A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes

Mario Botsch, Leif P. Kobbelt

Computer Graphics Group, RWTH Aachen,
{kobbelt,botsch}@cs.rwth-aachen.de

Abstract

When using triangle meshes in numerical simulations or other sophisticated downstream applications, we have to guarantee that no degenerate faces are present since they have, e.g., no well defined normal vectors. In this paper we present a simple but effective algorithm to remove such artifacts from a given triangle mesh. The central problem is to make this algorithm numerically robust because degenerate triangles are usually the source for all kinds of numerical instabilities. Our algorithm is based on a slicing technique that cuts a set of planes through the given polygonal model. The mesh slicing operator only uses numerically stable predicates and therefore is able to split faces in a controlled manner. In combination with a custom tailored mesh decimation scheme we are able to remove the degenerate faces from meshes like those typically generated by tesselation units in CAD systems.

1 Introduction

In recent years computer graphics techniques have become increasingly popular in engineering applications. Algorithms for the efficient processing of polygonal meshes which have become available through the adaption of multiresolution concepts to geometric object representations, can be used to speed up the computation of all sorts of numerical procedures for simulation and interaction. Traditionally triangle or quadrangle meshes are the preferred surface representation for most numerical simulations.

If a mesh is to be used for numerical computation, robustness is essential for the results to be reliable. This robustness can only partially be guaranteed by developing stable algorithms, the input data also has to meet some quality requirements. We will show several ways how to measure the quality of a triangle mesh in section 2. With simulation in mind, our quality criterion will strongly relate to the shape or roundness of the individual triangles.

Because 3D range scanners are getting cheaper as well as increasingly accurate, they are a popular source for input data. A lot of research has been done on reconstructing polygonal mesh models from sample points or range images [16, 4, 2, 1]. Because of the huge number of faces in the resulting meshes, some kind of mesh decimation scheme has to be used in order to reduce the model to a complexity the simulation can handle.

In most engineering applications surfaces are designed by using sophisticated CAD/CAM systems. Hence they have to be converted to a polygonal mesh representation in order to be used in numerical simulations — an option included in almost every CAD software. These tesselation modules produce a polygonal mesh approximating the geometric shape within a prescribed error tolerance.

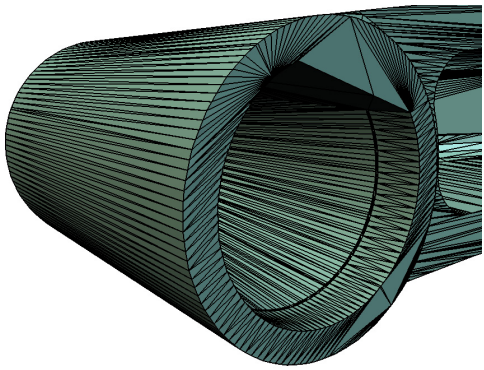

Figure 1: This tesselation of a cylindrical part has a low polygon count and a small approximation error, but unfortunately also very skinny triangles.

There are many ways to approximate a given geometrical shape as a polygonal mesh. Both mesh

decimation and surface tesselation will guarantee a certain approximation error, but even within this tolerance volume one can optimize the mesh according to different requirements. For example a low polygon count is preferrable, if the mesh has to be rendered as fast as possible. On the other hand, this may lead to skinny triangles and makes it almost impossible to get reliable simulation results for such meshes. While one has the possibility to trim the mesh decimation process to aim for well shaped triangles, one has no influence on the tesselating algorithm of the CAD software in use. Unfortunately these methods usually produce meshes not suitable for robust computations, like the cylindrical part shown in Fig. 1.

Therefore a polygonal mesh has to be prepared for the target application, i.e. it has to be *remeshed*: the given mesh has to be resampled in such a way that the new triangulation still approximates the geometric shape, but also fullfills special requirements depending on the application [8].

One class of remeshing algorithms converts an arbitrary input mesh into a mesh with semi–regular subdivison connectivity [5, 13, 12]. Here remeshing is used to get the necessary structure to apply special multiresolution operators to the input geometry.

