Permission Grids: Practical, Error-Bounded Simplification

STEVE ZELINKA
and
MICHAEL GARLAND
University of Illinois at Urbana-Champaign

We introduce the *permission grid*, a spatial occupancy grid which can be used to guide almost any standard polygonal surface simplification algorithm into generating an approximation with a guaranteed geometric error bound. In particular, all points on the approximation are guaranteed to be within some user-specified distance from the original surface. Such bounds are notably absent from many current simplification methods, and are becoming increasingly important for applications in scientific computing and adaptive level of detail control. Conceptually simple, the permission grid defines a volume in which the approximation must lie, and does not permit the underlying simplification algorithm to generate approximations outside the volume.

The permission grid makes three important, practical improvements over current error-bounded simplification methods. First, it works on arbitrary triangular models, handling all manners of mesh degeneracies gracefully. Further, the error tolerance may be easily expanded as simplification proceeds, allowing the construction of an error-bounded level of detail hierarchy with vertex correspondences among all levels of detail. And finally, the permission grid has a representation complexity independent of the size of the input model, and a small running time overhead, making it more practical and efficient than current methods with similar guarantees.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid and object representations*

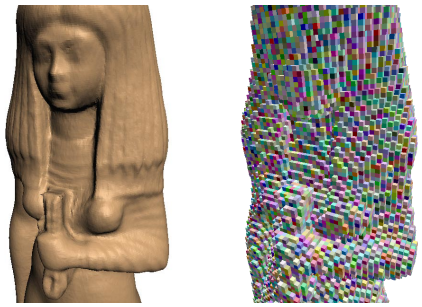
General Terms: Algorithms, Theory, Performance

Additional Key Words and Phrases: error bounds, level of detail, surface simplification

1. INTRODUCTION

Simplification of polygonal surfaces has been a well-studied area in recent years [Garland 1999], with a number of fast and/or high quality methods emerging [Garland and Heckbert 1997; Lindstrom and Turk 1998; Hoppe 1996]. However, most of these methods provide no meaningful guarantee over the quality of the approximation produced. Methods providing good error bounds are orders of magnitude slower than the fastest methods, and do not necessarily generate higher quality results [Cignoni et al. 1998]. This is problematic since many applications require some error guarantee in the approximation. For example, if a simplified model is to be used for fast collision detection, a guaranteed geometric bound on the error in the simplified mesh is required for accurate results. Some finite element

Author's address: Steve Zelinka, Department of Computer Science, Room 3315, 1304 W. Springfield Ave., Urbana, IL, 61801. zelinka@uiuc.edu



(a) (b)

Fig. 1. Visualization of a permission grid with a tolerance of 1% of the length of the bounding box diagonal. (a) A portion of the Isis model. (b) All occupiable voxels of the permission grid are drawn as randomly-coloured cubes. During simplification, approximating triangles that intersect empty voxels are not permitted.

problems can be solved quickly using multi-resolution models, but similar error bounds are required in order to ensure good efficiency improvements. The largest data-sets, from terrain data, medical imaging, or scientific computing, require simplification in order to be handled at all, and guaranteed error bounds in order to be useful after simplification. Ideally, one would like to be able to use a very fast or high quality method for simplification, while still having a good, meaningful error bound.

We have developed a new technique, called the *permission grid*, which augments current simplification algorithms to provide such a guaranteed error bound on the approximation produced, at a small additional cost in memory and running time. In particular, every point on the output approximation is guaranteed to lie within some user-specified distance ϵ from the original surface. This has been called the one-sided Hausdorff distance from the approximation to the original surface [Klein et al. 1996], and it should be noted that this is generally not equivalent to the one-sided Hausdorff distance between the original surface and the approximation. While the latter is more intuitive, it often overly restricts the amount of simplification possible. For example, long thin tendrils of a surface can never be removed. Also, it can be exceedingly difficult to maintain the latter bound efficiently while allowing topological changes in the approximation.

Conceptually, the permission grid is very simple: it is a 3D spatial occupancy grid surrounding the object to be simplified. Each voxel of the grid may be either *occupiable* or *empty*. Before any simplification begins, voxels which are entirely within ϵ of the mesh are marked as occupiable. As simplification proceeds, each new triangle generated by the underlying simplification algorithm is tested against the permission grid; if a triangle intersects any *empty* cells, the operation which would have created the triangle is rejected. Figure 1 shows a visualization of the permission grid for a model of a statue of Isis. Section 4 expands on the creation and testing algorithms for permission grids.

A prime attraction of the permission grid is its simplicity and practicality. As shown in our results (Section 7), the overhead of the grid is very low, especially when compared to current methods. It is easily understood and simple to implement, applying well-studied techniques in voxelization and volume graphics, and can be used in conjunction with most current simplification algorithms. The only requirement on the underlying simplification

algorithm is that it "ask permission" before generating any approximating triangles.

The topology of the approximation can be arbitrarily changed during simplification without affecting the permission grid. This is important because topological simplification, including removing holes and connecting separate components, is required for achieving significant levels of simplification, since topological features inherently require more triangles to represent. Scanning artifacts can lead to surfaces with genus in the hundreds [Guskov and Wood 2001], so requiring that topology be preserved may be burdensome even at moderate degrees of simplification.

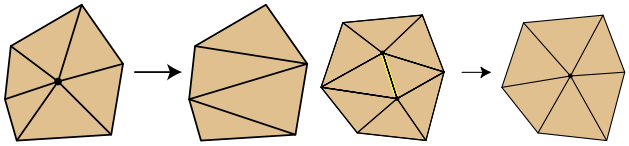
The only assumption placed on the input data is that it is a triangle mesh; there are no restrictions on connectivity. The permission grid works despite any kind of mesh degeneracy, such as T-junctions, self-intersections, non-manifold areas, and even non-orientable meshes. These are key features from a user stand-point, since many meshes generated as a result of range-scanning or CAD packages contain these kinds of degeneracies, and fixing them can be a non-trivial task, despite work in automatic model repair [Nooruddin and Turk 1999; Barequet and Kumar 1997].

The permission grid additionally enables a number of extensions. First, in Section 5, we show how the permission grid can be trivially extended by use of extra supplementary grids to preserve border curves and arbitrary feature curves along edges of the mesh; this is a common requirement for terrain applications and when dealing with incomplete range scan data.

An important application of surface simplification today is in the generation of a level of detail hierarchy for use with view-dependent refinement [Xia and Varshney 1996; Hoppe 1997; Luebke and Erikson 1997], finite element methods such as radiosity [Willmott et al. 1999], or digital geometry processing [Lee et al. 1998]. A key component of such a level of detail hierarchy is a correspondence between vertices and/or faces at different levels of the hierarchy. Without this correspondence, refinement or coarsening of the mesh may require an expensive, non-local operation, and producing *geomorphs* [Hoppe 1996], arbitrary blends of two similar meshes, is difficult, if even possible. Section 6 shows how the permission grid enables such correspondences to be maintained, using simple operations on the grid to increase the effective ϵ as simplification proceeds. For example, given a model M_0, one could generate an approximation M_1 at ϵ_1, modify the grid to have a tolerance of ϵ_2, and continue simplifying M_1 to obtain M_2 with a maximal error of ϵ_2. In this way, a level of detail hierarchy may be augmented with error information, and vertex or face correspondences may be maintained throughout the simplification process.

2. TERMINOLOGY

For our purposes, a *polygonal mesh* is assumed to be a collection of vertices in three-dimensional space, and an associated collection of triangles composed of those vertices. A simplification algorithm is intended to simplify a polygonal mesh by reducing the number of triangles in the mesh, while retaining as much fidelity to the original mesh as possible, for some application-dependent definition of fidelity. An *error-bounded* simplification algorithm provides some sort of guarantee on the fidelity of the approximation. Typically, this involves a geometric guarantee, such as every point in the approximation being within some user-specified tolerance of the original mesh, or vice versa. Often, the output of a simplification algorithm is a number of successive approximations of a mesh, and not just a single model. In this case, each approximation is referred to as a *level of detail*, and the



(a) Vertex Removal (b) Edge Collapse

Fig. 2. Typical simplification operations. Iterative simplification algorithms proceed by repeatedly performing an operation over different neighbourhoods of the input mesh, in some heuristic order.

collection of all models is often called a *level of detail hierarchy*.

Most current simplification algorithms are iterative in nature, applying small, local changes to the mesh, each of which removes a small set of triangles and replaces it with a different, smaller set of triangles. The simplification operation applied varies from algorithm to algorithm, with the most popular being vertex removal [Schroeder et al. 1992; Cohen et al. 1996; El-Sana and Varshney 1997] and edge collapse [Garland and Heckbert 1997; Lindstrom and Turk 1998; Hoppe 1996; Guéziec 1996]. Vertex removal, shown in Figure 2(a), removes a vertex and all triangles incident upon it, and then triangulates the resultant hole in the model. Edge collapse, illustrated in Figure 2(b), combines the two vertices of an edge into one, removing all resulting degenerate faces and edges. The order in which operations are applied is a key differentiating feature among algorithms, and is typically based upon weighting each potential operation, and greedily performing the operation with lowest cost. With edge collapse algorithms, there is also the question of where to place the collapsed vertex. *Subset placement* requires it to be in the same position as one of the original vertices. Other alternatives include allowing positions anywhere along the edge, or anywhere at all in space (which usually is guided by some heuristic, such as minimizing quadric error [Garland and Heckbert 1997]).

