.3 The GenerateTGMap function

The GenerateTGMap function obtains the MDCO representation of the initial polyhedron  $P$ . The MDCO generation from a BRep is a well known problem and is based on a simultaneous space subdivision and clipping of the boundary of  $P$  [BJN<sup>+</sup>88]. See [CH88] for a revision of other methods of octree construction from BRep schemes. This function is called only once since the rest of MDCO are all obtained from the first one by means of a pruning process, following a top-down DPS approach.

Although only *BTG* nodes will contain a part of the boundary of the reconstructed polyhedron, the MDCO stores the four types of nodes (B,W,G and TG) since they will be required for the DirectReconstruction function.

The TGMap is obtained as a subproduct of the MDCO construction. The LFaces mapping is easily obtained since in the subdivision process, the geometry (faces, edges and vertices) inside a node is known, and this information is inherited by the descendants.

The other component of the TGMap, LNodes, is obtained associating a list of nodes to every face of  $P$ . At the beginning of the MDCO construction, the MDCO has a single grey node which is included in all the lists. In the recursive subdivision process, the node lists of all faces associated to every son node are updated adding all the son nodes traversed by the face and removing the parent node.

The TG nodes are obtained by an algorithm very similar to the construction of a polyhedron from an Extended Octree [Nav86]; the rest of terminal nodes (B/W segmentation) are generated by a seed algorithm starting from nodes that are known to be W, instead of using a less robust point-inside-solid classification.

### 3.9.4 The Prune function

The Prune procedure performs a one-level pruning operation on the current MDCO. Let  $k$  the deepest level of the MDCO. The nodes involved in the pruning operation are these from levels  $k$  and  $k - 1$ . Nodes from level  $k$  are removed and grey nodes from level  $k - 1$  become TG nodes. As a result of the pruning operation, the size of TG nodes is doubled in each direction and the resulting number of TG nodes is the number of terminal nodes of level  $k$  divided by eight.

### 3.9.5 The DirectReconstruction function

The function DirectReconstruction uses a simple and direct method to obtain the OOS from  $O_k$ , which will be the first approximation  $P_k^D$  of the final polyhedron. The algorithm first calculates the set of *BTG* nodes from the set of TG nodes by examining the 26-neighbor nodes. The geometry inside each *BTG* is obtained with the help of a look-up table, similar to that of [AAB95] but indexed by 26-neighborhood configuration instead of vertex colors configurations. First, *BTG* configurations that contain orthogonal vertices are identified, and the set of final vertices is calculated, including vertex coordinates, plane equations of

#levels	#f Direct	#t mesh decim.	max. error
9	178 (360 t)	406	0.3%
8	122 (248 t)	294	0.6%
7	86 (176 t)	217	1.2%
6	64 (132 t)	161	2.1%
5	56 (116 t)	80	4.5%
4	34 ( 68 t)	59	6.1%

Table 2: Number of faces returned by Direct DPS (*second column*) and mesh decimation (*third column*) on model A of Figure 28

its three/six incident faces and convex/concave edge configurations, which are obtained from the look-up table. After all vertices have been extracted, they are combined to form large, orthogonal faces with the domino-matching algorithm presented in [Nav86], and hence it does not contain adjacent and coplanar faces, so there is no need of a compactation post-process. The TGMap  $TGMD_k$ , which is associated to  $PD_k$ , is also obtained by the DirectReconstruction function. For details on the UDMC reconstruction, see [MSS94].

### 3.10 Results and Discussion

We tested Direct DPS on several test models. Model A is a genus 2 CAD mechanical piece with 1,366 faces (Figure 28). Results of model A have been compared with the results of a free implementation of mesh decimation. To draw the comparison we had to triangulate both model A (2,728 triangles) and the model returned by Direct DPS. We have selected the tool described in [CS96] for computing the resulting approximation error on the outputs of both methods. The comparison is shown on Table 3.10. Figures 38-47 show the results of each method compared to the original model, where approximation error is shown by color mapping.

Taking the number of triangles as the basis for the comparison, Since the model has a very simple topology, Direct DPS produced less triangles than mesh decimation in accurate approximations but more triangles in coarse ones.

Regarding the number of faces, Direct DPS always produced less faces than mesh decimation (faces of arbitrary complexity). Mesh decimation produced self-intersections specially in coarse levels, while Direct DPS always preserved validity. Mesh decimation did not simplify topology while Direct DPS reduced the genus. Since Direct DPS faces were Delaunay-triangulated after simplification, resulting triangles had better aspect ratio.

Model B is a building with many nearby shells involving 4,866 faces. Results with several octree depths are shown on Table 3.10. Model B shows the power of Direct DPS on topologically very complex models, resulting in high reduction rates.

Model C and D, a slot machine and a dart with 1,279 and 2,032 triangles resp. have been simplified using Direct DPS and mesh decimation (Figures 30 and 31).

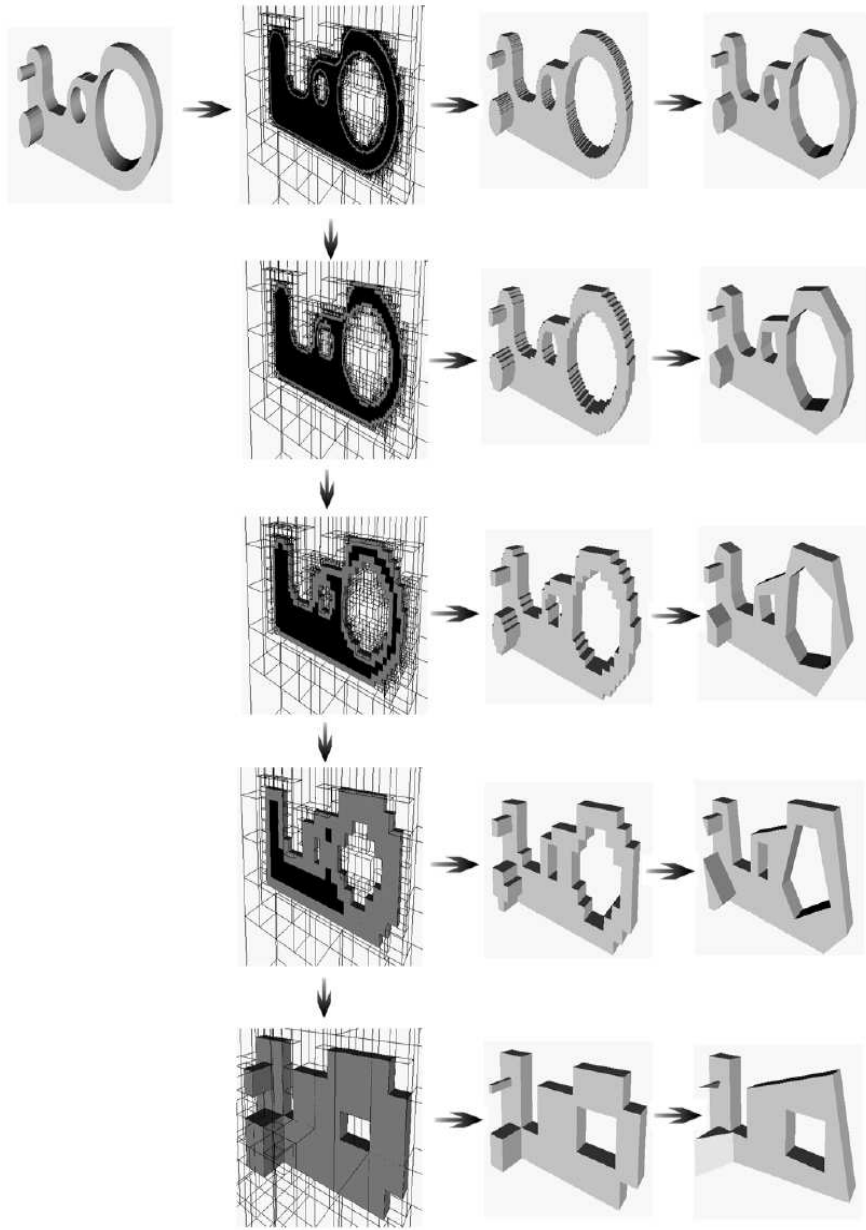


Figure 28: *From left to right*: input solid polyhedron, multi-resolution of octrees (front grey nodes have been culled to keep black nodes visible), reconstructed solids and final multi-resolution

