1991

R.C. Veltkamp

2D and 3D object reconstruction with the γ – neighborhood graph

Computer Science/Department of Interactive Systems Report CS-R9116 March

CWI, nationaal instituut voor onderzoek op het gebied van wiskunde en informatica

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

2D and 3D Object Reconstruction with the γ-Neighborhood Graph

Remco C. Veltkamp

CWI, Department of Interactive Systems

Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

e-mail: remco@cwi.nl

Abstract

This paper presents a method for the (re)construction of a simple polygon (2D) or polyhedron (3D) passing through all the points of a given set. The points are assumed to lie on the boundary of a closed object without holes, but no assumptions on the relative order of the boundary points are made. The reconstruction technique is based on a parameterized geometric graph, the γ-neighborhood graph. This graph unifies a range of geometrical graphs of which the convex hull, the Delaunay triangulation, and the Gabriel graph are well known instances. The hull of the γ-neighborhood graph is constricted, exploiting geometric information incorporated in the graph. Constriction on the basis of the γ-neighborhood graph succeeds in cases where constricting the Delaunay triangulation or growing the Voronoi skeleton fails, either because the constriction process gets locked, or because the Delaunay triangulation does not contain a Hamilton polygon/polyhedron. Object reconstruction from the γ-neighborhood graph gives correct and smooth boundaries when compared to other methods, as is shown by several examples.

1991 CR Categories:
I.3.5 [Computer Graphics] Computational Geometry and Object Modeling
G.2.2 [Discrete Mathematics] Graph Theory
I.4.8 [Image Processing] Scene Analysis

Key words: computational geometry, computational morphology, object reconstruction, γ-neighborhood graph, Hamiltonicity, range data.

1. *Introduction*

In many applications in geometric modeling, computer graphics, computer vision, distance map image processing and pattern recognition, the input is a set of points of an object's boundary surface, while some boundary of the point set defining an unambiguous object model is needed for further processing. The boundary points can be obtained in a variety of ways [Jain and Jain, 90]. Reconstruction typically yields a piecewise linear boundary, consisting of line segments in a two-dimensional, and triangles in three-dimensional embedding space. The boundary can then be used for visually appealing or realistic display [Glassner, 89], analysis, recognition and classification [Jain and Jain, 90], smooth interpolation [Herron, 85], or hierarchical representation and approximation [Veltkamp, 90a].

1.1. *Statement of the problem*

We first state the precise problem addressed here. The input is a set of N points in two-dimensional or three-dimensional Euclidean space. In the following we mean with point, a point from the input set, as opposed to an arbitrary point in space. The points are assumed to lie on the boundary of a closed object without holes. In the two-dimensional case the output is required to be a simple closed polygon passing through all points; the points are the vertices of the polygon. "Simple" here means that the polygon is connected, edges have only vertices in common, and every vertex has exactly two incident edges. In graph-theory, this amounts to a definition of a Hamilton cycle. In the three-dimensional case, the output must be a simple, triangular, and closed polyhedron, passing through all the points; the points are the vertices of the polyhedron. That is, the facets form a closed connected triangulated two-dimensional manifold which can be mapped onto a sphere. As a result, the polygonal/hedral boundary may not touch itself, which would give a thickness of zero.

We will use expressions like Hamilton polygon, Hamilton polyhedron, and Hamilton boundary for the required output. Note carefully that a Hamilton polyhedron is not the same as a Hamilton cycle in a graph in 3D embedding space. What we really need is a triangularly faceted boundary surface.

1.2. *Motivation*

It is not simple to determine a boundary of a set of points, when no structural relation between the points is known in advance. In contrast, it is relatively easy when the input is a planar contour chain; the sequential order of the points implicitly defines a boundary. For a pile of contours, for example taken from object cross sections, there are many methods to accomplish a correspondence between points of adjacent contours giving triangle strips, see [Keppel, 75], [Fuchs et al., 77], [Christiansen and Sederberg, 78], [Ganapathy and Dennehy, 82]. Also in a row by row scan of boundary points of a two-dimensional object, knowledge about the way of acquisition provides a way to connect the right points along the boundary.