Of particular relevance to a number of simplification algorithms is the concept of a *manifold*. Every neighbourhood on a manifold surface is topologically equivalent to a disk. A *manifold with boundary* is a manifold everywhere, except in some places where it is topologically equivalent to a half-disk.

In practice, few complex polygonal datasets are manifolds or manifolds with boundaries. Virtually all models, regardless of origin, contain a number of *mesh degeneracies*, such as T-junctions or inter-penetrating triangles. Just identifying mesh degeneracies can be problematic, as the imprecision of floating point arithmetic can become an issue. For example, very thin areas of the model may contain inter-penetrating triangles solely due to the limited precision of the coordinate representations. Even once degeneracies are identified, it can be difficult to correct them.

Another relevant property of a mesh is whether it is *orientable*. A mesh is orientable if all of the face normals of a mesh may be oriented consistently. Although not as common as other mesh degeneracies, there are meshes which are not orientable, with the most common example being a Möebius strip.

3. RELATED WORK

A number of related methods for error-bounded simplification have been published in recent years. This section explores some of these previous methods and their relation to

permission grids, concentrating on error-bounded and volumetric methods.

3.1 Error-Bounded Simplification

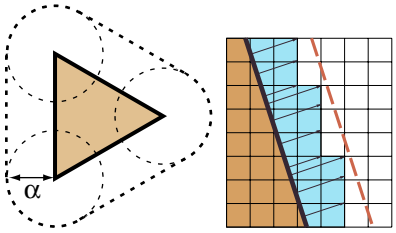
Most error-bounded methods can be broadly classified into the following categories:

3.1.1 *Vertex Clustering.* One of the earliest methods to guarantee an error bound, vertex clustering [Rossignac and Borrel 1993] groups vertices into clusters according to a regular grid, and reduces each cluster to a single vertex by computing a weighted average of the vertices in the cluster. Each vertex is then guaranteed to move by at most the diameter of its cluster, and the method is one of the fastest, but it produces noticeable gridding artifacts. By weighting vertices first, and clustering around highly-weighted vertices [Low and Tan 1997], some gridding artifacts can be reduced, but the results are still relatively low quality. The idea has recently been extended to large out-of-core datasets [Lindstrom 2000], using quadric error to position the vertex for each cluster. For all of these methods, in order to achieve a signficant degree of simplification, the cell size must be relatively large, so the error bound achieved may not be useful. Since large changes in the mesh are generated at each step, these methods are not suitable for continuous level of detail generation.

3.1.2 *Planar Approximations.* Superfaces [Kalvin and Taylor 1996] clusters faces of the input model into nearly co-planar sets, and replaces each of the face clusters by a single planar polygon. A bounded error is enforced by requiring all vertices in a cluster to lie within some ϵ of the approximating plane of the polygon. The method cannot simplify the topology, and since it induces large-scale changes to the mesh at each step, it is likely not suitable for generating continuous level of detail models.

3.1.3 *Local Reprojection.* These methods provide conservative bounds by locally reprojecting the original mesh vertices onto the new mesh after each simplification operation, requiring the projected vertices to never lie too far from their original positions. These methods are input-sensitive, as potentially all of the original mesh data must be reprojected at each step. None of these methods is designed to handle topological changes, and only Jade [Ciampalini et al. 1997] can simplify non-manifold meshes, but it only approximates its error bound. With significant computational effort, the two-sided Hausdorff distance may be enforced [Klein et al. 1996], as well as a complete mapping of the surfaces before and after each simplification step [Bajaj and Schikore 1996; Cohen et al. 1997]. Alternately, the one-sided Hausdorff distance from the original mesh vertices only, to the approximation, may be enforced [Kobbelt et al. 1998], a bound complimentary to ours. Another method [Ronfard and Rossignac 1996] only projects vertices onto supporting planes defined by each adjacent face, but the error bound achieved is dependent on the dihedral angles of the faces of the input.

3.1.4 *Tolerance Volumes.* This class of method is conceptually based upon constraining surfaces to lie within certain prescribed volumes. Simplification Envelopes [Cohen et al. 1996] offsets each vertex along its normal directions to construct two offset surfaces, and intersects new faces with these envelopes to determine validity. The method is input-sensitive, as each envelope is the size of the original mesh, and it requires a manifold, orientable mesh for the envelopes to be well-defined. It also cannot handle topological changes or create continuous level of detail models. Guéziec [Guéziec 1996] places error spheres at each vertex, which represent the current error at that vertex and grow as simpli-



(a) (b)

Fig. 3. Approximating α-prisms. (a) A two dimensional α-prism. The dashed area encloses the α-prism of the triangle. In three dimensions, the prism extends into and out of the page, rounded along each edge. For our purposes, $\alpha = \epsilon$, usually. (b) Our discrete approximation causes the effective error tolerance to vary. The brown triangle's α-prism extends to the dashed red line, but only the blue voxels are occupiable.

fication proceeds, such that linearly interpolating the spheres of each triangle is guaranteed to enclose the original mesh. However, the method is quite complex, appears computationally expensive, and cannot handle changes in topology. Note that both of these methods use surface-based representations for the tolerance volume.

The permission grid can be viewed as a tolerance volume method, according to the above classification. The shape of the volume is a discretization of the union of the set of α-*prisms* [El-Sana and Varshney 1997] of the mesh (see Figure 3(a)). The α-prism of a triangle is the Minkowski sum of the triangle and a sphere of radius α. Thus, the permission grid volume is similar to Guéziec's tolerance volume if all vertex tolerances are equal. El-Sana and Varshney use α-prisms to perform controlled topological simplification of a mesh, constructing the union of α-prisms in topologically-interesting regions, and filling in holes subsumed in the union. In this way, only topological features smaller than α are destroyed. The complexity of this union of α-prisms can be as bad as the complexity of the input, however, making their algorithm input-sensitive.

3.2 Volumetric Methods

In contrast to other tolerance volume methods, the permission grid uses an explicit volumetric representation. Other methods making use of explicit volumetric representations for simplification use the volume primarily for topological control.

One of the first methods to voxelize the input mesh [He et al. 1996] uses multi-resolution sampling and filtering, followed by an adaptive marching cubes algorithm to extract approximations. However, the method is restricted to closed surfaces, and cannot generate continuous level of detail models, since each model is extracted separately.

More recent work [Nooruddin and Turk 1999] uses morphological operations on the voxelization of the input mesh to perform topological control and mesh repair. Afterward, standard marching cubes is applied to extract an approximation, which often requires further topology-preserving simplification. The quality of results is excellent, although it appears more work is performed than is strictly necessary.

We know of only one method, other than the permission grid, to take advantage of both a volumetric and surface representation of the object simultaneously [Andujar et al. 1999]. The mesh (which must be a closed surface) is voxelized into an octree, and simplification is performed by merging octree nodes. An error bound related to the cell diameters is

enforced by constraining a marching cubes-like algorithm to extract surfaces only within their octree cells. Although continuous level of detail generation is supported, the error bound must double between each level of detail, since the octree cell size doubles at each level of the octree. Their results give no indication of running times, and the method appears difficult to implement.

4. PERMISSION GRIDS

The permission grid, as outlined above, is simply a spatial occupancy grid, where each cell is occupied only if the surface is allowed to pass through that cell. For simplicity, each cell is assumed to be a uniform size, and the grid an axis-aligned box. By constructing an appropriate set of occupied cells, we can guarantee that the approximating mesh lies within a fixed distance of the original mesh. The overall algorithm is as follows:

(1) Determine the grid resolution (§4.1).
(2) Voxelize each original mesh triangle's α-prism. (§4.2).
(3) While there are candidate simplification operations:
    (a) Test all triangles generated by the next simplification operation, ensuring that every voxel intersected by them is *occupiable* (§4.3).
    (b) If any triangle failed, disallow the operation.
    (c) Otherwise, perform the operation.

Note that step 3 implies a certain amount of tie-in with the underlying simplification algorithm. A number of implementation details are summarized in Section 4.4. We discuss various features and drawbacks of the permission grid in Section 4.5.

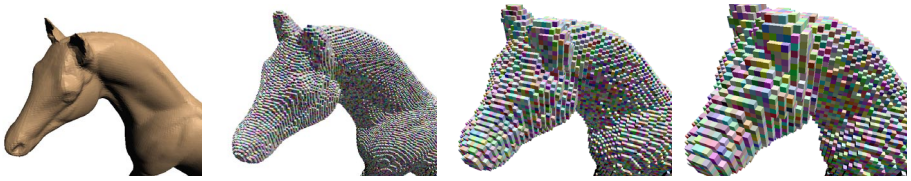
For the remainder of this paper, the tolerance, ϵ, is assumed to be some distance in object space. As is common practice, we typically specify ϵ as a percentage of the diagonal of the model's bounding box.

