Robust Tetrahedral Meshing of Triangle Soups

J. Spillmann M. Wagner M. Teschner

Computer Graphics
University of Freiburg, Germany

Abstract

We propose a novel approach to generate coarse tetrahedral meshes which can be used in interactive simulation frameworks. The proposed algorithm processes unconstrained, i. e. unorientable and non-manifold triangle soups. Since the volume bounded by an unconstrained surface is not defined, we tetrahedralize the pseudo volume of the surface, namely the space that is intuitively occupied by the surface. Therefore, a new signed distance field approach is employed and a tetrahedral lattice is laid onto the distance field. Elements outside the pseudo volume are discarded and a smoothing filter is applied to the mesh boundary as a postprocessing step.

Using our approach, we can generate coarse tetrahedral meshes from damaged surfaces and even triangle soups without any connectivity. Various examples underline the robustness of our approach. The usability of the resulting meshes is illustrated in the context of interactive deformable modeling.

1 Introduction

Tetrahedral meshes are commonly used to represent the interior of volumetric objects for physically-based simulations, e. g. in order to represent deformable objects. To compute mesh deformations, a variety of approaches have been presented ranging from finite element methods [MMD*02] to mass-spring methods [BW98]. Recently, techniques have been presented that allow to compute the deformations of tetrahedral meshes at interactive rates [THMG04, MG04]. In [GEW05, GW05a, GW05b], efficient GPU implementations of deformable models are presented. Further, there exist methods to change the mesh topology, e. g. by fracturing [OH99, OBH02], cutting [SHGS06], refinement [DDCB01] and simplification [SG98, CDM04].

Generating a tetrahedral mesh from a boundary surface is a non-trivial task. Most mesh generators assume that the boundary surface is a closed and orientable manifold. However, many surfaces do not obey these criteria and the enclosed volume is not defined. Surfaces that are obtained by laser scans often contain holes and cracks (see Fig. 1), making it non-manifold. Models that have been constructed using CAD software are composed of interpenetrating subparts (see Fig. 2). In theses cases, traditional volumization approaches have difficulties to determine the object volume which has to be tetrahedralized. Moreover, there exist object representations that are modeled from unconnected triangles (see Fig. 3). While a human observer can intuitively recognize the space occupied by this structure, a volumization approach fails to compute a plausible volumetric representation which hinders the generation of a tetrahedral mesh.

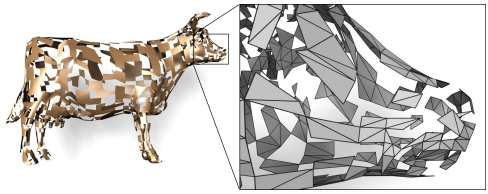


Figure 1: Surfaces obtained by laser scans can contain holes and cracks. In this example, 50% of the triangles have been removed. Our scheme still computes a plausible tetrahedral mesh for this representation.

Motivated by the growing need for tetrahedral meshes in interactive animations, we propose a method that generates meshes from arbitrary, unconstrained surfaces. Since the resulting meshes are intended to be used in interactive simulations, we focus on coarse meshes.

Our contribution. We present a scheme for robust tetrahedral mesh generation from triangle soups. We do not impose any constraints on the ob-

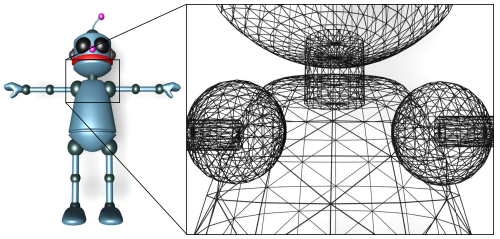


Figure 2: Models that have been constructed using a CAD software usually consist of interpenetrating subparts. Traditional volumization approaches cannot handle such models properly.

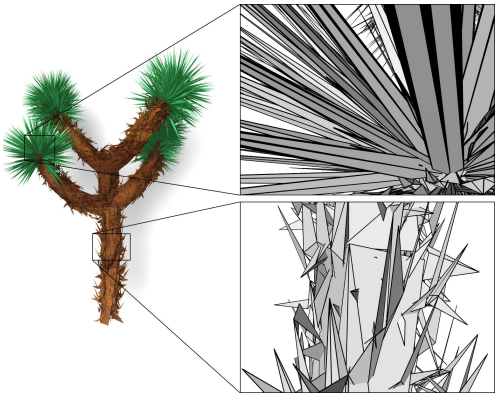


Figure 3: This object has been modeled from unconnected triangles. Although the resulting surface is unoriented and non-manifold, our approach can generate a plausible tetrahedral mesh.

ject surface, i. e. we can process unorientable and non-manifold surfaces. Since these surfaces in general do not enclose a volume, we introduce the term pseudo volume. The pseudo volume corresponds to the space that is intuitively occupied by the input surface. We show how to generate a well-shaped tetrahedral that approximates the pseudo volume. The meshing process requires user interaction and an intuitive way is provided to control the shape and size of the resulting mesh.

