-

Shrinkwrap: An efficient adaptive algorithm for triangulating an iso-surface

C.W.A.M. van Overveld
Department of Mathematics and Computing Science
Eindhoven University of Technology
PO Box 513, 5600 MB Eindhoven, The Netherlands
Email: wsinkvo@win.tue.nl
and
B.Wyvill
Department of Computer Science
University of Calgary
Address: 2500 University Drive N.W.,
Calgary, Alberta,Canada, T2N 1N4
Email: blob@cpsc.ucalgary.ca

Abstract

An algorithm is presented which generates a triangular mesh to approximate an iso-surface. It starts with a triangulation of a sphere and next applies a series of deformations to this triangulation to transform it into the required surface. These deformations leave the topology invariant, so the final iso-surface should be homeomorphic with a sphere. The algorithm is adaptive in the sense that the lengths of the sides of the triangles in the mesh vary with the local curvature of the underlying surface. A quantitative analysis of the accuracy of the algorithm is given along with an empirical comparison with earlier algorithms.

keywords:

iso-surfaces, implicit surfaces, polygonization

1 Introduction

For a scalar function V defined on $\mathbb{R}^3$, an iso-surface[1] is the collection of points r where $V(r)$ takes a given value, $V(r) = V_0$. Iso-surfaces play a significant role in computer visualisation. Various researchers have used skeletal iso-surfaces for modelling: ([?], [?], [?], [?]) and several of the surfaces that are of interest in pure mathematics, physics, or chemistry are iso-surfaces ([?]).

In order to visualise iso-surfaces, ray tracing is an often used technique ([?]) which produces high-quality images. In the case of all but trivial cases, however, it is an arduous task to

[1] Also called *implicit surface*

compute points on the iso-surface and the computational burden of ray tracing often imposes restraints on its application.

An alternative to ray tracing is the conversion of the iso-surface into polygon meshes which are rendered afterwards; an additional advantage of this approach is the availability of a full 3-D approximation of the iso-surface which allows for fast viewing from arbitrary directions, as well as the application of post-processing techniques such as mesh reduction ([**?**]), relaxation ([**?**]) or free-form deformation ([**?**]). Also, the resulting polygon mesh is a closed manifold, so it may be used in B-rep-based CSG operations ([**?**]).

Most currently existing techniques for the polygonisation of iso-surfaces are based on data structures that allow spatial indexing: either a voxel-based structure ([**?**]) or the hash-table structure of ([**?**]) may be used.

Some inherent disadvantages of these data structures exist:

First, the data structure comprises a partitioning of the space rather a tesselation of the surfaces to be polygonalised. Especially in the case of animation (e.g. in the computer animation "The great train rubbery", [**?**]), this is likely to cause geometric artifacts that are fixed with respect to space, thus moving in an incoherent way over every moving surface.

Second, there is an apparent mismatch between the number of triangles that is generated by these algorithms and the complexity of the surface that is approximated: even relatively smooth and flat segments of an iso-surface usually result in large amounts of facets. Bloomental ([**?**]) uses an adaptive version of the spatial indexing data structures, an octree in order to reduce the amount of polygons produced in tesselating an iso-surface. This indeed reduces the amount of polygons generated, but full advantage of large cells can only be taken if the flat regions of the surface happen to fall entirely within the appropriate octants. The same observation applies to the approach to adaptive tesselation using the geometry of a tetrahedron rather than a cubical structure.

Third, in earlier approaches using space partitioning with packed cubes, a value is sampled at each cube vertex and a check is made to see if the vertex is inside or outside the surface ([**?**]). A table determines the polygons which are used to replace each cube. Unfortunately this method has various ambiguous cases where there are alternative configurations for certain in-out combinations. Ning and Bloomenthal ([**?**]) explore this problem and point out tht each cube may be subdivided in 5 or 6 tetrahedra which may be replaced unambiguously by triangles. However, such an algorithm produces many unnecessary triangles. The algorithm presented in this paper does not suffer from such ambiguities.

In this paper, a different approach to the tesselation of a class of iso-surfaces is taken. This class is defined in **??**. The proposed approach is adaptive to the local behavior of the surface rather than being imposed by an octtree with a priori defined cutting planes; this causes the tesselation to move along with the surface in the case of (smooth) animations. Finally, as a side effect of the algorithm we propose, a coordinate system can easily be introduced on the surface which can be useful e.g. for applying surface texturing.

In section **??** the intuition behind the algorithm is sketched, and the algorithm proper is presented. It makes use of the notion of "acceptable edge", and this will be discussed in a more quantitative manner in relation to the accuracy of the algorithm in section **??**. The

results are summarised in section **??**.

2 An algorithm to arrive at an adaptive triangulation for an iso-surface

First we define the class of iso-surfaces for which our algorithm should produce triangulations. Next the algorithm, together with an intuitive motivation are given, and some aspects are discussed in further detail .

2.1 The definition of the iso-surface

An iso-surface is the collection of points r in 3-D space such that a given function $V(r) = V_0$. We consider the class of functions V where $V(r) = \sum_i \frac{\rho_i}{|r-R_i|}$. The summation over i indicates that the function is composed of a number of components with relative strength (weight) ρ_i. Components can be thought to be generated by different types of geometric primitives: points, line segments or convex polygons.

If a component is generated by a point, then R_i is the location of this point, and the set $\frac{\rho_i}{|r-R_i|} = V_0$ designates a sphere with radius ρ_i/V_0 round R_i.

The element can also be a line segment, say ab. In that case R_i depends on r; R_i is the projection of r onto ab and the set $\frac{\rho_i}{|r-R_i|} = V_0$ designates a cylinder with hemi-spherical caps with radius ρ_i/V_0 and ab as the axis.

Similarly, elements can be triangles or other convex polygons in which case a projection of r onto that polygon has to take place in order to obtain R_i. In these cases the designated surface is an offset surface of the original polygon with cylindrically rounded edges. Another way to envisage the designated surface is as the Minkowsky sum of a sphere with radius ρ_i/V_0 and the geometric object (point, line segment or convex polygon).

The collection of points, line segments and convex polygons that define $V(r)$ is called the skeleton; each geometric object in the skeleton is called a skeletal element ([**?**]).

The addition of the several components results in an iso-surface which is a smoothly blended union of the several iso-surfaces associated with the individual skeleton elements.

2.2 Intuitive introduction of the shrinkwrap algorithm

The basic idea that underlies the algorithm is that the iso-surface is to be sampled with sufficient density in order to capture all shape detail; further, that these samples are connected by edges to form a triangular mesh. In voxel-based methods, such edges result from intersecting the surface with voxel boundaries, and hence their directions all lay in one of three orthogonal planes (the XY, YZ or ZX planes of some world coordinate system). These directions have no apparent relations with the iso-surface, and therefore many unnecessarily short edges, and consequently, small triangles, result. In the algorithm proposed here, the

edges should adjust themselves to the shape of the surface. This means that e.g. in a cylindrical part of the surface, there should be relatively long edges directed more or less along the axis of the cylinder and relatively short edges in directions perpendicular to the axis. This allows for an adaptive triangulation. In order to let edges gradually adjust themselves to the shape of the surface, we develop an iterative approach where the surface develops in several steps from a sphere to the final shape. Indeed, the original shape (a sphere) has uniform curvature, and hence adaptivity plays no role there; the triangulation of a sphere in order to achieve an initial estimate for the mesh topology is simple. There are two natural ways to have an iso-surface develop itself from a sphere: One method operates by first collapsing the entire skeleton into a point and gradually expanding back ("inflating") to the desired skeleton geometry. The second method ("shrinking"), which allows a slightly simpler mathematical analysis, and which will be chosen therefore for our algorithm, is based on the following observation.

Consider color plate 1. It depicts a 2-dimensional cross section through a distribution of some point skeleton elements; colors indicate the function value $V(r)$ in a point. The iso-contour we are interested in is at the boundary between the yellow and the magenta regions. We observe that iso-surfaces

$$\{r \in \mathbf{R}^3 | \sum_i \frac{\rho_i}{|r - R_i|} = V_0\}$$

with $V_0 < 1$ have a shape which is less involved, whereas iso-surfaces with $V_0 > 1$ are more involved. An extreme case of the first example is $V_0 = 0$, which produces a sphere with an infinitely large radius.