4.1 Permission Grid Resolution

The first question one must ask when dealing with grids is what resolution to use; in particular, we are interested in the voxel size, ℓ, given in terms of the length of any edge of the voxel, and how many such voxels will be required to contain the entire model's permission grid. In order to respect an error tolerance ϵ, an occupiable voxel in the permission grid must be entirely within ϵ of the original mesh. Since two points in a given voxel may be a distance apart of at most $\ell\sqrt{3}$, ℓ must be at most $\epsilon/\sqrt{3}$ in order to be able to fill any voxel through which the mesh actually passes. Otherwise, as demonstrated in Figure 5(b), voxels which actually contain the mesh will also contain points farther than ϵ from the mesh, and therefore can't be marked occupiable. This would be undesirable in general, having the effect of preventing any simplification at all in such areas. In addition, a coarser grid resolution leads to a wider variance in actual error tolerated over the mesh, as shown in Figure 3(b).

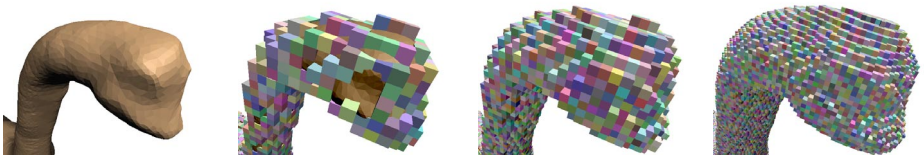
In our implementation, we leave the resolution to be specified by a *precision* parameter, ρ, which is given as the ratio between ϵ and voxel size, ℓ. The effects of varying ϵ and ρ on the shape of the permission grid are demonstrated in Figures 4 and 5, respectively.

Available memory places a practical limit on how fine the resolution of the grid may be, and this in turn places a limit on how small the error tolerance may be. We can derive an expression for the worst-case grid size for a given ϵ and ρ. The grid must be largest when



(a) Orig (b) $\epsilon = 1\%$ (c) $\epsilon = 2\%$ (d) $\epsilon = 3\%$

Fig. 4. Permission grid visualization at varying ϵ and fixed precision $\rho = 4$. Note how the occupiable regions inflate with larger tolerances, allowing a greater degree of simplification.



(a) Orig (b) $\rho = 1.25$ (c) $\rho = 2$ (d) $\rho = 4$

Fig. 5. Permission grid visualization at varying precisions on the dinosaur model. The tolerance is held fixed at 1%. The higher precisions fill in the rough edges on the lower precisions, allowing a greater freedom for the simplification algorithm. Note how with $\rho < \sqrt{3}$, voxels which contain the surface may not necessarily be occupiable, as shown in (b), where portions of the underlying mesh show through.

it has equal lengths in all dimensions; that is, when the bounding box is a perfect cube. Given ϵ as a fraction of the diagonal, and a precision ρ, ℓ by definition is ϵ/ρ. Scaling the bounding box diagonal to have a length of 1, each side of the bounding box has a length of $1/\sqrt{3}$. The resolution, R, of the grid on each side is simply the length of the side divided by the voxel size; therefore, we have:

$$R = \frac{\frac{1}{\sqrt{3}}}{\ell} = \frac{\rho}{\epsilon\sqrt{3}} \tag{1}$$

In our implementation, we use a full uniform grid, for simplicity, though other schemes such as octrees or *kd*-trees are possible as well. This requires one bit of storage per voxel, in order to distinguish between empty and occupiable voxels. Assuming a fixed precision of $2\sqrt{3}$, we can further derive an expression for the lowest possible ϵ given a memory limit of Ω bytes. The total memory usage is simply R^3 bits, so we have $\epsilon = \frac{1}{\sqrt[3]{\Omega}}$ is the minimum tolerance. Note that Ω is expressed in bytes. For example, given a memory limit of 256 MB, we are limited to an epsilon of approximately 0.155% in the worst case. Note it may not be quite this bad, since the bounding box is usually not an exact cube.

4.2 Permission Grid Creation

Once the resolution is decided, the permission grid is initialized to be empty, and the α-prism (see Figure 3(a)) of each mesh triangle is voxelized into the grid. Voxelization of such shapes is actually a well-studied area in recent years [Jones 1996; Huang et al.

1998], in the context of anti-aliased or transparent volume rendering. The shape of the 3D anti-aliasing filter for a triangle corresponds to the α-prism. We use a recent algorithm [Dachille and Kaufman 2000] to perform the voxelization. The basic idea of their algorithm is to traverse all voxels in the bounding box of a triangle, while maintaining distances to several planes oriented about the triangle; these distances change linearly over space, and can thus be updated with scan-line techniques. The signs of the distances tell us which component of the triangle (i.e., vertex, edge, or face) is closest to the voxel, so that a correct distance calculation can be applied. Note that since the distance is only calculated with respect to the center of the voxel, we cannot use a tolerance of ϵ when voxelizing. Each voxel contains points up to half the diagonal away from the center, which means any distance calculation with respect to the center can be off by up to $\ell\sqrt{3}/2$. Thus, we instead use a tolerance of $\alpha = \epsilon - \ell\sqrt{3}/2$ when voxelizing the α-prisms, to guarantee the correctness of the grid. An alternative would be to calculate at the voxel corners, and only mark as occupiable those voxels for which all corners were within the tolerance, but this is slightly more computationally and storage intensive. Note that the union of all α-prisms is calculated automatically by the permission grid representation; in effect, overlapping α-prisms simply have overlapping voxels marked twice.

4.2.1 *Two-Sided Error Bounds.* It is possible to modify the permission grid creation algorithm in order to produce an approximation which respects the full two-sided Hausdorff distance to the original mesh, at greater additional cost in memory and computation time. To do so requires an additional constraint on the underlying simplification algorithm: it must operate by taking a continuous patch of the surface, and replacing it with another continuous patch of the surface. In particular, it may not introduce holes or tears in the surface, or modify the topology (which, as we have seen above, is a typical requirement for maintaining a two-sided error bound).

In order to respect a two-sided error bound, two modifications are sufficient: any voxel touching the medial axis [Amenta et al. 1998] of the original mesh is cleared; and ϵ is reduced to $1/\sqrt{3}$ times its original value during voxelization. The proof of this assertion is provided in the appendix. Note that other modifications are possible, as suggested in our future directions (see Section 8).

While computing the cost of the medial axis is generally high, it is likely that a discrete approximation is sufficient. Further, it should be possible to produce such an approximation during the permission grid creation stage, by taking advantage of the distance values computed by the voxelization algorithm. And, of course, we only need compute the medial axis in occupiable regions of the grid. Even so, we have not implemented this feature, as we feel that enforcing the two-sided error bound loses many of the benefits of the permission grid algorithm: the possibility of topological simplification, the wide class of input surfaces, and the overall simplicity of the implementation.

4.3 Triangle Permission Testing

In order to guarantee that an approximation is faithful to the permission grid, we test every approximating triangle as it is generated, and disallow any operation which would create a violating triangle. In this section, we detail how the permission test is performed.

Testing whether a triangle is permitted by the grid amounts to voxelizing the triangle, ensuring no *empty* voxels are touched. This is much simpler than the grid creation, as we only need to examine voxels which are actually intersected by the triangle. Our implemen-

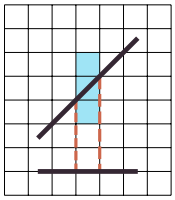


Fig. 6. 2D analogue to 3D scan-line triangle voxelization. In 2D, the problem is to rasterize a line segment, by projecting it onto a co-ordinate axis, and reprojecting segments of the axis into pixel columns, as shown. This shows the worst-case of reprojection, when the line has roughly equal projections onto each axis; in this case, as can be seen, up to 3 pixels need to be tested (in blue) due to numerical imprecision. The usual case will be only 1 or 2 pixels need testing.

tation is based on a 3D generalization of 2D scan-line rasterization [Kaufman and Shimony 1986], illustrated in two dimensions in Figure 6. First, the triangle is projected onto the 2D co-ordinate axes upon which its projection is largest, and then a 2D rasterization is applied. Each pixel generated by the 2D rasterization is "lifted up" into 3D, generating a small column of voxels to be tested. Small care must be taken to ensure all intersecting voxels are tested, since the standard algorithm attempts to generate a set of voxels which "look nice".

Note that the permission testing can be optimized somewhat by taking advantage of the fact that an entire neighbourhood of triangles typically needs to be tested at once, but the testing is sufficiently fast that we have not had need to do so.

4.4 Implementation Details

Our implementation uses a greedy edge collapse-based algorithm, QSlim [Garland and Heckbert 1997], for the underlying simplification engine. QSlim ranks all possible edge collapses by a cost metric, which is based primarily on the quadric error caused by the edge collapse. This can be understood as minimizing the sum of squared distances from the collapsed vertex position to the plane of each face incident to each of the vertices of the edge collapse. All edges are inserted into a priority queue, keyed on their cost, and the engine iteratively chooses to perform the edge collapse with the lowest cost. Once the engine selects an edge collapse to perform, QSlim would go ahead and perform the edge collapse, and then calculate new costs for all the edges in the neighbourhood of the collapse, adjusting their positions in the priority queue as necessary.

