Using Implicit Skeleton Shape Representations for Volumentric Shell Meshing

Kenny Erleben

Technical Report no. 07/04
ISSN: 0107-8283

Dept. of Computer Science
University of Copenhagen • Universitetsparken 1
DK-2100 Copenhagen • Denmark

Using Implicit Skeleton Shape Representations for Volumetric Shell Meshing

Kenny Erleben*
Department of Computer Science, University of Copenhagen, Denmark

September 16, 2007

Abstract

Low count tetrahedral meshes is desirable for animating deformable objects where accuracy is less important and to produce shell maps. This report develops a new method for creating a thick shell tetrahedral mesh from a triangular surface mesh. We propose to use signed distance fields to implicit represent the medial surface representation of a surface mesh. An iteratively algorithm is developed using line-stepping as the main ingredient. Line-stepping does not suffer from the convergence problems of past work and is parallel in nature.

1 Introduction

Many graphical models of solid objects are given as surface meshes [22, 3], since this is an economical representation for visualization and easily obtainable by laser scanning of real objects or by hand-modeling using a 3 dimensional drawing tool. However, animating deformations of solid objects requires a notion of inner structure which is surprisingly difficult to obtain. Medical scanners such as MR and CT, do offer a 3 dimensional volume measurement, but such apparatus are expensive to operate and not commonly available for non-medical applications. Existing algorithms such as [23] are difficult to implement and do not use the natural, intrinsic representation of shape by symmetry sets [10, 27].

Existing tetrahedral mesh generation methods in the literature typically create an initial, blocked tetrahedral mesh from a voxelization or signed distance field. Afterward, nodes are iteratively repositioned, while tetrahedra are subsampled in-order to improve mesh quality [24, 26, 23], or the variational gradient of an energy functional is used to move vertices [1]. In contrast to these methods, our is surface-based.

Shell meshes are attractive since they give a volume representation of a surface mesh with a very low tetrahedral count, which is desirable for animation

*{kenny}@diku.dk

or similar purposes, where speed is preferred over accuracy of deformation. The shell mesh finds applications in animating solid objects, for shell maps [28], and for the calculation of signed distance field [14, 29]. Recently a thin-shell of rigid prisms [4] have been used for shape deformation. Figure 1 shows examples of volumetric shell meshes.

In this report we will present extensions of [13, 15, 16]. In order to show the efficiency of our method, our main goal is to create the thickest possible shell mesh with the lowest possible tetrahedral count. I.e. given a polygonal surface mesh, we create a tetrahedra volume mesh representing a thick version of the surface mesh, a shell mesh. In past work vertices of the polygonal surface mesh are displaced inward, thereby creating a new version of the surface mesh. This operation is in the literature termed inward extrusion or just simply extrusion, although intrusion seems a better term. Following an inward extrusion the original surface mesh is used to generate the outside of the shell and the extruded surface mesh is used to generate the inside of the shell mesh. The two meshes is then used to create a triangle prism shell mesh. Finally, the triangle prism mesh are converted into a consistent tetrahedral mesh, also known as tetrahedral tessellation.

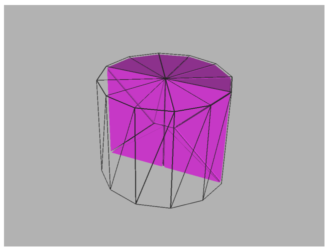
The suggested prism generation is reminiscent of an erosion operation with a spherical structural element on the polygonal model. The radius of the sphere corresponds to the extrusion length. It is well known that working directly on the boundary representation [30] is fast and simple, but topological problems arises easily such as shocks [19]. The counter-part to shocks are degenerated prisms, that is prisms with less than 6 vertices. These shocks turn out to be the limit on the extrusions lengths.

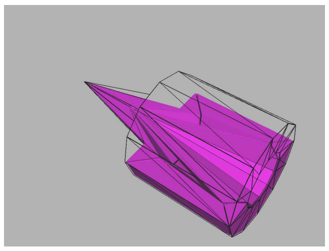
A linear randomized tessellation algorithm, the ripple tessellation, was developed in [13]. The ripple method suffers from several problems. It is not deterministic, but relies on picking random ripple directions to fix inconsistencies. Further, no proof has been given on existence of a consistent tetrahedral tessellation of the triangle prism shell. A safe conservative, upper extrusion length limit is used in [13] and later improved in [15]. Nevertheless, both algorithms are very slow due to bad and unpredictable convergence of the bisection search method. In present article we disregard the consistent tetrahedra tessellation problem and present new extrusion approaches.

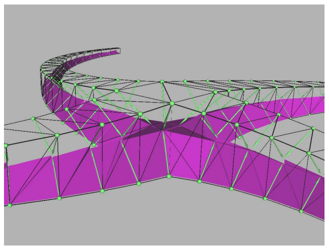
Throughout this report it is implicitly assumed that the reader is familiar with signed distance fields [30, 25], medial axis/surface representations (MREPs), or the more general concept of symmetry sets. It is assumed that the reader is familiar with polygonal and tetrahedra meshes. It would be beneficial to read our past work on the subject [13, 15, 16]

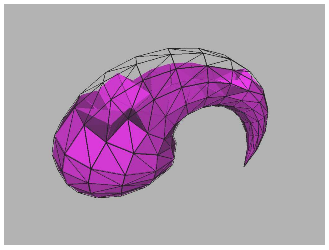
2 Line-Stepping in Signed Distance Field

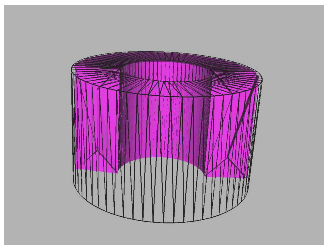
Intuition dictates that we really want to take a surface point and hit the corresponding point on the medial surface, since it seems logical that this would be the maximum extrusion length. However, obtaining the medial surface is

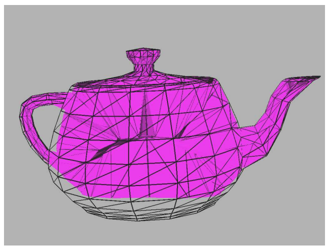

(a) Cylinder


(b) Pointy


(c) Star


(d) Horn


(e) Cylinder with hole


(f) Teapot

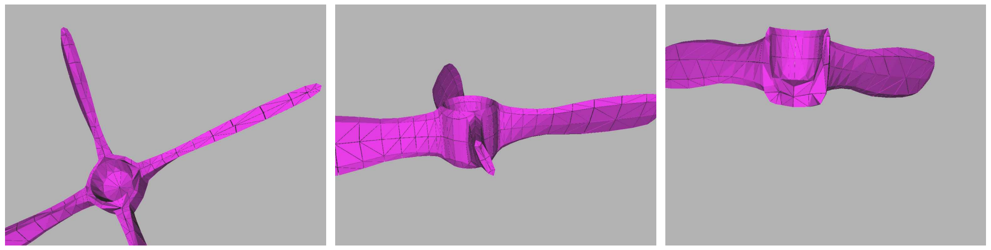

(g) Propeller

Figure 1: Cut-views showing the shell layers inside the volumetric meshes generated using our method: Force-following in a signed distance field from section 5. Notice that even when only using a single shell layer almost no internal empty space are present. This demonstrates the ability to achieve the maximum possible inward extrusion of the surface mesh.

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 as we will explain shortly.

In [16] line-stepping was shown to be more efficient than any root-search approach. Here we will briefly explain the main idea behind line-stepping and refer to [16] for details.

When doing line-stepping, we extrude a surface point $\vec{p}$ along a line. The position at the i'th step along the line is denoted by $\vec{q}^{\,i}$. To obtain the position $\vec{q}^{\,i}$ a fixed incremental step of length $\Delta\varepsilon$ is taken in the opposite direction of the surface normal $\vec{n}$. Given a signed distance field, ϕ, the surface normal is given by the gradient of the signed distance field at the surface position,

$$\vec{n} = \nabla\phi(\vec{p}). \tag{1}$$

The position $\vec{q}^{\,i}$ can be seen as a function of the extrusion length ε^i. That is

$$\vec{q}^{\,i} = \vec{q}(\varepsilon^i) \tag{2}$$

In the i'th step the extrusion length is updated by

$$\varepsilon^i = \varepsilon^{i-1} + \Delta\varepsilon \tag{3}$$

and the current extrusion point, $\vec{q}(\varepsilon^i)$, is found by

$$\vec{q}(\varepsilon^i) = \vec{p} - \varepsilon^i \nabla\phi(\vec{p}) \tag{4}$$

The stepping is performed as long as $\nabla\phi(\vec{q}^{\,i})$ points in the same direction as $\nabla\phi(\vec{p})$. If the cell size of the regular sampled signed distance field is given by Δx, Δy, and Δz then the increment is chosen as

$$\Delta\varepsilon = \frac{\min(\Delta x, \Delta y, \Delta z)}{2} \tag{5}$$

This ensures that we do not step along the extrusion line faster than the information changes in the signed distance field. This works due to spatial coherence of the values in the signed distance field. The value at a neighboring grid node in the signed distance field is different by at most

$$\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{6}$$

The stopping criteria we use is to keep on increasing ε^i while

$$\nabla\phi(\vec{p}) \cdot \nabla\phi(\vec{q}^{\,i}) > \rho \tag{7}$$

where $\rho > 0$ is a user specified threshold to control accuracy. We call this the normal-test. The intuition behind the normal-test is that if we pass the medial surface from one side of an object to the opposite side, then the gradients in the signed distance field will flip directions. That is the sign of the normal-test will flip from positive to negative.