So an algorithm could start by setting V_0 to a value close to 0, and providing a triangulation for the resulting sphere. This triangulation should consist of approximately equilateral triangles, since the curvature is the same anywhere.

Next the value of V_0 should be increased, in a number of steps, and with each step the surface shrinks a bit towards its final shape and size. With every shrink step, the vertices of the triangulation should move towards the new surface. This is discussed in section **??**. Also, in order to meet with accuracy requirements, to be discussed in **??**, it might be necessary to split edges and triangles. But we note that edges and triangles are only split when necessary, so there should be fewer unnecessarily small triangles at the end of the porcess than with voxel-based methods.

Of course, this gradually shrinking of the triangulation will only work as long as the topological structure of the surface stays equivalent (*homeomorphic*) to the sphere that we start with. The issue of topological changes during the shrink process has been studied in ([**?**]) and will be published elsewhere ([**?**]). However to make this paper self-contained, we give a coarse outline of the proposed technique to deal with topological changes (ruptures and holes) in Appendix B (topological changes).

2.3 getting the vertices onto the surface

Using the stepwise approach, and assuming the difference in iso-values V_0 from one step to the next sufficiently small[2], we will use a Newton-Raphson method to displace vertices. So we make use of a first order[3] Taylor expansion to compute a first estimate for the new location of a vertex when increasing V_0 with an amount ΔV to $V_0 + \Delta V$, and, when necessary, iterate. Assume r is on the $V = V_0$ surface: $V(r) = V_0$. Next we look for a new location, $r + \delta$, such that

$$V(r + \delta) = V_0 + \Delta V.$$

Taylor expansion round $\delta = 0$ gives:

$$V(r + \delta) = V(r) + (\delta \cdot \nabla V(r)) + O(|\delta|^2) = V_0 + \Delta V,$$

or

$$\Delta V \approx (\delta \cdot \nabla V(r)).$$

Of course, this does not tell us in which direction the step δ should be taken. A reasonable choice seems to be to set

$$\delta = \lambda \nabla V(r)$$

which gives

$$\lambda = \frac{\Delta V}{(\nabla V(r) \cdot \nabla V(r))}.$$

So the new location is

$$r + \frac{\Delta V \nabla V(r)}{(\nabla V(r) \cdot \nabla V(r))} \tag{1}$$

2.4 The shrinkwrap algorithm

We are now able to write down the total shrinkwrap algorithm. Vertices are defined as tuples $(r, E, V, d) \in \mathbf{R}^3 \times \mathbb{R}^3 \times \mathbf{R} \times \mathbf{R}^3$; the r-attribute is the location; E is the gradient $\nabla V(r)$; V is the value of the function $V(r)$, and d is the displacement vector that should apply to this vertex in order to get it to the surface with next higher iso-value.

Edges contain two references e1 and e2 to the vertices in the extremes, and two references to the two adjacent triangles. Furthermore, an edge has a boolean n to indicate if it is

[2] and assuming that no topological changes occur between this step and the next

[3] See section ?? for a proposal for a more sophisticated approach.

non-acceptable (the acceptability of an edge is discussed in section **??**; for now it suffices to observe that edges should be in some way close to the underlying surface in order to make quantitative statements about the accuracy of the triangulation; this can be obtained by splitting edges that are non-acceptable).

The difference ΔV is an entire fraction of V_0, say $\Delta V = V_0/N_{steps}$, and for V_0 the value $V_0 = 1$ will be used. When using this convention, the interpretation of the value ρ_i is simply "the radius of the offset surface if component i was the only one skeleton element".

The issue of the number of steps N_{steps} in relation to the robustness of the algorithm is discussed in section **??**.

Initially, the set of vertices consists of the vertices in the initial object (a triangulated sphere with more or less equilateral triangles); this object is assumed to be sufficiently large to be outside the entire iso-surface even for iso-value $V_0 = 1/N_{steps}$. All vertices v have $v.V = 0$; $v.E$ is pointing radially outwards, and $v.d$ is proportional to $v.E/(v.E \cdot v.E)$ (as derived in **??**).

The global structure of the algorithm (in pseudo-Pascal) reads as follows:

```
begin
  V0:=0;
  while V0 < 1.0 do
  begin
    V0:=V0+DeltaV;
    S1;
    { all vertices are on the surface;
      for every vertex v we
      have v.V=V(v.r)
           v.E=grad V(v.r)
           v.d=(V0-v.V)*v.E/(v.E,v.E)}
    S2;
    {...and all edges are acceptable}
  end;
  { all vertices are on the surface;
    all edges are acceptable,
    and V0=1.0 }
end;
```

The statement `S1` reads:

```
for all vertices v do
  repeat
    v.r:=v.r+v.d;
    v.V:=V(v.r);
    v.E:=grad V(v.r);
    v.d:=(V0-v.V)*v.E/(v.E,v.E);
  until |v.d|< Epsilon;
```

The statement S2 reads:

```
unlabel all triangles;
for all edges c do
begin
  c.n:="c is non-acceptable";
  { Use the analysis of
    section 3.2 and 3.3 to decide
    acceptability.}
end;
while edges are unacceptable do
begin
  for all edges c with c.n=true do
  begin
    create a new vertex w for edge
    c by splitting the edge;
    move w to the surface similar as
    in S1;
    label the two triangles
    adjacent to e; create two
    new edges for c's
    fragments, c1 and c2;
    c1.n:="c1 is non-acceptable";
    c2.n:="c2 is non-acceptable";
    remove edge c;
  end;
  { all initially unacceptable edges
    have been split, but new unacceptable
    edges may have been introduced }
  for all labeled triangles t do
  begin
    split triangle t
    create 2, 3 or 4 new triangles;
    unlabel the new triangles;
    create 1, 2 or 3 new edges, ci;
    for the new edge(s) ci, do ci.n:="ci
    is non-acceptable";
    remove triangle t;
  end;
  { all triangles bounded by initially
    unacceptable edges
    have been split,
    but new unacceptable edges may
    have occurred }
end;
{ end of the while loop; all
```

```
edges are accpetable}
```

In the next two sections we discuss how to split edges and how to split triangles.

2.5 How to split edges

We have not defined yet what an acceptable edge is. For now it suffices to state that an acceptable edge should be short enough so that the surface cannot bend away too much between the two extremes of the edge. Conversely, an unacceptable edge is an edge which is too long. So we see that the remedy to an unacceptable edge is to split it, and to make sure that the new midpoint is again on the iso-surface. A naive way to do so is depicted in the left column of figure 1 (figs. 1a-1d). The original edge is $A-B_1$ in fig. 1a; $M_1 = (A+B_1)/2$.

The array of dotted curves indicate the direction of the gradient of the function $V(r)$ in the neighbourhood of the iso-surface; the thick curve represents the iso-surface proper. If we move the point M_1 in accordance with the local gradient, we arrive at M_1', as indicated by the dash-dotted arrow. Note that although M_1' will be close to the surface, since we only use a linear approximation for $V = V(r)$, it will not lie on the surface in general. In order to get it closer to the surface, we have to iterate as in statement `S1` above. Now M_1' will be a new vertex. The edge $M_1' - B_1$ is very likely acceptable, but it may be much shorter than needed. On the other hand, $A - M_1'$ is probably still unacceptable. As shown in fig. 1b, we therefore have to repeat the process on edge $A - B_2$ (B_2 is the M_1' from the previous phase), which yields M_2'. As a result, we end up with a series of unnecessary short edges, as depicted in fig. 1d. The main cause for this unfortunate behaviour is that we use information about the geometry of the function $V(r)$ and its gradients in the points M_1, M_2, M_3, ..., which are possibly far from the surface. Evaluating the gradients in these points may yield misleading information on the geometry of the iso-surface, causing a slow convergence and many unnecessary short edges.