Mesh decimation schemes [7, 6, 11, 3] are a special class of remeshing algorithms: they change a given mesh in order to decrease its complexity, but do it by subsampling rather than more general resampling.

Some remeshing methods are based on a particle systems approach [9, 15]. Sample points are randomly distributed on the given input mesh, yielding a new tesselation. In a second step a global relaxation procedure balances the distribution of these sample points by shifting them on the surface.

Remeshing may also be achieved by using a distance field to convert a given mesh into a volume representation. Applying iso–surface extraction to get the zero–level surface will result in a remeshed version of the input mesh. In this case special care must be taken not to lose geometric features of the original mesh [17, 10].

We will concentrate on remeshing with the goal to prepare meshes for engineering simulations. This is why we want polygonal meshes without degenerate triangles. Because this type of remeshing is especially necessary for meshes containing many degenerate faces, the employed algorithms have to be particularly robust. For example one cannot rely on geometric information like area or normal vectors of the faces, because there is no way to evaluate them in a robust manner.

Most simulation packages already include a pre–processor that adapts the input mesh to the simulation's needs. Unfortunately even these pre–processors are often not able to deal with degenerate triangles. Other standard methods that prepare meshes for FEM simulations, like e.g. Delaunay–Refinement [14], heavily rely on geometric computations (like the center of the circum–circle) which become unstable for degenerate triangles.

In this paper we present a method that uses a combination of *mesh slicing* and *mesh decimation* to enhance a given polygonal mesh. In the first step all degenerate triangles that won't be removed by a mesh decimation process (so called *caps*) will be eliminated. A custom tailored mesh reduction algorithm removes the remaining skinny triangles while additionally taking the resulting triangles' shape into account. While these two steps won't necessarily produce meshes suitable for the simulation the resulting meshes do no longer contain degenerate triangles. Therefore standard techniques like the built–in pre–processor or Delaunay–Refinement methods are now able to process these meshes in a robust manner.

2 Mesh Quality Measures

A quality measure for triangle meshes should be adapted to the target application. As mentioned in the introduction one may aim at a low polygon count and smooth appearance for rendering purposes or at well shaped faces for numerically robust computations.

Different quality measures will lead to different tesselations of the same geometric shape even at the same triangle count: if the goal is smooth appearance triangles will need to be stretched along the direction of minimal curvature. For FEM simlations the shape or roundness of triangles is more important, because computations have to be done in a numerically robust way. Examples of these conflicting optimizations are given in Fig. 2.

Our goal is to get a triangle mesh with all faces well shaped. There are several ways to define the quality of a triangle's shape. The ratio of the short-

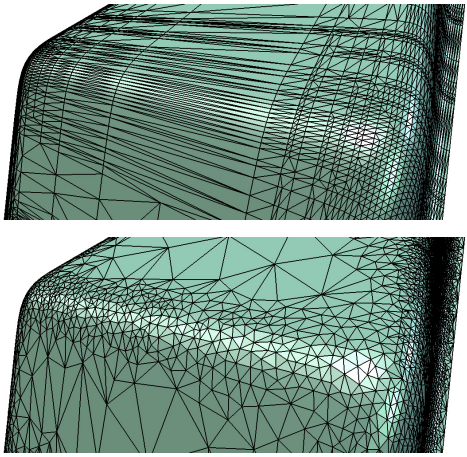

Figure 2: Two different quality measures: the same model optimized for smooth appearance, leading to stretched triangles (top), and for triangle shape, leading to surface shading artifacts (bottom).

est to the longest edge (*aspect ratio*) is a commonly used measure. In the mesh generation context the ratio of the shortest edge to the radius of the triangle's circumcircle is used to express the quality of a face [14].

Degenerate triangles have their shortest edge much shorter than their circumradius. These skinny triangles can be classified as

- *Caps*: triangles with an angle close to 180°.
- *Needles*: triangles, whose longest edge is much longer than the shortest one.

The definitions of caps and needles are not mutually exclusive (cf. Fig. 3). We will refer to caps as "pure caps", i.e. caps that are no needles.