On the other hand, a laser range system scanning points on the surface of an object will scan hidden surface parts only after moving around the object, when neighborly points have already been scanned. Also when surface points are extracted from a sequence of images of a moving object, or calculated from pairs of stereographic images (X-ray for example), the order in which points appear in the input gives no clue to any mutual correspondence. In this paper we therefore take the most general approach and assume no a priori imposed relation between the input points.

As pointed out in [O'Rourke et al., 87] a brute force method trying all combinations of N edges out of all $\binom{N}{2}$ edges from the complete graph results in

$$\binom{\binom{N}{2}}{N} = O(N^N)$$

time complexity. This is clearly infeasible, so that usually some geometric property is exploited. In a first attempt one can think of the distance property, but the

Figure 1: Nearest neighbors need not be consecutive along the boundary.

Euclidean distance does not always suitably approximate the metric of the boundary. Nearest neighbors, points having smaller distance to each other than to any other point, need not be consecutive points on the boundary, especially at strongly curved parts, see for example figure 1. Therefore a geometrical structure giving more information than some distance property, such as the Delaunay triangulation, can be useful in deciding which points are likely to be neighbors on the boundary [O'Rourke et al., 87] [Boissonnat, 84a]. In this paper we exploit the so called γ-neighborhood graph [Veltkamp, 88]. When a computational geometrical structure is used to extract shape information of a point set, it is called a computational morphology task [Toussaint, 80], [Toussaint, 88].

Also a brute force search in the Delaunay triangulation is not feasible: since the planar Delaunay triangulation has $O(3N-6)$ edges, a time complexity of

$$O\left(\binom{3N-6}{N}\right) = O(3^N)$$

results [O'Rourke et al., 87]. In three-space the situation is even worse: a triangular simple polyhedron through all points has $O(N)$ triangles, while the three-dimensional Delaunay triangulation has $O(N^2)$ triangles, yielding

$$O\left(\binom{N^2}{N}\right) = O(N^N)$$

time complexity. Also because the notion of a likely shape is an informal one, we are condemned to heuristics. Qualitative descriptions of the visual environment and heuristic approaches have recently received great interest in the computer vision community, see for example [QuaVis, 90].

1.3. Structure of the article

The rest of this article is structured in the following manner. The next section gives an overview of related work and results. Section 3 introduces the γ-neighborhood

graph, which is the basis of the presented reconstruction technique. The general technique is explained in section 4, where also example results are shown. Section 4.3 explains how we can always find a solution with the γ-graph, and presents examples of situations where a number of other methods fail. Section 5 presents the algorithm and discusses its complexity and implementation. Section 6 compares our technique with two other methods that we implemented. Finally section 7 mentions a few related applications and some conclusions.

2. *Related work*

2.1. Clustering

Much work has been done on finding a boundary of a set of points in the plane, generally not passing through all points, as in our application. In [Medek, 81], each point has an associated disc touching its nearest neighbor; the union of all discs defines clusters of points. [Edelsbrunner et al., 83] introduces a parameterized generalized disc; the intersection of all discs defines the so-called α-shape, which is closely related to the Delaunay triangulation. The 0-shape reduces to the convex hull.

Voronoi polygons are used as neighborhoods of points in [Ahuja, 82]. Clustering is then performed by grouping together Voronoi polygons having similar geometrical properties. The property actually used depends on the application.

2.2. Contour from rays

A unique planar contour can be reconstructed from rays [Alevizos et al., 87]. Rays are semi-infinite curves originating at contour points, and are supposed not to intersect the object. They represent for example the direction from which the points is seen, or the path of a robot arm that has sensed the point. Rays are thus more restrictive than mere points. The unique solution can be found in $O(N \log N)$ time, and in as much time can be verified whether the rays do not conflict, and thus correctly define a contour.

2.3. Contour chains

In 2D, there is no reconstruction problem when the input is a contour, since the contour itself is what we are looking for. It is needed though, to clean up a manually specified contour [Huijsmans, 83]. As mentioned before, there are several methods to obtain a boundary from a pile of cross sectional contours. Both [Keppel, 75] and [Fuchs et al., 77] construct a polyhedron that is globally optimal with respect to some criterion. In [Keppel, 75] it is proposed to consider the polyhedron with maximal volume as a natural object approximation. In [Fuchs et al., 77], the polyhedron of minimal surface area is chosen. In [Christiansen and Sederberg, 78] and [Ganapathy and Dennehy, 82] heuristic algorithms are presented that are only locally optimal with respect to some goal function. All these methods successively triangulate pairs of contours, yielding a ribbon of triangles between each contour pair.