A more efficient splitting strategy therefore should make use of reliable information only, that is information in points that are already on the surface. In fig. 1e, the same configuration is shown as in fg. 1a. The dashed thick curve is a curve which passes through the extremes A and B and is perpendicular to the normal vectors n_A and n_B, respectively. Moreover, it is the smoothest curve with these properties; Appendix A contains a derivation of an analytical expression for this curve. This curve serves to approximate a curve that lies in the iso-surface $V(r) = V_0$ through A and B, i.e. the thick solid curve in the figure. Based on the dashed thick curve, we propose point M, i.e. its parametric midpoint as a next point to evaluate the function's gradient. Point M in fig. 1e is likely to be closer to the iso-surface than M_1 in fig. 1a, so the gradient computed in M is likely to be more adequate to get acceptable edges than the gradient in M_1. In this case, $M - B$ already might be acceptable (the edge $C - B$ in fig. 1f); $A - M$ (the edge $A - C$ in fig. 1f) might need one more subdivision as depicted in fig. 1f.

2.6 How to split triangles

In case one or more edges are unacceptable, they have to be split. Fig. 2 shows a splitting scheme which illustrates how a triangle can be subdivided into smaller triangles. In the top row one of the edges is subdivided; the bottom row shows the case of three subdivided edges. In the case of two subdivided edges, two possible schemes exist; in case $|M_{AC} - B| < |M_{BC} - A|$ we choose the first alternative; otherwise we choose the second one.

3 Robustness and accuracy

In order to make quantitative statements about the behaviour of the shrinkwrap algorithm, we have to address several issues:

- a. what assumptions have to be made about the iso-surface $V(r) = V_0$;
- b. based on the assumptions of (a), how can we assure that the shrinkwrap process captures all large-scale structure of the developing iso-surface, in other words how can we prevent a situation like figure 3 happening (this regards the *robustness* of the algorithm);
- c. assuming we can guarantee robustness with respect to (b), how can we enforce bounds on the difference between the triangulation and the underlying surface (this regards the *accuracy* of the algorithm);
- d. what is the minimal value for N_{steps};

These items will be discussed in the subsequent sub-sections.

3.1 Requirements of the iso-surface

The shrinkwrap algorithm constructs a discrete model of a continuous object by means of sampling. This means that the sampling density should be sufficiently high in order to capture all shape detail of the surface. Since in shrinkwrap the samples are vertices of a piecewise planar mesh, all curved regions of the surface require samples to be sufficiently close together. If the local curvature radius of the surface is known to be nowhere smaller than a given value β, then we can define a sample density (that is, the maximal radius of a sphere round any sample such that no other sample lies within that sphere) that is guaranteed sufficient to capture all surface detail.

So we assume that for every value of $V_{0;k}$ that is to be used as an iso-value in the k-th step of shrinkwrap, a value of β_k is available such that the radius of curvature of the surface $V(r) = V_{0;k}$ is nowhere less than β_k.

The value of the local curvature in a point r, $V(r) = V_{0;k}$ can be computed using standard calculus, by fitting a quadratic polygonomial $V_{quad}(r) = (r.x, r.y, r.z, 1)A(R.x, r.y, r.z, 1)^T$ with suitable coefficient matrix A which approximates $V(r)$ in the neighbourhood of r,

and deducing the curvature of this quadric using the coefficients in A. However, for an arbitrary iso-surface $V(r) = V_{0;k}$ it is in general not possible to compute the globally smallest curvature radius analytically. Exhaustively sampling the space in the vicinity of the surface to estimate this curvature would be computationally prohibitive. Instead, the values of β_k *should be provided beforehand by the user.* The requirement that the values β_k be known beforehand seems to be impractically restrictive. However, three observations apply.

- First, this requirement is not unique for shrinkwrap. The correctness of all purely point-sampling-based methods for visualising iso-surfaces, such as ray tracing, uniform voxel-space algorithms and occtree- or tetrahedron-based algorithms, relies on the assumption that the curvature radius of the surface is bounded by a given constant. More sophisticated sampling techniques, such as interval artihmetic and affine arithmetic may provide useful here, but these techniques apply in the context of shrinkwrap as well.
- For a function $V(r)$ as defined in **??** that consists of one single component, the $\beta_{k;i}$ for that component i can be straightforwardly given. Indeed, since the iso-surface is then an offset surface[4] with radius $\rho_i/V_{0;k}$ defined on the skeleton element, $\beta_{k;i} = \rho_i/V_{0;k}$. Superposition of several elements (all with positive ρ_i) usually causes the minimal curvature radius to become larger, and in these cases $\beta_k = MIN(\beta_{k;0}, \beta_{k;1}, \beta_{k;2}, \cdots)$ is appropriate. Care should be taken, however for configurations such as depicted in figure 4 where the value for β_k for the higher values $V_{0;k}$ should be taken smaller. Finally, notice that the user should only be concerned about the value β_k for the value of $V_{0;k} = 1$ for the final step. For an earlier step k', we can set $\beta_{k'} = \frac{\beta_k V_{0;k}}{V_{0;k'}}$.
- In case the curvature radius of the surface should be locally smaller than the value for β_k provided by the user, then this only deteriorates the quality of the approximation in the vicinity of this region with high curvature: the rest of the surface (with sufficient large curvature radius) is not affected.

3.2 Capturing large-scale structure

Assume that the iso-surface has nowhere a smaller curvature radius than β_k. Consider a triangle ABC of adjacent samples A, B, and C that are all on the surface. Let the gradients in these points be n_A, n_B, n_C; they are normalised such that $|n_A| = |n_B| = |n_C| = 1$. Consider the points $a = A - \beta_k n_A$, $b = B - \beta_k n_B$ and $c = C - \beta_k n_C$ (see figure 5). These points will be called the *adjoint* points of A, B, and C, respectively. Because the surface has nowhere a curvature radius less than β_k, it stays outside ("above" in figure 5) the three spheres with radius β_k centered around a, b and c. Now if in the triangle abc the circle sectors with radius β_k and centres a, b and c cover triangle abc (that is, no point of the triangle abc falls outside all three circle sectors), the iso-surface cannot penetrate the triangle abc. Indeed: in order to penetrate the triangle abc there should be a region in abc

[4] in other words: since it is the Minkowski sum of the skeletal element and a sphere with radius $\rho_i/V_{0;k}$, it follows that for convex skeletal elements, the curvature radius is nowhere smaller than the radius of this sphere.

which is outside the spheres with radius β_k and centres a, b, and c as is illustrated for a 2-D case in figure 5b. Now since the iso-surface cannot penetrate triangle abc, it can not move too far from triangle ABC, because ABC and abc cannot be too far from each other. Hence the portion of the iso-surface that is approximated by ABC is bounded from below by triangle abc. More precisely: since no point of ABC is further away than β_k from triangle abc, the entire triangle ABC can be nowhere further away than β_k from the iso-surface. In other words: there cannot be a portion of the surface that "escapes" between the samples A, B and C as in figure 3.

Now a sufficient (although in general too restrictive) condition for abc to be covered by three circles with radius β_k round its corners a, b, and c is that $|a-b| < \beta_k\sqrt{3}$ and similar for bc and ca, as follows from the geometry of an equilateral triangle (see figure 5c).

A similar construction should be made for a triangle $a'b'c'$ which is on the other side of ABC so that $a' = A + \beta_k n_A$ etc., to make sure the iso-surface cannot "escape" in the upward direction. So the portion of the iso-surface approximated by ABC is bounded between abc and $a'b'c'$.

Thus in order to guarantee that the large-scale structure of the surface is captured, an edge AB should be sufficiently[5] short that

$$|A - \beta_k n_A - B + \beta_k n_B| < \beta_k\sqrt{3}. \tag{2}$$

$$|A + \beta_k n_A - B - \beta_k n_B| < \beta_k\sqrt{3}. \tag{3}$$

This is the first condition (robustness) for acceptability for an edge. This condition should hold throughout the shrinkwrap process for every iteration k. Notice that we used no assumptions about the iso-surface except that its curvature is bounded by β_k, so these conditions are valid for all types of skeletal elements.