Since a variety of arbitrarily triangulated object surface can be processed, our approach is particularly interesting for game design. The resulting coarse tetrahedral meshes can be used in interactive simulations of rigid and deformable solids. The presented approach is efficient. A surface consisting of 70K triangles can be tetrahedralized in less than three minutes.

Organization. Our approach consists of three steps, namely the computation of the pseudo volume, the generation of the tetrahedra, and the post-processing of the mesh.

In section 3, we show how to compute the signs of the distance field of the object representation. We propose a novel scheme that considers distance field densities to generate the signs. From the resulting pseudo volume, we generate a tetrahedral lattice and discard those elements that lie outside the object as described in section 4. Further, some post-processing issues are discussed. The resulting mesh approximates the pseudo volume of the input. In section 5, we present some resulting meshes.

Fig. 4 illustrates the mesh generation process.

2 Related Work

First, we discuss existing approaches that tetrahedralize a spatial domain. Second, we focus on tetrahedralization approaches that are based on distance fields. Since these approaches commonly assume that the surface bounding the domain is closed and orientable, the third part discusses model repair methods that process unconstrained surfaces.

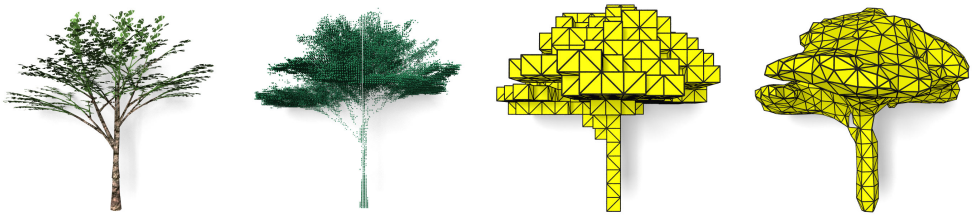
2.1 Volumetric mesh generation

Delaunay schemes [ACYD05] result in very well-shaped meshes [She02]. However, water-tight input surfaces are required.

In [SG95], a 3D domain is packed with spheres. The centers of the spheres are the nodes of the generated tetrahedral mesh. Advancing front methods [Sch97, ISS04] start from the boundary and attach new tetrahedra to existing ones.

Other approaches start with a coarse volumetric mesh. The cells are subdivided and the boundary of the resulting mesh has to be constrained to the input surface. In [DDCB01], a coarse tetrahedral mesh is subdivided to improve the precision. However, the authors report difficulties to preserve the mesh quality in the subdivided parts.

Further, there exist approaches that start with a structured grid consisting of cubic cells which are partitioned into tetrahedra. In this case, the main challenge is to constrain the tetrahedral mesh to the input surface, see e.g. [BW90, VTG97]. In [MBTF03], three approaches are discussed to



(a) (b) (c) (d)

Figure 4: Generating a tetrahedral mesh from a triangle soup (a). First, a distance field is computed. A novel method generates the signs of the distance field values. Negative values represent the space that is intuitively occupied by the surface, i. e. the pseudo volume (b). A tetrahedral lattice is laid onto the pseudo volume (c) and a smoothing filter is applied to obtain a mesh (d) that is appropriate for interactive simulations.

conform the tetrahedral mesh to the input surface: Either by simulating a mass-spring model or FE methods, or based on optimization constraints. However, the surface is required to be well-shaped in order to avoid undesirable results. In contrast, our post-processing step does not consider the input surface. Therefore, we can process unconstrained surfaces.

Some authors propose to subdivide the tetrahedra that interfere with the input surface. In [MT03], this idea is employed to generate tetrahedral meshes for medical simulations. Since a parity check is used to classify the tetrahedra, this approach is limited to water-tight surfaces. In [CDM*02], tetrahedral meshes for solid, layered volumes are generated and a subdivision is done along the volume layers. However, this approach can lead to poorly shaped elements. Further, the element size can differ significantly which is undesirable for dynamic simulations. Simplification and improvement strategies for these meshes are proposed in [CDM04].