In order to adapt QSlim for the permission grid, two modifications are needed. First, before any simplification begins, the permission grid is constructed and the input model is voxelized using the creation algorithm given above. Secondly, after it has chosen an edge collapse to perform, and has updated the mesh to reflect the edge collapse, the permission grid testing algorithm is invoked. At this point, the permission grid voxelizes all of the triangles in the neighbourhood of the collapse. If any triangles violate the permission grid, then the collapse is rolled back, and QSlim continues with the next-lowest-cost edge collapse.

Note that if a given edge collapse is rejected by the permission grid, it is possible that it will be reconsidered and accepted later on in the course of simplifying the model. This is due to the recalculation of edge costs for all edges in the neighbourhood of a collapse; *all* edges in the neighbourhood are recalculated and re-inserted into the priority queue, in-

cluding edges which may have been rejected before. While this could be wasteful in some cases, it is entirely possible that the neighbouring collapse moves the violating triangles which caused the rejection into occupiable regions of the grid.

4.5 Discussion

The creation algorithm treats input triangles independently, which has a number of important consequences. Most importantly, it allows the permission grid to work in the presence of any sort of mesh degeneracies. A non-manifold edge simply has several voxels written more than twice, once from each adjacent face. Similar effects occur from T-junctions, non-manifold vertices, self-intersections, and non-orientable regions. All of these degeneracies are properties of an aggregate collection of triangles, which means they are irrelevant to our algorithm, which never examines triangles as aggregate collections.

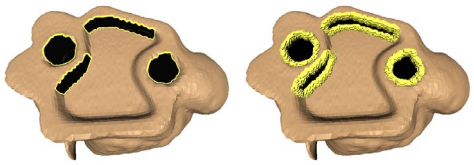
At worst, these kinds of degeneracies may cause a slightly odd shape to the permission grid in the neighbourhood of the degeneracy. This may cause the simplification algorithm to perform sub-optimally in these areas, but we have not observed any serious problems in practice. This is a significant improvement over current error-bounded simplification algorithms, virtually all of which depend on the connectivity and integrity of the mesh in some way, and refuse to operate on degenerate meshes at all. Note, however, that in order to gain this benefit of generality, the underlying simplification algorithm must be able to handle such mesh degeneracies as well. There are a number of fast algorithms [Garland and Heckbert 1997; Lindstrom and Turk 1998] which do not guarantee any error bounds and can operate on general meshes, and may therefore benefit from being coupled with a permission grid.

Additionally, the independent treatment of triangles means the voxelization may be parallelized, and even implemented in hardware [Dachille and Kaufman 2000], making further efficiency gains possible.

It is trivially possible to vary ϵ over the model in any user-specified manner [Cohen et al. 1996], to allow different portions of the model to be simplified to different error tolerances. The grid resolution must be set to handle the most restrictive tolerance, and then the tolerance may be changed on a per-face basis during the voxelization phase, essentially varying the α of the α-prisms.

In addition, the permission grid may be arbitrarily modified using well-founded principles of volume graphics and image processing, in order to accomodate other sources of information into simplification. For example, we may have a building embedded in a terrain in a non-trivial way, and wish to prevent interpenetration between the building and terrain during simplification of the building. This could be done by voxelizing the terrain after the building's permission grid is constructed, but clearing voxels during the voxelization, instead of marking them occupiable. All new approximations of the building would then be guaranteed never to penetrate a voxel containing the terrain, so no interpenetration could possibly occur.

Finally, it is important to note that the complexity of the grid is independent of the complexity of the input model. The permission grid representation is dependent mostly on the spatial occupancy of the model, so even a model of millions of small triangles may have a permission grid with a relatively small complexity. This independence on the input size is in contrast with previous simplification methods [Cohen et al. 1996; El-Sana and Varshney 1997], whose surface-based representations apparently cannot be reduced in complexity. In fact, the overhead of permission grids generally *decreases*, as a fraction of total runtime,



(a) (b)

Fig. 7. Border permission grids. (a) The underside of the Stanford bunny, with boundary edges in yellow. (b) The boundary grid visualized in yellow at 1% tolerance.

as the model size increases.

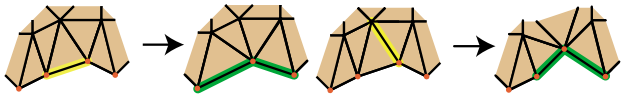
However, the permission grid may not be useful for certain classes of models. In particular, models with enough close, disjoint pieces of surface that the permission grid becomes space-filling may not find any advantage to using the permission grid. As previously discussed in Section 4.1, the grid complexity *is* dependent on the desired error tolerance. Smaller tolerances require larger grids, and increase the overhead significantly, both theoretically and in practice. However, the overhead is generally acceptable for a practical range of tolerances.

5. BOUNDARY PRESERVATION

For almost all applications, it is important that boundary edges be preserved during simplification. The permission grid as given is not enough to guarantee such preservation, as the boundary may retreat inside the grid arbitrarily. This depends on the behaviour of the underlying simplification algorithm, of course; QSlim, for example, includes a boundary preservation bias but lacks any formal guarantee of preservation. Alternately, one may simply want a smaller error tolerance around boundaries than across the entire model. Fortunately, it is easy to define a supplementary grid to handle such border curves. Figure 7 shows the borders on the Stanford bunny model, and the boundary grid constructed for them with a 1% tolerance. Note that these grids are shaped similarly to boundary envelopes [Cohen et al. 1996].

Boundary grids require the same care in terms of resolution and precision as regular permission grids. Instead of voxelizing triangles, however, we wish to voxelize the boundary edges. To do so, we adapt the grid creation algorithm, so that instead of maintaining distances to planes, we maintain the projection of the voxel center onto the line defined by the edge. With the line in the parametric form $(\mathbf{b} - \mathbf{a})t + \mathbf{a}$, where $\mathbf{a}$ and $\mathbf{b}$ are the endpoints of the edge, we maintain the t of the projection of the voxel center, which varies linearly over space. If $t \leq 0$, it's closest to $\mathbf{a}$; if $t \geq 1$, it's closest to $\mathbf{b}$; otherwise, it's closest to its projection on the line. Given the closest point on the edge, a distance test determines the voxel's occupiability. The algorithm otherwise proceeds in a manner similar to the regular grid creation algorithm.

Testing that a given edge doesn't violate the grid can be achieved with a simple 3D DDA algorithm [Fujimoto et al. 1986]. One problem, however, is deciding which edges should be tested; this is somewhat dependent on the simplification method used. The method outlined below is for edge collapse algorithms, though it should be fairly evident how to extend or adapt it for other kinds of simplification operations.



(a) (b)

Fig. 8. Propagating border vertices through an edge collapse. On the left, the yellow edges are about to be collapsed, in the neighbourhood of border edges and vertices marked in red. On the right, the mesh after the edge collapse is performed, with edges which would require testing marked in green.

First, every vertex adjacent to a border edge of the original mesh is designated as a border vertex. Then, any edge collapse which doesn't involve at least one border vertex need not be considered for border edge testing. This is because an edge collapse only changes the edges which are actually adjacent to one of the edge collapse's vertices. If neither vertex is a border vertex then by definition there can't be any border edges incident, and therefore no border will be changed. Otherwise, the new vertex is marked as a border vertex. As noted above, only edges incident to the new vertex could possibly need testing, and in fact, only edges with both vertices marked as border vertices require testing against the boundary grid. Both endpoints of the edge must be border vertices for the edge to be an actual border. The process is outlined in Figure 8, for the two main cases: collapsing an actual border edge, and collapsing an edge with one vertex on the border (the case of collapsing an edge with no vertices on the border is trivial, of course).

However, while this condition is sufficient for identifying a border edge, it is not necessary; any triangle which has two actual border edges will have all three vertices of the triangle marked border vertices. Therefore, the third edge will be incorrectly assumed to be a border edge in any edge collapse involving it. Since this only causes the algorithm to be more conservative in these areas, it is a minor issue in practice; these triangles usually denote sharp border corners and should be preserved in any case.

Note that this algorithm is trivially extensible to arbitrary feature curves over the surface of the mesh. A user could mark out a river or a political boundary along a terrain, for example, and ensure that it is not moved significantly during the simplification process.

6. LEVEL OF DETAIL HIERARCHIES

In this section, we explore how permission grids enable the efficient construction of level of detail hierarchies, with correspondences between vertices at different levels of detail. First, we review some key concepts in Section 6.1, and motivate the requirement for vertex correspondences. Then, Section 6.2 details the algorithmic details of generating level of detail models with a permission grid.

6.1 Vertex Hierarchies

One of the primary motivations for mesh simplification is to generate a level of detail hierarchy, such that meshes at different levels of detail can be smoothly blended for rendering and other applications. Such hierarchies require mappings between the vertices at different levels of detail to allow smooth transitions. Iterative simplification algorithms facilitate these mappings naturally, as each change to the mesh creates a correspondence between a small set of vertices and/or faces before and after the change. These mappings can be com-

bined and extended into a tree-like *vertex hierarchy*, where the nodes represent vertices, and the descendants of a node are all of the finer-level vertices which correspond to that vertex. Any cut through the vertex tree forms a valid mesh, so long as all simplification operations below the cut have been performed. This means local refinement or coarsening of the mesh may occur depending on the needs of the application. For example, a renderer need only descend through the hierarchy as far as is needed to achieve an accurate rendering, which may depend on how close portions of the mesh are to the viewpoint or to the silhouette [Hoppe 1997; Xia and Varshney 1996; Luebke and Erikson 1997].