3.3 Error bounds

In this section we derive a condition for the length of an edge AB, such that no point in the interior of a triangle ABC is further away than a predefined distance $\epsilon\beta_k$, $\epsilon << 1$, from the iso-surface. Similar as in the previous section, we construct a plane that bounds the iso-surface from below (and one to bound the surface from above). This time the plane has to be at distance $\epsilon\beta_k$ from ABC. This means that we have to introduce points a", b" and c" with $a" = A - n_A\epsilon\beta_k / \sin\delta_A$ where δ_A is the angle between the gradient n_A and the plane through ABC. Similar for b" and c". The triangle $a"b"c"$ is also intersected by the three spheres, centered round a, b, and c with radius β_k, and we have to derive a condition such that the intersected sectors cover the entire area of $a"b"c"$ in order to enforce the iso-surface to be close to ABC. Therefore we consider the plane α through A and a, perpendicular to the plane through ABC. See figure 6. The point labeled m indicates the intersection of α with $a"b"c"$ and the sphere round a, and it can be seen that a circular sector with radius

[5] observe that this condition always can be satisfied: if we take A and B closer together, as is the case when splitting edges, n_A and n_B have to get closer together, too.

$|a" - m|$ round $a"$ in $a"b"c"$ falls entirely within the intersecting sector between $a"b"c"$ and the sphere round a with radius β_k. The distance $x = |a" - m|$ can be computed from the cosine rule in triangle $a"ma$. We have that

$$|a" - a| = \beta_k(1 - \epsilon/\sin\delta_A) \tag{4}$$

and

$$|m - a| = \beta_k, \tag{5}$$

so

$$\beta^2{}_k = x^2 + \beta^2{}_k(1 - \epsilon/\sin\delta_A)^2 + x\beta_k(1 - \epsilon/\sin\delta_A)\cos(\delta_A). \tag{6}$$

This gives for x:

$$x = \beta_k(\frac{\epsilon}{\sin\delta_A} - 1)\cos\delta_A + \beta_k\sqrt{2\epsilon\sin\delta_A - \epsilon^2 + \cos^2\delta_A}. \tag{7}$$

or

$$x = \beta_k(\frac{\epsilon}{\sin\delta_A} - 1)\cos\delta_A - \beta_k\sqrt{2\epsilon\sin\delta_A - \epsilon^2 + \cos^2\delta_A}. \tag{8}$$

By substituting for $\epsilon = 0$ we should find that $x = 0$, so the +-sign option in **??** applies.

We note that the radius of the intersecting sectors is bounded by a number which is dependent of δ_A, and this angle can of course be different from δ_B and δ_C. To facilitate our further analysis, we will assume that $\delta_M = MAX(\delta_A, \delta_B, \delta_C)$ and derive a safe condition stating that the iso-surface does not penetrate $a"b"c"$. Notice that replacing δ_A, δ_B, and δ_C by their maximum means that we should actually compute the δ_M for every triangle before we can apply the acceptability test.

Then again we have, from the geometry of the equilateral triangle, that $|a" - b"| < x\sqrt{3}$ and similar for $b"c"$ and $"c"a"$ where x follows from **??**. A similar argumentation holds for the plane $a'''b'''c'''$ where $a''' = A + n_A\epsilon\beta_k/\sin\delta_A$, etc.

So in order to guarantee that the surface is nowhere further away from the triangular mesh than $\epsilon\beta_k$, an edge AB should be sufficiently[6] short that

$$|A - n_A\epsilon\beta_k/\sin\delta_A - B + n_B\epsilon\beta_k/\sin\delta_B| < x\sqrt{3}, \tag{9}$$

$$|A + n_A\epsilon\beta_k/\sin\delta_A - B - n_B\epsilon\beta_k/\sin\delta_B| < x\sqrt{3}, \tag{10}$$

.

[6] observe that this condition also always can be satisfied for the same reasons as in section **??**

Now have arrived at the second condition for acceptability for an edge (accuracy). This condition should only hold towards the end of the shrinkwrap process when $V_{0;k}$ approaches the final value of 1. A useful strategy therefore is to decrease ϵ gradually from 1 to the desired final value during iteration. This causes the first iterations to take place with relatively few triangles.

Notice that conditions **??** and **??** are less critical than **??** and **??**: failure of **??** or **??** does not jeopardize the global structure of the triangulation. Therefore, instead of the global value for β_k, we may dare to use a local approximation of β_k, which may be different for each triangle ABC, derived from the angles between n_A, n_B and n_C and the plane through A, B, and C. In this way, again triangles in relatively flat areas of the surface are allowed to be larger which results in an even better adaptation. The experiments in the results-section were made using this optimisation.

3.4 The number of steps

One parameter of the shrinkwrap algorithm has received no attention yet: the choice for N_{steps}. On the one hand, N_{steps} should be as small as possible to give a small number of evaluations and hence an efficient algorithm. On the other hand, a large value of ΔV may impede the Newton Raphson iteration in statement `S1` from converging. Therefore a possible implementation could be to apply relatively large steps of ΔV and as soon as a convergence problem occurs, go back one iteration and reduce ΔV.

However in practice it works better to take a value of ΔV slightly smaller than the maximally allowable value for Newton Raphson convergence: even if convergence occurs, it may happen that the distribution of small and large triangles is not very well adapted to the curvature of the surface if the vertices have to move a large distance between each iteration and the next one (this is e.g. the case with the penguin in table **??** for 3 and 4 iterations). Moreover, due to edge splitting underway, it is obvious that during the first $N_{steps} - 3$ or so iterations significantly less vertices have to be updated than during the last few iterations. This means that the amount of processing is very much concentrated in the last 3 or 4 iterations, whatever the number of iterations is. So if N is a minimal value for N_{steps} such that convergence marginally occurs, then $N + 2$ or $N + 3$ is a much safer value which only costs a small amount of additional computing effort (this can be seen from table **??**). This issue is discussed further in section **??**.

4 Results, future research

The algorithm described has been implemented. In color plate 2, its qualitative behaviour is depicted: while increasing V_0 in 9 steps from 0.1 to 1.0, we see a penguin emerge from what starts of as a feature-less large spherical shape. Note how gradually the details become visible, first the most protruded ones (for this reason we equipped the penguin with an oversized bill), later on the more subtle ones. Color plate 3 shows the final object together with the triangle mesh. Here, the adaptiveness of the algorithm is clearly visible, e.g. the bill consists of mostly very slender triangles, whereas in the spherical part of the head we

find more obtuse ones. Also, in the concave regions, having a relatively high curvature, the triangle mesh has a much higher resolution than elsewhere.

Some of the quantitative features are presented in the graph in figure 7. Here, we see the increase of the number of triangles while iterating the penguin. It follows that the majority of the triangles is only created towards the final (few) iterations. As a result, the efficiency of the algorithm is not as bad as one might expect from an iterative approach: the work of the first few iterations is very small compared to the work in the final phase of the process. For instance, when we apply 7 steps, the total number of function evaluations called is 7640, whereas the final version contains 2152 triangles; that is 3.55 function evaluation per triangle. If we apply smaller steps for V_0, the result is of course worse; e.g. with 10 iterations we have 9389 function evaluations and 2118 triangles in the final object, so 4.43 overhead.

So, as can be seen in table 1, the efficiency of the method is not extremely sensitive to the number of iterations we use.

4.1 Comparison with earlier algorithms

In this version of the algorithm we do not attempt to reproduce the precise shapes as produced by other algorithms; to provide a full comparison, the faithfulness to the surface of the polygonal approximation would have to be measured. This could be done by taking samples for each triangle and computing the error from the true surface. The sum of such an error could be averaged over all the triangles. Polygonisation algorithms for iso-surfaces can be divided into two classes, adaptive and non-adaptive. For a given number of triangles an adaptive algorithm should have a smaller error than for a non-adaptive algorithm.

In order to find the surface, polygonisation algorithms such as the one described in this paper, must evaluate a function $V(r)$ for a number of points in space. An exhaustive comparitive analysis is beyond the scope of this paper, however it is likely (and certainly true for the algorithms we have tested) that the performance of such algorithms depends on the number of such function evaluations (*FE*'s), averaged over the number of triangles. (FE per triangle - FEPT).

