The Visual Computer manuscript No.
(will be inserted by the editor)

Kenny Erleben

Volumetric Shells using Path Tracing

Abstract Low count tetrahedral meshes are desirable for producing shell maps or animating deformable objects, where accuracy is less important. This paper presents a new method for creating a shell tetrahedral mesh from a triangular surface mesh. We propose to use signed distance fields to implicitly represent the medial surface of a surface mesh combined with a divergence based medial surface representation. An iterative path tracing algorithm is the main result, which is easy to parallelize. The new method respects the structure of the object shape, when generating a shell mesh. We demonstrate the capability of our method to create thick shells with almost no residual space inside our test objects.

Keywords Shell Meshing · Path Tracing

1 Introduction

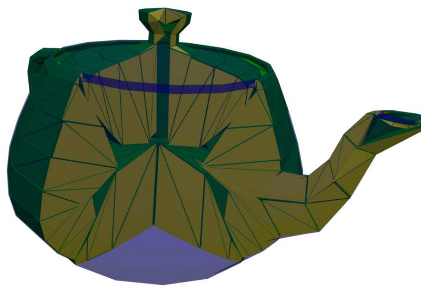
Volumetric shell meshes are attractive, since they give a volumetric representation of a surface mesh with a very low tetrahedral count. Low tetrahedral count is desirable for animation or similar purposes, where speed is preferred over accuracy of deformation. Volumetric shell meshes are becoming more attractive for animation of deformable objects [12] or shell maps [19]. Figure 1 shows examples of volumetric shell meshes generated with our method.

Existing algorithms such as [15] are difficult to implement and do not use the natural, intrinsic representation of shape by medial surfaces [6,18]. Existing tetrahedral mesh generating methods typically create an initial, blocked tetrahedral mesh from a voxelization or signed distance field. Afterward, nodes are iteratively repositioned, while tetrahedrons are sub-sampled in order to improve mesh quality [16,21], or the variational gradient of an energy functional is used to move vertices [1]. In contrast to these methods, ours is surface based. Further, our method can create a shell mesh, whereas the other methods will create meshes with a full coverage of the interior.

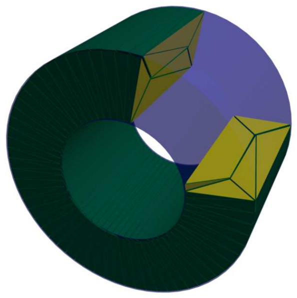
K. Erleben
Department of Computer Science, University of Copenhagen
Tel.: +45-32351400
Fax: +45-32351401
E-mail: kenny@diku.dk

A linear randomized tessellation algorithm, the ripple tessellation, was developed in [8]. The ripple method suffers from several problems. It is not deterministic, but relies on picking random ripple directions to fix inconsistencies. A safe, conservative, upper limit on the extrusion length is used in [8] and later improved in [9]. Nevertheless, both algorithms are very slow due to bad and unpredictable convergence behavior of the bisection search method. In [10] line stepping was shown to be a more efficient numerical method for the displacement computation. All the surface based methods [8–10] suffer from one drawback: at extreme curvatures the shell meshes become paper thin. A paper thin shell may be undesirable for many reasons. For example, the small thickness would limit the size of features rendered by a shell map or cause unwanted tunneling effects when doing collision detection during a simulation. Further, large area surface triangles with small extrusion thickness would result in poor mesh quality due to the obvious creation of slivers and needles in the resulting shell mesh. Thus, it is interesting to try to find the thickest possible shell mesh. This is the problem we address in this paper.

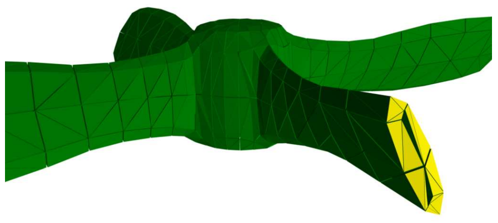
Our main goal is to create the thickest possible shell mesh with the lowest possible tetrahedral count. That is, given a polygonal surface mesh, we create a tetrahedral volume mesh representing a thick version of the surface mesh, called a shell mesh. The vertices of the polygonal surface mesh are displaced inwards along a path trace, thereby creating a new version of the surface mesh. Following the displacement the original surface mesh is used to generate the outside of the shell and the displaced surface mesh is used to generate the inside of the shell mesh. The two meshes are then used to create a triangle prism shell mesh. Finally, the triangle prism mesh is converted to a consistent tetrahedral mesh, also known as a tetrahedral tessellation. In this work we only focus on the displacement problem and use the tetrahedral tessellation algorithm of [8].



(a) Teapot



(b) Tube



(c) Propeller

Fig. 1 Volumetric Shell meshes visualized by partly removing some of the tetrahedrons of the shell mesh, exposing the inside (yellow color) of the shell mesh.