6.2 Permission Grid Level of Detail Generation

A key problem with current error-bounded methods is that they are designed to do as much as possible, given a single error tolerance ϵ_1. To generate an approximation for a new, coarser tolerance ϵ_2, often requires starting from scratch. In doing so, correspondences between the ϵ_1 approximation and the ϵ_2 approximation are lost, since the simplification will almost certainly follow different paths in each case, as the algorithm takes advantage of the greater freedom of ϵ_2. Correspondences between vertices at different levels of detail are then lost, and are difficult to recover [Cohen et al. 1996].

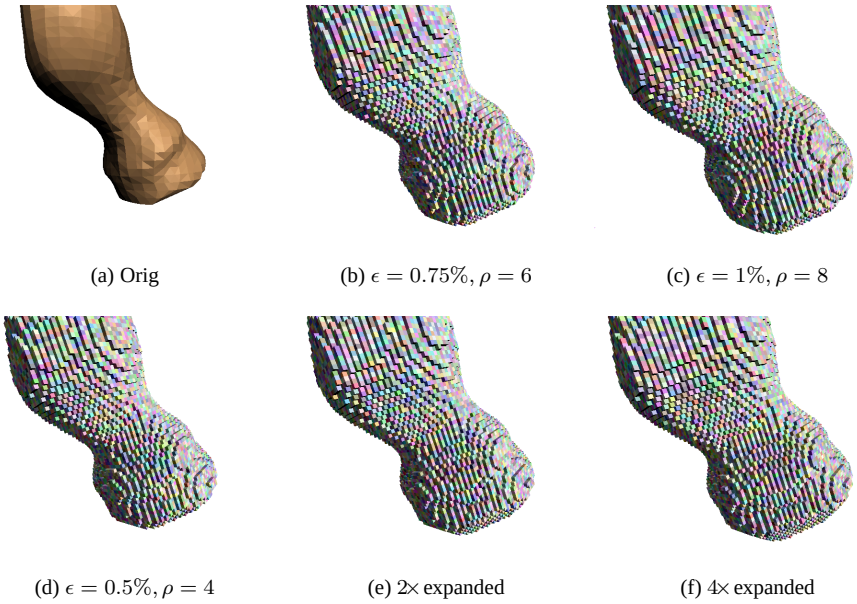
Permission grids, on the other hand, allow simple operations on the volumetric grid to be performed to increase the error tolerance, which means simplification may continue past the point at which the original tolerance prevents any further simplification. Once such an expansion has been performed, the underlying simplification algorithm is restarted, using the current approximation as the starting point. This means that the approximation for ϵ_2 is further simplified from the approximation for ϵ_1. In this way, the vertex correspondences remain natural throughout the entire simplification process, and an entire continuous level of detail hierarchy may be generated, with each level of the hierarchy augmented with error information as well.

The simplest method for increasing the error tolerance, which we have implemented, is to expand all of the occupiable grid cells by one extra voxel (simpler yet would be to revoxelize the entire grid at the new tolerance, but that would be very expensive). For every grid cell which is already occupiable, we mark all 6 of its face-neighbours as occupiable as well, increasing the current ϵ of the permission grid by the voxel size. One drawback to this approach is that it is slightly conservative, as we are effectively using the L_1 norm (Manhattan distance) rather than the L_2 norm (Euclidean distance), which results in some inaccuracy, but at least errs on the side of conservatism. However, the inaccuracy is minor; Figure 9 shows the grids resulting from expanding ϵ in the manner described above, compared to those generated from scratch, on a close-up view of a hoof of the horse model.

In our current implementation, the expansion may be invoked whenever simplification stops due to all candidate simplification operations being rejected by the permission grid. Other schemes are certainly possible, such as expanding once the proportion of rejected operations to allowed operations becomes high. The appropriate scheme again seems rather application- and model-dependent.

7. RESULTS AND DISCUSSION

We have implemented the permission grid to work in conjunction with Michael Garland's QSlim software, which uses the quadric error metric [Garland and Heckbert 1997] to guide simplification through edge collapse operations. For simplicity, we have chosen to represent the grid as a full, uniform voxel grid, using 1 bit per voxel. All timings presented in



(a) Orig (b) $\epsilon = 0.75\%, \rho = 6$ (c) $\epsilon = 1\%, \rho = 8$

(d) $\epsilon = 0.5\%, \rho = 4$ (e) $2\times$ expanded (f) $4\times$ expanded

Fig. 9. Conservatism of Grid Expansion. (a) The original model, for reference. (b, c) Permission grids generated at tolerances of 0.75% and 1%, respectively, with precisions of 6 and 8, in order to have equal voxel sizes with (d–f). (d) Permission grid at 0.5%, used as a base for expansion. (e, f) Expanded versions of (d), to give effective tolerances of 0.75% and 1% respectively. Note that while there are definitely occupiable voxels from (b, c) missing in (e, f), the differences are relatively difficult to note, only really noticeable in the shapes of the "top-most" contours near the top center and bottom right.

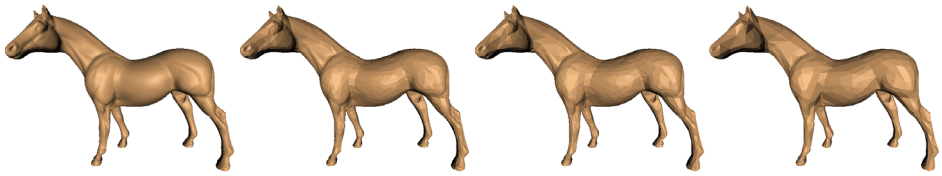
this section were performed on an Intel Pentium III 933 Mhz PC, with 512 MB of RAM, running Windows 2000. Maximum and average error values were calculated using a sampling approach; distances were calculated between the original mesh and vertex set of the approximation, plus a set of 10 random samples per face of the approximation. Error tolerances and values are specified as a percentage of the diagonal of the model's bounding box, and average errors are given as root-mean-squared (RMS) errors. All figures are flat-shaded to highlight mesh structure. Note that we show coarse models since it is only at extreme levels of simplification that clear differentiations among algorithms can be made; an experiment showed collapsing a random sequence of edges, at small to medium degrees of simplification, can often generate adequate results.

First, we explore the quality of results as the parameters of the permission grid are changed. Figure 10 shows the results of simplifying a horse model as ϵ is varied from 0.175% to 2%, while the precision ρ is held fixed at 4. Numerical statistics are summarized in Table I for this model. Since finer grid resolutions are required with smaller tolerances, more data must be dealt with, and thus the algorithm runs slower as ϵ decreases. The limit of practical tolerances on our test machine were generally in the range of 0.15% for the models we tested. Given that many datasets today tend to be vastly over-sampled range or volume data on which significant simplification is required, we haven't found this to be a problem.

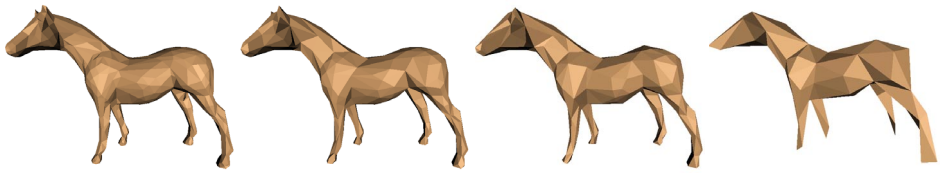
In contrast, we hold ϵ fixed at 1% and vary ρ from $\sqrt{3}$ to 16 on a dinosaur model

Table I. Simplification of 96,966 face horse model with varying tolerance ϵ and fixed precision $\rho = 4$.