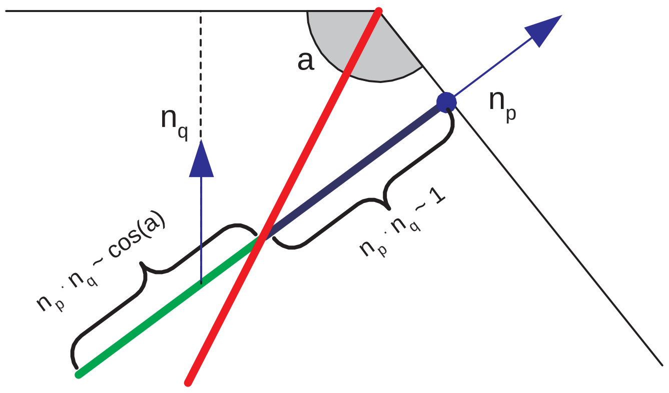


Figure 2: An extrusion line can cross over the symmetry set if the surface angle a is less than the accepted angle difference between $\nabla\phi(\vec{q})$ and $\vec{n}$. The directional derivative of $\nabla\phi(\vec{q}) \cdot \vec{n}$ is 1 until the symmetry line is hit. However, the distance value is still decreasing while stepping along on the other side of the symmetry line. To stop at the symmetry line we must detect the change in the value of the directional derivative.

It is worth noting that the normal-test is in fact the directional derivative at the position $\vec{q}^{\,i}$. The sign of the directional derivative therefore tell us something about how ϕ changes, as we move in the opposite direction of the surface normal vector $\vec{n}$. In our specific case the following rules applies:

$$\nabla\phi(\vec{q}^{\,i}) \cdot \vec{n} \begin{cases} < 0 & \phi \text{ is increasing} \\ > 0 & \phi \text{ is decreasing} \\ = 0 & \phi \text{ is constant} \end{cases} . \tag{8}$$

The actual value of the directional derivative tell us something about how fast ϕ changes, while we step in the normal direction. Intuitively, we want to keep on extruding inward as long as ϕ is decreasing. However, this is not quite enough, because we do not want to pass over the symmetry set of the object.

An extrusion line may cross over a symmetry line, if we only require the directional derivative to be positive. This is illustrated in Figure 2. To avoid this problem we must require that the gradient of the signed distance field do not differ from the surface normal by more than some specified angle.

$$\nabla\phi(\vec{q}^{\,i}) \cdot \vec{n} > \rho \tag{9}$$

where ρ is the cosine of the accepted angle difference, α.

$$\rho = \cos(\alpha) \tag{10}$$

In our test examples we used $\alpha = 0.4363$ radians (approximately 25 degrees), which means $\rho \approx 0.9$. The cross-over can still occur and would cause overlapping regions of the resulting shell mesh. All we have achieved is to reduce the number of cases, where a cross-over would occur. Depending on the resolution and accuracy of the signed distance field, equation (9) can be extremely sensitive to numerical errors. In such cases setting ρ too tight would result in almost no extrusion.

In Figure 3 we have shown planar cross-intersections of a few shell meshes generated using the line-stepping method. The more complex teapot shape do suffer from a cross-over problem. However, aside from this it is clear that the presented extrusion method is capable of filling the internal void inside the surface meshes (shown as black wire-frame).

The time-complexity of the proposed method is govern by two parameters, the signed distance field resolution, N, and the number of vertices, V. Each extrusion line is treated independently of each other. Thus, the algorithm scales linear in the number of vertices. The number of steps that can be taken along an extrusion line is bounded above by the maximum number of grid nodes encountered on the diagonal of the regular grid. Thus,

$$O(V\sqrt{3\tfrac{1}{2}N^2}) \approx O(VN). \tag{11}$$

This is extremely fast since the operations done during each step have very low constants (computing the gradient of a scalar field sampled on a regular grid). The major performance cost is the computation of the signed distance fields. We refer to the paper [14] for details on the performance of signed distance field computations.

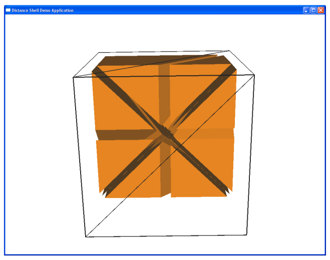
2.1 Pre-processing

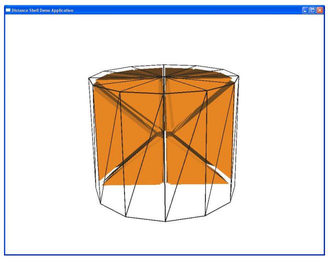
A simple pre-processing strategy has been adopted, where the surface mesh is iteratively changed to closely approximate a constrained surface Delaunay triangulation. Initially mesh faces with an area larger than a user specified threshold is sub-divided using the edge mid-points to generate a new triangulation. This has the benefit of sub-dividing non-planer edges. Hereafter edge-flipping is performed on the planar edges of the sub-divided surface mesh. If an edge has two triangular face neighbors then an edge flip is performed whenever a non-shared triangle vertex of one of the triangle faces is outside the circumscribed sphere of the vertices of the other triangle face. A dihedral angle threshold is used to test edges for planarity.

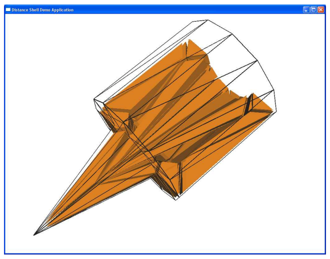
It is a brute and simple approach for improving the surface mesh quality. Results are shown in Figure 4. The method could be substantially improved by using a priority queue to pick the next face/edge to subdivide/flip. However, it serves our purpose of illustrating how sampling artifacts due to poor tessellation can be improved.

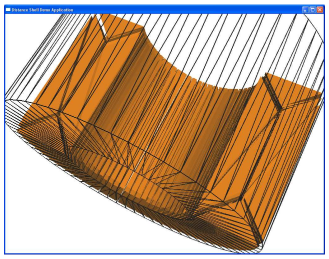
2.2 Post-processing

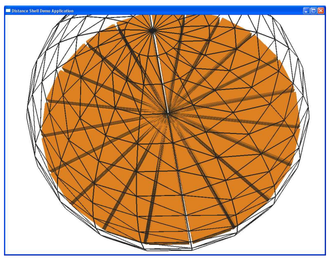
After having build a tetrahedral shell mesh a post-processing step can be performed to improve the mesh quality. It may well be that vertices are extruded

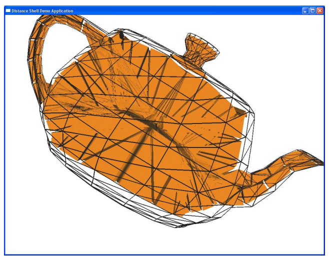

(a) Box. Note all extrusions is along symmetry lines.

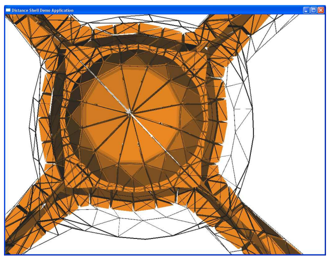

(b) Cylinder.


(c) Pointy, suffering from void sampling artifact.


(d) Tube.


(e) Sphere.


(f) Teapot, suffering from cross-over problem.


(g) Propeller 1. Quality of thin regions are highly dependent of signed distance field resolution.

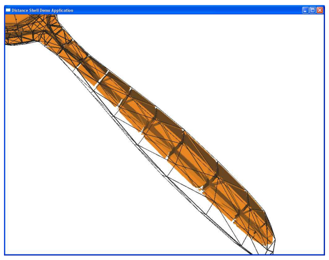

(h) Propeller 2.

Figure 3: Results of the line-stepping method. Using signed distance field of resolution 256^3, and a normal-test with $\rho = 0.9$. Observe the overlapping shell mesh of the teapot.

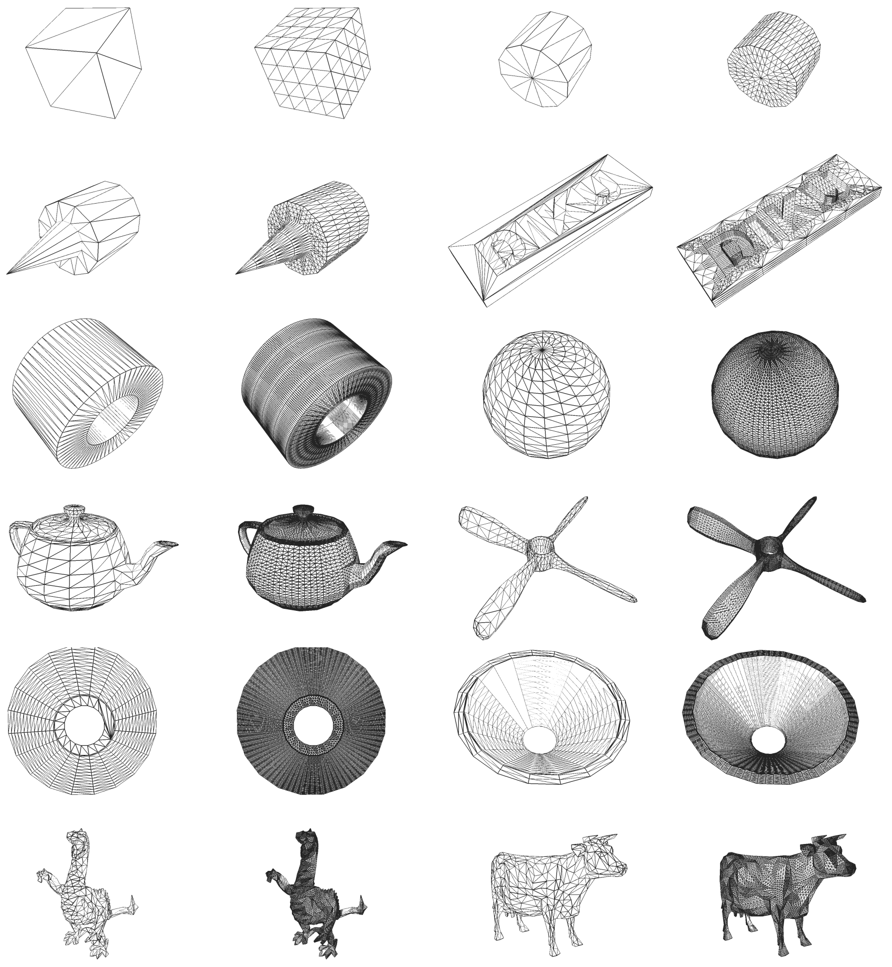


Figure 4: Results of pre-processing surface meshes. Using two sub-division iterations interleaved by three constrained Delaunay triangulations. Observe that after pre-processing the surface triangles appear more regular. A needle like surface triangle would result in a flat tetrahedra in the resulting shell mesh.

from opposite sides of the original surface mesh are meeting along the symmetry set, implying they have the same coordinates (within numerical precision). That is to say we have redundant vertices in our shell mesh.