Intuition hints that we want to find the point on the medial surface that corresponds to a given surface point. By definition the medial surface consists of the set of center points of all interior maximal balls. However, obtaining the medial surface is computational difficult and notoriously error prone due to discretization errors. Instead, we will use a signed distance field of the surface mesh. The signed distance field of the surface contains the medial surface implicitly, and this can be exploited to result in a very simple solution to be explained shortly.

Working directly on the boundary representation is fast and simple [20], but topological problems such as shocks [13] arise easily. Our method is based on the inwards extrusions, also know as intrusions, of the surface triangles, thus producing prisms. At shocks these prisms will degenerate to contain less than 6 vertices. These shocks turn out to be the limit on the intrusion lengths in our method.

Distance fields have been used prior to our work [17]. However, [17] uses generalized Newton potentials, which is basically equivalent to regularizing the Euclidean distance field, such that the medial surface collapses into a line-skeleton [4]. In our work we use the Euclidean distance field, and we disprove the claim from [17] that the Euclidean distance fields cannot be used for shell generation. Besides, [17] suffers from folding at extreme shell thickness. Thus [17] only shows extrusions for modest shell thickness. As we demonstrate, our method does not suffer from this artifact. Lastly, it should be pointed out that the stopping criteria used in the numerical method of [17] rely on the zero gradient of the distance field or exceeding a maximum allowed thickness. As we show, such a method will result in shells with folds. In addition, we show a mesh quality analysis at extreme thickness.

2 The Path Tracing Algorithm

To inwardly extrude surface triangles we intrude the triangle vertices as close as possible to the medial surface without crossing it. Path tracing is a very robust method for this and will be described in the following subsections. Overall, our method can be described as follows

– Perform path tracing of all non-planar surface points.
– Perform path tracing of all planar surface points.
– Create triangular prisms by using line segments from the original surface points to the end positions of the corresponding path traces.
– Convert the triangle prism shell into a tetrahedral shell mesh by a tessellation method.

In Section 2.1 we present our novel stopping criteria and the intuition behind them. In Section 2.2 we will present all the details of the path tracing algorithm.

2.1 The Non-folding Stopping Criteria

We trace a surface point $\mathbf{p}_j$ along a curved path in space. The gradient of a signed distance field, ϕ, of the surface mesh is used to determine the trajectory of the point, as the point traces out the path. In the general case the path trace is stopped, when a critical-point in the signed distance field is passed. This is detected by testing, whether the magnitude of the gradient of the signed distance field drops below some threshold-value. Due to discretization and interpolation errors the gradient of a regular sampling of a signed distance field is not precisely one everywhere. The regular sampling and numerical diffusion tend to smooth the signed distance field. Therefore, in our implementation we used a fraction of the gradient at the initial surface position as the threshold-value. Two more stopping criteria are also applied:

1. If a path trace hits another path trace, then the path trace is stopped. This action often corresponds to having detected a junction in the medial surface of the object. The merge point of the path with the minimum path length is stored together with each path.
2. If the initial surface point is planar, the path trace will not start on a seam of the medial surface. This implies that the path trace will not necessarily hit a junction of the medial surface. Therefore the path trace is halted, once the trace hits the medial surface.

The purpose of criterion 1 is to enhance performance, when path traces are done sequentially. The idea is that if a merge point is detected, the path trace will trace out the same trajectory as the merging path.

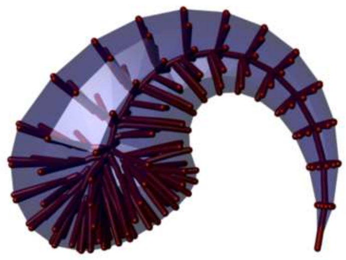
In order to detect the merge points we used spatial hashing [23] to test whether a point trace is sufficiently close to a point trace from another path. Spatial hashing is simple to implement, since it requires only a hashing function to look up cells in a 3D grid. In our test cases we rescaled all surface meshes to be within the unit-cube and used a distance value equal to the length of the cell diagonal in the regular sampled disance field to detect close-by point traces.

Figure 2 shows the actual path traces and merge points of a more complex horn-like object. Note that if the tip of the horn was not stopped at the first merge point, one would have created a triangle prism starting from the tip and ending in the middle of the base of the horn. Such a prism would go outside the horn-like object and therefore is clearly undesirable.

Observe that path traces of planar surface points may have merge points that are not junctions of the medial surface. As a consequence, merge points of path traces of non-planar vertices must be detected prior to tracing any paths of planar vertices. This is illustrated in Figure 3. Figure 4 shows real-life examples of path traces using our method. Observe that the path traces obey the stopping criterion for planar vertices.

2.2 The Details of our Path Tracing Algorithm

The algorithm works as follows: The position of the j'th surface vertex at the i'th step along the path is denoted by $\mathbf{q}_j^i$ and $\mathbf{q}_j^0 = \mathbf{p}_j$. To obtain the position $\mathbf{q}_j^i$ a fixed incremental step of length $\Delta\varepsilon$ is taken in the opposite direction of the gradient of the signed distance field, $\nabla\phi(\mathbf{q}_j^i)$,


(a) Skeleton of horn object.