			Error		
ϵ	Resolution	Face Count	Max.	RMS	Time (s)
0.175	$840 \times 1744 \times 1464$	4060	0.14	0.041	89
0.2	$744 \times 1536 \times 1288$	3466	0.15	0.054	76
0.333	$480 \times 952 \times 808$	1984	0.24	0.090	44
0.5	$352 \times 664 \times 568$	1250	0.39	0.14	32
0.75	$264 \times 472 \times 408$	752	0.53	0.21	26
1	$216 \times 376 \times 328$	486	0.79	0.27	23
2	$152 \times 232 \times 208$	148	1.5	0.50	19



(a) Orig (b) 0.175% (c) 0.2% (d) 0.333%

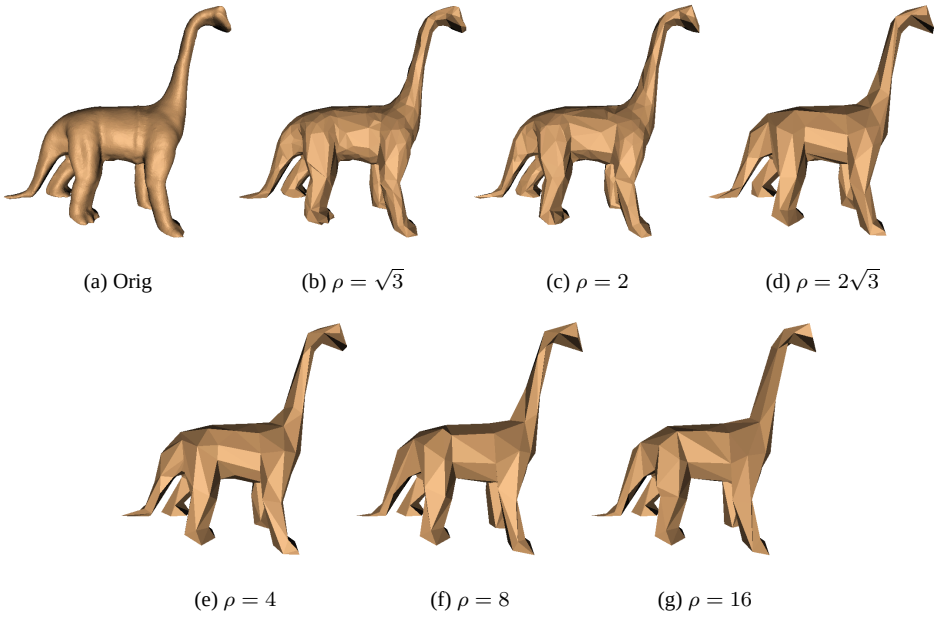


(e) 0.5% (f) 0.75% (g) 1% (h) 2%

Fig. 10. Simplification results at varying tolerances ϵ, while precision is held fixed at 4. Detailed statistics are in Table I.

Table II. Simplification of 47,904 face dinosaur model with varying precision ρ and fixed tolerance $\epsilon = 1\%$. Note the diminishing utility of higher precisions; running times increase significantly and result in much smaller improvements in error fidelity.

			Error		
ρ	Resolution	Face Count	Max.	RMS	Time (s)
$\sqrt{3}$	$160 \times 152 \times 80$	1226	0.52	0.078	4.4
2	$184 \times 176 \times 96$	942	0.59	0.10	4.9
$2\sqrt{3}$	$320 \times 304 \times 160$	318	0.70	0.20	9.2
4	$368 \times 352 \times 184$	282	0.73	0.22	11.9
8	$728 \times 696 \times 360$	228	0.85	0.28	66
16	$1456 \times 1384 \times 712$	214	0.89	0.31	489



(a) Orig (b) $\rho = \sqrt{3}$ (c) $\rho = 2$ (d) $\rho = 2\sqrt{3}$

(e) $\rho = 4$ (f) $\rho = 8$ (g) $\rho = 16$

Fig. 11. Simplification results with varying precision ρ, while the tolerance is held fixed at 1%. Note that the higher precisions allow greater freedom for the simplification algorithm, and thus greater degrees of simplification. Detailed statistics are in Table II.

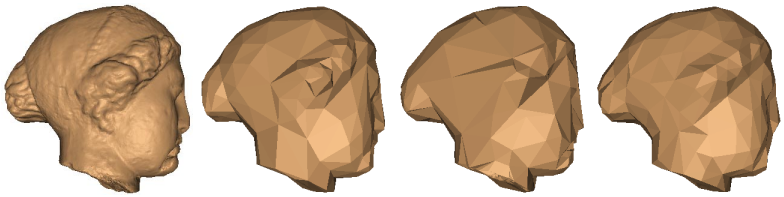
in Figure 11, with numerical statistics in Table II. As can be expected, the very coarse precisions do not allow nearly so much simplification to occur, and stop well short of the error bound, while the higher precisions slow the algorithm fairly significantly. There is a typical quality/time trade-off inherent in the use of permission grids.

Next, we present a comparison between our method, Simplification Envelopes [Cohen et al. 1996], and regular QSlim [Garland and Heckbert 1997]. We chose simplification envelopes since it offers a similarly strong error bound as the permission grid, and QSlim to show the effect the permission grid has on the underlying simplification algorithm; also, source code for both of these is freely available. All comparisons were done using an error tolerance of 1%, and a precision of 4 in the permission grid (we have found a precision of 4 to offer a good trade-off between quality and speed/memory usage). To compare against QSlim, which has no error bounds, we generated QSlim approximations with the same face count as the permission grid-generated approximation.

Table III presents the details of the results, showing that permission grids provide a strong quantitative improvement; up to an order of magnitude faster, with roughly comparable or better results in terms of error quality, over Simplification Envelopes. Note that the average error values on the Venus and Isis models are worse, for the permission grid, as it achieved significantly more simplification than simplification envelopes. An interesting thing to note is that the running time for the permission grid increases with finer tolerances, while simplification envelopes becomes faster. This is likely due to the numerical computation of the envelope having less potential for problems from self-intersections at the smaller tolerances. However, the larger models clearly show that the running time of

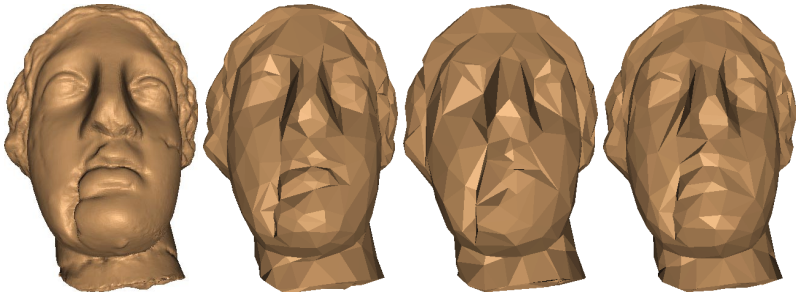
Table III. Performance comparison of Simplification Envelopes, Permission Grids, QSlim, and Subset-placement-based Permission Grids and QSlim, with tolerances as noted and precision $\rho = 4$.

		Error		
Algorithm	Face Count	Max.	RMS	Time (s)
Bunny (69,451 faces), $\epsilon = 1\%$				
Simplification Envelopes	529	0.99	0.37	178
Permission Grid	489	0.70	0.22	19.2
QSlim	488	2.0	0.26	4.7
Subset Permission Grid	687	0.74	0.24	19.4
Subset QSlim	687	1.6	0.29	4.4
Venus (268,714 faces), $\epsilon = 0.5\%$				
Simplification Envelopes	1014	0.49	0.18	761
Permission Grid	1498	0.36	0.13	81
QSlim	1498	0.87	0.16	21
Subset Permission Grid	1388	0.35	0.12	80
Subset QSlim	1388	0.86	0.15	18
Venus (268,714 faces), $\epsilon = 1\%$				
Simplification Envelopes	832	0.99	0.25	945
Permission Grid	596	0.75	0.28	61
QSlim	596	1.6	0.32	21
Subset Permission Grid	504	0.73	0.24	59
Subset QSlim	504	1.8	0.31	18
Teeth (233,204 faces), $\epsilon = 1\%$				
Simplification Envelopes	990	0.97	0.26	945
Permission Grid	706	0.83	0.28	57
QSlim	706	1.6	0.32	18
Subset Permission Grid	670	0.75	0.25	55
Subset QSlim	670	1.6	0.32	16
Isis (375,736 faces), $\epsilon = 1\%$				
Simplification Envelopes	2198	0.94	0.12	2303
Permission Grid	276	0.69	0.25	77
QSlim	276	1.1	0.28	30
Subset Permission Grid	348	0.78	0.24	76
Subset QSlim	348	1.6	0.29	27
Buddha (1,085,634 faces), $\epsilon = 1\%$				
Simplification Envelopes	— input model too large to process —			
Permission Grid	1230	0.80	0.25	219
QSlim	1230	1.7	0.29	97
Subset Permission Grid	1130	0.82	0.24	223
Subset QSlim	1130	1.6	0.31	88



(a) Orig (b) P.G. (c) S.E. (d) QSlim

Fig. 12. The head of the statue of Venus is simplified with $\epsilon = 1\%$. Note the better preservation of the curls and nose-line in the permission grid result.



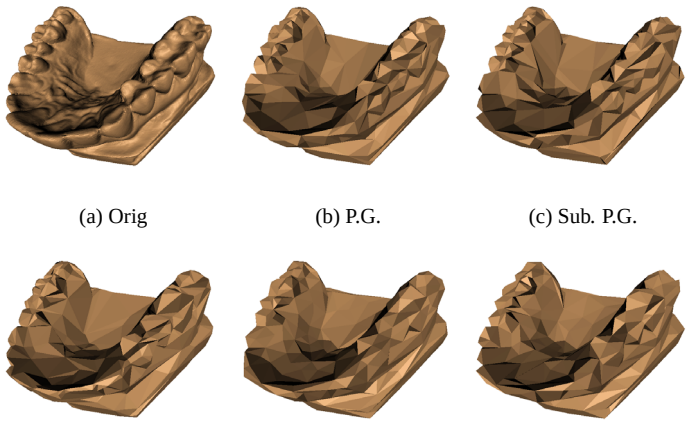
(a) Orig (b) P.G. (c) S.E. (d) QSlim

Fig. 13. The head of the statue of Venus is simplified with $\epsilon = 0.5\%$. The permission grid preserves extra detail along the cleft near the mouth, the lips, and the eyebrows.

simplification envelopes increases quickly with model size, while permission grids enjoy a roughly linear increase in runtime.