The redundant vertices can be eliminated quickly using optimal spatial hashing [34]. Here we initially map all vertices into the hashtabel and then perform box queries. For each vertex we construct an axis aligned bounding box by adding and subtracting a small value from the coordinates of the vertex. In our examples we used a value of the same order as the grid-spacing in the signed distance fields.

During the queries a new tetrahedra mesh is built. While performing the box queries, a vertex is inserted into the new mesh, only when the bounding box of the vertex contains no other vertices. Finally the tetrahedra is inserted in the new mesh after having inserted all the non-redundant vertices. Prior to inserting a tetrahedra it is tested, whether its four vertices are all distinct. This is because removable of redundant vertices will cause zero-sized (within precision) tetrahedra to have non-distinct vertices.

3 Divergence of Signed Distance Field Gradient

As can be seen from our test results the line-stepping approach is not perfect. The extrusion is not as aggressive as we hoped. This was due to the rather large threshold on the directional derivative. Even worse the threshold value adds an element of parameter tuning. There were other problems too, which caused degenerated extrusions (tentacles) and sampling artifacts (voids and overlaps, see [16] for details). In this section we will propose another approach to the extrusion problem. Before doing so we will summarize, what we have learned about line-stepping so far. The line-stepping stopping criteria can be summarized in two simple rules

(1) If the initial surface point is lying on the medial surface, then inward extrusion is performed until that point, where the extrusion line leaves the medial surface or a junction (in case we are stepping along a seam) is encountered, or where a seam is encountered (in case the extrusion line is embedded in a sheet). In other words line-stepping is done until the point where the medial surface "changes".

(2) If the initial surface point is not part of the medial surface then line-stepping is done until the extrusion line hits the medial surface.

Note that (1) means that the initial surface point is a strictly convex point or a saddle-point of the surface and (2) means the point is either flat or strictly concave. The rule that needs to be applied to a surface point can be determined from the surface geometry only, the medial surface is in fact not needed for this purpose.

The above two rules form a simple stopping criteria that is easy to understand, but not easy to implement. Obtaining the medial surface is known to be a very difficult problem, and many settle for simplifications or approximations of the medial surface, such as the simplified medial axis [32, 17, 31], the power crust algorithm [2], or divergence based medial surfaces [5, 12, 11]. Thus getting an exact representation of the medial surface is a problem in itself.

Even if we were able to obtain a perfect and exact medial surface of our object, it would still be problematic to apply the first rule of the stopping criteria, because it relies on testing, whether an infinite thin line is embedded in an infinitely thin seam or sheet. Alone the numerical inaccuracy involved in updating the extrusion point would drive the extruded point off the medial surface. Besides, we must expect some error in the initial estimation of the extrusion direction, making the problem of unintentionally slipping off the medial surface even more difficult. Thus, for a practical implementation we need a numerical more robust approach. We reason as follows:

- A signed distance field is in a sense a dual representation of the medial surface, since the medial surface can be found looking at extrema and discontinuities in the signed distance field.
- A signed distance field is often sampled on a regular lattice. Thus, it is difficult to reconstruct the exact position of the medial surface that is implicitly represented by the signed distance field.
- However, it is fairly easy to build a new field telling us whether the medial surface passes through a given voxel.

The θ-SMA algorithm [31] can compute such a field. However, we have instead 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 [5] or the average outward flux (AOF) field [12, 11], and is defined mathematically as follows:

$$F(i, j, k) = \int_S \nabla\phi \cdot \vec{ds} \tag{12}$$

Numerically we evaluate F for each grid node as follows: the surface S is chosen as a cubical surface, having the node (i, j, k) at 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 it corner points. The surface integral is evaluated using a summation approximation

$$F(i, j, k) = \sum_{a \in N(i,j,k)} \nabla\phi(a) \cdot \vec{n}(a) \tag{13}$$

where $N(i, j, k)$ is the index set of the $3 \times 3 \times 3$ neighboring voxels (corners, face mid-points, and edge-midpoints of the surface cube), and $\vec{n}(a)$ is the outward cube surface normal. For $\nabla\phi$ we used a first order central difference approximation. Thus the flux at a single node is based on a measure from a neighborhood of $5 \times 5 \times 5$ voxels. In our experience this adds a lot of 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.

If F is zero it is a clear indication that the medial surface is not passing through the cube surface. If the value is negative then we know that the cube surface contains a discontinuous line originating from a concave surface point, strictly speaking such lines are not part of the medial surface, thus we can ignore them. The last possibility is if F is positive this means that the cube surface contains part of the medial surface. Larger positive values indicate greater discontinuity, thus a small value would imply a sheet, larger value a seam, and even larger values a junction. We can think of this as a likelihood of encountering the medial surface in a given voxel. The larger positive value the more certain it is that we have the medial surface.

The cube surfaces of neighboring voxels overlap by one-voxel and the $5 \times 5 \times 5$ stencil adds a certain amount of smearing thus we will have a thicken representation of the medial surface. From our viewpoint this is an advantage since it helps avoid the line-stepping to unintentionally falling off the medial surface in case of the first stopping criteria rule.

Our initial idea is simply to use F for the line-stepping, the value of F is recorded at the initial surface point, during line-stepping it is tested if the value of F at the extruded point deviates to much from the initial value. If this is the case we halt the line-stepping. Otherwise we keep on stepping along. When a non-medial surface vertex hits the medial surface then a sub-pixel accuracy can be achieved by searching for the position between the old step position and the new step position that maximizes the AOF value. Seams and sheets may have higher order curves, therefore we may need to adjust the line-step direction when we trace a vertex lying on the medial surface. The line-step direction correction is treated in detail later in Section 3.1.

Having taken care of the possible curved nature of the medial surface, we need to consider how to detect if we slipped off the medial surface or when we hit something on the medial surface that should cause us to stop our search. The first problem is dealt with easily by testing whether the AOF value dropped below a certain value at the current step position. The other problem is non-trivial, intuition tell us that we should test for when we hit a lower dimensional feature of the medial surface. For instance if we are tracing along a seam we want to stop the stepping at the end junction point. If we are tracing inside a seam we would like to halt when we hit a seam or junction. These locations are detected in the AOF field as discontinuities. We therefore apply a relative test between the AOF value at the initial surface point and the current step-position. It would be interesting to classify voxels in the AOF field as being junctions, seams, and sheets. However, we have not seen any 3D extensions on the work done in [12, 11] on flux invariants. This may be an interesting research problem in itself, and we leave it for future work.

We have outlined the AOF line-step algorithm in Figure 5. Notice that traces along the medial surface may curve. Therefore we perform a final correction of the extrusion direction and the total extrusion length.

```
Algorithm line-step-AOF
  for each extrusion line $l$ do
    do
      $\vec{q}_{fake} = \vec{q}(l) - \vec{n}(l)\Delta\varepsilon$

      if $\text{flux}(\vec{p}(l)) < \delta$ and $\text{flux}(\vec{q}_{fake}) > \delta$ then
        $\vec{q}_{fake}$ =maximum point from $q$ to $\vec{q}_{fake}$
        break // non MREP-vertex hits MREP
      end if

      if $\text{flux}(\vec{p}(l)) \geq \delta$ and $\text{flux}(\vec{q}_{fake}) < \delta$ then
        $\vec{q}_{fake} = \text{line-search-correction}(l)$
        if $\text{flux}(\vec{q}_{fake}) < \delta$ then
          break // MREP-vertex slipped off MREP
        end if
      end if

      if $\text{flux}(\vec{p}(l)) \geq \delta$ and $|\text{flux}(\vec{q}_{fake}) - \text{flux}(\vec{p}(l))/\text{flux}(\vec{p}(l))| > \varepsilon$ then
        break  // MREP-vertex hit discontinuity in AOF
      end if

    while forever
    $\vec{q} = \vec{q}_{fake}$
    $\vec{n} = unit(\vec{q} - \vec{p})$
    $\varepsilon = \|\vec{q} - \vec{p}\|$
  next l
End algorithm
```

Figure 5: Line-stepping in AOF field

One major drawback of our method remains: fine detailed objects or objects with thin structures need high resolution of the signed distance field. Figure 6 and 7 shows results of three simple surface meshes using different field resolutions for the AOF computation. We did not perform any line-step direction correction in any of these tests.