In all cases, the fact that the original polyhedron is not used beyond discretization carries many advantages: the running times of reconstruction and face reduction do not depend on the complexity of the original polyhedron, so cheap

#levels	#f Direct	#v Direct	max. error
Original	4,866	4,134	N/A
7	140	248	1.2%
6	84	146	2.1%
5	61	100	4.4%

Table 3: Number of faces and vertices returned by Direct DPS (*third column*) on model B of Figure 29

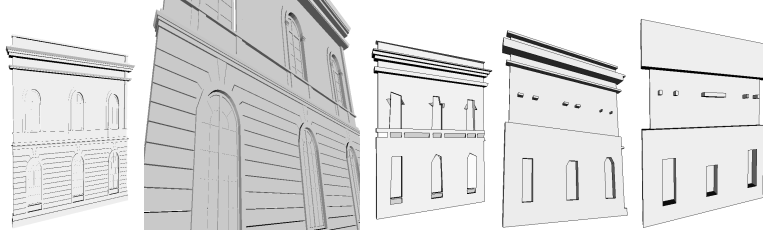


Figure 29: From left to right: original building (4,866 faces), detail, and simplifications with 140, 84 and 61 faces.

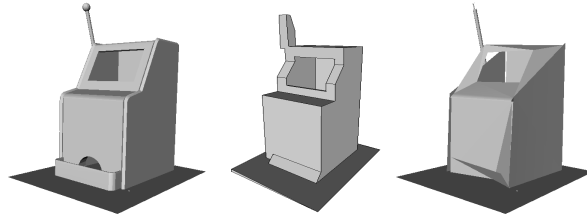


Figure 30: From left to right: original slot machine (1,279 triangles), Direct DPS result (37 faces) and mesh decimation output (210 triangles).

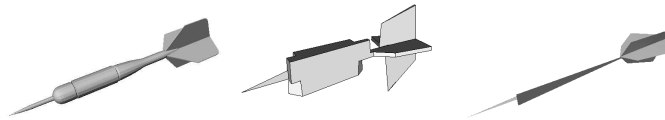


Figure 31: From left to right: original dart (2,032 triangles), Direct DPS result (34 faces) and mesh decimation output (48 triangles).

simplifications are cheap to compute (e.g. 3D thumbnails generation); Direct DPS is not restricted to BReps and it can simplify any model which can be converted to an octree; finally, it is not affected by degeneracies of the input surface, and produces two-manifold solids even with non-manifold inputs so Direct DPS can be viewed as a lossy conversion from non-manifolds to manifolds.

Compared to mesh simplification, Direct DPS minimizes the number of planes



by using arbitrary faces, so it is especially suitable for BSP-based applications. In occlusion analysis, Direct DPS provides a fast and sound way of computing big occluders from a complex scene, specially in reverberation path calculation, used in acoustic modeling, where significant high tolerances can be used.

Before the OOS construction, if TG are converted to B and W nodes 26-adjacent to new B are converted to TG, then OOS is a bounding volume of the input solid. Symmetrically, if TG become W and 26-adjacent B become TG, then OOS will be completely contained in the input solid (Figures 32-34). Due to its ability to produce bounding or bounded approximations, it is also suitable for real-time acceleration of collision detection.

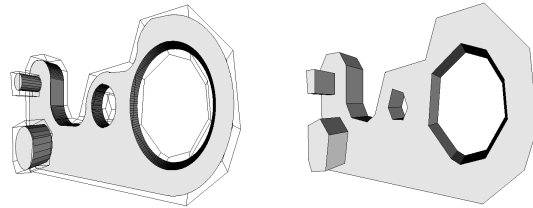


Figure 32: Original object (shaded on the left) and bounding simplification with 47 faces (wire on the left, shaded on the right).

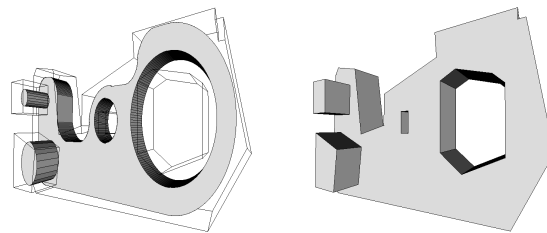


Figure 33: Original object (shaded on the left) and bounding simplification with 38 faces (wire on the left, shaded on the right).

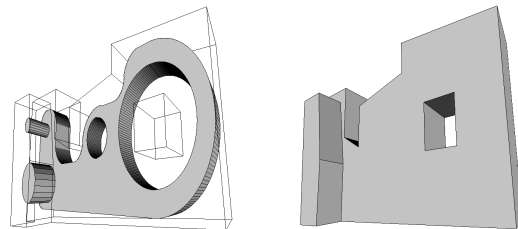


Figure 34: Original object (shaded on the left) and bounding simplification with 21 faces (wire on the left, shaded on the right).

Meanwhile, the Direct DPS method has several limitations. The cost of the Direct DPS method depends on the maximum subdivision level of the MDCO,

which in real world applications is determined by the error tolerance. For very small error tolerances (less than 0.05%), a surface simplification method is probably better, since not much topology changes would be expected and the running times of most surface simplification methods decrease with the tolerance <sup>2</sup>.

Regarded as a surface simplification method, the fact that the Direct MDCO DPS is unable to use information of the original polyhedron in the reconstruction step carries some drawbacks. The use of the original (or previously generated) polyhedron in the face reduction step would enable the preservation of the orientation of important faces, which is a major topic in rendering-oriented simplifications to avoid the visual artifacts produced by the shading of slightly different planes. The Direct DPS method does not provide any support to photometry attributes. More precisely, it does not provide a suitable framework for identifying and keeping unsimplified faces with very different values of photometry attributes. The extension outlined in the next section overcomes these limitations.

---

<sup>2</sup>However, the running times of some incremental simplification methods are quite independent of the tolerance since the pre-computation of a queue prioritizing the entities to be collapsed dominates the overall cost.

## 4 The Hybrid MDCO DPS method

### 4.1 Introduction

As stated in the previous section, the Direct MDCO method is suitable for those applications where surface appearance, such as curvature and color information, does not need to be preserved. Examples of these applications are found in visibility analysis, collision detection and global illumination. However, one of the most important applications of multi-resolution models, the LOD-based visualization [HG94], [MS95], [FS93], requires the overall surface appearance to be preserved in the simplified models.

In this section we introduce the Hybrid MDCO DPS, which is another member of the family of simplification methods modeled by the DPS framework. The Hybrid MDCO DPS is a visualization oriented extension of the Direct MDCO DPS in the sense that it provides mechanisms for surface appearance preservation.

### 4.2 Visualization-oriented simplification

Multi-resolution models for visualization applications must fulfill the following extra requirements:

- *Normal preservation*

The overall orientation of faces must be preserved, since on-screen color in most lighting models depends on the surface normal. Radiosity models, which store final color information in a per-vertex basis, are an exception to these rules. In this case, the simplified representations must preserve the per-vertex or per-corner color information. A method for surface appearance simplification which does not require the preservation of overall orientation of faces is presented in [COM98]. The method is based on a parameterization of the triangulated surface and the storage of surface appearance attributes in normal and color maps. Although it achieves very realistic approximations of complex shapes and provides a bound in the resulting image error, normal maps are still available only in prototyped hardware [OL98], and texture maps increase rasterization cost and can decrease overall performance if the graphics subsystem is raster-limited.

- *Color preservation*

The other surface appearance attributes, such as color and texture parameters, must be preserved. When these parameters come on a per-face basis, faces with different values for such attributes can be merged together only if the impact on the final image is acceptable. Only a few of the existing surface simplification methods deal with such appearance attributes [CPD<sup>+</sup>96], [Hop97].