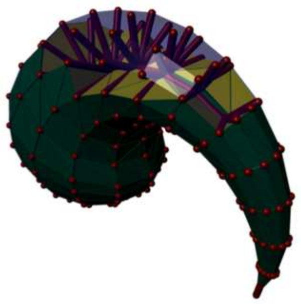
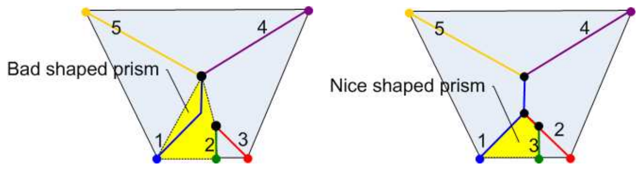

(b) Tetrahedra generated from the prisms of the horn object.

Fig. 2 Observe how the path trace of the pointy spike traces out a spline through the center line of the object. All traces along the tip of the horn are merged correctly with the spline.



(a) (b)

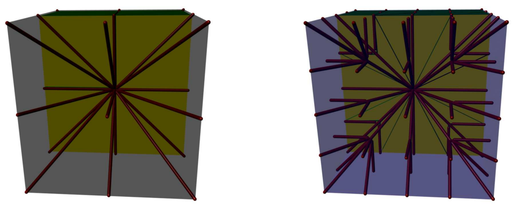
Fig. 3 Path tracing problems that must be handled correctly. Numbers indicate the path trace order. (a) Problem: The path traces of 2 and 3 result in a merge point that is not a junction. If planar vertices on the line between 1 and 5 are traced, they are likely to intersect the prism generated by paths 1 and 2. (b) Solution: By tracing planar surface points last we guarantee that prisms respect the medial surface.

$$\mathbf{q}_j^i = \mathbf{q}_j^{i-1} - \Delta\varepsilon\nabla\phi(\mathbf{q}_j^{i-1}). \tag{1}$$

If the cell size of the regular sampled signed distance field is given by Δx, Δy, and Δz then the fixed size increment is chosen as

$$\Delta\varepsilon = \frac{\min(\Delta x, \Delta y, \Delta z)}{2}. \tag{2}$$

This ensures that we do not step along the path faster than the information changes in the signed distance field. This


(a) Box Object

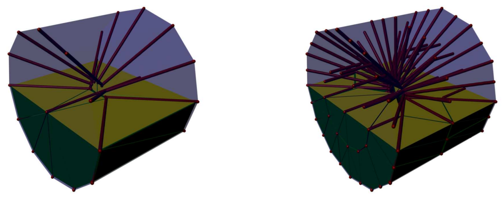

(b) Cylinder Object

Fig. 4 Increasing the resolution of the surface meshes causes planar vertices to be created in the surface meshes. The left column shows path traces and corresponding shell meshes of low-resolution surface meshes. The right column shows how our method correctly deals with the planar vertices at the increased resolution. Note that the planar vertices do not cross the medial surface of the objects.

works due to spatial coherence of the values in the signed distance field. The value at a neighboring node in the signed distance field is different by at most

$$\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}. \tag{3}$$

We keep on stepping, until the stopping criterion,

$$\nabla\phi(\mathbf{q}_j^i)^T \nabla\phi(\mathbf{q}_j^i) < c\nabla\phi(\mathbf{p}_j)^T \nabla\phi(\mathbf{p}_j), \tag{4}$$

is meet. Where we use $c = 10^{-3}$ in our experiments. To counter numerical diffusion we normalize the gradient of the signed distance field, whenever $\nabla\phi(\mathbf{q}_j^i)^T \nabla\phi(\mathbf{q}_j^i) > 1$. Finally, the corresponding path length is updated accordingly

$$\varepsilon^i = \varepsilon^{i-1} + \left\| \mathbf{q}_j^i - \mathbf{q}_j^{i-1} \right\|. \tag{5}$$

The path length is used to find the first merge point along a path trace.

The stopping criterion for the planar surface points is easy to understand, but not easy to implement. Obtaining the medial surface is known to be a very difficult problem, and many simplifications or approximations of the medial surface exist, such as the simplified medial axis [22], the power crust algorithm [2], or divergence based medial surfaces [3, 7]. Thus, getting an exact representation of the medial surface is a problem in itself. However, it is fairly easy to build a new field, telling us, whether the medial surface passes through a given location. The θ-SMA algorithm [22] can compute such a field. However, for simplicity we have chosen to compute the divergence of the gradient field of the signed distance field, i.e. the flux of the gradient field. This representation is often termed the divergence based medial surface [3] or the average outward flux (AOF) field [7]. The flux field is defined mathematically as follows:

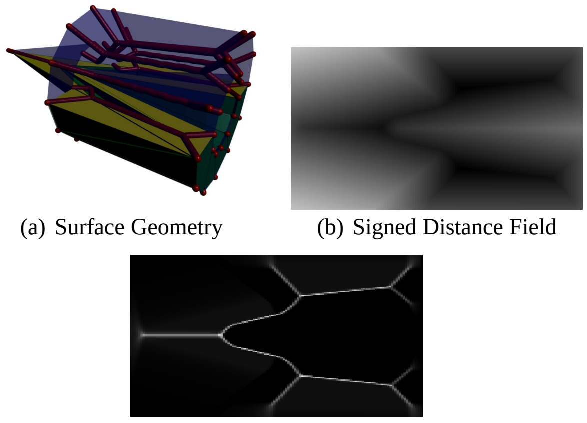

(a) Surface Geometry (b) Signed Distance Field (c) Flux Field

Fig. 5 Illustration of a signed distance field and the corresponding divergence based medial surface (flux field). Observe how the flux field corresponds to the shape of the path traces.

$$\text{flux}(i,j,k) = \int_S \nabla\phi \cdot \mathbf{d}s. \tag{6}$$

Numerically, we evaluate the flux for each node as follows: the surface S is chosen as a cubical surface, having the node (i,j,k) as its center. The cube has the eight neighboring nodes $(i-1,j-1,k-1)$, $(i-1,j-1,k+1)$, $(i-1,j+1,k-1)$, ..., $(i+1,j+1,k+1)$ as its corner points. The surface integral is evaluated using a summation approximation

$$\text{flux}(i,j,k) = \sum_{a \in N(i,j,k)} \nabla\phi(a) \cdot \mathbf{n}(a), \tag{7}$$

where $N(i,j,k)$ is the index set of the $3\times3\times3$ neighboring nodes (corners, face mid-points, and edge-midpoints of the surface cube), and $\mathbf{n}(a)$ is the outward normal of the cube surface. For $\nabla\phi$ we used a first order central difference approximation. Thus the flux at a single node is based on a measurement from a neighborhood of $5\times5\times5$ nodes. In our experience this adds robustness to our method. Once the flux field have been obtained, it can be used as an indicator for where the medial surface is located. Figure 5 illustrates a flux field of a simple shape. If the flux is zero, then it is a clear indication that the medial surface is not passing through the cube surface. If the flux is positive, it means that the cube surface contains part of the medial surface. Now, a path trace originating from a planar surface point is stopped, when the flux-value is sufficiently positive,

$$\text{flux}(\mathbf{q}_i^j) > \rho, \tag{8}$$

where ρ is a small user-specified threshold to counter numerical precision and round-off problems from the computation of the flux field. The entire path tracing algorithm is summarized in the pseudo code shown in Figure 6.

```
Algorithm path-trace( j'th path )
  $i = 0$
  set $q_j^i$ to surface point of path
  while forever
    add $q_j^i$ to path
    $\varepsilon^i = \varepsilon^{i-1} + \left\| \mathbf{q}_j^i - \mathbf{q}_j^{i-1} \right\|$
    $g = \nabla\phi(\mathbf{q}_j^i)$
    if planar and $\text{flux}(\mathbf{q}_j^i) > \rho$ or
       $g^T g < c\nabla\phi(\mathbf{q}_j^0)^T \nabla\phi(\mathbf{q}_j^0)$ or
       merge point detected at $\mathbf{q}_j^i$ then
      break
    end if
    if $g^T g > 1$ then
      $g = g / \|g\|$
    end if
    $\mathbf{q}_j^{i+1} = \mathbf{q}_j^i - \Delta\varepsilon g$
    $i = i + 1$
  end while
End algorithm
```

Fig. 6 Pseudo code version of our path trace algorithm. Notice that three different stopping criteria are used.

name	F	time	T	point size	step size
box	48	0.04	144	0.0150	0.0043
box	192	0.06	576	0.0150	0.0043
cylinder	48	0.04	144	0.0141	0.0035
cylinder	192	0.09	576	0.0141	0.0035
pointy	96	0.15	288	0.0110	0.0024
star	588	0.57	1764	0.0123	0.0006
horn	408	0.11	1224	0.0116	0.0020
tube	512	0.17	1536	0.0135	0.0028
sphere	760	561.66	2280	0.0150	0.0043
teapot	1056	567.09	3168	0.0115	0.0024
propeller	1200	569.79	3600	0.0124	0.0009
funnel	1280	559.61	3840	0.0127	0.0016
dragon	1496	558.47	4488	0.0128	0.003
bowl	2680	562.22	8040	0.0124	0.0010

Table 1 Statistics of test examples using our shell method. Time is given in seconds. Here F is the number of surface triangles and T is the number of tetrahedrons.

3 Results

In all the test examples presented in this paper we have used a regular sampling of the signed distance fields with resolution 128^3. All surface meshes were isotropically scaled to be within a unit-cube centered around the origin. In Table 1 we have listed timing results together with other features of our test examples.

In Figure 7 some complex shapes have been tested with our method. The Star-shape is stressing the node spacing in the regular sampling. Likewise, the small details on the dragon (fingers, the backside horns etc..) are difficult to capture with as coarse a grid-resolution as ours.