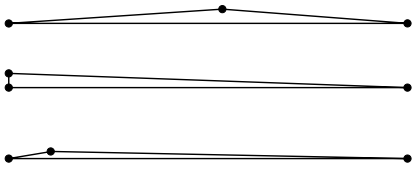

Figure 3: Classifications of skinny triangles: Caps have an angle close to 180° (top), needles a bad shortest to longest edge ratio (middle). A triangle can be both a cap and a needle (bottom).

Both types of degeneracies do not carry any relevant geometric information since their surface area is close to zero. It turns out that caps are much harder to eliminate than needles: While needles can usually be removed by simply collapsing their shortest edge, the situation for caps is more complicated, since collapsing their relatively long edges can cause neighboring triangles to degenerate. Hence an alternative operation is needed for the elimination of the caps. Once we manage to do this, a suitable mesh decimation algorithm will be capable of removing the remaining needles.

3 Caps Removal

Caps are defined as having their maximum angle close to 180°, leading to a relatively long edge on the opposite side of the triangle. As mentioned they cannot be removed by collapsing one of their edges. Therefore new points have to be inserted to split up faces resulting in smaller angles. In order to keep the mesh consistent also the neighboring triangle has to be split and must be taken into consideration (cf. Fig. 4).

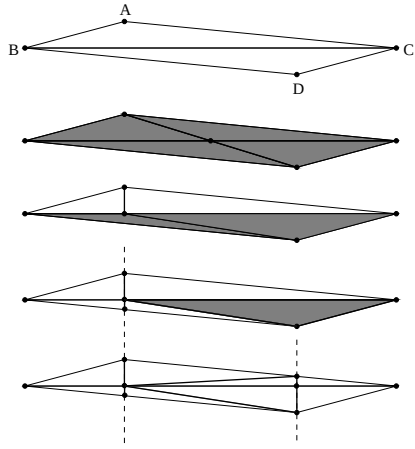

Figure 4: The two neighboring caps at the top can neither be eliminated by subdiving the base at its midpoint nor by splitting the edge at the orthogonal projection of the upper vertex, because this may lead to new caps, marked in grey. The only way is cutting the whole mesh at every cap vertex having a large angle, like A and D.

The first guess would be to subdivide the longest edge BC at its midpoint. Depending on the angle between the two new edges and the baseline this may introduce up to 4 new caps (marked in grey).

Splitting the edge BC at the othogonal projection of A creates a perpendicular edge, but also an arbitrarily large angle on the opposite side. This operation will remove the upper cap, but may split the lower one into 2 new caps.

If we define a plane perpendicular to the longest edge BC, centered at the point A, and use this plane to cut through the whole mesh the upper cap will certainly be eliminated. Cutting the cap BDC by this plane will split it into two needles and one cap. So the number of caps will decrease by one. By analogously cutting through D in a second pass we can eliminate this new cap and get only needles, but no caps, as depicted in the bottom drawing of Fig. 4.

In order to make the cutting algorithm robust, it will be applied globally and not only locally to the mesh. But then additional care must be taken not to create new caps at other regions of the mesh.

4 Mesh Slicing

If we want to utilize mesh slicing to remove caps, we first have to make sure that this operation can be performed in a stable way even for meshes containing degenerate faces. Also we have to guarantee that no new caps will inadvertently be created by cutting other triangles in the mesh.

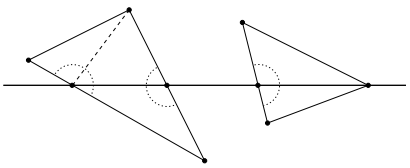

Figure 5: Cutting a triangle produces two or three new triangles.

Cutting a mesh by a given plane is a global operation that acts on the whole mesh. In a first pass all vertices are classified as being above, below or on the plane (up to some ϵ). In the second pass all faces are subdivided into two or three triangles, depending on whether one or two of their edges intersect the plane (cf. Fig. 5).

To test for an intersection the vertex distances are evaluated: if the distances at the edge's endpoints have different signs an intersection occurs and the intersection point can robustly be computed by a convex combination of the endpoints.