- *Coherence between consecutive representations*

In addition to the increasing bound on the appearance error between the original representation and each approximation, there should be a fixed,

user-defined bound on the appearance error between two consecutive representations. This requirement is very important to limit the impact of the image artifacts caused by the real-time switching between consecutive representations. Some surface simplification methods allow a smooth transition between two consecutive LOD representations [RB93], [Tur92], [HHK<sup>+</sup>95], [aa95], [DLW94] but the cost of computation of the intermediate representations, which must be done in visualization time, often exceeds the speed-up of the LOD-based visualization. Some high-end API's provide a mechanism to reduce these artifacts. In [Eck97] an image-space smooth transition is achieved by a weighted blending of both representations in a small interval around the switch point. The previous techniques are not mutually exclusive, and both object-space and image-space smoothing techniques should be used together to reduce the perception of these image artifacts.

Note that the solid simplification metrics, such as the In-metrics introduced along with the DPS framework, are still valid for visualization purposes. As it was shown in Figure 2, distances based on points on the surface unnecessarily limit face reduction and yield to poor results on objects with complex topology.

A key difference with respect to all-purpose solid simplification is that visualization oriented applications are less concerned in maintaining large, complex faces whenever possible, since polyhedra are in general triangulated for visualization purposes, and triangle meshes are efficiently handled by most API's and graphics subsystems [Ros96].

### 4.3 Combining surface and solid simplification strategies

One of the most important results of the comparative of the Direct MDCO DPS against surface simplification methods is that the former is more suitable when the object's topology is very complex or the tolerance value is high, due to its powerful capability of simplifying the object's topology maintaining its validity. With very small tolerances, or when the objects have a simple topology, surface simplification methods will suffice.

Since the union of the optimum domain of each approach covers the whole polyhedra domain, we could figure out a *smart* scene simplification algorithm which requests one representative of each family for the simplification of each object, founding this decision on its topological complexity with respect to the user-defined tolerance. As we will show later, the Hybrid MDCO approach is, in fact, an adaptive version of this smart algorithm, but operating in a region-based basis instead of in a object-based one, where regions correspond to subsets of the  $BTG^W$  nodes of the MDCO, and hence it is suitable for the whole domain of polyhedra.

### 4.4 Features of the Hybrid DPS Method

As MDCO-based approaches, both the Direct MDCO and the Hybrid MDCO methods share the decomposition scheme definition and interpretation, the full-

closed MDCO. They differ basically in the reconstruction step: the former uses a direct approach, i.e. all the information for the reconstruction step is obtained from the MDCO alone, and the latter uses a hybrid approach, i.e. the reconstruction is based on the MDCO but directed by the polyhedron generated in the previous iteration. Since face orientation information is lost in the decomposition scheme, no Direct reconstruction method is capable of appearance preservation.

As an extension of the Direct MDCO, the Hybrid method inherits most of its advantages (generation of error-bounded, low genus, two-manifold solids) but solves most of its limitations: unlike the Direct MDCO, the Hybrid MDCO makes the most of the algorithm’s coherence between iterations, resulting in both a speed-up of the running times and a bounded transition between consecutive LOD’s, and supports appearance preservation.

## 4.5 The Hybrid MDCO DPS algorithm

The key ingredient of the Hybrid MDCO DPS is the application of a surface simplification algorithm to the object’s surface inside some regions and the application of a solid simplification inside the rest of regions, basing this decision on local topology changes (by local we mean inside a set of  $BTG^W$  nodes). By virtue of this process, the boundary of the generated simplification involves patches from two different sources, the surface of the previous generated polyhedron, and that of the orthogonal reconstruction, OOS. This approach is outlined in Figure 35.

Due to its bias towards visualization, and unlike the Direct MDCO DPS, the Hybrid MDCO triangulates all faces of the original polyhedron and produces only triangle meshes, which are fully supported by most surface simplification strategies.

The top-level Hybrid MDCO algorithm is identical to that of the Direct MDCO, except for the fact that the faces of the original polyhedron have been triangulated, such as in [RB93]. The only different function is the reconstruction step, whose algorithm follows. Note that  $O_{k+1}$  and  $P_{k+1}$  are the MDCO model and the mesh of the previous iteration, respectively.

```

function HybridReconstruction( $O_k, O_{k+1}$ : MDCO;  $P_{k+1}$ : TMesh)
    < BestOrtho, BestInput >:=classify_BTG_nodes( $O_k, O_{k+1}$ )
    divide_regions(BestOrtho, BestInput)
    OrthoPatches:=direct_reconstruction( $O_k, BestOrtho$ )
    InputPatches:=import_surface( $P_{k+1}, BestInput$ )
     $P_k$ :=merge(BestOrtho, OrthoPatches, InputPatches)
    surface_simplification( $P_k$ )
    return $P_k$ 
end

```

The  $BTG$  nodes of the current MDCO  $O_k$  are classified into two categories. The first category is called *best ortho*, and contains all the  $BTG$  nodes with topology changes of the previous polyhedron  $P_{k+1}$  with respect to the OOS corresponding

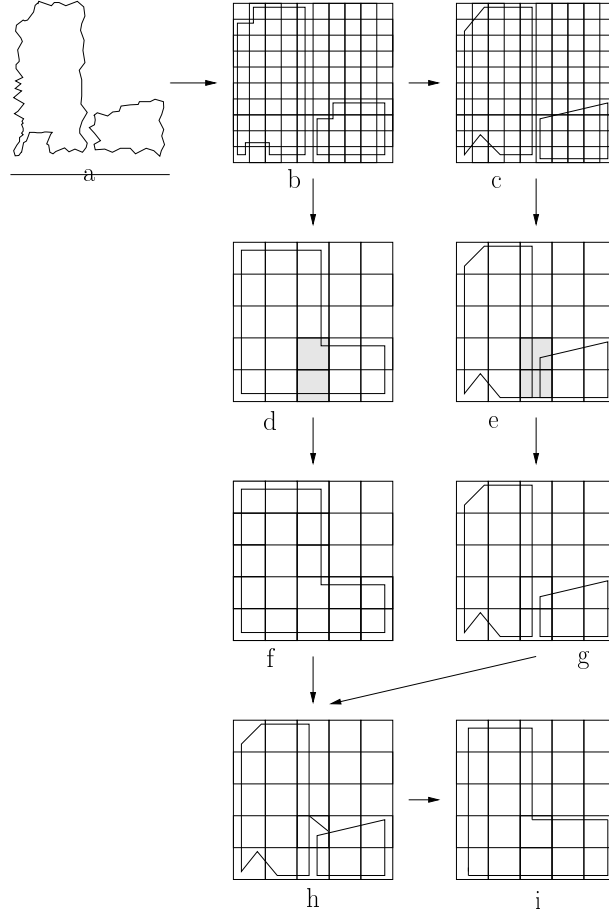


Figure 35: Hybrid reconstruction Overview: a) original solid; b)  $O_{k+1}$ ; c)  $P_{k+1}$ ; d)  $O_k$ ; e)  $P_{k+1}$  (same as c) but viewed inside nodes of a higher level; shaded nodes correspond to nodes with topology changes; f) the surface inside nodes with topology changes is reconstructed with the  $O_k$ ; g) the surface inside nodes without topology changes is reconstructed with the  $P_{k+1}$ ; h) the previous surfaces are merged in a polyhedron topologically equivalent to  $O_k$  but geometrically closer to  $P_{k+1}$ ; i) resulting polyhedron after surface simplification.

to its one-lower level MDCO,  $O_k$ . The second category is called *best input*, and contains the rest of *BTG* nodes, i.e., nodes whose interior surface cannot be topologically simplified and therefore a surface simplification of the surface of the previous iteration polyhedron is preferred.

Once *BTG* nodes are classified, each subset is partitioned in 6-connected regions. The surface inside best ortho regions is reconstructed by clipping the orthogonal reconstruction of the *BTG* nodes of that region, producing the *ortho patches*. Similarly, the surface inside best input regions is reconstructed by clipping the surface of the previous generated polyhedron, producing the *input patches*. At this stage, topology reduction has been done (on best ortho regions), but the

surface is invalid since it has discontinuities in the union of the ortho and input patches.

The next step is the merge of input-patches and ortho-patches, in order to ensure the surface continuity between nodes. The merge creates new triangles and generates a two-manifold, feasible boundary; it may contain more faces than the original, but a simpler topology.