Figures 12 and 13 show qualitative results on the bust of Venus; note that our method in general produces higher quality approximations than simplification envelopes, though this is possibly an artifact of the fact that simplification envelopes uses vertex removal, and therefore the approximation uses only a subset of the original vertices of the model (subset placement), while QSlim attempts to optimize the mesh vertices according to their quadric error. Oddly enough, however, the error bounds and face count are actually better with subset placement for some models. Another comparison among the algorithms is shown on a dental model in Figure 14. Here, again, the permission grid approximation is clearly better than the other algorithms in terms of quality, with a mild overhead on runtime as compared to QSlim.

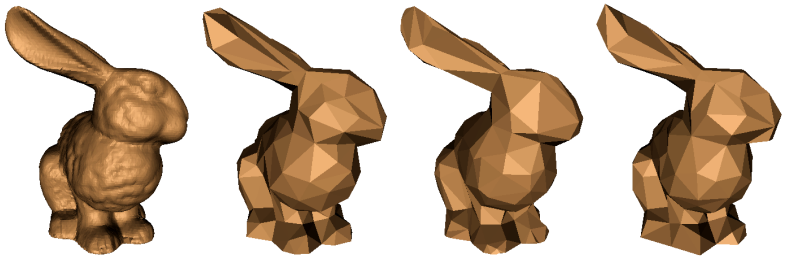
Figure 15 shows the Stanford Bunny model, with little qualitative difference between the various methods. However, as can be seen in Figure 16, the borders, which occur underneath the bunny, are handled much differently. Simplification Envelopes loses a lot of detail in the cavity, generally filling it in to form a flat bottom, though it does a good job in simplifying the border shapes. Permission grids, on the other hand, preserve the overall shape and border curves, including the cavity, quite well. Note that QSlim alone behaves



(a) Orig (b) P.G. (c) Sub. P.G.

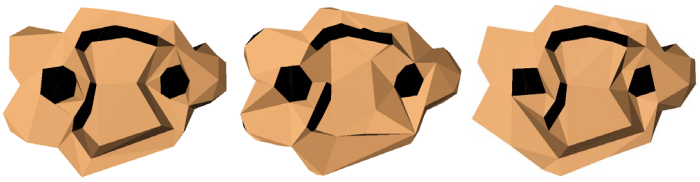
(d) S.E. (e) QSlim (f) Sub. QSlim

Fig. 14. Comparison of simplification results on a dental model. Note the shape of the teeth and the gum-line is more faithful to the original in the permission grid approximation. Also, the results generated using subset-placement for vertices are of slightly lower quality, but the differences are minor.



(a) Orig (b) P.G. (c) S.E. (d) QSlim

Fig. 15. Comparison of results on the bunny model. This model is generally quite smooth, and all methods work relatively well on it.



(a) P.G. (b) S.E. (c) QSlim

Fig. 16. Border maintenance on the underside of the Stanford bunny. The original is in Figure 7(a).

Table IV. Level of detail generation on the horse model. The model simplified at $\epsilon = 0.5\%$ was further simplified to greater tolerances, as shown, using the grid expansion method of Section 6. The increased effective precision (6, and 8, respectively) overcomes the inaccuracy of the expansion algorithm, and produces even lower face-counts than the previous results starting from scratch with lower precisions (see Table I).

		Error	
ϵ	Face Count	Max.	RMS
0.75	626	0.59	0.23
1	380	0.82	0.30

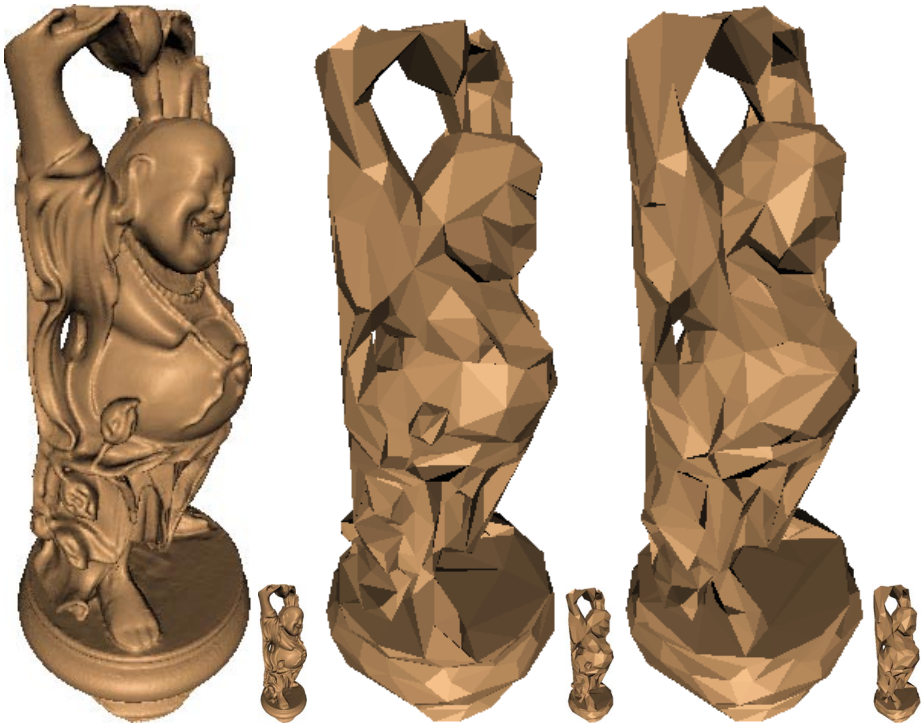
quite poorly in these border regions, tending to fill them in; this has been documented before [Lindstrom and Turk 1998], and as this result shows, permission grids can be an effective solution. Note that for these results, we have enabled QSlim's boundary plane constraints.

To explore the effectiveness of generating multiple levels of detail via our simple grid expansion method, we generated two successive approximations based upon the horse model, originally simplified at $\epsilon = 0.5\%$, with grid precision $\rho = 4$, as in Table I. Given the 1250 face model, we expanded the grid twice to an effective ϵ of 0.75%, and continued the underlying simplification algorithm. Once it stopped, we again expanded the grid twice, for effective $\epsilon = 1\%$, and again simplified as much as possible. Qualitatively, the results are similar to those obtained starting from scratch and simplifying to the given tolerances, Quantitative results are shown in Table IV. As was expected, the differences are minor, and in fact the increased effective precision completely offsets any conservatism in the expansion algorithm; the face counts are in fact lower with the expanded grid.

Lastly, Figure 17 shows results on a large model of a Buddha statue, with approximately 1.1 million faces. This was in fact too large for simplification envelopes to handle on our test machine, even if the model did not contain a number of non-manifold vertices and edges which prevented using simplification envelopes in the first place. In fact, this manifold requirement caused us some fair difficulty in finding models with which to compare at all; most publicly-available data contains such degeneracies. As can be seen, the permission grid preserves a greater amount of detail than QSlim alone, especially in the flowers along the leg and in the feet. Meanwhile, the permission grid only doubles the runtime, much less than the orders of magnitude of overhead of previous methods.

8. CONCLUSIONS AND FUTURE WORK

We have presented a novel method for guaranteeing a meaningful geometric error bound during simplification. The permission grid is useful for a practical range of error tolerances, with efficiency penalties modest enough to make it faster than any other method with similar error guarantees. Even ignoring the error bounds, the resulting approximations are qualitatively better with the permission grid than with a non-error-bounded method. That it can work over arbitrary triangular models, and can be expanded to create hierarchical error-bounded levels of detail with vertex correspondences, mark significant, novel improvements over current methods. Real-world data tends to contain all kinds of mesh degeneracies, and level of detail representations are becoming critical in enabling the solution or display of ever-increasingly complex problems or datasets. The permission grid



(a) Orig (b) P. Grid (c) QSlim

Fig. 17. The buddha model was too large (1.1 million faces) for simplification envelopes to handle. The flowers along the leg and the feet are better-preserved with the permission grid. In the thumbnails, the face and hands are more detailed in the permission grid, as well.

solves these problems yet remains conceptually simple and easy to implement, making it a practical, attractive technique for a variety of applications.

We see a number of avenues of related work to be pursued. One significant problem is the memory consumption of the grid; the most obvious solution is to use some method for compressing the grid. A simple run-length encoding may be sufficient, or sparse grid methods may work (in practice, we observe roughly a tenth of the cells are occupiable at 1% tolerance, which should naturally drop off at smaller tolerances). Multi-resolution representations or adaptive techniques such as octrees could yield vast improvements as well, at the cost of complicating the testing and construction algorithms: such representations would require fast clipping or box intersection algorithms, rather than voxelization algorithms.