Regular sampling is problematic, and the resolution should depend on the curvature of the shapes. Adaptive fields [11] may turn out to be a future improvement, or other field types [5] that can be evaluated from closed forms might be interesting.

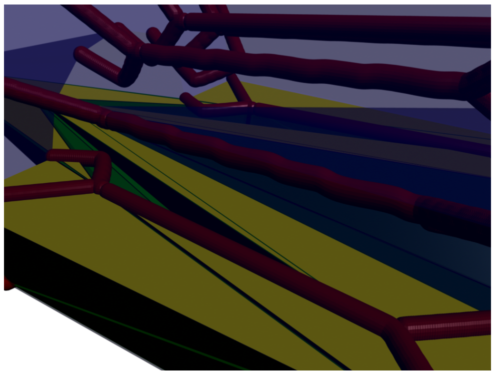


Fig. 8 First order explicit Euler stepping causes path traces to twist and bend in cases, where the path traces are expected to be straight lines.

The path tracing is done with fixed increments in the gradient direction. This is similar to solving an ordinary differential equation using an explicit first order Euler method. The step-size contributes with an $O(\Delta\varepsilon)$ discretization error. The effect is that the path traces are not as straight and smooth as one would expect. This can be observed in Figure 5(a), where the long path traces are not completely straight lines; a close-up is shown in Figure 8.

Computing the gradient of a regular sampling using central differences contributes with a $O(\Delta\varepsilon^2)$ discretization error, and finally the interpolation at non-nodal positions adds numerical diffusion. In combination, this means that sub-cell accuracy of the regular sampling is difficult to achieve, and path traces are therefore sometimes stopped too early. The effect of the regular sampling on path traces is seen in the case of the bowl and funnel objects in Figure 9.

The last parameter in our implementation is the point-size. The point-size is basically the accuracy by which we detect merge points. Setting the parameter too high may cause path traces to merge too quickly. The sphere shape in Figure 9 illustrates this. Setting the value too low might mean that merge points are not detected when expected. This is because the discretization error of the explicit Euler stepping is larger than the point-size. One remedy may be to detect merge points using a k-nearest neighbor search or to use adaptive step-size control for the path trace. We leave these ideas for future work.

As seen from Table 1 meshes of size 1000-2000 take roughly 500 seconds to compute with our current implementation. The computational complexity of our method is bilinear in the number of vertices and the resolution of the regular sampling. In theory each path trace can be computed in parallel. Note that the point-size of all the points of the path traces are the same, thus yielding optimal query time of the spatial hashing scheme.

After having built a shell mesh, a post-processing step can be performed to improve the mesh quality. Vertices extruded from opposite sides of the original surface mesh may meet

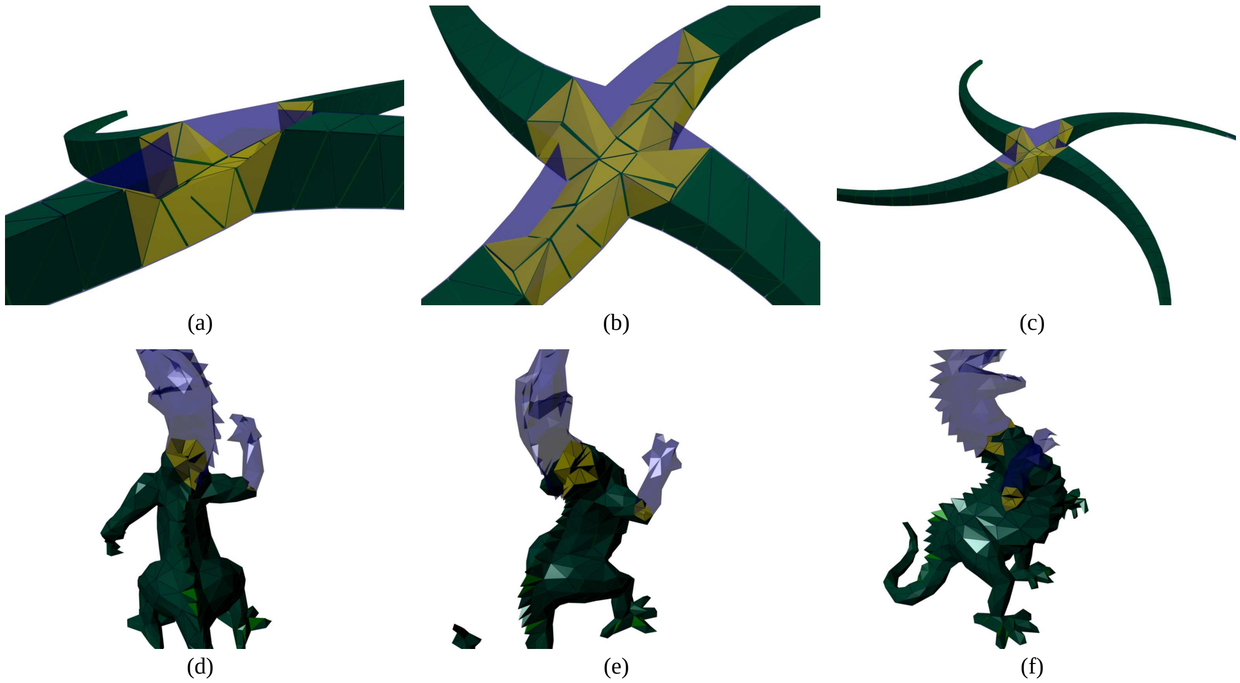


Fig. 7 Examples of shell meshes of more complex shapes: (a)-(c) is the Star object, and (d)-(f) is the Dragon object seen from various directions.