Examining the close-up views in Figure 7 it becomes evident that the direction of the extrusion lines are misaligned with the medial surface. Implying that we unintentionally slip off the medial surface while doing the line-stepping. However, it is clear that the average outward flux (AOF) is a great way to identify voxels containing the medial surface. To circumvent this problem we propose to add a line-correction processing step, which we will explain in the next section.

3.1 Correction of Line-Stepping Direction

Assume we are line-stepping along a extrusion line with initial vertex position $\vec{p}$, surface normal $\vec{n}$ and an AOF value, $\theta_p = \text{flux}(\vec{p})$, at $\vec{p}$ larger than some small threshold, δ. If we during the line-stepping hits a position $\vec{q}^i$ of the extrusion line where the AOF value is below the threshold δ then this indicates that we have slipped off the medial surface. To remedy an unintentional slip-off we will try to find a better $\vec{q}^i$-position and then afterward correct the $\vec{n}$-normal which indicates the line-stepping direction.

Here is how it works, when doing the line-stepping we first make a fake-position update

$$\vec{q}_{\text{fake}} = \vec{q}^i - \vec{n}\Delta\varepsilon \tag{14}$$

then we lookup the AOF value at the fake position, $\theta_q = flux(\vec{q}_{\text{fake}})$, If $\theta_q < \delta$ and $\theta_p > \delta$ then we have a slip-off. In which case we compute

$$\vec{t} = \nabla\phi(\vec{q}_{\text{fake}}) \tag{15}$$

$$\vec{q}_a = \vec{q}_{\text{fake}} \tag{16}$$

$$\vec{q}_b = \vec{q}_{\text{fake}} - \vec{t}(2\Delta\varepsilon) \tag{17}$$

That is we have created a small line segment originating from the fake position and going sufficiently far in the opposite distance field gradient direction.

The reasoning being that we know that $\vec{q}^i$ is lying within distance $\Delta\varepsilon$ of the medial surface, thus $\vec{q}_{\text{fake}}$ cannot have moved farther away from the medial surface than $\Delta\varepsilon$. To make sure we walk at most twice that distance toward the medial surface. The opposite gradient direction will point in the direction of the medial surface, but not necessary yielding the shortest direction.

A linear search can be done along the line segment from $\vec{q}_a$ to $\vec{q}_b$ looking for the position that maximize the flux value θ. The linear search is illustrated in Figure 8. A more elegant solution would use a binary search. This could be done as shown in Figure 9. Once we have found a new better position $\vec{q}_{\text{fake}}$ we can correct the line-stepping direction by using a back-projection onto the sphere centered at $\vec{q}^i$ and having radius $\Delta\varepsilon$

$$\vec{n} = -\text{unit}(\vec{q}_{\text{fake}} - \vec{q}^i) \tag{18}$$

$$\vec{q}_{\text{fake}} = \vec{q}^i - \vec{n}\Delta\varepsilon \tag{19}$$

Now as a last test we verify whether the AOF value at the fake position, $\vec{q}_{\text{fake}}$, is still less than the small threshold value δ. If this is the case we have truly slipped-off the medial surface and should halt the line-stepping. On the other hand if we pass the test we can simply continue our line-stepping.

3.2 Ignoring Medial Surface Vertices

The line-stepping correction improve results significantly, but unfortunately we still have a problem. The problem is illustrated in Figure 10. The problem is that line-stepping along the symmetry set do not respect the surface tessellation. Ideally the resulting volume mesh should have the same shape at different resolutions. However, as the figure shows this is no longer true.

This lead us to the following solution to the problem:

- Make sure flat surfaces of the object holds a sufficient number of vertices. This can be achieved by using our pre-processing solution from Section 2.1.

These types of vertices are easily dealt with by line-stepping. Thus, we are guaranteed a fine extrusion length computation of such vertices.

- Drop the medial-surface line-stepping vertices. That is if an extrusion line originates from a mesh vertex having positive AOF value then the extrusion line is ignored. These vertices are problematic, so we simply ignore them. The reasoning behind this is that we can just make the subdivision in the pre-processing step as fine at we want. In the limiting case the "missing" medial surface vertices will not be noticed.

To get sub-pixel accuracy we perform a line-search for the medial surface when we hit it. If we went too far then we know our last position was valid, and so we set up a binary search between the last good position and the current bad position in order to find the optimal point.

From a mathematical viewpoint the AOF only have non-zero values on infinitely thin manifolds (sheets, seams, and junctions of the medial surface) in space. We have sampled the AOF on a finite regular grid, due to this discretization we have a smooth representation of the AOF. One could call it a thickening or averaging. This implies that the true infinitely thin AOF is located at the local maxima of the discrete AOF.

Due to our line-stepping we know that we have one point (the old position) lying on one side of the maximum AOF value and another point (the current position) lying on the other side of the maximum AOF value. Thus, we do a binary search between the old and current positions where we try to maximize the AOF value along the line-stepping direction. This should result in subpixel accuracy (within numerical precision). We have listed the algorithm in Figure 11.

We have shown our results using this algorithm in Figure 12. Observe that the box-example looks perfect. However, the cylinder do not look as nice, there seem to be something going wrong at the corners where extrusion lines are ignored. This is because surface triangles with three convex or saddle-point vertices will not be extruded. This results in a zero-length extruded prism and causes an internal void inside the object. In fact if the box where not subdivided none of the triangles would be extruded. Thus there are two major drawbacks of this solution

- Internal voids may appear.
- Fine subdivision are needed to avoid too big internal voids.

One possible resolution to the internal void problem may be to displace the triangle vertices toward their centers before doing the actual extrusion, this would in our opinion correspond to cutting off the corners of the meshes. However, since a vertex may be part of more than one triangle there is some inherent ambiguity that need to be resolved. This could probably be resolved by making a random choice. Another solution may be to simply add noise to the medial surface vertices until they all are classified as non-medial surface vertices according to their AOF field value. However, the extrusion length we would achieve would be of the same order as the noise displacement. Thus, we end up with the too thin shell problem again. Our shell generation algorithm were intended for computer graphics, therefore the second drawback with having fine subdivision resolutions is totally un-acceptable. This will create a shell mesh with a high number of tetrahedra elements. In conclusion, it is not advisable to ignore medial surface vertices of the surface mesh, and subdivision only results in a final shell that do not fulfill our wish for a low tetrahedra element count. Thus, we are facing the problem of how to deal correctly with medial surface vertices. In Section 4 and 5 we will propose different approaches for line-stepping.

4 The Junctions Method

It is evident that correct handling of medial surface vertices is extremely important. It is also evident that line-stepping of non-medial surface vertices is very robust using the AOF field. From earlier attempts it is clear that line-stepping approaches for medial surface vertices is not very robust. Thus a slightly different approach could be used:

- Do AOF line-stepping on non-medial surface vertices. Store all the extruded positions in a point set $\boldsymbol{E}$
- Find all internal junctions and store them in a point set, $\boldsymbol{J}$.
- Create the point set
$$\boldsymbol{P} = \boldsymbol{E} \cup \boldsymbol{J} \tag{20}$$
- For each non-medial surface vertex, v, find the closest point in $\boldsymbol{P}$ and use this as the extrusion point for v.

Naturally, we need to decide how we can compute junctions in the medial surface from the shape representations we have been working with so far.

Traditionally when working with AOF (or signed distance) fields [35, 33, 5] junction detection is done by

- Obtaining voxels close to the medial surface, for instance all positive AOF voxels.
- Perform a thinning operation of the voxels.
- Convert the voxels into a graph representation.
- Compute a minimum spanning tree (MST) of the graph representation.
- Merge “straight” edges of the MST.
- Finally analyze branches in the MST, to determine the junctions.

This is a tedious approach, it requires many sub-steps and each sub-step is dependent on a number of user-specified parameters. In our opinion the approach is more accurately classified as voxel manipulation on a higher level. The amount of tweaking and tuning parameters immediately suggest that this is a very bad approach. A similar technique exist using signed distance fields instead of the AOF field [20, 35, 33].

We desire a different approach for junction detection which do not require too many parameters or too much tweaking. Since an automated "tool-box" for shell generation is wanted we favor simplicity and robustness.

As a first approach we tried to classify AOF voxels by placing a slightly enlarged sphere at the AOF voxel center. The radius of the sphere were equal to the signed distance field value at the voxel center. The idea was that if the sphere made at least four intersections with the polygonal surface then the voxel center would be classified as a junction candidate. Unfortunately this approach did not work. Too many voxels where being classified as junctions, because a planar surface consist of multiple triangles and sphere intersections are reported with all of these.

Obviously local minima (also termed critical points) in the signed distance field is related to the positions of a subset of the internal junctions in the medial surface. Initially we searched for strict local minima. However, as seen in Figure 13, some objects do not pose any strict local minima, but still have internal junctions. In other cases one can find a strict local minima that would be a useful extrusion point. In other cases such an strict local minima would cause the shell to penetrate the surface mesh. Even more importantly in the latter case we need other points than the junction points to ensure no surface penetration. Figure 13 illustrate the cases in 2D. In 3D we may want to add points on the medial surface seams as well. Points on the seams are also characterized by not being strict local minima in the signed distance field.

To test our hypothesis we apply a simple strategy for junction detection in a signed distance field. Initially we loop over all the voxels and for each voxel we look at its 26 neighboring voxels to determine whether any of these have a value less than the current voxel. If this is the case we know that the voxel we are looking at can not be a local minima.

Results of local minima detection in signed distance field is shown in Figure 14. From the figure it is clearly seen that our method for detecting local minima are too noisy. Basically it is not easy to see critical points with sub-pixel accuracy. This is not very surprising as is stated in [9]:

> Critical points are difficult to locate in a vector field, particularly because they do not necessarily occur at the given sample locations, but often occur in between sampling points. A good heuristic for detecting critical points is described in [18]: a zero in the vector field occurs when all three components of the force vector (x, y and z) vanish, thus, if we can identify a region where each vector component changes sign, the region is a candidate for containing a critical point.