An alternative approach is presented in [SOS04]. They compute a smooth implicit surface from a polygon soup. The implicit surface is polygonized and a tetrahedral mesh is constructed. The results look promising. However, the authors state that additional normal constraints have to be imposed to get good results. Thus they require consistent normal orientations. In contrast, our approach does not take the surface normals into account.

2.2 Distance field computation

There exist numerous methods to compute distance fields. In [HYFK98], a method for computing unsigned distance fields is presented. Since the distance field computation is expensive, several optimizations have been proposed. [Mau00] computes Voronoi diagrams of surface features which allows a fast computation of the closest feature. Further, [MJRR03] proposes a fast algorithm that computes distance fields from voxelized surfaces. In [WK93], an alias-free voxelized surfaces is produced. However, the volume is not explicitly computed.

There exist efficient GPU implementations for the computation of distance fields. In [SPG03], a signed 512^3 distance field is computed in six seconds from a surface consisting of 32K faces. They rasterize Voronoi regions of the surface features. A different approach is presented in [Br05] where the surface is rasterized into LDIs. However, since the normal of the surface triangles is used to compute the sign, this method is not suitable for surfaces with inconsistent normals.

2.3 Model repair

Surfaces that are obtained by 3D range scans tend to be ill-shaped, i. e. they are non-manifold and contain holes and gaps. Since mesh generators usually assume closed and orientable manifolds, the surfaces have to be repaired.

There exists a variety of model repair methods that process polygonal surfaces [TL94, BNK02, Lie03]. Although these mesh-based methods provide good results, they cannot guarantee that the output surfaces are closed everywhere. Further, there exist models that consist of triangles without connectivity (see Fig. 3) and a mesh-based method would fail to repair such input data. [NT03]

and [Ju04] use distance fields to repair models. However, these approaches cannot be applied to unconnected triangle soups as depicted in Fig. 3.

Bischoff et al. [BPK05] present an approach that is similar to ours since they construct a volumetric representation of a polygonal input mesh, which is then employed to restore the mesh topology. However, it is unclear whether this approach can handle totally unstructured triangle soups.

3 Volume representation

Computing a tetrahedral mesh from a surface corresponds to computing a volume Ω in $\Re^3$ from a boundary $\partial\Omega$. This volume is well-defined if $\partial\Omega$ is a closed manifold and orientable. Both criterions are necessary since. Although a Klein-bottle is a closed manifold, it cannot be oriented and has infinite volume.

Mesh generators commonly assume that the above criteria hold for the input surface. However, our approach particularly addresses objects that do not meet these criteria. Since the object volume might be undefined, we cannot consider approaches that determine the object volume. Instead, we have to find a way to approximate the space occupied by the object representation which is referred to as pseudo volume. In order to compute the pseudo volume, we consider the signed distance field of the input surface. A signed distance field corresponds to a distance field where the sign of the distance value indicates whether a voxel is inside or outside the volume bounded by the zero isocontour. We propose a novel scheme to generate the signs of the distance values such that the resulting set of voxels with negative signs corresponds to the pseudo volume.

A standard approach such as [Br05] can be used to compute the distances. To generate the signs, we use an adaption of the parity count scheme proposed by [NT03]. They cast rays in a given direction through the surface and count the number surface intersections. Voxels with an odd number of intersections are then classified as interior. Since holes can lead to a misclassification of a span, rays are shot from 13 different directions, and the majority vote decides on the sign of a voxel. This method works well for surfaces that contain a small number of holes, but are closed elsewhere. However, it cannot be applied to unconstrained surfaces as depicted in Fig. 3 since the parity only changes if a ray actually interferes with the surface.

In the following discussion, we denote a parity change as a volume event. A parity change from even to odd is denoted as an out-in volume event, and the the parity change from odd to even as an in-out volume event. We assume that the distance field is defined within the axis-aligned bounding box of the surface. We then interpret the distance d_{min} from the center of a voxel v to its closest surface feature as a probability of a volume event:

$$P_v(\text{volume event}) = 1 - c|d_{min}|$$

where c is a global normalization factor such that $0 \leq P_v \leq 1$ for all voxels v in the distance field. To decide whether an in-out or out-in event occurs, we consider the computed probability in each voxel: If $P_v > k$, a volume event is reported. A subsequent volume event is only reported if the probability P_v dropped below k in the meantime. Fig. 5 shows an example. The threshold value k is user-defined and $k = 0.8$ is a good choice in many cases. Considering the volume event probability is more robust compared to methods that consider actual intersections of the ray with the object representation. Following [NT03], the robustness of the scheme is further improved by shooting rays from different directions. As a consequence, the generated pseudo volume is not larger than the convex hull of the surface.