Although vertex classification is a geometric operation, it is robust, because it acts on vertices, not on faces. It also can be done consistently: Even if a vertex is classified incorrectly because of rounding errors, this information will be used by all incident faces in a consistent manner. Edge classification and actual triangle splitting are purely topological operations and not influenced by mesh degeneracies, therefore preserving the consistency of the mesh. Thus mesh slicing can be applied to degenerate meshes without any numerical instabilities.

We now must ensure that slicing the mesh in order to remove one cap won't create any new caps by splitting other triangles. If a vertex' distance to the plane is less than the snapping bound ϵ, no intersections will be computed for edges containing this vertex. We will use this *snapping* to avoid creating new caps. As depicted in Fig. 5 the angles at the original points of sliced triangles either stay the same or decrease. The only possibility for new caps to be generated are large angles at the intersection points (marked by arcs in Fig. 5). Large angles only occur if the intersected edge forms a large obtuse angle with the cutting plane. If we guarantee that no such edge will be intersected we can be sure that no additional caps will be created by slicing the mesh. Therefore we include an additional test to the vertex classification stage: for each vertex all emanating edges that intersect the plane are analyzed. If an edge builds an angle close to 90° with the plane normal the vertex will be snapped onto the plane, i.e. its distance will be set to zero thereby avoiding the critical intersection.

Now for each cap a suitable plane is used to slice the mesh and split this cap up into two needles. In order to ensure that this cap will be eliminated the endpoints of its longest edge are not allowed to snap onto the plane. A potentially bad edge emanating from one of these endpoints will only have one vertex snapped to the plane, resulting in an uncritical bisection.

Hence mesh slicing is a robust operation that eliminates (at least) one cap at a time. It may produce new skinny triangles, but these are guaranteed to be no caps and therefore have to be needles. Nee-

dles, in turn, can easily be removed by a mesh decimation process after all the caps have been eliminated.

5 Mesh Enhancement

Having a robust tool for cap elimination we can now formulate an algorithm for the automatic removal of degenerate faces from a mesh. In a first pass all caps are split up into needles using the slicing method:

```
for each cap do
  1) define a plane
     eliminating this cap
  2) classify all vertices,
     possibly snap them
  3) slice the mesh
done
```

In the resulting mesh all remaining degenerate triangles are needles and can be removed by a suitable mesh decimation algorithm. We use one that stays within a prescribed approximation error and controls the decimation process to optimize for well shaped triangles.

Mesh slicing can additionally be used to super–sample the object in order to provide more degrees of freedom for the decimation stage. In this case we slice the mesh by a set of manually or automatically constructed planes, thereby enhancing it by the new intersection points. This may be necessary if some needles cannot be collapsed without violating the error tolerance, e.g. because a fan of needles describes a trimming boundary of some detailed contour. Another example would be a cylinder like the one shown in Fig. 1, that may be cut by several planes perpendicular to its axis. For automatic slicing three perpendicular sets of evenly spaced axis–aligned planes can be used to cut the bounding cube into equally sized cells. This would provide sample points on a regular grid as well as control the maximum size of the triangles.

After we have eliminated all caps and provided enough degrees of freedom, the mesh decimation process can do the rest of the work, yielding a result that stays within a prescribed error tolerance and additionally satisfies the quality requirements with respect to triangle shape.

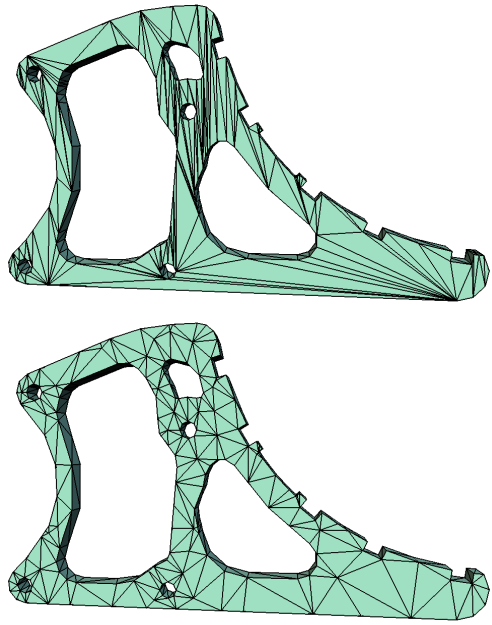

Figure 6: 700 triangle CAD model of a blower mount (5 needles, 20 caps) has been enhanced to a mesh without degeneracies and about 860 triangles.