We could have applied a similar strategy to detect local maxima in the AOF field, since these would indicate junctions. To add further robustness to our method we could also have applied an analysis of the shape-matrix. That is looking at the eigenvalues of the Hessian matrix.

Our approach for junction detection is flawed in more than one way. As stated detecting extrema with sub-pixel accuracy is very hard when looking at a voxel neighborhood. Thus, we need a completely different approach for such a task. Another problem is also quite obvious, not all junctions are identified by an extrema.

5 Force Following Approaches

We will now introduce a conceptual different approach to overcome the problems of robust detection of extrema based on classifying voxels. The method will also handle the cases where junctions are not defined by extrema. The basic idea is to perform line-stepping of all the surface vertices in a signed distance field. This type of method is also called path-following or force-following. The line-stepping is halted when a critical-point is passed. This is detected by testing whether the magnitude of the gradient of the signed distance drops below a 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 tends to smooth the signed distance field a little. Therefore, in our implementation we used a fraction of 1000 of the gradient at the initial surface position as the threshold value. To overcome further numerical problems we have smoothed the signed distance field using a curvature flow.

This line-stepping approach is similar to what we did in Section 2. However, this time we are really looking for when the line-traces are merging together and for the critical points at which the line-traces stop. At these positions we will find the "junctions" we are looking for.

In Figure 15 we have shown results for tracing all the surface vertices to the strict local minima in the signed distance field. From the figure it is seen that tracing signed distance fields looks nice, but it becomes difficult when sheets are encountered. Line-traces in the medial surface sheets seems to lack robustness, since the line-traces can merge unexpected or crawl towards another line-trace. This is especially seen for the pointy object in Figure 15(a).

One observation can be made, our strategy for picking nearest junction as the extrusion point is not correct. We need to trace vertices to the proper junction point. This is seen in Figure 15(d). In conclusion we need to keep track of which junction or critical point the line-trace of a surface vertex encounters first. This is shown in Figure 1.

5.1 Curve-Skeletons

The curve-skeleton is defined by line-stepping all the medial-surface vertices toward the critical points of a generalized energy potential field. The line-traces represent the curve-skeleton segments. When line-segments merge together a junction of the curve-skeleton will have been detected.

In [8] several applications of curve-skeletons are surveyed and methods for computing curve-skeletons are compared. In [6] a closed-form solution is presented for computing the generalized potential of a polygonal surface. It is used to trace out the curve-skeleton by straightforward line-stepping along the gradient direction. In [7] a different approach is taken, the generalized potential is computed on a voxel grid using the method of [6]. Afterward critical points are detected in the voxel grid and seed points are traced using the voxel grid, thus generating the curve segments. A more detailed description of the method can be found in [9], where the authors extend their work to include a hierarchical representation based on curve-skeletons. In [21] another approach is presented for computing the generalized potential field and the gradient hereof. It is also our method of choice in this report. Here we will briefly describe our approach.

For given (user-specified) positive integer value m, the generalized energy potential, E, in a point, $\vec{q} \in \mathbb{R}^3$, of space is defined as

$$E(\vec{q}) = \int_S \frac{dS}{\|\vec{p}(s) - \vec{q}\|^m} \tag{21}$$

where S denotes the surface of the object and $\vec{p}(s) \in \mathbb{R}^3$ is the point in space corresponding to the "surface" point $s \in S \in \mathbb{R}^2$. Straightforward computation yields the gradient of the energy potential function.

$$\begin{aligned} \nabla E(\vec{q}) &= \nabla \int_S \frac{dS}{\|\vec{p}(\vec{s}) - \vec{q}\|^m} & (22) \\ &= \int_S \nabla \frac{dS}{\|\vec{p}(\vec{s}) - \vec{q}\|^m} & (23) \\ &\approx \sum_{i \in N, q_i \in S} \nabla \frac{1}{\|\vec{p}_i - \vec{q}\|^m} & (24) \end{aligned}$$

In the last step we have approximated the integral by an summation of "infinite" set, N, of sample points. In practice we do of course not use a infinite number of points, but merely pick a sufficiently high number (more on this later). The approximation is somewhat strange. Notice that if N goes to infinity the gradient magnitude can grow without limit. There should have been a ΔS_i factor or something similar in the approximation.

$$\nabla E(\vec{q}) \approx \sum_{i \in N, q_i \in S} \nabla \frac{\Delta S_i}{\|\vec{p}_i - \vec{q}\|^m} \tag{25}$$

Here ΔS_i represents the small "surface area" being hit by the i'th ray. However, we can drop the ΔS_i term since we normalize the gradient for the line-tracing. We are currently only interested in the direction, not the magnitude of the gradient. Now looking at $\nabla \frac{1}{\|\vec{p}_i - \vec{q}\|^m}$ we rewrite it as

$$\nabla \|\vec{p}_i - \vec{q}\|^{-m} = \nabla \left(\vec{p}_i - \vec{q}\right)^T \left(\vec{p}_i - \vec{q}\right)^{-\frac{m}{2}} \tag{26}$$

Using the chain-rule we have

$$\nabla\left(\vec{p}_i-\vec{q}\right)^T\left(\vec{p}_i-\vec{q}\right)^{-\frac{m}{2}} = -\frac{m}{2}\left(\vec{p}_i-\vec{q}\right)^T\left(\vec{p}_i-\vec{q}\right)^{-\frac{m}{2}-1}\nabla\left(\vec{p}_i-\vec{q}\right)^T\left(\vec{p}_i-\vec{q}\right) \tag{27}$$

$$= -\frac{m}{2}\left(\vec{p}_i-\vec{q}\right)^T\left(\vec{p}_i-\vec{q}\right)^{-\frac{m}{2}-1}\left(2q^T\nabla q - 2p_i^T\nabla q\right) \tag{28}$$

$$= -m\left(\vec{p}_i-\vec{q}\right)^T\left(\vec{p}_i-\vec{q}\right)^{-\frac{m}{2}-1}\left(q^T - p_i^T\right) \tag{29}$$

$$= -m\frac{1}{\left\|\vec{p}_i-\vec{q}\right\|^{m+2}}\left(\vec{q}^T - \vec{p}_i^T\right) \tag{30}$$

$$= m\frac{\left(\vec{p}_i-\vec{q}\right)}{\left\|\vec{p}_i-\vec{q}\right\|^{m+2}} \tag{31}$$

Define the unit direction to the surface points as $\vec{d}_i = \left(\vec{p}_i-\vec{q}\right)/\left\|\vec{p}_i-\vec{q}\right\|$ then

$$\nabla\left(\vec{p}_i-\vec{q}\right)^T\left(\vec{p}_i-\vec{q}\right)^{-\frac{m}{2}} = m\frac{\vec{d}_i}{\left\|\vec{p}_i-\vec{q}\right\|^{m+1}} \tag{32}$$

So we end up with having

$$\nabla E(\vec{q}) \approx \sum_{i\in N,\vec{p}_i\in S} m\frac{\vec{d}_i}{\left\|\vec{p}_i-\vec{q}\right\|^{m+1}} \tag{33}$$

In a practical implementation the set N is chosen as a set of evenly scattered direction vectors on the unit sphere. The $\vec{p}_i$'s are found by performing ray-triangle intersections of the i'th ray emitted at the point $\vec{q}$ and in the direction $\vec{d}_i$. Only the first penetration point is used, since according to [21] the best result is obtained by thinking of the polygonal surface as an insulator. For the i'th vertex point, $\vec{q}_i$, we update its position in the j'th step according to

$$\vec{q}_i^{\,j+1} = \vec{q}_i^{\,j} - \Delta t\frac{\nabla E(\vec{q}_i^{\,j})}{\left\|\nabla E(\vec{q}_i^{\,j})\right\|} \tag{34}$$

where Δt is a sufficiently small fixed step-size. We continue stepping like this until the stopping criteria

$$\nabla E(\vec{q}_i^{\,j}) \cdot \nabla E(\vec{q}_i^{\,j-1}) < 0, \tag{35}$$

is fulfilled. This criteria is a crude indicator that we have passed a critical point in the generalized energy potential. The pseudo code in Figure 16 summarizes the line-stepping approach we have used.

In Figure 17 we have shown results of line-stepping generating the $\vec{d}_i$'s from a unit sphere by subdividing an icosahedron three times (642 samples). This creates a much more nice and even distribution of the ray samples. As seen from the figure we now have nice smooth straight traces. Although there are still some problems left. Due to sampling artifacts and using a fixed step-size some traces can escape the surface to the outside and get trapped outside. This artifact is seen in the case of the teapot and the dragon object.

Line-stepping the generalized potential energy field looks extremely well. It appears that all the deficiencies that appeared from line-stepping the signed distance field have disappeared. One benefit we immediately gain from using the generalized energy potential is that we no longer need to work on a regular sampling. Thus we are freed from the interpolation issues and from having to consider the trade-off between sampling resolution and memory-usage etc.. Furthermore signed distance field computations often rely on the polygonal surface mesh to be a watertight mesh. No such criteria is required for the energy potential approach. In fact the energy potential approach could be applied to general polygon soups.

However, it is also apparent that we need to look into methods that do not have the sampling and fixed step-size artifacts. This calls for closed-form methods for computing the energy potential and the gradient, and for adaptive step-size algorithms. We leave this for future research.

5.2 The Algorithm

We propose the following algorithm for finding good extrusion points for shell generation.