Also, we believe there exist simpler and easier methods to extend the permission grid to allow a two-sided error bound, with fewer restrictions. In an analogous 2D algorithm for curve simplification, it is possible to reduce the tolerance by a factor dependent on the angle between line segments; this factor is often much less than the $\sqrt{3}$ reduction factor we have given. This should be extensible to examining the solid angle of the cone around a vertex in 3D.

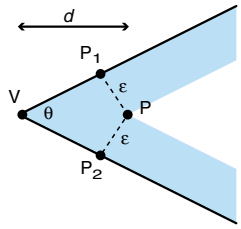


Fig. 18. Overlap within the permission grid. The lines indicate faces viewed from the side; the blue area is voxelized by the standard permission grid algorithm. The final approximation may be a distance of at most *d* from the original mesh edge between the two faces.

APPENDIX

Here, we sketch a proof of the fact that voxelizing the medial axis of the original mesh, and reducing the tolerance by a factor of $1/\sqrt{3}$ causes the permission grid algorithm to respect a global, two-sided Hausdorff error bound.

The primary cause of violation of the two-sided bound using the standard permission grid is from overlap between disjoint portions of the surface. For example, two disjoint but near portions of surface may result in a grid which is completely occupiable between them, and thus the surface may move twice the normal prescribed distance from its original position (in fact, it may move arbitrarily far, assuming enough disjoint pieces "lined up" in a row, and close together). Since a portion of the medial axis must lie between two disjoint pieces of surface (or, indeed, even between two folds in a surface), there is no way for this to occur once the voxels intersecting the medial axis have been cleared.

The other possible case for violation is in the presence of sharp dihedral angles between faces, or "spikes" in the mesh. In this case, the edge or vertex joining such faces may be moved farther than ϵ from their original positions, since there will be a fully occupiable path of such a length through the bisector of the angle, as shown in two dimensions in Figure 18. The distance which the edge or vertex may move (*d* in the Figure), is clearly $\epsilon/sin(\theta/2)$, from examination of triangle VPP_1. However, if $\theta \leq \pi/2$, the medial axis is guaranteed to lie near the surface, as an arbitrarily small empty ball can be wedged between the faces. Otherwise, the minimum of $sin(\theta/2)$ occurs at $\pi/2$, with a value of $1/\sqrt{2}$, which means the distance *d* can be most $\epsilon * \sqrt{2}$. Choosing ϵ to be $\epsilon'/\sqrt{2}$ limits this distance, and causes the grid to respect a global two-sided error bound of ϵ'.

Note that the generalization to three dimensions changes the $\sqrt{2}$ factor to $\sqrt{3}$; at a 3D vertex, the worst-case overlap where the medial axis does not touch the vertex occurs when three almost mutually-orthogonal edges are adjacent to the vertex, similar to a corner of a cube.

ACKNOWLEDGMENTS

Special thanks to Frank Dachille for the voxelization code, and to Jon Cohen and co-workers for releasing the simplification envelopes source code. Also thanks to the Stanford Computer Graphics Laboratory and Cyberware for the freely-available data-sets, and to the anonymous reviewers for their valuable comments and suggestions. This work was funded in part by a grant from the NSF (CCR-0086084).

REFERENCES

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new Voronoi-based surface reconstruction algorithm. *Proceedings of SIGGRAPH 98*, 415–422.

ANDUJAR, C., AYALA, D., AND BRUNET, P. 1999. The discretized polyhedra simplification (dps): A framework for polyhedra simplification based on decomposition schemes. Tech. rep., Universitat Politècnica de Catalunya. LSI-99-3-R, http://www.lsi.upc.es/dept/techreps/ps/R99-3.ps.gz.

BAJAJ, C. AND SCHIKORE, D. 1996. Error-bounded reduction of triangle meshes with multivariate data. *SPIE 2656*, 34–45.

BAREQUET, G. AND KUMAR, S. 1997. Repairing CAD models. In *Visualization*. Phoenix, AZ, 363–370.

CIAMPALINI, A., CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 1997. Multiresolution decimation based on global error. *The Visual Computer 13*, 5, 228–246.

CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 1998. A comparison of mesh simplification algorithms. *Computers & Graphics 22*, 1, 37–54.

COHEN, J., MANOCHA, D., AND OLANO, M. 1997. Simplifying polygonal models using successive mappings. In *Proceedings IEEE Visualization '97*. 395–402.

COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., BROOKS, F., AND WRIGHT, W. 1996. Simplification envelopes. In *SIGGRAPH '96 Proc.* 119–128. http://www.cs.unc.edu/~geom/envelope.html.

DACHILLE, F. AND KAUFMAN, A. 2000. Incremental triangle voxelization. In *Graphics Interface*. 205–212. http://www.graphicsinterface.org/proceedings/2000/108.

EL-SANA, J. AND VARSHNEY, A. 1997. Controlled simplification of genus for polygonal models. In *IEEE Visualization 97 Conference Proceedings*. 403–410.

FUJIMOTO, A., TANAKA, T., AND IWATA, K. 1986. Arts: Accelerated ray-tracing system. *IEEE Computer Graphics and Applications*, 16–26.

GARLAND, M. 1999. Multiresolution modeling: Survey & future opportunities. In *State of the Art Report*. Eurographics, 111–131. http://www.uiuc.edu/~garland/papers.html.

GARLAND, M. AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Proc.* 209–216. http://www.uiuc.edu/~garland/research/quadrics.html.

GUÉZIEC, A. 1996. Surface simplification inside a tolerance volume. Tech. rep., Yorktown Heights, NY 10598. Mar. IBM Research Report RC 20440, http://www.watson.ibm.com:8080/search_paper.shtml.

GUSKOV, I. AND WOOD, Z. 2001. Topological noise removal. In *Proceedings of Graphics Interface 2001*. To appear. http://www.cs.caltech.edu/~ivguskov/papers.html.

HE, T., HONG, L., VARSHNEY, A., AND WANG, S. 1996. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (June), 171–184.

HOPPE, H. 1996. Progressive meshes. In *SIGGRAPH '96 Proc.* 99–108. http://research.microsoft.com/~hoppe/.

HOPPE, H. 1997. View-dependent refinement of progressive meshes. In *SIGGRAPH 97 Proc.* 189–198. http://research.microsoft.com/~hoppe/.

HUANG, J., YAGEL, R., FILIPPOV, V., AND KURZION, Y. 1998. An accurate method to voxelize polygonal meshes. In *1998 Volume Visualization Symposium*. 119–126.

JONES, M. W. 1996. The production of volume data from triangular meshes using voxelization. *Computer Graphics Forum 15*, 5, 311–318.

KALVIN, A. D. AND TAYLOR, R. H. 1996. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl. 16*, 3 (May). http://www.computer.org/pubs/cg&a/articles/g30064.pdf.

KAUFMAN, A. AND SHIMONY, E. 1986. 3d scan-conversion algorithms for voxel-based graphics. In *Proc. 1986 Workshop on Interactive 3D Graphics*. 45–75.

KLEIN, R., LIEBICH, G., AND STRASSER, W. 1996. Mesh reduction with error control. In *Proceedings of Visualization '96*. 311–318.

KOBBELT, L., CAMPAGNA, S., AND PETER SEIDEL, H. 1998. A general framework for mesh decimation. In *Proc. Graphics Interface '98*. 43–50.

LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proc. SIGGRAPH 98*. 95–104.

LINDSTROM, P. 2000. Out-of-core simplification of large polygonal models. *Proceedings of SIGGRAPH 2000*, 259–262.

LINDSTROM, P. AND TURK, G. 1998. Fast and memory efficient polygonal simplification. In *IEEE Visualization 98 Conference Proceedings*. 279–286,544.

LOW, K.-L. AND TAN, T.-S. 1997. Model simplification using vertex-clustering. In *1997 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH. http://www.iscs.nus.sg/~tants/.

LUEBKE, D. AND ERIKSON, C. 1997. View-dependent simplification of arbitrary polygonal environments. In *SIGGRAPH 97 Proc*. 199–208.

NOORUDDIN, F. S. AND TURK, G. 1999. Simplification and repair of polygonal models using volumetric techniques. Tech. Rep. GIT-GVU-99-37, GVU Center, Georgia Institute of Technology. http://www.gvu.gatech.edu/gvu/reports/1999/abstracts/99-37.html.

RONFARD, R. AND ROSSIGNAC, J. 1996. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum 15,* 3 (Aug.). Proc. Eurographics '96.

ROSSIGNAC, J. AND BORREL, P. 1993. Multi-resolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Applications*, B. Falcidieno and T. Kunii, Eds. 455–465.

SCHROEDER, W. J., ZARGE, J. A., AND LORENSEN, W. E. 1992. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proc.) 26*, 2 (July), 65–70.

WILLMOTT, A. J., HECKBERT, P. S., AND GARLAND, M. 1999. Face cluster radiosity. In *Eurographics Workshop on Rendering*. http://www.cs.cmu.edu/~ajw/paper/fcr-eg99/.

XIA, J. C. AND VARSHNEY, A. 1996. Dynamic view-dependent simplification for polygonal models. In *Proceedings of Visualization '96*. 327–334.

Received September 2001; revised January 2002; accepted February 2002