The last step is the incremental surface simplification method, which performs a topology-preserving face reduction. Each of these steps is described in depth in the following subsections.

## 4.6 Classifying the BTG nodes

The classification of the *BTG* nodes takes into account topology, geometry and photometry properties. In order to explain how these elements are used, the following definitions must be introduced:

**Definition** *In a given iteration  $i$ ,  $i$  being the depth of the MDCO, the Input-surface( $i$ ) is  $P_{i+1}$ , i.e., the polyhedron generated in the previous iteration (the previous LOD to that being generated).*

Figure 35 c) shows the input-surface for a given level. Note that in the first iteration, Input-surface(1) is the original polyhedron  $P$ .

**Definition** *In a given iteration  $i$ ,  $i$  being the depth of the MDCO, the Ortho-surface( $i$ ) is  $OOS(MDCO(P, i))$ , i.e., the orthogonal reconstruction corresponding to the MDCO with depth  $i$ .*

Figure 35 b) shows the ortho-surface for a given level. Note that, in the Direct MDCO DPS, the result of the reconstruction process at iteration  $i$  is simply *Ortho – surface( $i$ )*.

**Definition** *Given a BTG node  $n$ , the Input-surface( $i, n$ ) is the Input-surface( $i$ ) clipped to the cubic region associated to  $n$ .*

**Definition** *Given a BTG node  $n$ , the Ortho-surface( $i, n$ ) is the Ortho-surface( $i$ ) clipped to the cubic region associated to  $n$ .*

**Definition** *Let  $k$  be the current iteration depth. A  $BTG^W$  node  $n$  of  $MDCO(P, k)$  is classified as best-input only if the following conditions are satisfied: a) Input-surface( $k, n$ ) and Ortho-surface( $k, n$ ) have the same number of sheets, and b) every sheet of Input-surface( $k, n$ ) intersects the same faces of  $n$  than the corresponding sheet of Ortho-surface( $k, n$ ). Otherwise, the node is best-ortho.*

Figure 35 d) and e) show the classification of *BTG* nodes in best-input and best-ortho nodes. Note that when a node is classified as best-input, the topology of the input-surface is the same as that of the ortho-surface. In other words, replacing Input-surface( $k, n$ ) with Ortho-surface( $k, n$ ) will not simplify the topology and will increase both the number of staircase-like set of faces and the surface appearance error. The best-input nodes are so called because it is more convenient to reconstruct the surface inside them using the input-surface instead of the ortho-surface.

*BTG* nodes  $n$  of  $MDCO(P, k)$  not satisfying the conditions of best-input nodes

are classified as best-ortho, so called because it is more convenient to reconstruct the surface inside them using the ortho-surface instead of the input-surface.

The following theorem is fundamental in the classification algorithm:

**Theorem 4.1** *If the face reduction step of the Hybrid DPS method preserves both the topology and the SOF feasibility of the mesh, then for every BTG node  $n$  of  $MDCO(P, k)$ ,  $Input\text{-}surface(k, n)$  and  $Ortho\text{-}surface(k-1, n)$  have the same number of sheets and intersect the same set faces of  $n$ .*

The face reduction step we will use in the Hybrid MDCO DPS satisfies both conditions, and hence the classification of *BTG* nodes can be done by comparing  $Ortho\text{-}surface(k, n)$  with  $Ortho\text{-}surface(k-1, n)$ . Since the complete geometry of both OOS can be completely reconstructed from a look-up table indexed by node neighborhood configurations, the classification of *BTG* nodes do not involve any explicit geometric test at all.

Both the number of best-ortho and the number of best-brep nodes can be zero for some input polyhedra (although not at the same time). In the former case, the Hybrid MDCO is reduced to a surface simplification algorithm, with no orthogonal reconstruction at all; on the latter case, it is a triangulated version of the Direct MDCO reconstruction.

## 4.7 Partitioning the BTG nodes

Once the *BTG* nodes have been classified in best-input and best-ortho nodes, each category must be partitioned into 6-connected components.

Then, some best-input nodes are converted into best-ortho until the partition satisfies these two conditions: a) if a best-ortho node  $n$  in level  $i$  has a best-ortho 26-neighbor  $m$  and  $ortho\text{-}surface(i, n)$  intersects the shared vertex, edge or face, then both  $n$  and  $m$  belong to the same region, and b) if a best-input node  $n$  has a best-input 26-neighbor  $m$  and  $input\text{-}surface(i, n)$  intersects the shared vertex, edge or face, then both  $n$  and  $m$  belong to the same region. The resulting regions are 6-connected, and two different best-input regions are not 26-adjacent unless there is not any surface crossing the common boundary. The same holds for two best-ortho regions. This transformation guarantees that  $C^0$  continuity between patches must be enforced only in shared faces of a best-input and a best-ortho region.

## 4.8 Extracting the patches

The input-patches are extracted from the surface of the previous iteration polyhedron. There is a brep-patch for each 6-connected best-input region. The patch is calculated by collecting all triangles of  $P_{k+1}$  completely inside the region, and clipping to the region's boundary those triangles partially inside the region. The resulting patches involve one or more sheets of the Input-surface,  $P_{k+1}$ . Since  $P_{k+1}$  is triangulated and the clipping region is orthogonal and with integer coordinates, this clipping is a simple case of the general polygon clipping problem.



The ortho-patches are extracted in a similar way, but using the Ortho-surface instead of the Input-surface. The resulting patches involve one or more orthogonal sheets of the Ortho-surface. Note that, although input-patches are triangular meshes, ortho-patches involve faces of arbitrary number of contours. They will be triangulated in the merge step.

Vertices of both the Ortho-patches and the Input-patches have a flag called *border* indicating whether they also belong to the ortho-surface (input-surface) or they appeared as a result of the clipping to the region's boundary.

## 4.9 The merge step

The merge process modifies the ortho patches so that the resulting surface has  $C^0$  continuity along the faces shared by a best-input and a best-ortho region.

There are five types of continuity between nodes: a) between nodes of a best-input region, b) between nodes of a best-ortho region, c) between nodes of different best-input regions, d) between nodes of different best-ortho regions, and e) between a best-input and a best-ortho node.

The first two still hold since the input and the ortho patches are  $C^0$ . The second pair can be ignored, since different best-input regions are not 26-adjacent unless there is not any surface crossing the common boundary, and the same holds for best-ortho regions. Thus the continuity must be ensured only in stabbed node faces shared by a best-input node and a best-ortho node.

The merge consists in a constrained triangulation of the ortho-patches by connecting the borders of the input-patches maintaining the ortho-patch incidence; the geometry of the surface inside best-input regions is preserved:

```

procedure Merge(BestOrtho, OrthoPatches, InputPatches)
  foreach patch in OrthoPatches do
    mapping(patch, InputPatches)
    foreach face f in patch do
      foreach contour c in face do
        replace_border_vertices(c, patch)
      end
      triangulate(f)
    end
  end
end

```

The mapping function  $\mathcal{M}$  maps each border vertex of the input-patch with at least one border vertex of the ortho-patch. Each border vertex of the input-patch has an order number  $ord(v_i)$  which is obtained by cyclically visiting each edge-connected contour of border vertices of the input-patch.

The mapping must fulfill the following requirements: a) each border vertex of the input-patch is mapped to at least one border vertex of the ortho-patch; b) each border vertex of the ortho-patch is referenced by an edge-connected sequence of at least one border vertex of the input-patch; and c) for each pair of border vertices  $v_i, v_j$  of an edge-connected contour of the input-patch, if

$ord(v_i) < ord(v_j)$  then  $ord(\mathcal{M}(v_i)) < ord(\mathcal{M}(v_j))$ .

The mapping can be implemented by simultaneous traversal of the border vertices of both the input-patch and the ortho-patch, where the current element is advanced in one of the two lists depending on geometric proximity and the mapping restrictions.

In the next step, each border vertex of the ortho-patch is replaced by the ordered sequence of edge-connected border vertices of the input patch given by the mapping. This replacement is done in a per-contour basis to guarantee the feasibility of the resulting surface.