along the medial surface. This implies that intruded vertices from opposite sides coincide within numerical accuracy. In a post-processing step, a new shell mesh can be build by performing two passes: In the first pass new unique vertices are created in the new shell mesh, while keeping track of the corresponding vertices in the old shell mesh. In the second pass tetrahedra are created in the new shell mesh. Prior to creating a tetrahedron it is verified that all its vertices are unique, and only in this case is the new tetrahedron created. The improvements in quality are shown in Figure 10. The figure clearly shows an improvement, but it also reveals that mesh quality is not good. In our case we focused on making the thickest possible shell mesh. Thus, for most cases this obviously results in slivers and needles. To achieve some regularity perhaps one should choose a local maximum thickness of each surface vertex related to the average edge-lengths of the neighboring edges of the vertex.

The mesh quality of the algorithm, we use [9], has never been examined. A possible improvement may be to use another tessellation method or to expand the post-processing by an optimization process [14]. This is quite extensive and not the focus of the work presented in this paper. We therefore, leave it for future work.

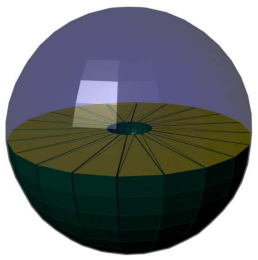
In Figure 11 we compare a result from our method with other methods. In Figure 12 we show the shell mesh of the horn object using our practical suggestion for choosing the shell thickness. The figure also illustrates the two main sources for the generation of silvers and needles. In the supplementary video we have used the shell mesh from the horn object in Figure 12 to animate a deformable object hitting a plane using FEM and linear damped penalty forces. Due to the thickness of the shell the object appears solid and not hollow even with a large residual space. The low tetrahedron count of the shell mesh yields real-time performance and reasonable animation quality even though the shell contains tetrahedra of poor quality.

4 Conclusion

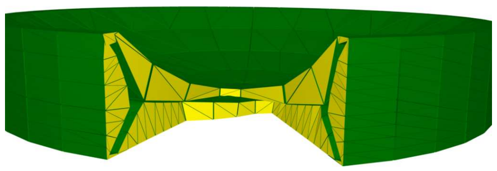
In this paper we have shown a proof of concept, of how to create volumetric shell meshes by performing path traces in signed distance and flux fields. The resulting shell meshes can be as thick as one wishes while still respecting the structure of the object shapes. Our main contribution is an algorithm using path traces to detect merge points, which yields the final displacements of points of the original surface mesh.

Our test results show that the tessellation algorithm used does not create a mesh of suitable quality for numerical simulation. However, we did demonstrate that simple post-processing could improve mesh quality. We leave the problem of finding a “good” tessellation for future work.

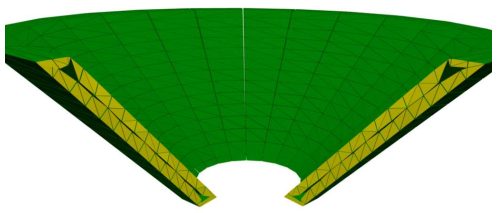
Our shell mesh generation method completely relies on the properties of the surface meshes. Especially, they must be non-intersecting twofolds, also know as water-tight surfaces. This is a major drawback, since models created by artists often have all kinds of degeneracies. We believe that our work can be extended to deal with unpleasant meshes using a coarse resolution pseudo volume as in [21]. From the



(a) Notice the empty space in the center of the sphere.



(b) Notice the empty space at the top of the bowl



(c) Notice the empty space at the top and bottom of the funnel.

Fig. 9 In these examples the point size parameter was set too high. This caused path traces to be merged before reaching their true critical points or too far from the precise merge point. The funnel and bowl are further troubled by having a thin structure comparable to the node spacing used.