- Calculate line-traces for all surface vertices using closed-form solutions for the generalized energy potential and an adaptive line-step algorithm.
- For all line-traces compute their intersection points, these are the junctions!
- For each vertex assign the first junction point encountered along the corresponding line trace as the extrusion point. If no such junction point exist simply use the ending position of the line-trace as the extrusion point.

It should be noted that this is not the final saying, we obviously get a center-line problem, which can be understood in terms of surface mesh sampling artifacts. This is similar to the problems encountered in Section 2. Looking at the bottom of Figure 17(d) it is seen that the line traces coming from the letters will intersect the extruded triangles coming from the bottom faces of the box. Thus, internal intersections may happen. As a solution to this problem we propose the idea of applying a post-processing step.

In the post-processing, after having performed a glue-operation as described in Section 2.2, one iterate over the extrusion lines one-by-one. For each extrusion line it is tested if it intersects any of the extruded surface faces except the extruded surface faces that the end-point of the current extrusion line is part of. This will remove any internal intersections. It may be that after this final step internal void-regions appear. These can be reduced by performing the pre-processing step outlined in Section 2.1.

6 Conclusion

Our work on using different shape representations and methods for computing extrusion points have resulted in many discoveries. We have learned many important properties of the shape representations and of the numerical methods applied to these representations. We have also learned about and evolved our understanding of what "defines" the inner structure, the extrusion points, of a shell. Here follows a detailed summary of the main points of discoveries:

- Line-stepping for finding the "center-position" is the best solution over any root search method.
- Extrusion based on signed distance fields is difficult, because:
  - Thresholding on normal testing is a necessary evil due to fix-precision floating point arithmetic. This leads to possible overlapping interior regions of the resulting shell mesh. In practice this calls for parameter tuning. The implication is that it is not always possible to find an acceptable bound on the normal test which restrict internal overlaps and allow for the most aggressive extrusion lengths.
  - Shell quality depends on the resolution of the signed distance field.
  - All extrusion lines can be computed in parallel.
  - Obtaining a signed distance field is not trivial. For instance it requires closed water tight surfaces.

  In conclusion we find this approach problematic.
- An average outward flux (AOF) field is a very robust way to locate voxels containing the medial surface.
  - Surface normals are not necessarily aligned with the medial surface structure. That is the seams having end-junctions at the surface vertices are not aligned with the surface normal of these vertices. Thus, one can not rely on surface normals to pick a line-step direction in agreement with the medial surface structure.
  - It is nearly impossible to perform line-stepping along the medial surface. Essentially the medial surface consist of a piece-wise continuous surface constructed from infinitely thin curved geometric surfaces. Any lattice based method (distance field or AOF field) is a regular sampling. Discretization and interpolation errors mean that we at best only have a smeared indication of the medial surface. It is therefore quite difficult to detect when one slips off a smoothed version of the medial surface or if it is just because the surface bends. In conclusion we need methods to correct the line-step direction.
  - Back-projection is a great way to correct line-step directions. The idea of correcting a fake-position update by setting up a tangent

search direction and search for a extrema position followed by a back-projection onto a step-size sphere results in a more “implicit” correction method.

  – Ignoring medial surface vertices is down-right wrong. The consequence is zero-length extrusion.

  – Line-stepping medial surface vertices independent of the extrusion of non-medial surface vertices will result in large internal overlaps in the shell mesh. This implies that medial-surface vertices can not be processed independently of other surface vertices.

- Local extrema detection in signed distance and AOF fields are difficult to detect because extrema are unlikely to occur at nodal positions in a regular sampling.

- Junctions are not defined by local extrema in the signed distance field or the AOF field. Besides for 3D objects medial surface vertices could just as easily be extruded to lie on a seam of the medial surface. This is true for vertices having a positive first principal curvature and a zero second principal curvature. That is vertices lying on a sharp edge of the surface mesh.

- Force-following in a signed distance field is not numerical robust, since there is too many degrees of freedom when hitting the medial surface sheets. Thus, unexpected merges of the trace-lines may occur.

- Force-following in curve-skeletons appear to be numerical well-defined compared to using signed distance fields. Besides the energy function is defined independently of the resolution of a grid and can be applied to general polygon soups.

- Connecting medial-surface vertices to nearest junction is not the correct approach. Instead one must examine what junctions are paired with the medial surface vertices by making a record of the traces.

Our next step in this project is to investigate better methods for computing the curve-skeletons before evaluating the proposed algorithm in Section 5.2.

References

[1] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. *ACM Trans. Graph.*, 24(3):617–625, 2005.

[2] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *SMA ’01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, New York, NY, USA, 2001. ACM Press.

[3] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. Openmesh–a generic and efficient polygon mesh data structure. In *Proc. Open SG Symposium*, 2002.

[4] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. Primo: Coupled prisms for intuitive surface modeling. *Symposium on Geometry Processing*, pages 11–20, 2006.

[5] Sylvain Bouix and Kaleem Siddiqi. Divergence-based medial surfaces. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 603–618, London, UK, 2000. Springer-Verlag.

[6] Jen-Hui Chuang, Chi-Hao Tsai, and Min-Chi Ko. Skeletonization of three-dimensional object using generalized potential field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1241–1251, 2000.

[7] Nicu D. Cornea, M. Fatih Demirci, Deborah Silver, Ali Shokoufandeh, Sven J. Dickinson, and Paul B. Kantor. 3d object retrieval using many-to-many matching of curve skeletons. *Shape Modeling and Applications, 2005 International Conference*, pages 366–371, june 2005.

[8] Nicu D. Cornea, Deborah Silver, and Patrick Min. Curve-skeleton applications. In *IEEE Visualization*, page 13, 2005.

[9] Nicu D. Cornea, Deborah Silver, Xiaosong Yuan, and Raman Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer*, 21(11):945–955, 2005.

[10] André Diatta and Peter Giblin. Pre-symmetry sets of 3D shapes. In *Proceedings of the 1st International Workshop on Deep Structure, Singularities, and Computer Vision*, volume 3753 of *Lecture Notes in Computer Science*, pages 36–49. Springer Verlag, 2005.

[11] Pavel Dimitrov. Flux invariants for shape. Master's thesis, Department of Computer Science, McGill University, Montreal, Quebec, Canada, July 2003.

[12] Pavel Dimitrov, James N. Damon, and Kaleem Siddiqi. Flux invariants for shape. In *CVPR (1)*, pages 835–841, 2003.

[13] Kenny Erleben and Henrik Dohlmann. The thin shell tetrahedral mesh. In Søren Ingvor Olsen, editor, *Proceedings of DSAGM*, pages 94–102, August 2004.

[14] Kenny Erleben and Henrik Dohlmann. Scan conversion of signed distance fields. In Søren Ingvor Olsen, editor, *Proceedings of DSAGM*, pages 81–91, August 2006.

[15] Kenny Erleben, Henrik Dohlmann, and Jon Sporring. The adaptive thin shell tetrahedral mesh. *Journal of WSCG*, pages 17–24, 2005.

[16] Kenny Erleben and Jon Sporring. Line-stepping for shell meshes. In Bjarne Ersbøll and Kim Steenstrup Pedersen, editors, *Scandinavian Conference on Image Analysis (SCIA '07)*, volume 4522 of *Lecture Notes in Computer Science*. Springer Verlag, June 2007. (to appear).

[17] Mark Foskey, Ming C. Lin, and Dinesh Manocha. Efficient computation of a simplified medial axis. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 96–107, New York, NY, USA, 2003. ACM Press.

[18] A. Globus, C. Levit, and T. Lasinski. A tool for visualizing the topology of three-dimensional vector fields. In *VIS '91: Proceedings of the 2nd conference on Visualization '91*, pages 33–40, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[19] Benjamin B. Kimia, Allan R. Tannenbaum, and Steven W. Zucker. Shapes, shocks, and deformations I: The components of two-dimensional shape and reaction-diffusion space. *International Journal of Computer Vision*, 15:189–224, 1995.

[20] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885, 1992.

[21] Pin-Chou Liu, Fu-Che Wu, Wan-Chun Ma, Rung-Huei Liang, and Ming Ouhyoung. Automatic animation skeleton construction using repulsive force field. *pg*, 00:409, 2003.

[22] Martti Mantyla. *Introduction to Solid Modeling.* W. H. Freeman & Co., New York, NY, USA, 1988.

[23] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. Adaptive physics based tetrahedral mesh generation using level sets. (in review), 2004.

[24] M. Müller and M. Teschner. Volumetric meshes for real-time medical simulations. In *Proc. BVM (Bildverarbeitung für die Medizin)*, pages 279–283, Erlangen Germany, March 2003.

[25] Stanley Osher and Ronald Fedkiw. *Level-set methods and Dynamic Implicit Surfaces*, volume 153 of *Applied Mathematical Sciences.* Springer, 2003.

[26] Per-Olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM Review*, 46(2):329–345, June 2004.

[27] Stephen M. Pizer, P. Thomas Fletcher, Sarang Joshi, Andrew Thall, James Z. Chen, Yonatan Fridman, Daniel S. Fritsch, A. Graham Gash, John M. Glotzer, Michael R. Jiroutek, Conglin Lu, Keith E. Muller, Gregg Tracton, Paul Yushkevich, and Edward L. Chaney. Deformable m-reps for 3d medical image segmentation. *International Journal of Computer Vision*, 55(2/3):85–106, 2003.

[28] Serban D. Porumbescu, Brian Budge, Louis Feng, and Kenneth I. Joy. Shell maps. *ACM Trans. Graph.*, 24(3):626–633, 2005.

[29] James A: Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999.

[30] James A. Sethian. *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.

[31] Avneesh Sud, Mark Foskey, and Dinesh Manocha. Homotopy-preserving medial axis simplification. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 39–50, New York, NY, USA, 2005. ACM Press.