The resulting pseudo volume does not conform to the analytical volume since such a volume does not exist for unconstrained surfaces. Further, the pseudo volume is not unique for a given surface. Instead, it depends on the choice of the threshold value k. If $k = 1$, then the pseudo volume corresponds to [NT03]. For $k < 1$, the effect is similar to applying a morphological closure operator to the negative signed voxels produced by [NT03]. In contrast to [NT03], we can handle all kinds of degeneracies in a unified scheme. Refer to Fig. 6 for examples on the computed pseudo volumes.

4 Mesh Generation

So far, we have generated a signed distance field from the input surface and the set of voxels with negative signs represents the pseudo volume. Now, we show how to generate a tetrahedral mesh from the pseudo volume.

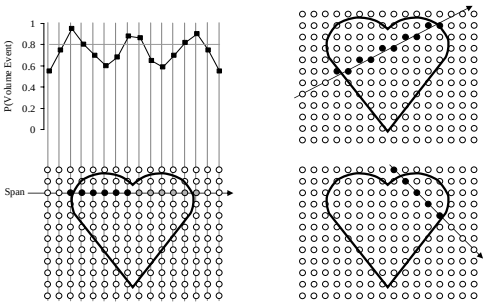


Figure 5: Sign generation and misclassification. Left: A ray and its per-voxel volume event probabilities. Because the volume event probability is higher than $k = 0.8$ for the voxel closest to the concave corner of the heart shape, a volume event is raised and the rest of the span is misclassified (gray circles). Right: Rays are shot from 26 different directions and the majority vote decides on the sign of the voxel. Thus, the voxels that are misclassified in one pass will nevertheless have correct signs.

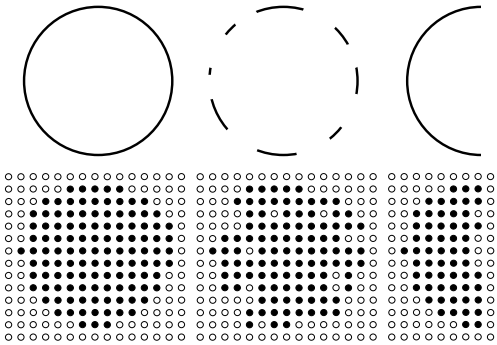


Figure 6: Generation of the signs of a 2D distance field for a damaged circle. Negative values represent the pseudo volume bounded by the surface. Left: Pseudo volume of the unmodified circle. In this case, the pseudo volume conforms to the actual voxelized volume bounded by the circle. Middle: Pseudo volume of the circle with 50% of its segments missing. Right: Pseudo volume of one half of the circle.

Therefore, we apply a structured method based on an axis-aligned grid [Blo94, CDM*02]. First, we lay a uniform lattice onto the distance field. The resolution of the lattice is user-defined. Each cubical cell is split into five tetrahedra. To ensure that neighboring cells match each other, the orientations of the tetrahedra alternate [CDM*02].

Second, we discard those elements that lie outside the volume. In contrast to [MT03, VTG97], we do neither consider the input surface nor the zero isocontour since the input surface can be unoriented. As a consequence, it is not possible to test an element against a surface normal to decide whether it is inside or outside the volume. Instead, we define the pseudo volume density $\rho(V)$ within an arbitrary volume V as

$$\rho(V) = \frac{m(V)}{M(V)}$$

where $m(V)$ is the number of voxels with negative sign within the volume V, and $M(V)$ is the total number of voxels within the volume V.

We impose the following rule to discard an element T with volume V_T: T is discarded if $\rho(V_T) < \rho_{min}$. ρ_{min} is a user-defined value that controls the volume of the final tetrahedral mesh.