A volumetric approach is presented in [Boissonnat, 88], where the Delaunay triangulation is used.

As stated before, in this paper we consider the more general situation where points do not necessarily lie on contours, but have arbitrary position.

2.4. Triangulation growth

Several methods expand some initial triangulation until all points are captured. In [Boissonnat, 82] the points are divided into subsets having the same sign of, or vanishing, Gaussian curvature. The subsets are triangulated separately. A current triangulation is grown by searching in the neighborhood of an edge in the contour of the current triangulation, for a point to create a new triangle with. These neighborhood points are mapped onto the Gaussian sphere. The point whose image forms a triangle with the edge in the convex hull of the image points on the Gaussian sphere is taken to construct a new triangle in the original triangulation.

In [Boissonnat, 84a] the points need not be divided into subsets according their Gaussian curvature. Now the points in the neighborhood of the edge are projected onto a tangent plane. The point that sees the edge under the largest angle is taken to create a new triangle.

[Choi et al., 88] performs a triangulation of points in three-space under the assumption that there is a viewpoint from which all these points are visible on the original surface, so that the original surface must be known in the first place. An initial triangulation is constructed and then grown, after which the resulting triangulation is improved according to some smoothness criterion. Surfaces not satisfying the assumption must be split, apparently manually.

2.5. Minimal area polyhedron

[O'Rourke, 81], just as [Fuchs et al., 77], proposes that a polyhedron of minimal surface area is the most natural polyhedral boundary of the point set. An argument for this proposition is that many real surfaces tend to a situation of minimal tension. Since the tension over the surface is proportional to its area, the object will take a shape with minimal surface area.

A heuristic algorithm is presented with starts with the convex hull, which is the minimal area polyhedron when all points would lie on the convex hull. Successively the internal point with the smallest value for

$$\sum(\text{area new faces})/(\text{area nearest face})$$

is added to the boundary, where the "nearest face" is the face on the current boundary that is closest to that internal point, and the "new faces" are the faces that must be created in order to include that point into the current polyhedron.

As shown in [Boissonnat, 84a], the minimal surface area criterion can give strange results.

2.6. Minimal area change constriction of Delaunay triangulation

In [Boissonnat, 84a] the Delaunay triangulation is taken, and boundary triangles (2D) or tetrahedra (3D) are successively deleted until all points are included in the

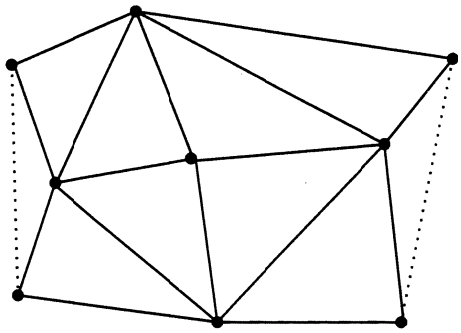

Figure 2: Constricting the Delaunay triangulation can get locked.

boundary. The boundary simplex (polytope of $(k+1)$ vertices in a kD space) to remove, is the one having the smallest or largest criterion value. In an attempt to minimize the modification of the current boundary, it is proposed [Boissonnat, 84b] to delete the simplex with the smallest value of

$$\frac{\sum(\text{area interior faces}) - \sum(\text{area boundary faces})}{\sum(\text{area all faces})}$$

where in 2D, face area means the length of the triangle side. This expression yields a small value when the boundary element is small relative to the inner faces of the tetrahedron.

However, the Delaunay constriction procedure can get locked, as shown in figure 2. After removing the flattest simplices, no more simplices can be deleted without yielding an invalid boundary. The point in the middle can then no longer be included into the boundary, although there exists a Hamilton polygon in the triangulation. There also exist Delaunay triangulations with no Hamilton polygon at all; more about both problems is said in section 4.3.