[32] Avneesh Sud, Miguel A. Otaduy, and Dinesh Manocha. Difi: Fast 3d distance field computation using graphics hardware. In M.-P. Cani and M. Slater, editors, *Proc. of Eurographics*, volume 23, Grenoble, France, 2004. http://gamma.cs.unc.edu/DiFi/.

[33] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 130, Washington, DC, USA, 2003. IEEE Computer Society.

[34] M. Teschner, M. Müller B. Heidelberger, D. Pomeranets, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization*, pages 47–54, Munich, Germany, November 2003.

[35] Lawson Wade and Richard E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer*, 18(2):97–110, April 2002.

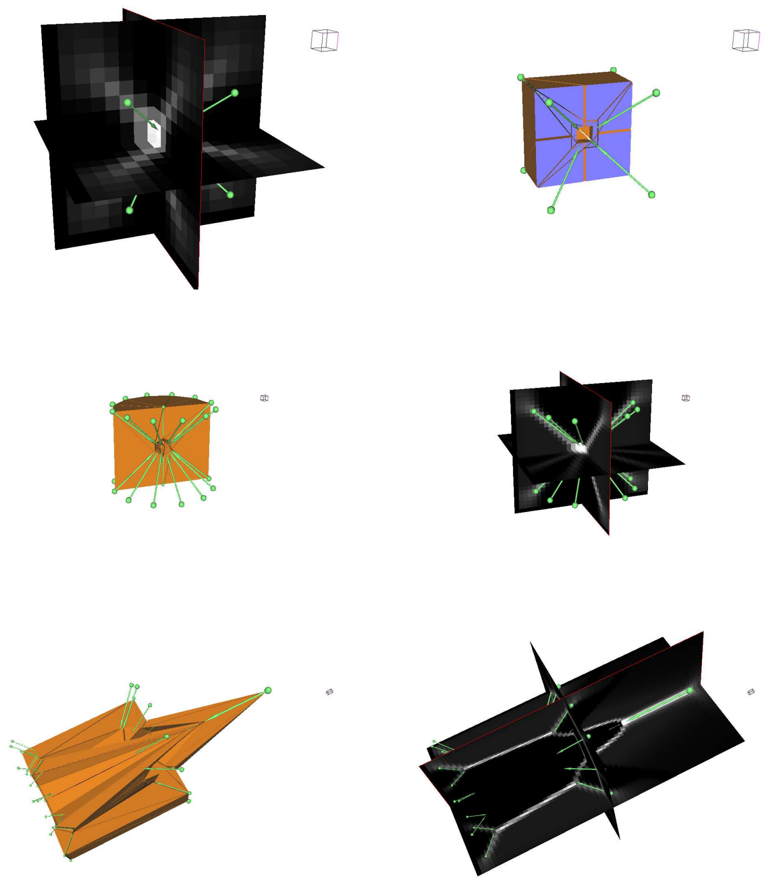
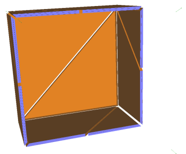
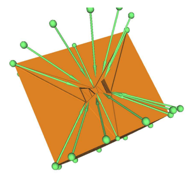
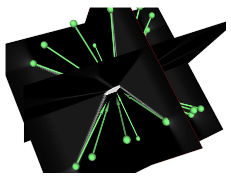
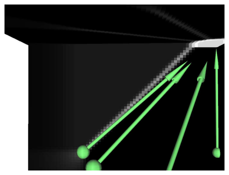


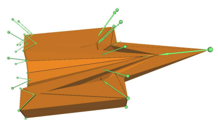
Figure 6: AOF line-stepping on a few simple surface meshes using a coarse grid from 16^3 to 64^3. Green arrows indicate extrusion lines.

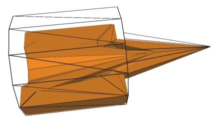

(a) Box extrusion stops too early due to misalignment.

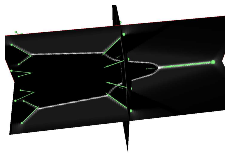

(b) Cylinder almost perfect.


(c) Cylinder extrusion overlay ed on AOF field.


(d) Close-up of overlaid cylinder extrusion lines.


(e) Extrusion lines overlaid on cut-view of shell mesh.


(f) Surface mesh overlaid on cut-view of shell mesh.


(g) Pointy extrusion lines overlaid on AOF field.

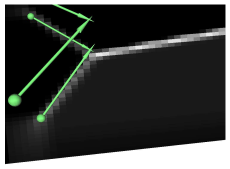

(h) Close-up of Pointy Extrusion lines. Observe misalignment.

Figure 7: AOF line-stepping on a few simple surface meshes using a fine resolution grid of 128^3. Green arrows indicate extrusion lines. Note that the green arrows are not running along the white lines representing the medial surface. Thus, surface normals are “misaligned” with medial surface structures.

```
$\theta_{\max} = \text{flux}(\vec{q}_a)$
$i_{\max} = 0$
$\Delta s = 2\Delta\varepsilon/N - 1$
for $i = 1$ to $N$ do
  $s = i\Delta s$
  $\theta = \text{flux}(\vec{q}_a - \vec{t}s)$
  if $\theta > \theta_{\max}$ then
    $i_{\max} = i$
    $\theta_{\max} = \theta$
  end if
next $i$
$\vec{q}_{\text{fake}} = \vec{q}_a - \vec{t}(i_{\max}\Delta s)$
```

Figure 8: Pseudo code for a linear search of a line-step direction correction.

```
do
  $\vec{q}_c = (\vec{q}_a + \vec{q}_b)/2$
  if $\|\vec{q}_a - \vec{q}_b\| < \varepsilon_{\text{too small}}$ then
    $\vec{q}_{\text{fake}} = \vec{q}_c$
    break;
  end if
  $\rho = \vec{t} \cdot \text{unit}(\nabla\phi(\vec{q}_c)))$
  if $|\rho| < \delta$ then
    $\vec{q}_{\text{fake}} = \vec{q}_c$
    break;
  end if
  if $|\rho| > 0$ then
    $\vec{q}_a = \vec{q}_c$
  end if
  if $|\rho| < 0$ then
    $\vec{q}_b = \vec{q}_c$
  end if
while forever
```

Figure 9: Pseudo code for a binary search for a line-step direction correction.

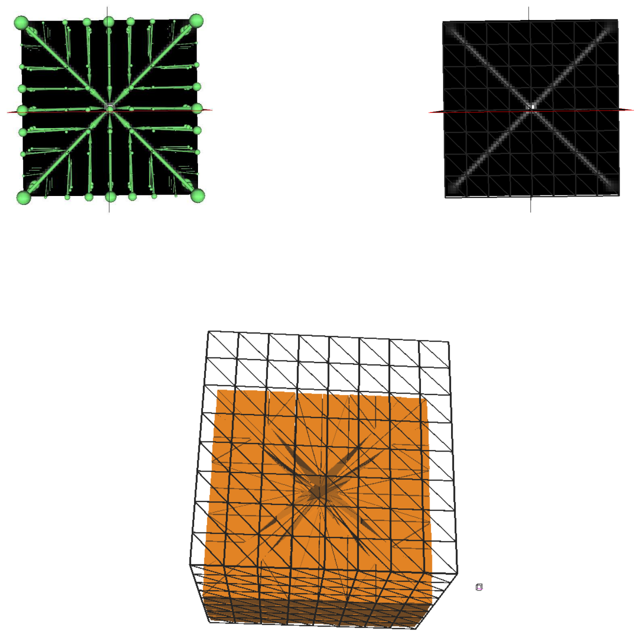


Figure 10: AOF line-stepping with line correction and finer tessellation. This shows that line-stepping along the symmetry set completely degenerate the resulting volume mesh if we refine the surface mesh.

```
Algorithm line-step-AOF
  for each extrusion line $l$ do
    if $flux(p(l)) > \delta$ then
      continue
    end if
    do
      $q(l) = q(l) - n(l)\Delta\varepsilon$
      if $flux(q(l)) > \delta$ and $n(l) \cdot \nabla flux(q_{fake}) > 0$ then
        // binary search for maximum AOF
        $q_a = q(l) + n(l)\Delta\varepsilon$
        $q_b = q(l)$
        do
          $\vec{q}_c = (\vec{q}_a + \vec{q}_b)/2$
          if $\|\vec{q}_a - \vec{q}_b\| < \varepsilon_{\text{too small}}$ then
            $\vec{q}(l) = \vec{q}_c$
            break
          end if
          $\rho = \vec{n}(l) \cdot \text{unit}(\nabla\text{flux}(\vec{q}_c)))$
          if $|\rho| < \delta$ then
            $\vec{q}(l) = \vec{q}_c$
            break
          end if
          if $|\rho| < 0$ then
            $\vec{q}_a = \vec{q}_c$
          end if
          if $|\rho| > 0$ then
            $\vec{q}_b = \vec{q}_c$
          end if
        while forever
      end if

      break

    while forever
    $\varepsilon = \|\vec{q}(l) - \vec{p}(l)\|$
  next $l$
End algorithm
```

Figure 11: Pseudo code for line-stepping in AOF field ignoring initial medial surface vertices.