Since the element discarding scheme does neither consider the orientation of the surface nor its zero isocontour, it is robust for unoriented and scattered input surfaces. Moreover, the user can control the shape of the resulting tetrahedral mesh. By adjusting the value ρ_{min}, an intuitive way is provided to control how much space around a scattered surface is covered by the tetrahedral mesh. In our experiments we use $\rho_{min} = 0.5$. If ρ_{min} is decreased, more tetrahedrons are added at the boundary of the mesh. Transformations of the lattice provide additional degrees of freedom. However, so far the user interface is restricted to translations.

At this point, the resulting meshes do still contain sharp creases and corners. We thus apply a smoothing filter similar to [DMSB99] to the mesh boundary to produce well-formed meshes. Hereby, the mesh is locally scaled back to its original volume in order to guarantee volume preservation. The resulting meshes are suitable for dynamic simulation and collision handling. Refer to Fig. 7 for an example.

5 Results

In this section, we present some meshes that we have generated using our approach. All tests have been performed on an Intel Xeon PC, 3.8 GHz.

Closed manifold surfaces. The torus surface shown in Fig. 8 is a closed manifold. It consists of 640 triangles. The resulting distance field is com-

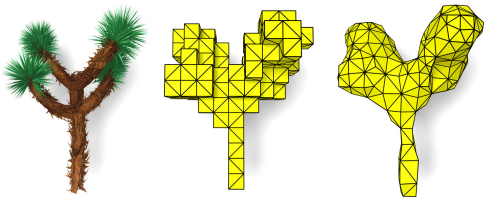


Figure 7: Surface smoothing. Left: The input surface. Middle: The tetrahedral lattice. Right: A smoothing filter is applied to the lattice in order to remove creases and corners. Basically, the volume of the mesh is preserved.

puted at a resolution of 150x150x37 voxels. Fig. 8 illustrates that our approach can generate meshes at different resolutions. The left mesh consists of 139 tetrahedra. Due to its coarseness, it is well-suited for interactive simulations. The right mesh consists of 8K tetrahedra. The cutaway view shows the regular structure of the interior elements.

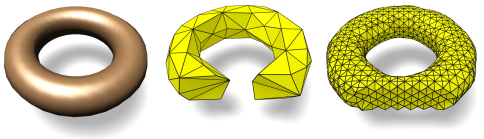


Figure 8: Torus. Our approach can generate tetrahedral meshes of varying resolutions. The left mesh consists of 139 elements and the right mesh consists of 8K elements.

Non-manifold surfaces. Our approach can also handle objects that are composed of interpenetrating subparts. The model illustrated in Fig. 9 and Fig. 2 consists of 16K triangles. While a human observer can intuitively identify the volume, algorithms might have problems to consider the union of two intersecting subparts as interior. However, our scheme produces plausible and well-shaped tetrahedral meshes for such object representations.

Damaged surfaces. To illustrate that our approach can handle even badly damaged models, we have removed 50% of the faces from a surface as shown in Fig. 10 and Fig. 1. Note that the element size of the generated tetrahedral mesh do not significantly vary, which is an important criterion for efficient dynamic simulation of such meshes.

Triangle soups. There exist objects that are composed of a set of unconnected triangles and many

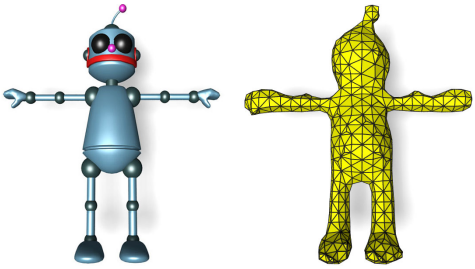


Figure 9: The input model is composed of interpenetrating subparts. However, our approach still computes a plausible tetrahedral mesh for this object. The mesh consists of 2K elements.

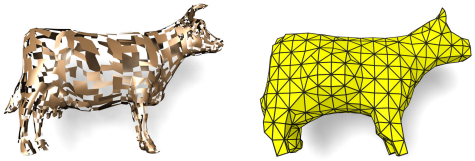


Figure 10: Our approach can be used to build tetrahedral meshes from damaged surfaces. 50% of the faces have been removed from the surface (left). However, a plausible tetrahedral mesh can still be produced for this surface (right). The mesh consists of 1158 elements.

existing techniques are not able to tetrahedralize such models. However, since our approach considers the pseudo volume density, we can tetrahedralize the space that is occupied by the surface. Fig. 11 illustrates the tetrahedralization of a tree model. The input surface consists of 8K triangles. The resulting mesh has 3K elements. Fig. 7 shows the tetrahedral mesh generated for a palm model that consists of 520 tetrahedra.