2.7. Shortest Voronoi skeleton

In [O'Rourke et al., 87] a simple polygon in the Delaunay triangulation is constructed by a growth algorithm. Since the dual of any triangulated simple polygon is a tree, the constructed polygon corresponds to a tree in the dual of the Delaunay triangulation, the Voronoi diagram. They argue that a natural polygon has a short Voronoi tree, acting as a skeleton or medial axis, and thus search for the minimal length tree in the Voronoi diagram. This method can also be performed in 3D, but both in 2D and in 3D the method seems to work properly only for objects with a clear skeleton. Especially in 3D the Voronoi tree is likely to twist around in order to reach all Voronoi vertices (or dual tetrahedra), which does not naturally correspond to a skeleton and will give unexpected results, see figure 11 in section 6. Note that the skeleton vertices need not lie inside the object.

Due to the restriction that the boundary must be Hamiltonian, they are not able to find a deterministic algorithm, but try each Voronoi vertex as a seed for a growth algorithm and record the shortest tree. This however leads to a worst-case time complexity of $O(N^3)$ in 2D, and $O(N^4)$ in 3D.

3. *The γ-neighborhood graph*

In this section we define the γ-neighborhood graph for arbitrary dimension k. In the following we will use 'γ-graph', '$\gamma(\gamma_0, \gamma_1)$-graph', or simply '$\gamma(\gamma_0, \gamma_1)$' and similar expressions, to denote the appropriate γ-neighborhood graph. In the definition of the γ-graph we use the following notation: for k arbitrary points $x_1, \ldots, x_k$ from the input set, and for $k \geq 2$, $r(x_1, \ldots, x_k)$ denotes the radius of the smallest sphere through $x_1, \ldots, x_k$. Thus for $k = 2$, $r(x_1, x_2)$ equals half the Euclidean distance between x_1 and x_2.

The neighborhood graph $\gamma(\gamma_0, \gamma_1)$ is defined for $-1 \leq \gamma_0, \gamma_1 \leq 1$, and $|\gamma_0| \leq |\gamma_1|$. In kD space, the graph connects points $x_1, \ldots, x_k$ pairwise, thus with $k(k-1)/2$ edges, if an associated neighborhood $\mathrm{N}(\gamma_0, \gamma_1)$ contains none of the other input points in its interior. In that case we call the neighborhood empty. $\mathrm{N}(\gamma_0, \gamma_1)$ is defined by two kD spheres in the following way:

1. the spheres have radii $r(x_1, \ldots, x_k)/(1 - |\gamma_0|)$ and $r(x_1, \ldots, x_k)/(1 - |\gamma_1|)$,
2. if $\gamma_0 \gamma_1 < 0$, the centers of the spheres lie on the same side of the plane through $x_1, \ldots, x_k$, if $\gamma_0 \gamma_1 > 0$ the centers lie on both sides of that plane,
3. if $\gamma_1 \leq 0$, we take the intersection of the two spheres, if $\gamma_1 \geq 0$, we take the union.

Note that there can be two neighborhoods if γ_0 and γ_1 are both non-zero. The graph connects $x_1, \ldots, x_k$, as soon as one of these neighborhoods is empty. Note further that this definition is valid for $\gamma_0 = \gamma_1 = 0$. In that case, the two spheres coincide, their common center lies in the plane through $x_1, \ldots, x_k$, and the intersection equals the union.

For $k = 2$ the definition involves two points and two circles, and $r(x_1, x_2)$ is scaled by factors $1/(1 - |\gamma_0|)$ and $1/(1 - |\gamma_1)|$. The planar $\gamma(\gamma_0, \gamma_1)$ reduces to well known geometric graphs for special values of γ_0 and γ_1:

- $\gamma_0 = \gamma_1 = 0$. The resulting neighborhood $\mathrm{N}(0, 0)$ is the smallest circle through x_1 and x_2, called the Gabriel neighborhood, after [Gabriel and Sokal, 69]. $\gamma(0, 0)$ is the Gabriel graph.
- $\gamma_0 = \gamma_1 = -1$. The intersection of the two half-planes yields the line through x_1 and x_2. If no three or more points are collinear, $\gamma(-1, -1)$ is the complete graph.
- $\gamma_0 = \gamma_1 = 1$. The union of the two half-planes gives the entire plane. If no three or more points are collinear, $\gamma(1, 1)$ is a void graph.
- $\gamma_0 = -1$, $\gamma_1 = 1$ and $\gamma_0 = 1$, $\gamma_1 = -1$. In both cases the two half-planes lie on the same side of the line through x_1 and x_2. Therefore they coincide (more general $\gamma(\gamma_0, -\gamma_0) = \gamma(-\gamma_0, \gamma_0)$. The neighborhood is empty if all other points lie on one side of the line through x_1 and x_2, or on the line, but outside the segment from x_1 to x_2. Therefore, $\gamma(-1, 1)$ and $\gamma(1, -1)$ are the convex hull.
- $\gamma_0 = \gamma_1$. The graph $\gamma(\gamma_0, \gamma_0)$ reduces to the circle-based β-skeleton, whose neighborhood consists of equally sized circles [Kirkpatrick and Radke, 85].