The result is a non-planar face, topologically equivalent to the original ortho-patch, but including only the non-border vertices of the ortho-patch and the border vertices of the input-patch. Finally, the non-planar face is triangulated. This process is illustrated on figure 36.

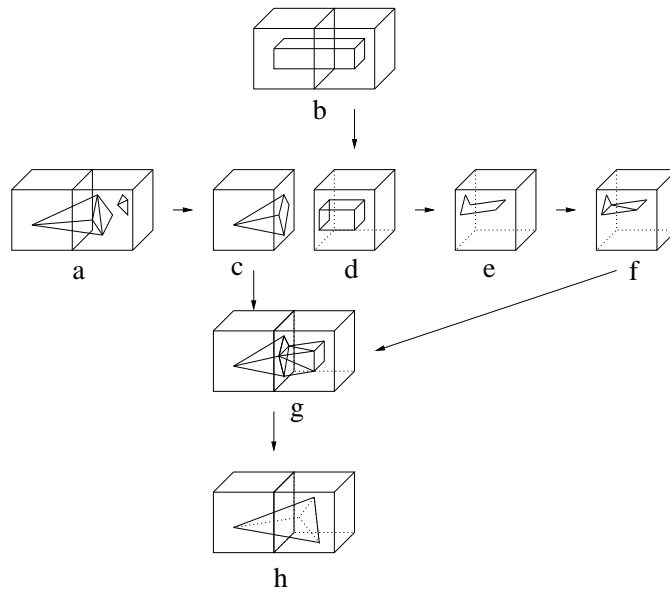


Figure 36: Merge of patches: a) the input-surface; b) the ortho-surface; c) input-patch; d) ortho-patch; e) non-planar contour including border vertices of the input-patch and non-border vertices of the ortho-patch; f) triangulated patch; g) resulting surface after merge; h) resulting surface after edge-collapses.

The result of the merge process is a triangle mesh topologically equivalent to the Ortho-surface, but geometrically closer to the Input-surface. In most cases its topology is simpler than that of the Input-surface.

#### 4.10 Patch simplification

Finally, the merged surface is object to a face reduction step. The surface simplification must fulfill the following requirements: a) the simplified surface must traverse exactly the same set of nodes, and b) the topology must be preserved.

The proposed Hybrid framework is not limited to a particular surface simplification method, and different methods can be used to produce different results (though some incremental methods are more suitable since can be easily extended to support the above restrictions).

There are two important properties which simplify the surface simplification. On one hand, the octree provides a suitable bound for the application of reduction operators, so only appearance-preserving criteria must be added to the surface simplification algorithm. On the other hand, unlike most incremental methods, the use of a queue prioritizing the operators does not make sense since all feasible reductions should be applied. We have selected the edge-collapse due to its simplicity and generality.

#### **4.11 Discussion**

The Hybrid MDCO method solves most of the pitfalls of the Direct MDCO while inheriting many of its advantages. Making a better use of coherence between iterations, it can reduce the disparity between LOD's, and provides a framework in which the collapse of faces with very different photometry attributes can be easily avoided. When simplifying simple objects with accurate tolerances, hybrid's behavior is more suitable.

## 5 Concluding Remarks and Future Work

In this report a framework for polyhedral solid simplification has been presented. Instances of this framework are simplification methods capable of modifying the topology of the solid while guaranteeing an interior-points error bound.

Both the Direct MDCO produces valid, two-manifold objects, even when the input is non-manifold. Unlike most of the current simplification methods, which are restricted to triangular meshes, the presented algorithms can deal and also produce faces with arbitrary geometry and topology, independently of the input surface subdivision.

Although other surface simplification methods might produce better-looking approximations of topologically simple objects from the point of view of LOD-based rendering, DPS methods are more suitable for LOD-based applications where surface appearance preservation is not as much important as topology reduction, such as collision detection, occlusion analysis and acoustic modeling [ea98].

Compared to triangular meshes, the unconstrained planar-faces boundary representation is a more concise representation of polyhedral objects, which is suitable in most of the applications listed above; real-time visualization is the only exception since many APIs and hardware subsystems are optimized for triangle meshes and triangle strips.

Future work includes the analysis of different variants of the face-merge operator, such as allowing the merge of faces sharing more than one edge (loop creation). We have shown that moving one face at a time we cannot achieve optimal results. Figure 37 shows a simple object that cannot be obtained from the OOS without moving two faces simultaneously, so future work also includes the study of an iterative face-merge version, which tries to move two or more faces at the same time. Shortcuts for the iterative application of face-merge operations should be studied (see figure 37).

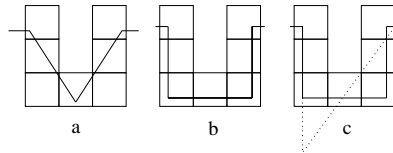


Figure 37: a) Original object b) OOS with two faces being merged c) a simpler OOS-feasible solid cannot be produced without simultaneously modifying the third face.

Regarding to the Hybrid MDCO, several surface simplification options should be studied, and an improved merge of ortho and brep patches should be developed.

## 6 Acknowledgements

The authors would like to thank the Geometry Group people at the GVU Center of the Georgia Institute of Technology for their helpful comments, especially Jarek Rossignac, Greg Turk and Renato Pajarola.

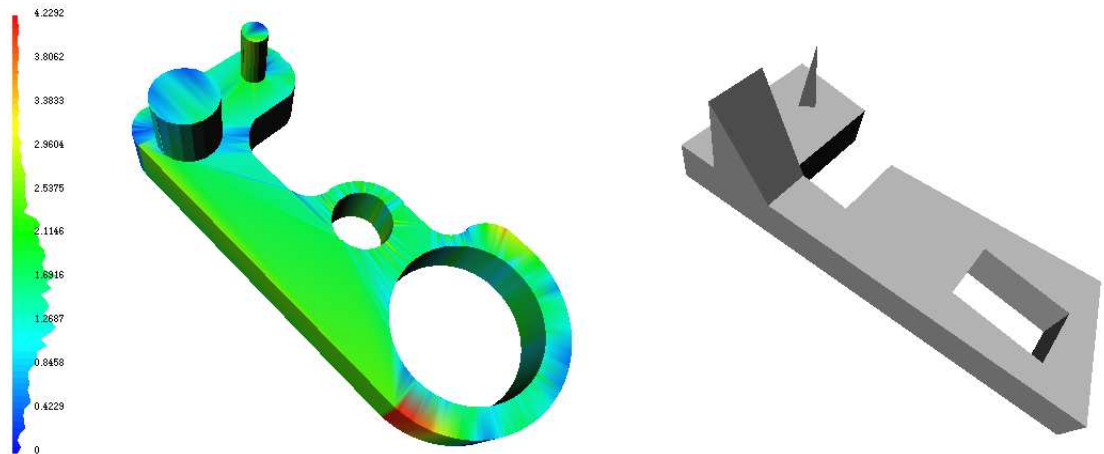


Figure 38: Test polyhedron (1366 faces) and output of Direct DPS (34 faces) using depth 4. Maximum error is 4.2%

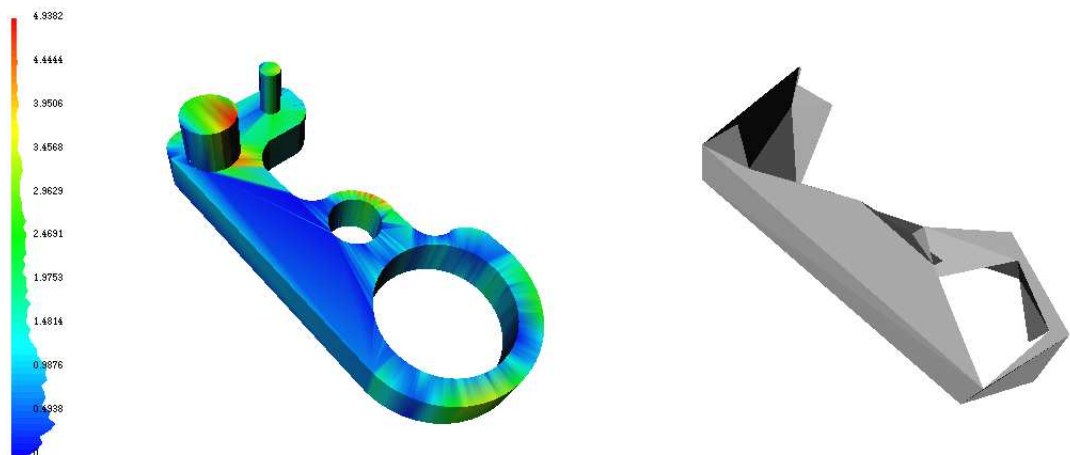


Figure 39: Test polyhedron and output of a typical mesh decimation (59 triangles). Maximum error is 4.9%