Performance. To compute the distance field, a standard approach such as [Br05] can be used. They can produce a distance field of resolution 256^3 from an input surface with 70K triangles in less than 20s. Determining the signs is independent of the number of surface triangles. In our implementation, tt takes about 90s to compute the signs for a distance field with a resolution of 256^3. Since user interaction is required in the mesh generation and postprocessing, exact measurements cannot be provided. However, based on the pseudo volume, the generation of 3K tetrahedra takes less than 7s, and one smoothing

Figure 11: Generating a tetrahedral mesh for an unconnected triangle soup. In this case, the resulting volumetric representation corresponds to the space that is intuitively occupied by the model. The mesh consists of 3K elements.

pass takes less than 3s.

Application. Since our approach can produce very coarse and nevertheless plausible tetrahedral meshes, they are are well-suited for interactive simulations and animations. To illustrate this fact, we have implemented the methods described in [THMG04]. In Fig. 12, we run an animation of four deformable robot models falling onto a deformable tree model. The total number of elements is 3K. The simulation runs at about 20 frames per second including visualization and collision handling. A massive scenario is depicted in the bottom row of Fig. 12 where 125 deformable robots are falling onto a deformable tree model. The total number of elements is 58K and the simulation runs at 0.3 frames per second including the rendering of two million surface triangles.

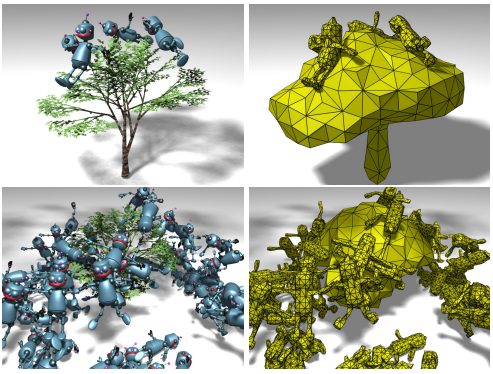


Figure 12: Simulation of deformable objects. The meshes have been generated using our approach. The scenario in the top row can be simulated at 20 frames per second using [THMG04]. A massive scenario, consisting of 125 objects and a total of 58K tetrahedra is shown in the bottom row.

6 Conclusion

We have presented a new approach that generates plausible tetrahedral meshes from arbitrary surfaces. Since unoriented and non-manifold triangle soups do not enclose a volume, we have introduced the pseudo volume to denote the space that is occupied by the surface. A signed distance field is used to approximate the pseudo volume. We have proposed a novel scheme that generates the signs of the distance field employing the probability of volume events. We have further shown how to generate a well-shaped tetrahedral mesh covering the pseudo volume. The boundary of the mesh is smoothed, making the resulting meshes suited for physical simulations. We have performed various experiments that emphasize the robustness of our scheme.

References

[ACYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Transactions on Graphics* (2005), 617–625.

[Blo94] BLOOMENTHAL J.: An implicit surface polygonizer. In *Graphics Gems IV*. Academic Press, 1994, pp. 324–349.

[BNK02] BORODIN P., NOVOTNI M., KLEIN R.: Progressive gap closing for mesh repairing. In *Advances in Modelling, Animation and Rendering* (2002), pp. 201–213.

[BPK05] BISCHOFF S., PAVIC D., KOBBELT L.: Automatic restoration of polygon models. *ACM Transactions on Graphics 24*, 4 (2005), 1332–1352.

[Br05] BRENTZEN J. A.: Robust generation of signed distance fields from triangle meshes. *Volume Graphics* (2005), 167–175.

[BW90] BLOOMENTHAL J., WYVILL B.: Interactive techniques for implicit modeling. *Computer Graphics* (1990), 109–116.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proc. SIGGRAPH* (1998), pp. 43–54.

[CDM*02] CUTLER B., DORSEY J., MCMILLAN L., MÜLLER M., JAGNOW R.: A procedural approach to authoring solid models. In *ACM Trans. on Graphics* (2002), pp. 302–311.

[CDM04] CUTLER B., DORSEY J., MCMILLAN L.: Simplification and improvement of tetrahedral models for simulation. In *Proc. Eurographics Symposium on Geometry Processing* (2004), pp. 95–104.

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space and time adaptive sampling. In *Proc. SIGGRAPH* (2001), pp. 31–36.