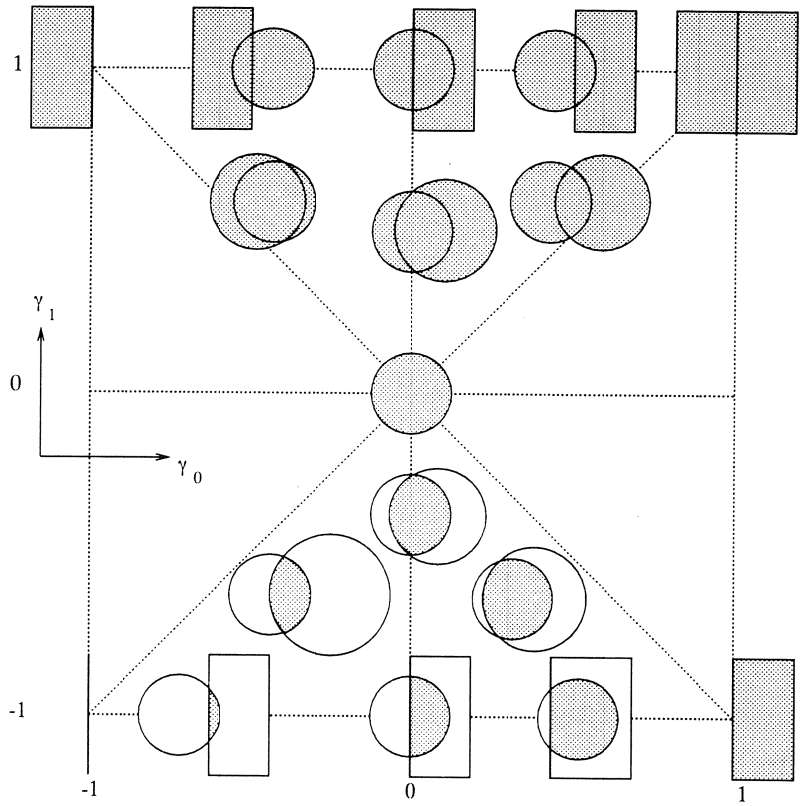

Figure 3: Overview of the spectrum of planar $\gamma(\gamma_0, \gamma_1)$-neighborhoods, $-1 \leq \gamma_0, \gamma_1 \leq 1$, and $-1 \leq \gamma_0/\gamma_1 \leq 1$ (rectangles denote half-spaces).

Figure 3 gives a graphical overview of the whole spectrum of planar neighborhoods.

In 3D space, the definition of the neighborhood involves three points and two spheres. The scale factors blow up the smallest possible sphere through x_1, x_2, and x_3, while the spheres still pass through these points. The three points are connected, forming a triangle, if the two spheres form an empty neighborhood. Not all of the geometric graphs in the list above generalize to 3D in the same way as the γ-graph, if at all.

So far we have considered fixed values of the γ-parameters. We can also look at the largest values of the γ-parameters, for which the corresponding neighborhood is still empty. That is the value for which the sphere touches a $(k+1)$th point or is either 1 or -1 if there is no such point. We define $\gamma([\gamma_0, \gamma_1], [\gamma_2, \gamma_3])$ to be the graph connecting points $x_1, \ldots, x_k$ with each other, if the largest γ-parameter values for which the corresponding neighborhood is still empty, lie in $[\gamma_0, \gamma_1]$ and $[\gamma_2, \gamma_3]$ respectively.