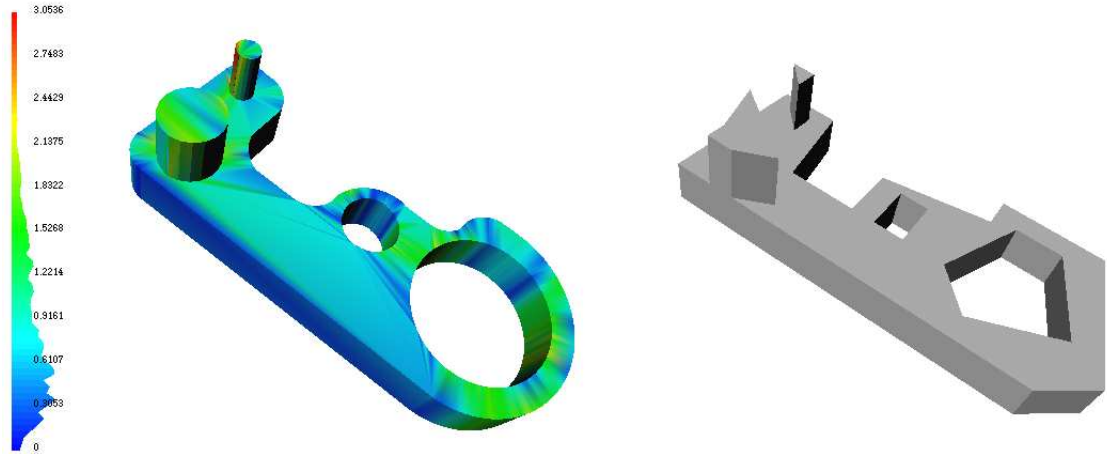


Figure 40: Test polyhedron and output of Direct DPS (56 faces) using depth 5. Maximum error is 3.1%

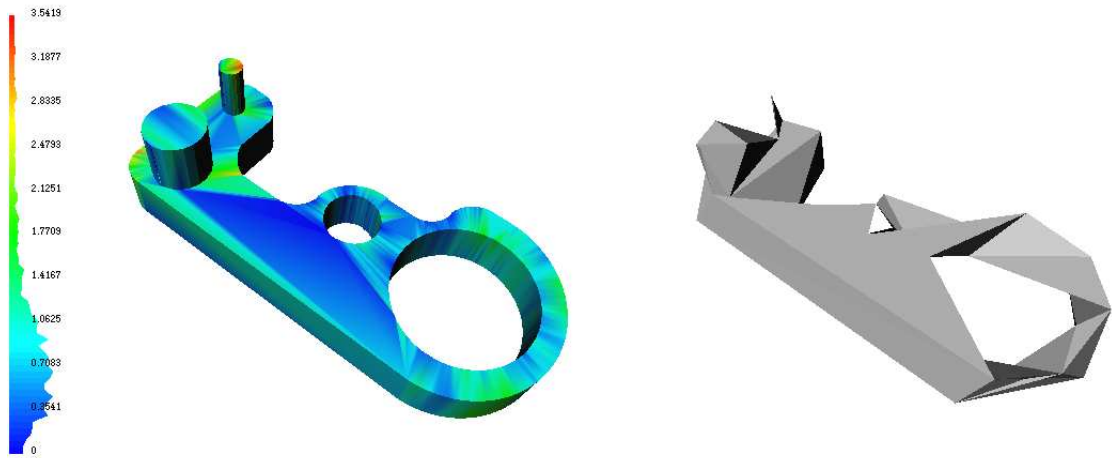


Figure 41: Test polyhedron and output of a typical mesh decimation (80 triangles). Maximum error is 3.5%

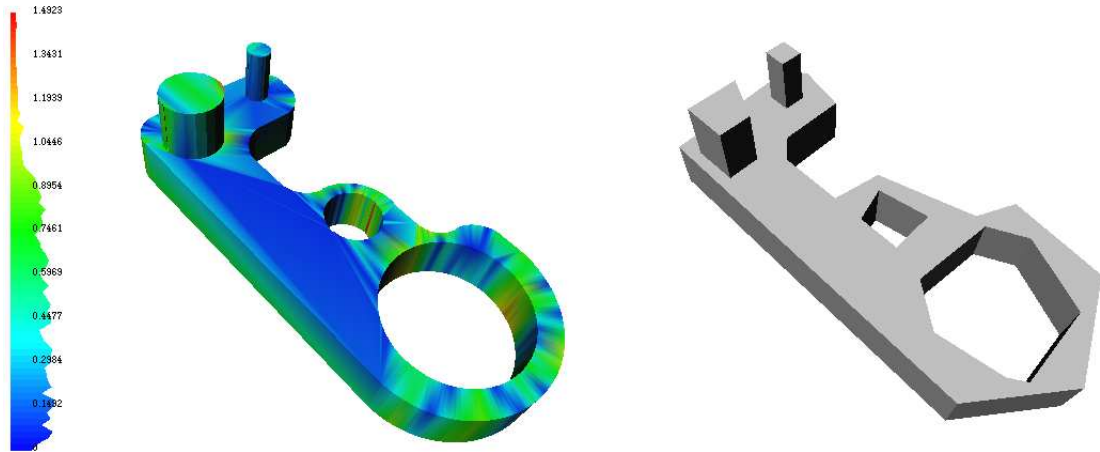


Figure 42: Test polyhedron and output of Direct DPS (64 faces) using depth 6. Maximum error is 1.5%

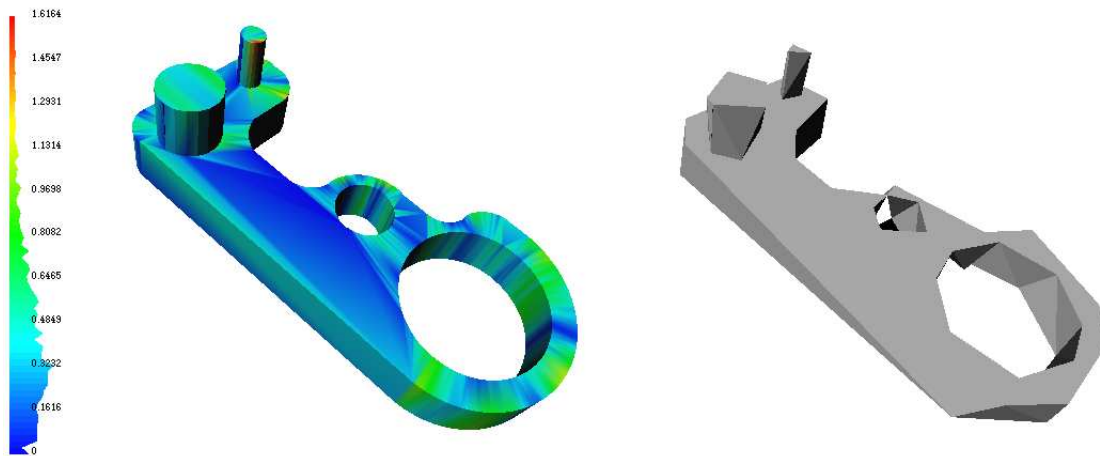


Figure 43: Test polyhedron and output of a typical mesh decimation (161 triangles). Maximum error is 1.6%