6 Results

Both examples shown in this section have been converted by a CAD tesselator and contain the skinny triangles typical for this class of converters. The tolerance angles for caps and needles have been set to 175° and 5°, respecively.

The CAD model of a blower mount (700 triangles) initially contained 2 needles and 20 caps (cf. Fig. 6). Removing all caps by mesh slicing led to 2500 triangles, an automatic cutting by stacks of 20 planes in x–, y– and z–direction resulted in a mesh of 8000 faces, containing no caps, but about 1000 needles. The final decimation process eliminated all needles producing an approximation within 1 promille of the bounding box, consisting of 858 triangles. The mesh slicing step took less than 5 seconds, the mesh decimation about 1 second on a 866 Mhz Pentium III.

The second example is another CAD model, consisting of about 9200 faces that contain 226 needles and 77 caps (cf. Fig. 7). Here we started by slicing

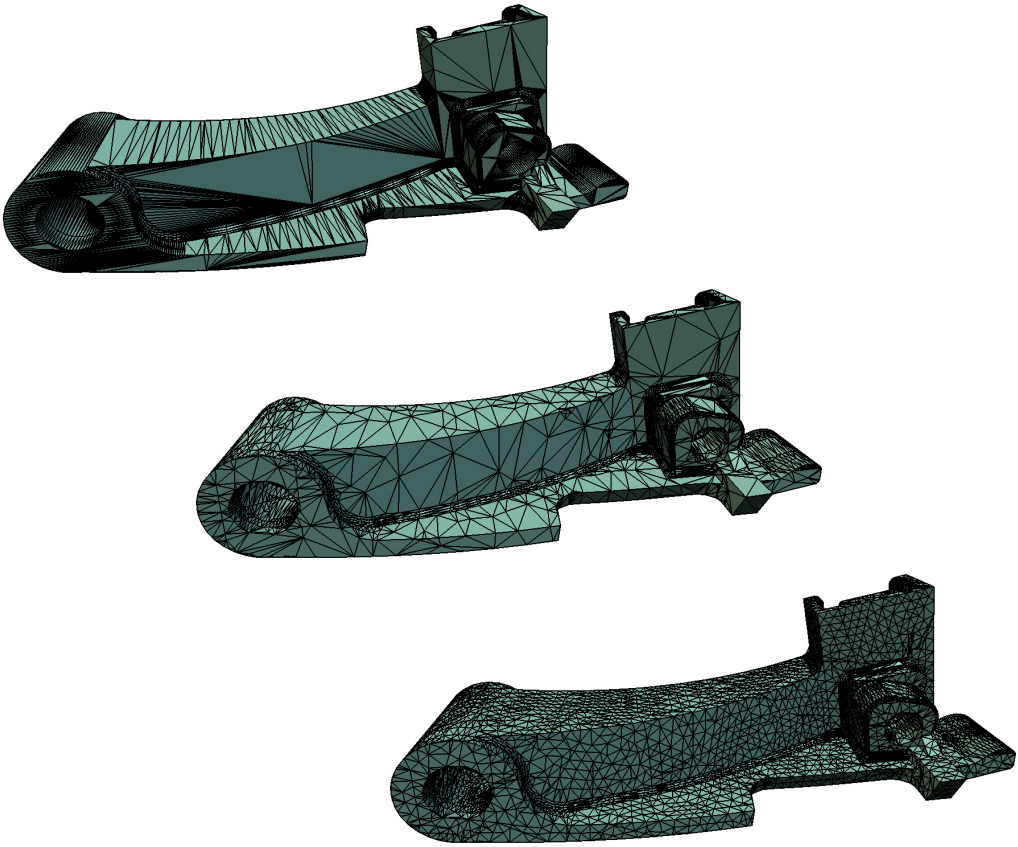

Figure 7: The upper mesh contains 226 needles and 77 caps in its 9200 faces; the enhanced version consists of 11k triangles and stays within an error bound of 1 promille. All angles are within 5 and 175 degrees. After all degeneracies have been removed the mesh can now be remeshed using standard techniques (e.g. in order to equalize triangle sizes).