Shrinkwrap was compared with an earlier algorithm as indicated below:

- method 1: *Shrinkwrap*, as described in this document.
- method 2: *Soft Objects*, based on ([**?**]).

The voxel-based algorithm that we consider for comparison performs some processing to find a point of the surface by intersecting the surface with one of the voxel boundaries. The point of intersection is then calculated. Since this can be done by a variety of techniques (e.g. regula falsi, binary search) this part of the algorithm should be the same for each, thus it is not counted in the total number of FEPTs. Another area where the algorithms differ is the method chosen for computing the normal. In the algorithm of this paper the normal is computed as a by-product of the search technique. The technique from ([**?**]) takes components computed from $V(X+h) - V(X-h)$ where h is 0.01 the length of a voxel

Algorithm	Sphere	2 Spheres	3 lines	penguin
Shrinkwrap	8.03	8.56	3.22	2.03 ($N_{steps} = 3$; incomplete triangulation)
Shrinkwrap				2.46 ($N_{steps} = 4$; incomplete triangulation)
Shrinkwrap				2.91 ($N_{steps} = 5$; triangulation OK)
Shrinkwrap				3.25 ($N_{steps} = 6$; triangulation OK)
Shrinkwrap				3.55 ($N_{steps} = 7$; triangulation OK)
Shrinkwrap				3.83 ($N_{steps} = 8$; triangulation OK)
Shrinkwrap				4.14 ($N_{steps} = 9$; triangulation OK)
Shrinkwrap				4.43 ($N_{steps} = 10$; triangulation OK)
Shrinkwrap				5.78 ($N_{steps} = 15$; triangulation OK)
Soft Objects	12.56	12.58	12.61	12.59

Table 1: Comparison of two polygonisation algorithms in terms of FEPTS

side. In fact algorithm 2 without computing normals has an FEPT count about 20 times less than with the normal calculation.

Three typical modelling situations were chosen (a single sphere, where there is no adaptivity; 2 spheres that blend together: and three blended line segments) to be tested along with a more complicated model, the penguin. Although the primitives produce similar models for both methods, the penguin surface depends on the style of blending chosen as well as the precise position for the final iso-surface. Thus figures for the penguin can only be thought of as a guide.

The data was used by both algorithms and the following results computed and shown in Table **??**. The numbers are FEPTS (Function Evaluations Per Triangle).

When the penguin is triangulated with 3 or 4 iterations, we find that the bill is not completely triangulated due to convergence failures with the Newton Raphson process. With 5 or more iterations, the triangulation is OK. Notice that the number of FEPTS increases sublinearly with the number of iterations.

4.2 Future research

Experiments show that the version of the shrinkwrap algorithm as discussed in this paper seems to be a promising approach to efficient adaptive triangulation of a relevant class of iso-surfaces. It is therefore interesting to look at some obvious improvements of the algorithm.

- the skeleton elements need not necessarily be limited to points, line segments and convex polygons. In fact, any geometric primitive that allows a differentiable distance function is appropriate.
- the method to move vertices onto the surface for every next value of $V_{0;k}$ uses a Newton-Raphson algorithm. This algorithm computes the distance a vertex should

move along a straight line (namely the gradient direction). If the distance between the iso-surface for k and $k+1$ is too large (that is, if ΔV is too large), this straight line can be a bad approximation of the curved field line that the vertex should ideally follow. Therefore, the steps ΔV cannot be too large, and hence N_{steps} cannot be too small. The algorithm therefore could be made much more efficient if the algorithm to move the vertices were not based on a first order Taylor expansion, but on a second order expansion. In that case, the curvature of the ideal trajectory of the moving vertex could be approximated better, and a vertex could move over a longer distance in one (curved) step. We are currently investigating this option.

- the criterion for acceptability of an edge is too strict. Indeed, irrespective of the local curvature, an edge (and hence, a triangle) is split as soon as one of the accuracy conditions is violated. It is, however, only essential that the surface is checked in samples that are not further away than the indicated distances; it is not necessary that the triangles are split if samples in the interior of such a triangle would indicate that the iso-surface is sufficiently close to the plane of the triangle. So a further reduction of the number of triangles in flat regions could be obtained by a slightly more sophisticated splitting criterion.

- along the same line, we can even afford to leave out sampling the surface with a sample density in accordance with the criteria **??** and **??** if a test could indicate that the entire region of an iso-surface within a triangle is sufficiently close to the plane of that triangle. Such a test should employ interval arithmetic or affine arithmetic on the bounding box of the triangle where a coordinate transformation should be applied such that the plane of the triangle becomes parallel to one of the coordinate planes. Then a very thin bounding box results, and the range of values of $V(r)$ on this box could be established with very few evaluations only.

- as indicated in Appendix B, the restriction of iso-surfaces that are homeomorphic to sphere can be easily lifted.

5 Conclusion

We have presented a new algorithm for polygonizing an iso-surface, defined by a set of skeletal elements. Whereas earlier approaches depend on a grid structure, the approach presented here is adaptive in the sense that the tiling is at arbitrary locations on the surface. The algorithm produces triangles of different sizes and shapes according to the local curvature of the surface offering an approximation with a known accuracy. The advantages of this technique are summarized as follows:

- Adaptive mesh
- No artifacts due to motion in fixed grid
- Vertex Normals are calculated essentially for 'free'
- Faster (fewer FEPTs) than previous adaptive algorithms.

A disadvantage of this algorithm, is that it can only cope with a single closed surface without holes. However, since the submission of the first version of this paper, we have devised an extension of this approach to solve this problem; this is outlined in Appendix B.

6 Acknowledgements

We would like to thank Jules Bloomenthal and Geoff Wyvill for their continuing interest and enthusiasm for this research. Also, the other members of the U. of Calgary graphics lab. (the GraphicsJungle) who have helped with the ongoing research on iso-surfaces. This research was partly sponsored by the Natural research and Engineering Council of Canada. One of the authors (CWAMvO) thanks the Department Board of Mathematics and Computing Science of EUT for giving him the opportunity of his sabbatical leave.

7 Appendix A: a smooth interpolation curve

The method for splitting edges as explained in **??** is based on finding a curve that should lay close to the iso-surface and that passes through two given points, say p_0 and p_3 (the A and B from section **??**). Since it should be close to the iso-surface, it should also be perpendicular to the normals in these two points, say n_0 and n_3. We approximate this curve by a cubic Bezier curve, and we demand it to be optimally smooth. Call this curve $p(t)$, $0 < t \leq 1$, with control points p_0, p_1, p_2, and p_3. Thus the extremes of the edge are p_0 and p_3; the normal vectors in these points are n_0 and n_3. The unknown variables are p_1 and p_2. The boundary conditions of perpendicularity to the normal vectors are expressed by

$$n_0 \cdot (p_1 - p_0) = 0 \tag{11}$$

and

$$n_3 \cdot (p_2 - p_3) = 0. \tag{12}$$

We solve this problem by minimising the average curvature of the curve subject to **??** and **??** by introducing the Lagrange multipliers μ_0 and μ_3. Then the minimisation problem is expressed as "minimise $\Phi(p_1, p_2)$", where

$$\Phi = \int_0^1 \mid \ddot{p}(t) \mid^2 dt + \mu_0(n_0 \cdot p_1 - p_0) + \mu_3(n_3 \cdot p_2 - p_3) \tag{13}$$

subject to **??** and **??**.