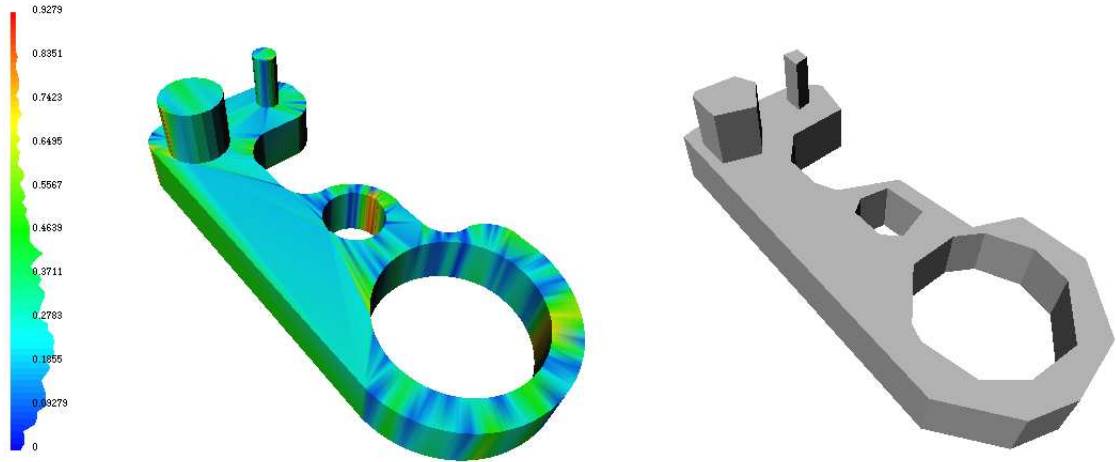


Figure 44: Test polyhedron and output of Direct DPS (86 faces) using depth 7. Maximum error is 0.9%

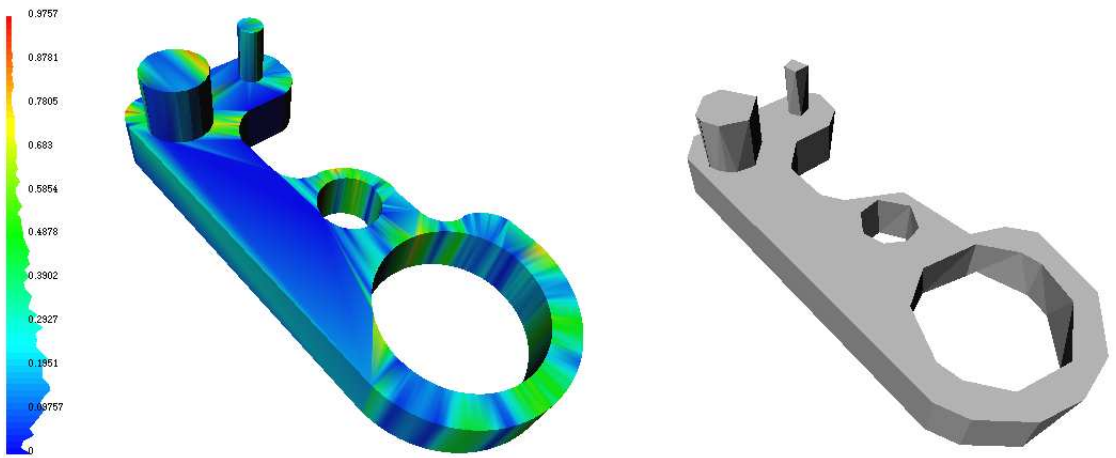


Figure 45: Test polyhedron and output of a typical mesh decimation (217 triangles). Maximum error is 0.9%

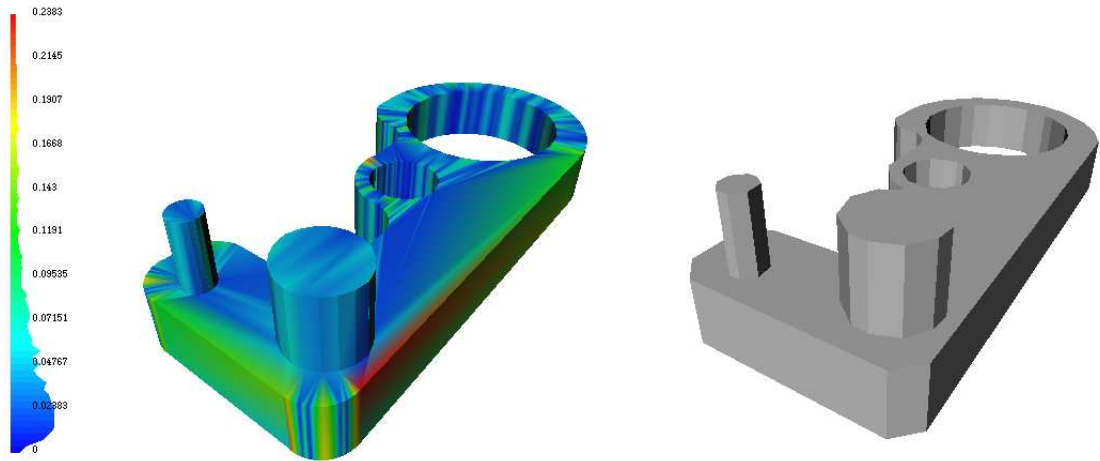


Figure 46: Test polyhedron and output of Direct DPS (178 faces) using depth 9. Maximum error is 0.2%

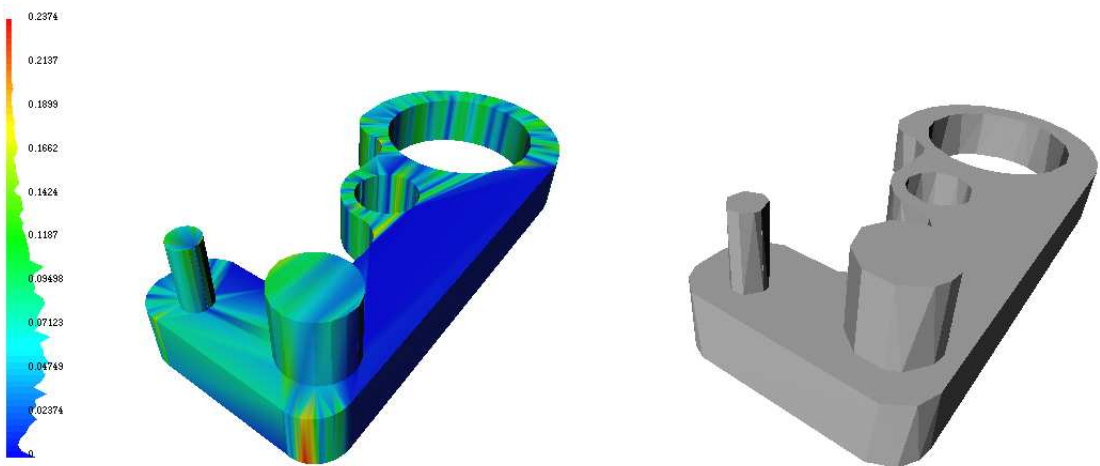


Figure 47: Test polyhedron and output of a typical mesh decimation (406 triangles). Maximum error is 0.2%