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. SIGGRAPH* (1999), pp. 317–324.

[GEW05] GEORGII J., ECHTLER F., WESTERMANN R.: Interactive simulation of deformable bodies on gpus. In *Simulation and Visualisation* (2005).

[GW05a] GEORGII J., WESTERMANN R.: Interactive simulation and rendering of heterogeneous deformable bodies. In *Vision, Modeling and Visualization* (2005).

[GW05b] GEORGII J., WESTERMANN R.: A multigrid framework for real-time simulation of deformable volumes. In *Workshop On Virtual Reality Interaction and Physical Simulation* (2005).

[HYFK98] HUANG J., YAGEL R., FILIPPOV V., KURZION Y.: An accurate method for voxelizing polygon meshes. In *IEEE Symposium on Volume Visualization* (1998), pp. 119–126.

[ISS04] ITO Y., SHIH A. M., SONI B. K.: Reliable isotropic tetrahedral mesh generation based on an advancing front method. In *Proc. 13th International Meshing Roundtable* (2004), pp. 95–106.

[Ju04] JU T.: Robust repair of polygonal models. *ACM Trans. on Graphics* (2004), 888–895.

[Lie03] LIEPA P.: Filling holes in meshes. In *Proc. Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2003), pp. 200–205.

[Mau00] MAUCH S.: A fast algorithm for computing the closest point and distance function. *Tech. Rept. CalTech., unpublished* (2000).

[MBTF03] MOLINO N., BRIDSON R., TERAN J., FEDKIW R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proc. 12th International Meshing Roundtable* (2003), pp. 103–114.

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proc. Graphics Interface* (2004), pp. 239–246.

[MJRR03] MAURER JR. C. R., RENSHENG Q., RAGHAVAN V.: A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2003), 265–270.

[MMD*02] MÜLLER M., MCMILLAN L., DORSEY J., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proc. Symposium on Computer Animation* (2002), pp. 49–54.

[MT03] MÜLLER M., TESCHNER M.: Volumetric meshes for real-time medical simulations. In *Proc. BVM* (2003), pp. 279–283.

[NT03] NOORUDDIN F. S., TURK G.: Simplification and repair of polygonal models using volumetric techniques. In *IEEE Trans. on Visualization and Computer Graphics* (2003), pp. 191–205.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. In *Proc. SIGGRAPH* (2002), pp. 291–294.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH* (1999), pp. 137–146.

[Sch97] SCHOBERL J.: Netgen - an advancing front 2d/3d-mesh generator based on abstract rules. *Comput. Visual. Sci.* (1997), 41–52.

[SG95] SHIMADA K., GOSSARD D. C.: Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. *ACM 3rd Symposium on Solid Modeling and Applications* (1995), 179–191.

[SG98] STAADT O. G., GROSS M. H.: Progressive tetrahedralizations. In *Proc. Visualization* (1998), pp. 397–404.

[She02] SHEWCHUK J. R.: What is a good linear element? - interpolation, conditioning, and quality measures. *Proc. 11th International Meshing Roundtable* (2002), 115–126.

[SHGS06] STEINEMANN D., HARDERS M., GROSS M., SZEKELY G.: Hybrid cutting of deformable models. In *IEEE VR 2006, to appear* (2006).

[SOS04] SHEN C., O'BRIEN J. F., SHEWCHUK J. R.: Interpolating and approximating implicit surfaces from polygon soup. In *Proc. SIGGRAPH* (2004), pp. 896–904.

[SPG03] SIGG C., PEIKERT R., GROSS M.: Signed distance transform using graphics hardware. In *Proc. IEEE Visualization* (2003), pp. 83–90.

[THMG04] TESCHNER M., HEIDELBERGER B., MÜLLER M., GROSS M. H.: A versatile and robust model for geometrically complex deformable solids. In *Computer Graphics International* (2004), pp. 312–319.

[TL94] TURK G., LEVOY M.: Zippered polygon meshes from range images. In *Proc. SIGGRAPH* (1994), pp. 311–318.

[VTG97] VELHO L., TERZOPOULOS D., GOMES J.: Implicit manifolds, triangulations and dynamics. *Journal of Neural, Parallel and Scientific Computations. Special Issue in Computer Aided Geometric Design* (1997), 103–120.

[WK93] WANG S., KAUFMAN A.: Volume sampled voxelization of geometric primitives. In *Proc. Visualization* (1993), pp. 78–84.