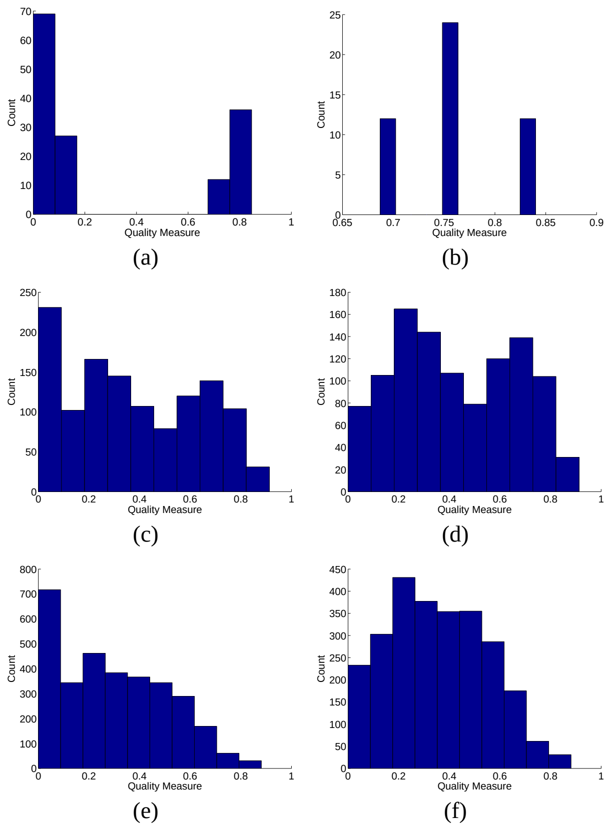


Fig. 10 Histograms showing how the volume length quality measure [14] is improved by removing redundant vertices of the shell meshes. In (a), (c), and (e) histograms of quality measures are shown for the box, horn and teapot objects respectively. In (b), (d), and (f) the corresponding histograms are shown after post-processing.

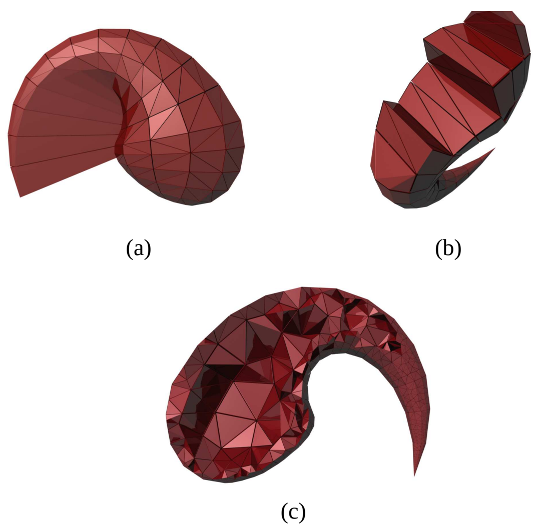


Fig. 11 Comparisons of generated tetrahedra meshes of the Horn object. In (a) the result of [17] using extreme thickness, (b) a constrained Delaunay Triangulation made using TetGen, and (c) a quality mesh using TetGen. Observe that (a) is clearly illegal. Notice that (b) and (c) cannot possibly be used as shell meshes. Observe the large number of elements in (c).

pseudo volume a twofold surface can be created by isosurface extraction. Further smoothing or constrained Delaunay re-triangulation can be applied to generate a pleasant low resolution water tight surface mesh. The coarse generated shell mesh can be coupled to the high-resolution artistic mesh for visualization purpose.

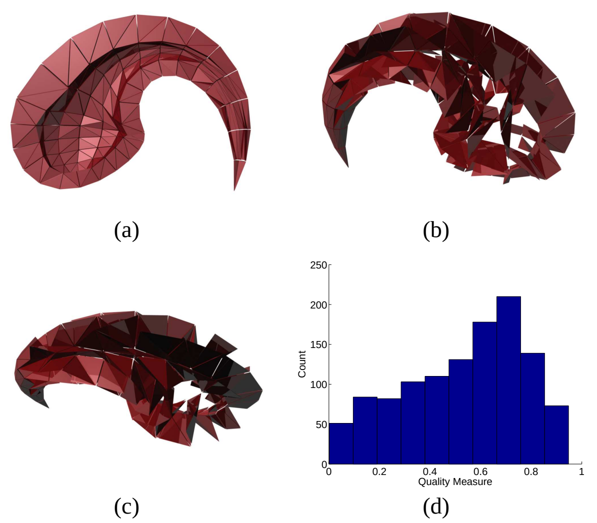


Fig. 12 In (a) a shell mesh is shown using our method. The thickness is chosen locally based on the average edge length. In (d) the corresponding volume length ratio quality measure histogram. Observe that the quality is better than in Figure 10 (c). In (b) and (c) all tetrahedra with a volume length ratio less than 0.5 are displayed. Notice that in the tail-part the generated tetrahedra are dominated by the original surface triangles, and the small thickness results in slivers and needles. Along the convex curve one observes almost paper thin tetrahedra. This is an artifact from the ripple-tessellation method.