The $\gamma([-1, 1], [0, 1])$-graph connects points $x_1, \ldots, x_k$ in kD space if there are two spheres through these sites, of arbitrary radius, such that the union is empty. If no more than $k+1$ points are cospherical, this amount to a definition of the Delaunay triangulation, giving a unique triangulation of the convex hull (in 3D we can also speak of a tetrahedralization). If there are more than $k+1$ cospherical points, then $\gamma([-1, 1], [0, 1])$ connects them all, whereas the Delaunay triangulation arbitrarily connects k points as long as the resulting $(k-1)$D-faces do not intersect.

The planar graph $\gamma([\gamma_0,\gamma_1],[\gamma_2,\gamma_3])$ can be constructed in $O(N \log N)$ time, and the k-dimensional one ($k > 2$) in $O(N^{1+\lceil k/2 \rceil})$ time, provided that the point set is non-degenerate, and $[\gamma_0,\gamma_1] \subseteq [-1,1]$ and $[\gamma_2,\gamma_3] \subseteq [0,1]$. All other k-dimensional γ-graphs can be computed in $O(N^{k+1})$ time.

The neighborhood graph presented here is called γ-graph because the notions of parameterized neighborhoods exploited by the α-hull [Edelsbrunner et al., 83] and the β-skeleton [Kirkpatrick and Radke, 85] are pushed further, resulting in a unification of a range of geometric graphs. See [Veltkamp, 90b] for a more detailed explanation of the relation with other geometric graphs, and the time complexities for computation.

4. Constricting the γ-graph

In the following we use the general notion *boundary segment* for an edge lying on the current boundary in 2D, and for a triangle on the current boundary in 3D. We use *boundary simplex* for a triangle having a boundary segment in 2D, and for a tetrahedron having a boundary segment in 3D.

In order to obtain a Hamilton boundary, we take $\gamma([-1,1],[0,1])$ and successively remove boundary simplices from the body of the object, by deleting boundary segments from the current boundary (initially the convex hull). The boundary is iteratively constricted until all vertices are included in the boundary.

First we assume that this graph is non-degenerate (and thus coincides with the Delaunay triangulation). Subsection 4.3 tells what to do when $\gamma([-1,1],[0,1])$ does not contain a Hamilton boundary. The solution also covers the case where the graph is degenerate, or the constriction process gets locked.

Removing a boundary simplex must not introduce an isolated vertex, dangling boundary segments, or a self-intersecting boundary. Thus, deletion is allowed only if:

- the simplex has one boundary segment, and the vertex opposite to that segment is not on the boundary, or
- (in 3D only) the tetrahedron has two boundary triangles, and the edge opposite to the common edge of these triangles does not lie on the boundary.

We will call a simplex removable if its removal is allowed according these rules.

In order to select the simplex to remove, we associate to all current boundary simplices a value that is based on the γ-values of the boundary segments. However, we keep the sign of the γ-values of the boundary segments consistent with the following rule: if $-1 \leq \gamma < 0$, the center of the associated circle/sphere lies on the side of the boundary segment that is outside the current boundary, and if $0 < \gamma \leq 1$, the center lies on the inside.

4.1. 2D

We first consider the 2D case, in which only triangles with one boundary edge may be deleted. The selection of the triangle to remove is based on the attempt