Let us express the curve $p(t)$ into the cubic Bernstein polynomials (they provide a complete base for cubic polynomials, and cubic polynomials are known to minimise the average curvature as expressed in the integral):

$$p(t) = p_0(1-t)^3 + 3p_1t(1-t)^2 + 3p_2t^2(1-t) + p_3t^3$$

This yields for the integral (apart from a multiplicative factor)

$$\int_0^1 \mid \ddot{p}(t) \mid^2 dt =$$

$$1/3 \mid p_3 - 3p_2 + 3p_1 - p_0 \mid^2 + \mid p_2 - 2p_1 + p_0 \mid^2 + (p_3 - 3p_2 + 3p_1 - p_0, p_2 - 2p_1 + p_0).$$

We proceed by demanding $\partial\Phi/\partial p_1$ and $\partial\Phi/\partial p_2$ to vanish; this gives

$$p_1 = \frac{2p_0 + p_3 - 2\mu_0 n_0 - \mu_3 n_3}{3} \tag{14}$$

$$p_2 = \frac{2p_3 + p_0 - 2\mu_3 n_3 - \mu_0 n_0}{3} \tag{15}$$

To establish the values of the multipliers, (**??**) is substituted back, yielding the following set of linear equations to be solved:

$$2\mu_0(n_0, n_0) + \mu_3(n_0, n_3) = (n_0, p_3 - p_0)$$

$$\mu_0(n_3, n_0) + 2\mu_3(n_3, n_3) = -(n_3, p_3 - p_0)$$

Observe that, in the case of normalised normal vectors, the coefficient matrix of this set takes the form

$$\begin{pmatrix} 2 & \cos\phi \\ \cos\phi & 2 \end{pmatrix},$$

where ϕ is the angle between the two normal vectors. This matrix is always non-singular.

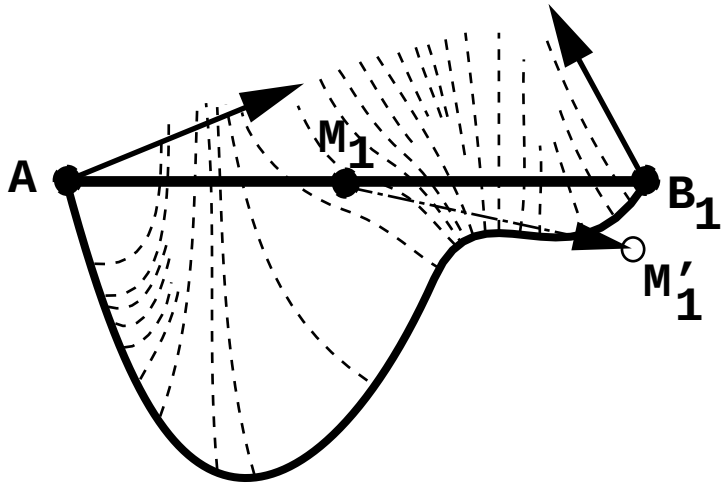
Finally, the desired midpoint M is found by taking $p(t = 1/2)$, so

$$M = \frac{p_0 + 3p_1 + 3p_2 + p_3}{8}.$$

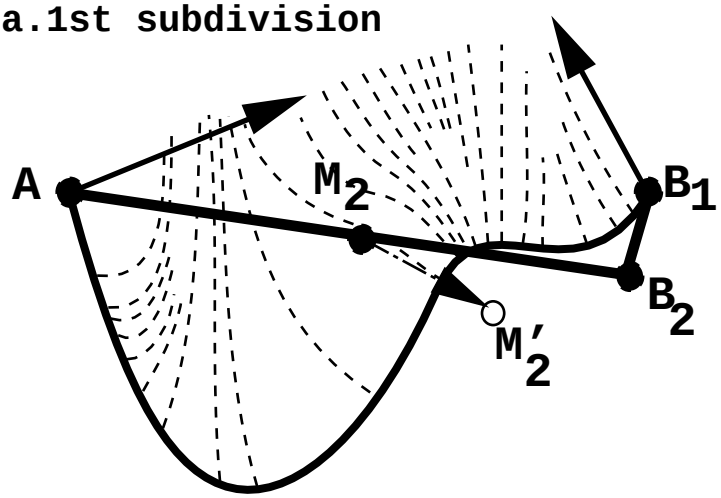
8 Appendix B: topological changes

The shrinkwrap algorithm as described in this paper assumes that the surface for $V_{0;k} = 1$ is homeomorphic to a sphere. It is not a priori possible to deduce from a given skeleton if this will be the case; moreover, the applicability of shrinkwrap would be significantly improved if it could deal with arbitrary topological configurations. Since the submission of the first version of this paper, an extension to shrinkwrap has been developed which allows such generalisation. It will be submitted for publication elsewhere; however we give a coarse outline of the proposed method here to make this paper self contained. The modifications to shrinkwrap are as follows:

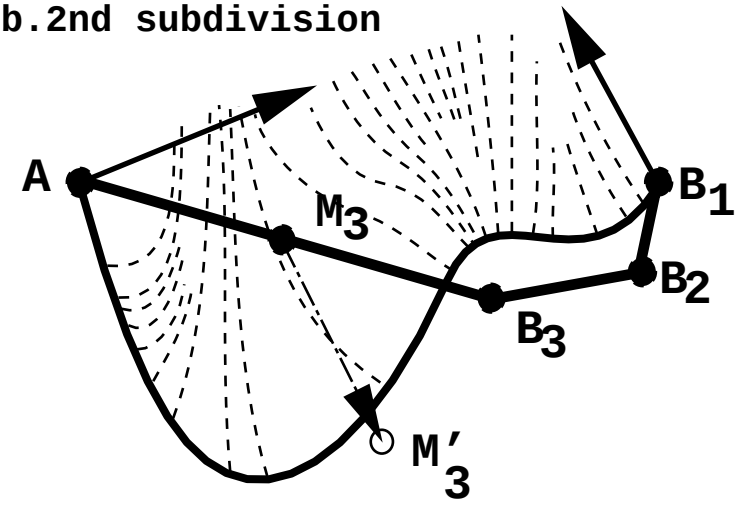
- a topological change occurs in a point (a saddle point) where $\nabla V(r) = 0$. So in principle, for every vertex r a search could be done to a nearby point r_{saddle} with $\nabla V(r_{saddle}) = 0$ using also a Newton Raphson iteration. Of course, it is useless to actually perform such an iteration for every vertex, and a condition is derived which can be used for a fast check to test if a saddle point is near a given vertex;
- once a saddle point is found, its function value is computed: $V_{saddle} = V(r_{saddle})$;
- this value V_{saddle} is inserted in the list of iso-values that is used for the shrink wrap iterations. Most often, V_{saddle} will be the next higher iso-value adjacent to the current one;
- when the vertices are displaced to arrive on the surface with iso-value V_{saddle}, make sure that a vertex lands in the saddle point;
- in this saddle point, distinguish between a rupture and a hole. This distinction can be made on the basis of a local approximation of the function near r_{saddle} as a quadratic polynomial in x, y, and z. The coefficients of this polynomial can be inspected in order to make the required distinction;
- disconnect the edges to the vertex r_{saddle} and re-connect them in order to form the right topological configuration;
- proceed with the iteration either to the next $V_{0;k}$ or the next V_{saddle} for an optional next saddle point.