References

1. Alliez, P., Cohen-Steiner, D., Yvinec, M., Desbrun, M.: Variational tetrahedral meshing. ACM Trans. Graph. **24**(3), 617–625 (2005). DOI http://doi.acm.org/10.1145/1073204.1073238
2. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. In: SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications, pp. 249–266. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/376957.376986
3. Bouix, S., Siddiqi, K.: Divergence-based medial surfaces. In: ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I, pp. 603–618. Springer-Verlag, London, UK (2000)
4. Chuang, J.H., Tsai, C.H., Ko, M.C.: Skeletonization of three-dimensional object using generalized potential field. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(11), 1241–1251 (2000). DOI http://doi.ieeecomputersociety.org/10.1109/34.888709
5. Cornea, N.D., Silver, D., Min, P.: Curve-skeleton applications. In: IEEE Visualization, p. 13 (2005)
6. Diatta, A., Giblin, P.: Pre-symmetry sets of 3D shapes. In: Proceedings of the 1st International Workshop on Deep Structure, Singularities, and Computer Vision, *Lecture Notes in Computer Science*, vol. 3753, pp. 36–49. Springer Verlag (2005)
7. Dimitrov, P., Damon, J.N., Siddiqi, K.: Flux invariants for shape. In: CVPR (1), pp. 835–841 (2003)
8. Erleben, K., Dohlmann, H.: The thin shell tetrahedral mesh. In: S.I. Olsen (ed.) Proceedings of DSAGM, pp. 94–102 (2004)
9. Erleben, K., Dohlmann, H., Sporring, J.: The adaptive thin shell tetrahedral mesh. Journal of WSCG pp. 17–24 (2005)
10. Erleben, K., Sporring, J.: Line-stepping for shell meshes. In: B. Ersbll, K.S. Pedersen (eds.) Scandinavian Conference on Image Analysis (SCIA '07), *Lecture Notes in Computer Science*, vol. 4522. Springer Verlag (2007). (to appear)
11. Frisken, S.F., Perry, R.N., Rockwood, A.P., Jones, T.R.: Adaptively sampled distance fields: a general representation of shape for computer graphics. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 249–254. ACM Press/Addison-Wesley Publishing Co. (2000). DOI http://doi.acm.org/10.1145/344779.344899
12. Galoppo, N., Otaduy, M.A., Tekin, S., Gross, M., Lin, M.C.: Soft articulated characters with fast contact handling. In: Computer Graphics Forum, vol. 26. Prague, Czech Rep. (2007)
13. Kimia, B.B., Tannenbaum, A.R., Zucker, S.W.: Shapes, shocks, and deformations I: The components of two-dimensional shape and reaction-diffusion space. International Journal of Computer Vision **15**, 189–224 (1995)
14. Klingner, B.M., Shewchuk, J.R.: Aggressive tetrahedral mesh improvement. In: Proceedings of the 16th International Meshing Roundtable. Seattle, Washington (2007)
15. Molino, N., Bridson, R., Teran, J., Fedkiw, R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In: International Meshing Roundtable, vol. 12, pp. 103–114 (2003)
16. Müller, M., Teschner, M.: Volumetric meshes for real-time medical simulations. In: Proc. BVM (Bildverarbeitung für die Medizin), pp. 279–283. Erlangen Germany (2003)
17. Peng, J., Kristjansson, D., Zorin, D.: Interactive modeling of topologically complex geometric detail. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, pp. 635–643. ACM, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1186562.1015773
18. Pizer, S.M., Fletcher, P.T., Joshi, S., Thall, A., Chen, J.Z., Fridman, Y., Fritsch, D.S., Gash, A.G., Glotzer, J.M., Jiroutek, M.R., Lu, C., Muller, K.E., Tracton, G., Yushkevich, P., Chaney, E.L.: Deformable m-reps for 3d medical image segmentation. International Journal of Computer Vision **55**(2/3), 85–106 (2003)
19. Porumbescu, S.D., Budge, B., Feng, L., Joy, K.I.: Shell maps. ACM Trans. Graph. **24**(3), 626–633 (2005). DOI http://doi.acm.org/10.1145/1073204.1073239
20. Sethian, J.A.: Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press (1999). Cambridge Monograph on Applied and Computational Mathematics
21. Spillmann, J., Wagner, M., Teschner, M.: Robust tetrahedral meshing of triangle soups. In: Vision, Modeling, Visualization VMV'06, pp. 9–16. Aachen, Germany (2006)
22. Sud, A., Foskey, M., Manocha, D.: Homotopy-preserving medial axis simplification. In: SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling, pp. 39–50. ACM Press, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1060244.1060250
23. Teschner, M., B. Heidelberger, M.M., Pomeranets, D., Gross, M.: Optimized spatial hashing for collision detection of deformable objects. In: Proc. Vision, Modeling, Visualization, pp. 47–54. Munich, Germany (2003)

Kenny Erleben After his studies Erleben was employed as full time researcher in the Company 3DFacto A/S for a period of 10 months. In 2001 Erleben started on his Ph.D. studies. During 2004 Erleben stayed 3 months at the Department of Mathematics, University of Iowa. Hereafter he received his PhD degree in the begining of 2005 and finally late 2005 Erleben was employed as an Assistant Professor at the Department of Computer Science, University of Copenhagen.