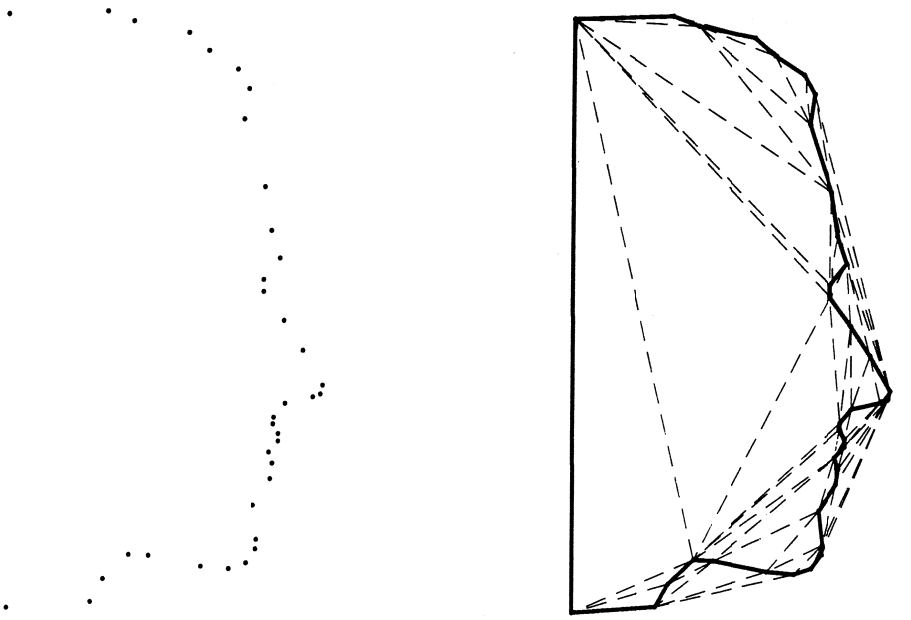

Figure 4: Planar example result: face of 37 points. Solid lines depict the boundary, dashed lines $\gamma([-1,1],[0,1])$.

to, informally speaking, change the shape of the current boundary not too much, relative to the size of the triangle. Formally, we choose the triangle with the largest interior angle at the vertex opposite to the boundary edge. A second motivation for this selection criterion is that the interior vertex with the widest angle has the largest probability among the interior vertices of all the current boundary triangles to be seen, or sensed, from outside.

Let R be the radius of the circle through the vertices of the triangle that we consider, γ the γ-value of the boundary edge corresponding to that triangle, and x_1 and x_2 the two vertices on the boundary. We abbreviate $r(x_1, x_2)$ to r.

Note first that r/R is independent of the size of the triangle. The angle at the interior vertex, ϕ, increases when r/R increases if γ is non-negative, and increases when $2 - r/R$ increases if γ is non-positive. The exact relation is given by the sine rule: $r/R = \sin\phi$. By definition, r/R equals $1-|\gamma|$, which is $1-\gamma$ for a non-negative γ. Similarly, $2 - r/R$ expands to $1 + |\gamma|$, which equals $1 - \gamma$ for a non-positive γ. This results in the following selection rule:

> delete the removable triangle whose boundary edge has the largest value for $1 - \gamma$ (or equivalently, the smallest value for γ).

Figure 4 and 5 show two examples: a point set from a face and from a chalice.

4.2. 3D

Analogous to the planar case, we wish to change the shape of the object not too much, when we successively delete boundary tetrahedra. A removable tetrahedron can have one or two boundary triangles, under the constraints given above. We first consider a tetrahedron having one boundary triangle. Let R be the radius of the sphere through the vertices of the tetrahedron, γ the γ-value of the boundary

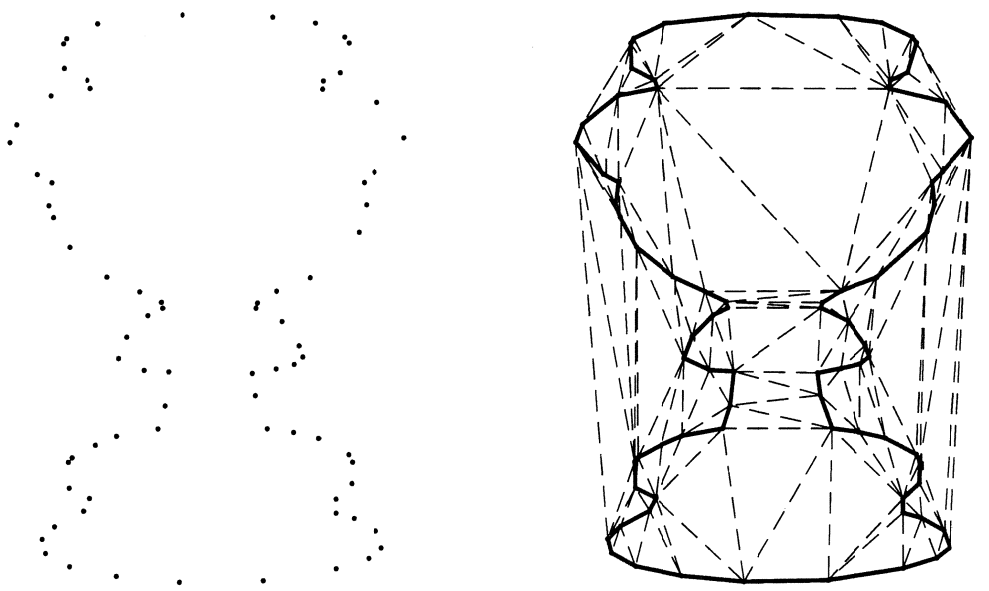

Figure 5: Planar example results: chalice. Solid lines depict the boundary, dashed lines $\gamma([-1,1],[0,1])$.