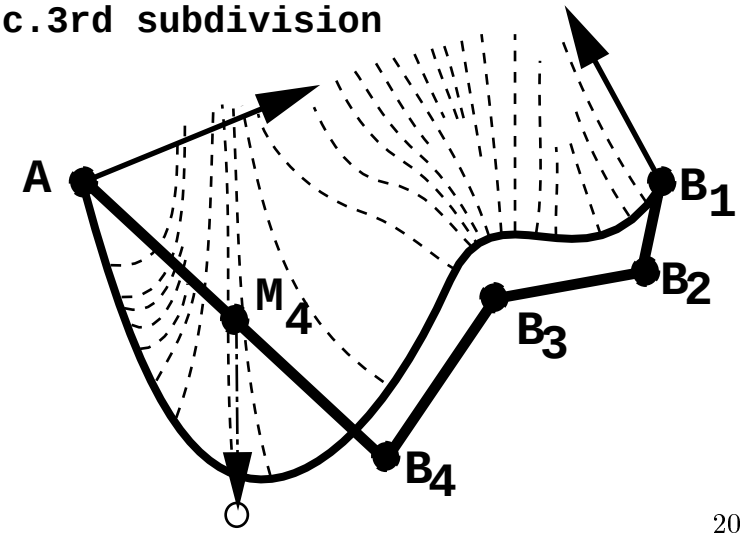
a.1st subdivision



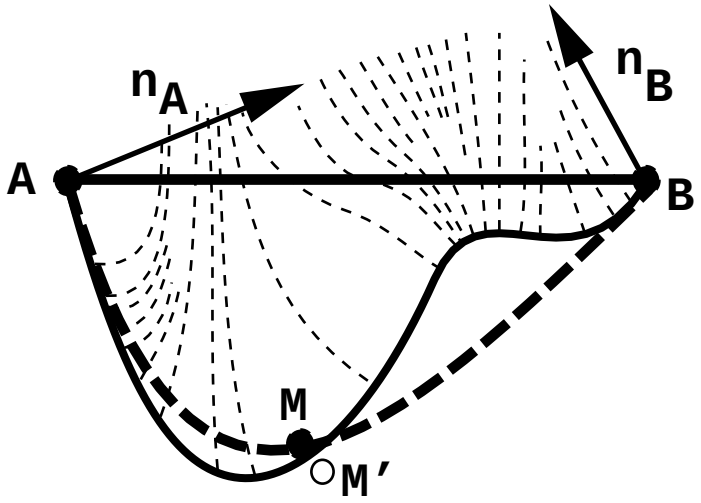
b.2nd subdivision



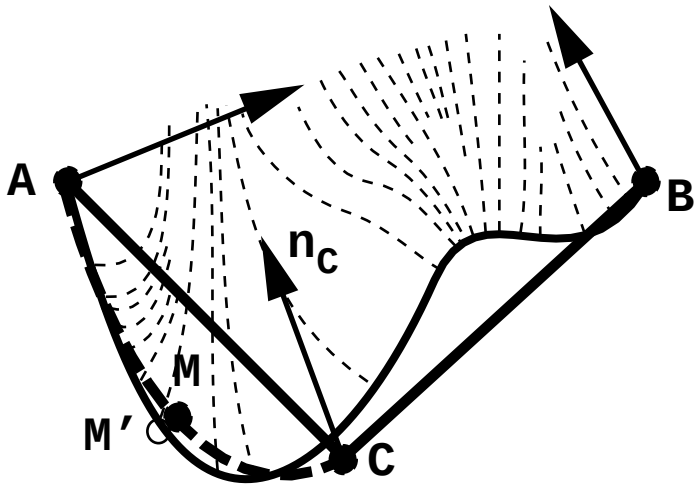
c.3rd subdivision



d.4th subdivision



e.subdivision using curve AMB



f. subdivision using curve AMC

figure 1.
Left:
constructing new vertices by repeated subdivision of an unacceptable edge and moving the new vertex according to the field's gradient may cause slow convergence and many spurious small chords.

Right:
constructing a new vertex, defined as the midpoint of the smoothest curve passing through A and B, perpendicular to the field's gradients in A and B. Similar for A and C.

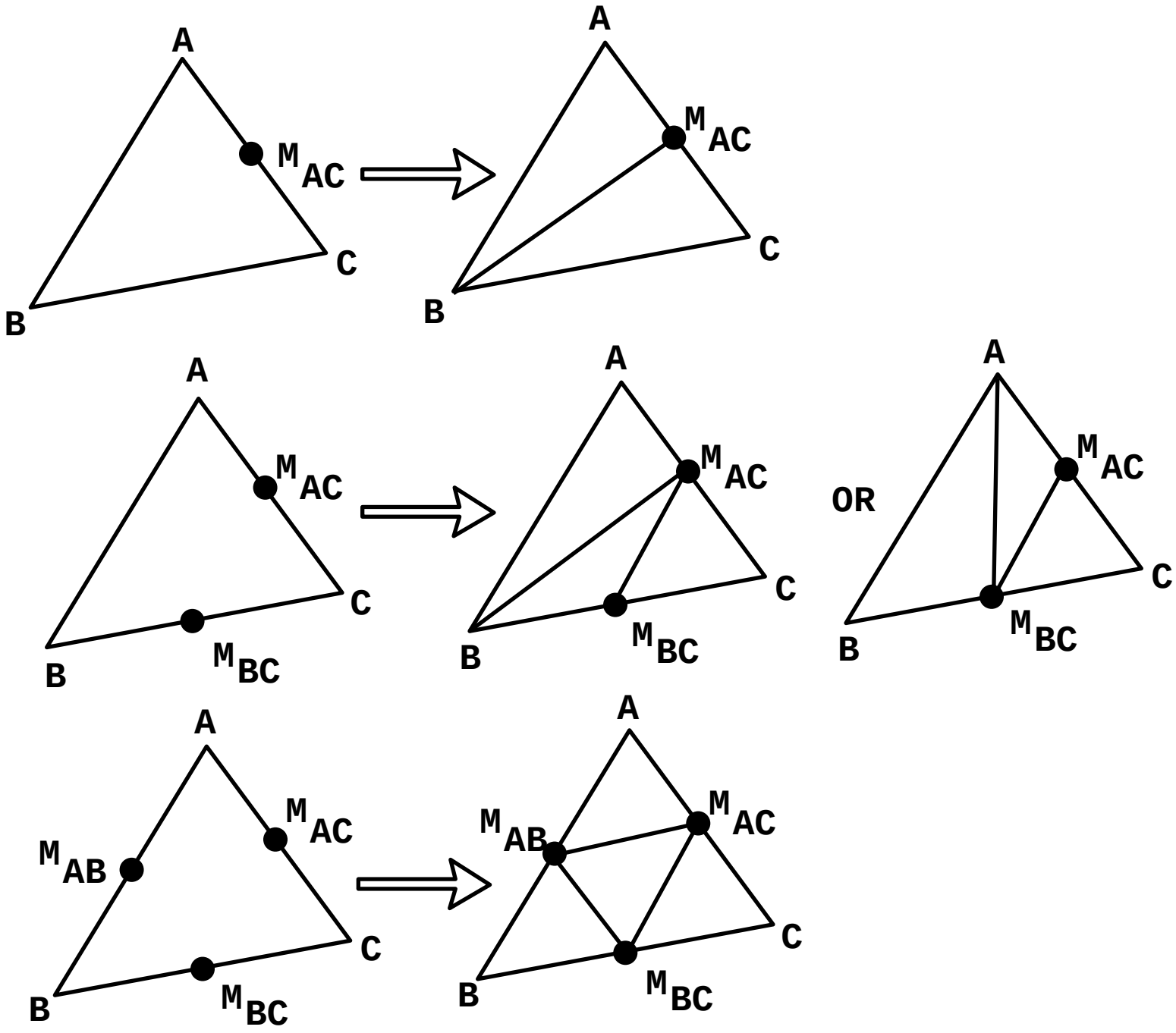
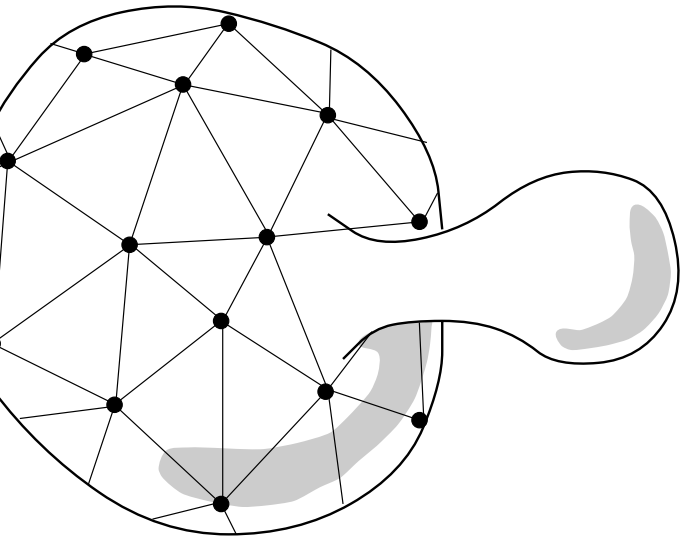
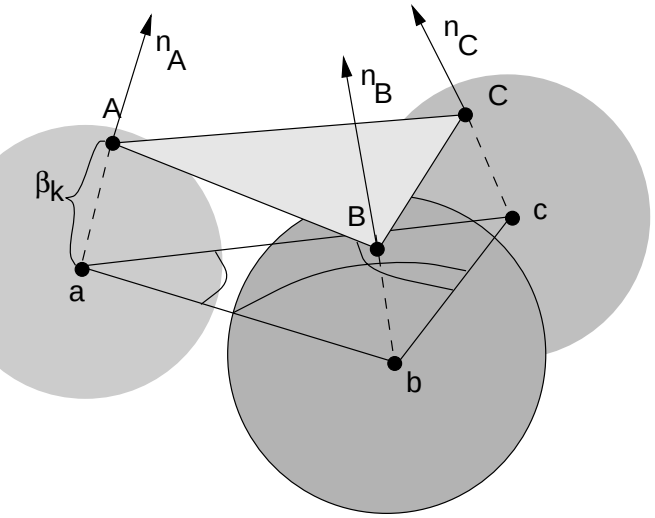


figure 2. The triangle subdivision scheme.