## References

- [aa95] Matthias Eck et al. Multiresolution analysis of arbitrary meshes. In *Proc. SIGGRAPH*, pages 173–182, 1995.
- [aa96] Taosong He et al. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–184, 1996.
- [AAB95] D. Ayala, C. Andújar, and P. Brunet. Automatic simplification of orthogonal polyhedra. In D.W. Fellner, editor, *Modelling Virtual Worlds Distributed Graphics*, pages 137–147. International Workshop MVD’95, Infix, 1995.
- [AAB<sup>+</sup>96] C. Andújar, D. Ayala, P. Brunet, R. Joan, and J. Solé. Automatic generation of multiresolution boundary representations. *Computer Graphics Forum*, 1996.
- [ABJN97] D. Ayala, P. Brunet, R. Joan, and I Navazo. Multiresolution approximation of polyhedral solids. In D. Roller and P. Brunet, editors, *CAD Systems Development: Tools and Methods*, pages 327–343. Springer-Verlag, 1997.
- [And98a] Carlos Andújar. The discretized polyhedra simplification: A framework for polyhedra simplification based on decomposition schemes. Technical report, Universitat Politècnica de Catalunya, LSI-98-XR, 1998.
- [And98b] Carlos Andujar. Simplificacion de modelos poliedricos (written in spanish). Technical report, Universitat Politècnica de Catalunya, LSI-98-1T, 1998.
- [ANM97] Andrea L. Ames, David R. Nadeau, and John L. Moreland. *The VRML 2.0 sourcebook*. Wiley, New York, NY, USA, second edition, 1997.
- [AS96] M.-E. Algorri and F. Schmitt. Mesh simplification. *Computer Graphics Forum*, 15(3):C77–C86, September 1996.
- [BBCS96] Chandrajit L. Bajaj, Fausto Bernardini, Jindong Chen, and Daniel Schikore. Automatic reconstruction of 3d cad models. In *Proc. of Theory and Practice of Geometric Modelling*, 1996.
- [BJN<sup>+</sup>88] P. Brunet, R. Juan, Isabel Navazo, J. Sole, and D. Tost. *Scientific Visualization. Advances and challenges*. Academic Press, 1988.
- [Bru90] P. Brunet. Face octrees. involved algorithms and applications. Report LSI-90-14, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, 1990.
- [CCMS97] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer, Springer International*, 13(5):228–246, 1997.

- [CH88] H. Chen and T. Huang. A survey of construction and manipulation of octrees. *Computer Vision, Graphics and Image Processing*, 43, 1988.
- [Cla76] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, 1976.
- [COM98] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. *Computer Graphics Proceedings, Annual Conference Series, 1998 (ACM SIGGRAPH '98 Proceedings)*, pages 115–122, 1998.
- [CPD<sup>+</sup>96] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *Proc. SIGGRAPH*, pages 91–99, 1996.
- [Cro82] F. C. Crow. A more flexible image generation environment. *Computer Graphics*, 16(3):9–18, 1982.
- [CS96] C. Rocchini P. Cignoni and R. Scopigno. Metro: measuring error on simplified surfaces. Technical report, Istituto I.E.I.-C.N.R., Pisa, Italy, January 1996.
- [DLW94] Tony D. DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. In *Proc. SIGGRAPH*, 1994. Course notes.
- [ea96] Jonathan Cohen et al. Simplification Envelopes. *ACM SIGGRAPH '96 Proceedings*, pages 119–128, 1996.
- [ea98] Thomas Funkhouser et al. A beam tracing approach to acoustic modeling for interactive virtual environments. *ACM SIGGRAPH '98 Proceedings*, 1998.
- [Eck97] George Eckel. *IRIS Performer Programmer's Guide*. Silicon Graphics, Inc. Document no. 007-1680-040, 1997.
- [Eri96] C. Erikson. Polyhedral simplification: An overview. Tr 96-016, Department of Computer Science, UNC-Chapel Hill, 1996.
- [FS93] T.A. Funkhouser and C.H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proc. SIGGRAPH*, pages 247–254, 1993. Computer Graphics Proceedings, Annual Conference Series.
- [GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Conference Proceedings*, pages 209–216. Addison Wesley, August 1997.
- [Grü67] Branko Grünbaum. *Convex Polytopes*. Interscience Publishers, New York, 1967.
- [GSG96] Markus H. Gross, Oliver G. Staadt, and Roger Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2), June 1996.

- [Gue96] Andre Gueziec. Surface simplification inside a tolerance volume. Technical report, Yorktown Heights, NY 10598, March 1996. IBM Research Report RC 20440.
- [GW94] Allen Van Gelder and Jane Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, 1994.
- [Ham94] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11:197–214, 1994.
- [HG94] P.S. Heckbert and M. Garland. Multiresolution modelling for fast rendering. *Proc. Graphics Interface 94*, pages 43–50, May 1994.
- [HHK<sup>+</sup>95] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel based object simplification. In G.M. Nielson and D. Silver, editors, *Visualization'95*, pages 296–303, Atlanta, GA, October 29 – November 3 1995.
- [Hop96] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.
- [Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH 97 Proceedings*, pages 189–198. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
- [JAS95] Robert Juan-Arinyo and Jaume Solé. Constructing face octrees from voxel-based volume representations. *Computer-Aided Design*, 27(10):783–791, 1995.
- [KCHN91] Alan D. Kalvin, Court B. Cutting, B. Haddad, and M. E. Noz. Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. In *Medical Imaging V: Image Processing*, volume 1445, pages 247–258. SPIE, February 1991.
- [KR89] T. Y. Kong and A. Rosenfeld. Digital topology: Introduction and survey. *Computer Vision, Graphics and Image Processing*, 48:357–393, 1989.
- [KT93] Alan Kalvin and Russell Taylor. Superfaces: Polyhedral approximation with bounded error. Technical Report RC19135, IBM Research Division. T.J. Watson Research Center., 1993.
- [KT96] Alan D. Kalvin and Russell H. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, ?(?):64–77, 1996.
- [LC87] William Lorensen and Harvey Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. SIGGRAPH*, pages 44–50, 1987. *Computer Graphics* vol. 21 no. 4.
- [Man88] Martti Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, MD, 1988.

- [Mat95] J Mattioli. Minkowski operations and vector spaces. *Set-Valued Analysis*, 3(1):33–50, 1995.
- [MBL<sup>+</sup>91] James Miller, David Breen, Willian Lorensen, Robert OBara, and Michael Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. In *Proc. SIGGRAPH*, pages 217–226, 1991. Computer Graphics, vol. 25 no. 4.
- [Meg83] N. Megiddo. Linear-time algorithms for linear programming in 3d and related problems. *Siam J. Computer*, 12(4):759–776, 1983.
- [MS95] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 95–102. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.
- [MSS94] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *Visualization'94*, pages 281–287. IEEE Computer Society Press, 1994.
- [Nav86] Isabel Navazo. *Contribucio a les tecniques de modelat geometric d'objectes polimerics usant la codificacio amb arbres octals (written in Catalan)*. PhD thesis, Universitat Politecnica de Catalunya, 1986.
- [OL98] Marc Olano and Anselmo Lastra. A shading language on graphics hardware: The pixelflow shading system. *Computer Graphics Proceedings, Annual Conference Series, 1998 (ACM SIGGRAPH '98 Proceedings)*, pages 159–168, 1998.
- [PM85] F. Preparata and M. Shamos. *Computational Geometry*. Springer Verlag, 1985.
- [RB93] J. Rossignac and P. Borrel. Multiresolution 3D aproximations for rendering complex scenes. In *Modeling in Computer Graphics*. Springer-Verlag, 1993.
- [Red96] M. Reddy. SCROOGE: Perceptually-driven polygon reduction. *Computer Graphics Forum*, 15(4):191–203, 1996. ISSN 0167-7055.
- [RH94] John Rohlf and James Helman. IRIS performer: A high performance multiprocessing toolkit for real-Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.
- [Ros96] Jarek Rossignac. Geometric simplification. In *ACM SIGGRAPH Course notes No. 35*. ACM Press, 1996.
- [RR96] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C67–C76, C462, September 1996.

- [SL96] Marc Soucy and Denis Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding: CVIU*, 63(1):1–14, January 1996.
- [Sri81] Sargur N. Srihari. Representation of three-dimensional digital images. *Computing Surveys*, 13(4):399–424, 1981.
- [SW70] Josef Stoer and Christoph Witzgall. *Convexity and Optimization in Finite Dimensions*, volume 163 of *Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellung*. Springer-Verlag, Berlin, 1970.
- [SZL92] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proc. SIGGRAPH*, pages 65–70, 1992. Computer Graphics vol. 26 no. 2.
- [Tur92] Greg Turk. Re-tiling polygonal surfaces. In *Proc. SIGGRAPH*, pages 55–64, 1992. Computer Graphics vol. 26 no. 2.
- [Vel92] R.C. Veltkamp. *Closed Object Boundaries from Scattered Points*. PhD thesis, Erasmus University Rotterdam, 1992.
- [Vel93] Remco C. Veltkamp. 3D computational morphology. In R. J. Hubbard and R. Juan, editors, *Eurographics '93*, pages 115–127, Oxford, UK, 1993. Eurographics, Blackwell Publishers.