3 stacks of 15 planes (11 sec) that removed 32 caps, and eliminated the remaining caps one by one in a second step (35 sec). The intermediate size of the model was about 130k triangles. The mesh reduction step removed all degeneracies in 18 seconds, yielding a mesh of about 11k triangles.

7 Conclusion

We presented a robust algorithm to remove both types of degenerate faces from triangle meshes in order to prepare them for reliable numerical simulations.

There are many ways to remesh a given input mesh for the use in numerical computations. Most of these methods cannot deal with mesh degeneracies. Since mesh slicing only uses robust geometric computations and uncritical topological operations it is not affected by these degeneracies and is able to remove them in a controlled manner.

While needles can be removed with relatively low effort, caps are much harder to eliminate. The proposed algorithm is guaranteed to remove all caps one by one by splitting them into needles, which subsequently are cleaned up by any suitable decimation scheme.

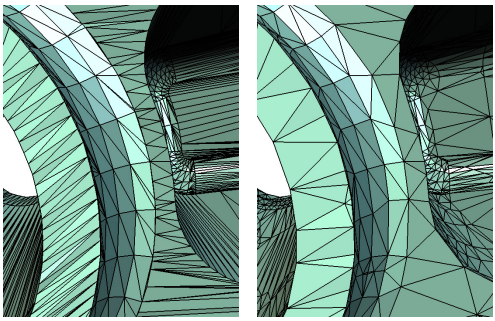

Figure 8: A closeup on some of the original mesh's caps (looking like T–junctions). These caps have all been eliminated in the enhanced version.

The main drawback and direction for future work is the large amount of needles generated by slicing the mesh. This intermediate mesh complexity could be kept low by interleaving cutting and collapsing steps (e.g. decimating along the intersection curve after each slicing), thereby reducing memory consumption and computation costs.

References

[1] F. Bernardini, C. L. Bajaj, J. Chen, and D. R. Schikore. Automatic reconstruction of 3D CAD models from digital scans. *International Journal of Computational Geometry and Applications*, 9(4&5):327–370, 1999.

[2] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball–pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[3] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

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

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

[6] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Conference Proceedings*, pages 209–216, 1997.

[7] H. Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 99–108, 1996.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *SIGGRAPH 93 Conference Proceedings)*, pages 19–26, 1993.

[9] L. Kobbelt, T. Bareuther, and H.-P. Seidel. Multiresolution shape deformations for meshes with dynamic connectivity. In *Computer Graphics Forum (Proc. Eurographics 2000)*, volume 19(3), pages 249–260, 2000.

[10] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH 01 Conference Proceedings*, to appear.

[11] L. Kobbelt, S. Campagna, and H.-P. Seidel. A general framework for mesh decimation. In *Graphics Interface*, pages 43–50, 1998.

[12] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel. A shrink wrapping approach to remeshing polygonal surfaces. *Computer Graphics Forum*, 18(3):119–130, 1999.

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

[14] Jonathan R. Shewchuk. Delaunay refinement mesh generation. *PhD–Thesis, Carnegie Mellon University, Pittsburg*, 1997.

[15] G. Turk. Re–tiling polygonal surfaces. In *Computer Graphics (SIGGRAPH 92 Conference Proceedings)*, volume 26, pages 55–64, 1992.

[16] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *SIGGRAPH 94 Conference Proceedings*, pages 311–318, 1994.

[17] Z. Wood, M. Debrun, P. Schröder, and D. Breen. Semi–regular mesh extraction from volumes. In *Proceedings of Visualization 2000*, 2000.