Robustness problem: due to too coarse sampling, the surface ’escapes’ between the mesh triangles.



5.: The iso-surface cannot ’escape’ through triangle if the indicated circular sectors fully cover *abc*

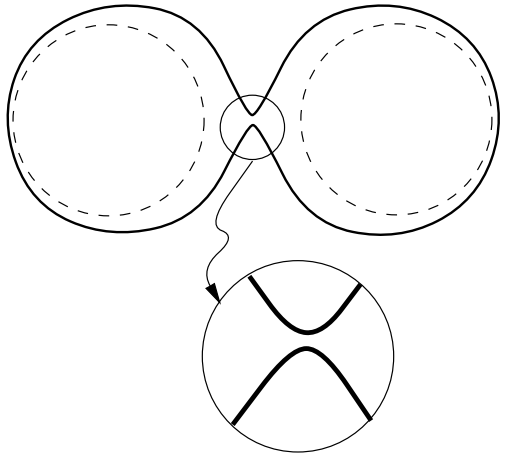


Fig. 4: in the vicinity of saddle points, small curvature radii can occur, even if the separate components have large curvature radii.

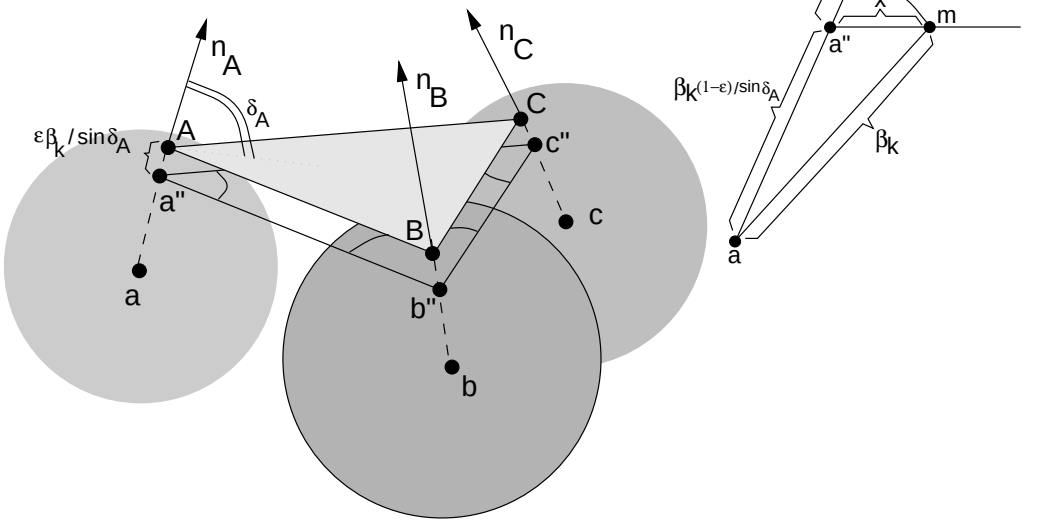
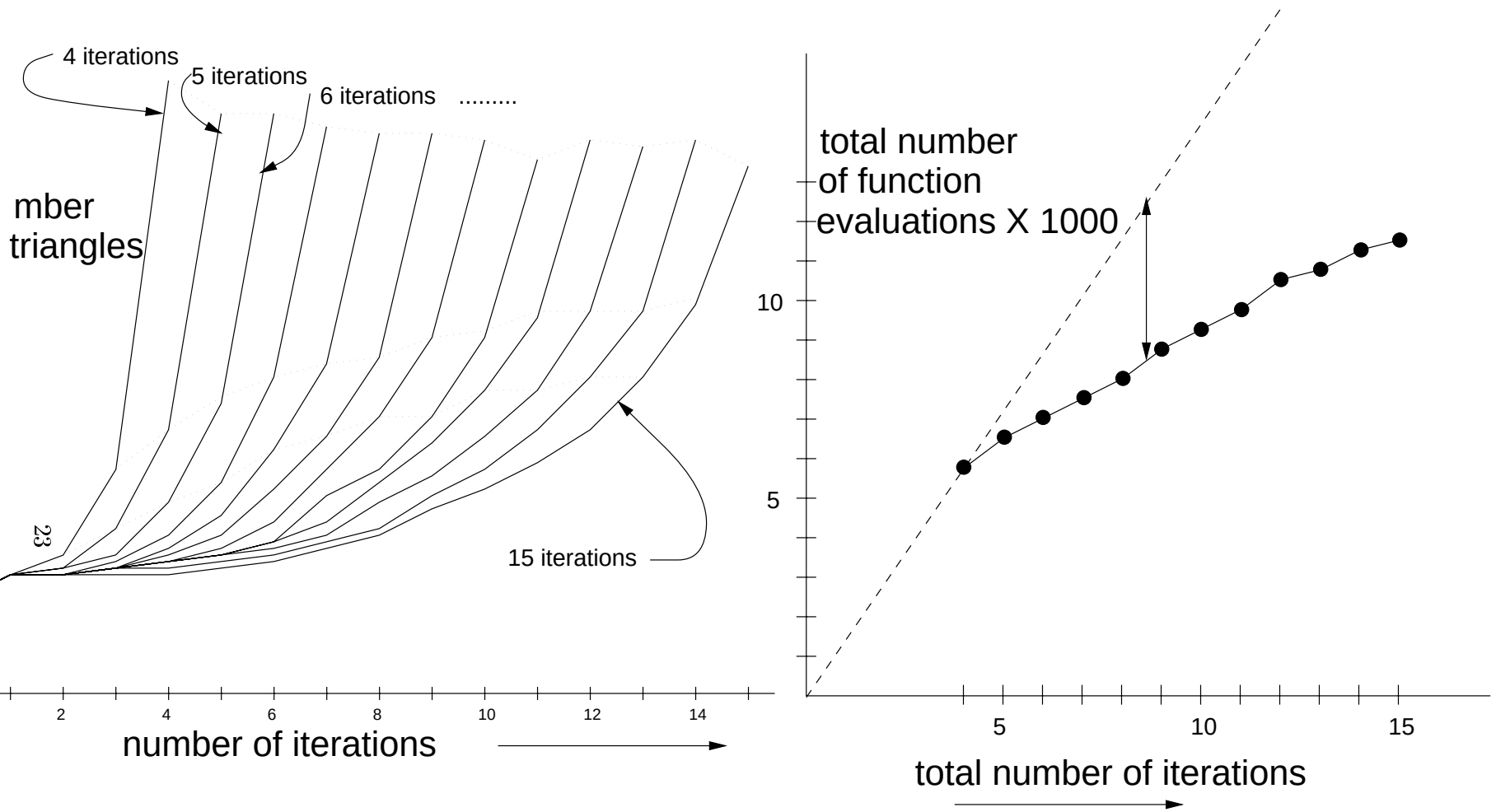


Fig 6: A similar construction as in fig. 5 applies to derive accuracy conditions. Right: the configuration a-a"-A-m



ig. 7: the number of generated triangles as a function of the
umber of iterations.

ight: the total number of function evaluations
creases less than linear with the total number
iterations used.