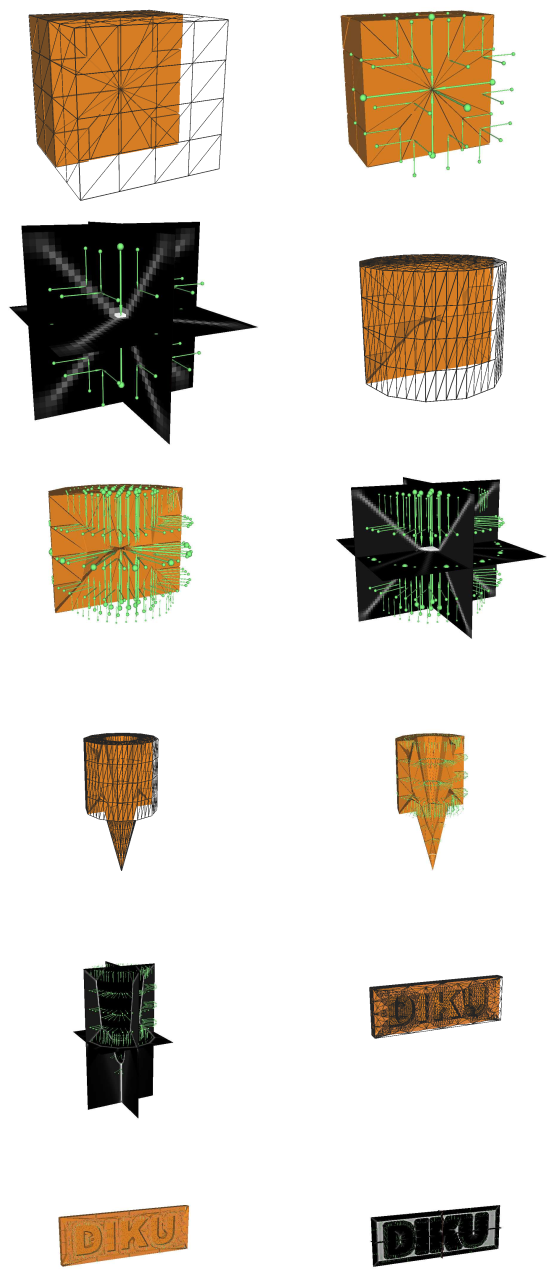
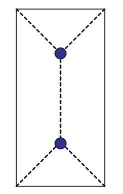
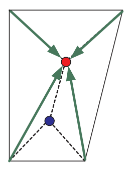


Figure 12: AOF line-stepping ignoring medial surface vertices. Notice the strange artifacts at the top/bottom of the cylinder.


(a) No global strict minima exist


(b) One global strict minima exist, but bottom-most junction is not even a minima

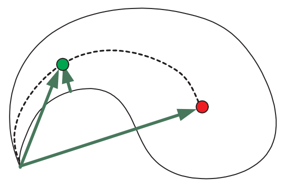

(c) Connecting medial surface vertex directly to junction may cause surface penetration. Correct solution is to use closest non-medial surface vertex extrusion point.

Figure 13: Various cases illustrating the usage of junctions and non-medial surface vertex extrusion points.

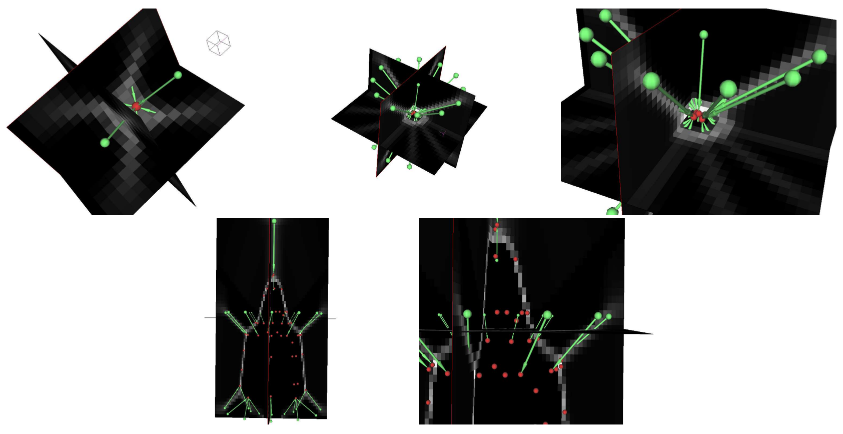
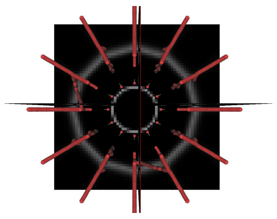
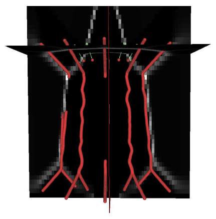
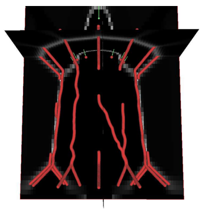
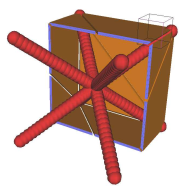
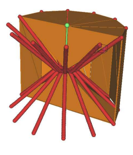


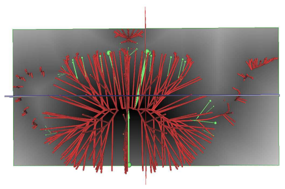
Figure 14: Junction detection using local minima detection in a signed distance field smoothed using curvature flow to reduce noise artifacts. Observe that too many points are classified as local minima.

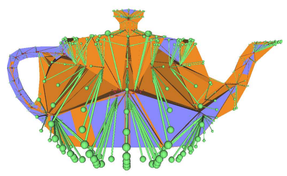
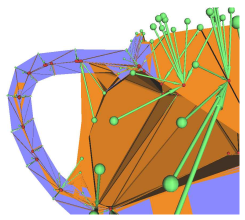
(a) Pointy object.

(b) Simple convex objects.



(c) Teapot line-traces.

(d) Resulting shell using closest junction as extrusion point.

Figure 15: Junction detection using a line-stepping method in a signed distance field smoothed using curvature flow. Notice on the pointy object that if sheets are hit the paths move in an unexpected way. Observe from the teapot object that the closest junction is not always the correct extrusion point.

```
for $i = 0$ to $N_{\text{max seeds}} - 1$
  $\vec{E}_{\text{old}} = \vec{0}$
  $\vec{E}_{\text{cur}} = \vec{0}$
  for $j = 0$ to $N_{\text{max steps}} - 1$
    $\vec{E}_{\text{cur}} = \nabla E(\vec{q}_i^j)$
    if $\vec{E}_{\text{old}} \cdot \vec{E}_{\text{cur}} < 0$ then
      break
    $q_i^{j+1} = q_i^j - (\vec{E}_{\text{cur}} / \left\| \vec{E}_{\text{cur}} \right\|)\Delta t$
    $\vec{E}_{\text{old}} = \vec{E}_{\text{cur}}$
  next $j$
next $i$
```

Figure 16: Pseudo code for line-stepping in generalized energy potential.

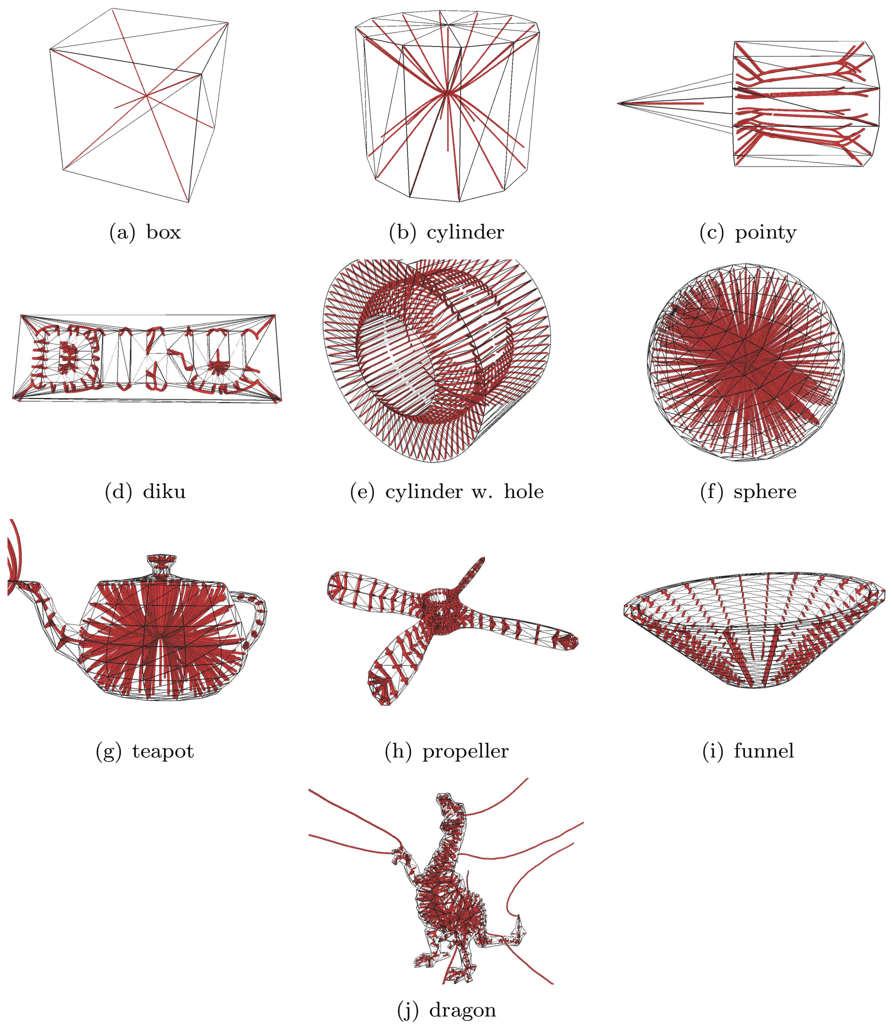


Figure 17: Examples when using subdivided icosahedron as sphere mesh for the ray samples when tracing the curve-skeletons with $m = 1$. Notice the added robustness of the pointy object compared to using the signed distance field approach. Observe the path tracing problem of the teapot and the dragon objects.