face that corresponds to that tetrahedron. We call the vertices on the boundary x_1, x_2, and x_3, and the area of triangle (x_1, x_2, x_3) A; we let y be the interior vertex, ϕ the solid angle at y (bounded by three sides of the tetrahedron), and abbreviate $r(x_1, x_2, x_3)$ to r.

To the best of my knowledge, there is no "3D sine rule", relating ϕ to R and r or A. However, ϕ does depend on the shape of triangle (x_1, x_2, x_3), and on how close y lies to triangle (x_1, x_2, x_3) (relative to the size of the tetrahedron or R). Note that analogous to the 2D case, r/R is independent of the size of the tetrahedron. Now, for a fixed shape of (x_1, x_2, x_3), $1 - \gamma$ expresses how close y lies to triangle (x_1, x_2, x_3) relative to the tetrahedron size. The larger $1 - \gamma$, ranging from 0 to 2, the closer y lies to the triangle and the wider ϕ is. Conversely, when γ is fixed, a "wider" triangle gives a larger ϕ.

We have used a criterion based on $1 - \gamma$ and A/R^2 [Veltkamp, 89]. It appeared that using $1 - \gamma$ alone gives a better result. Indeed, a typical convex hull has many thin triangles that should be deleted in the constriction process, see for example the convex hull in figure 7.

There are also tetrahedra that have two boundary faces. As stated before, such a tetrahedron is removable only if the edge opposite to the edge that is common to both boundary triangles, does not lie on the boundary. Since all four vertices already lie on the boundary, deletion of the tetrahedron adds no points to the boundary. However, it can result in an extra boundary tetrahedron, and moreover, deletion of the tetrahedron gives the two vertices opposite to the boundary triangles "more air", enlarging the probability that they are sensed from these directions. Because the solid angles at the two vertices bound two non-overlapping parts of space, it seems logical to use the sum of the values $1 - \gamma$ corresponding to both opposite boundary triangles, as the criterion value for the tetrahedron.

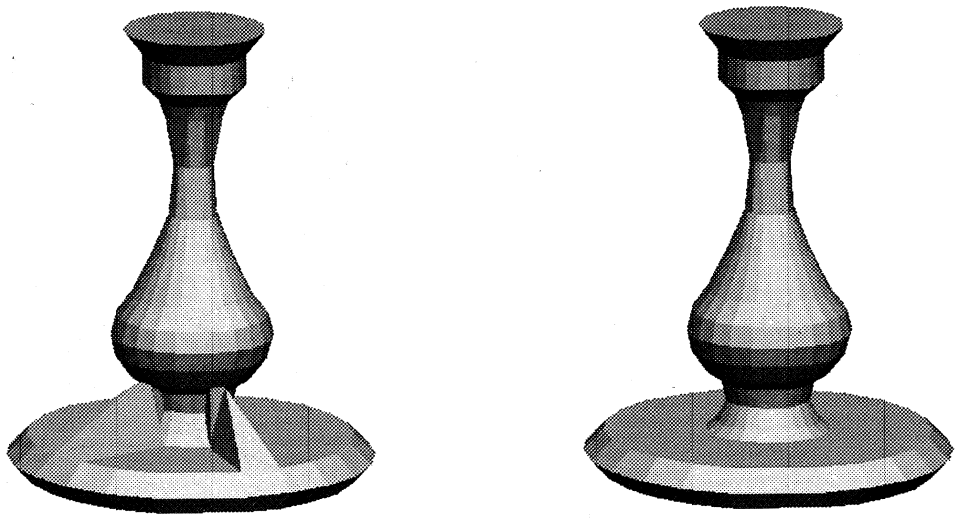

Figure 6: Synthetic object: candlestick consisting of 481 points and 958 triangles. Both objects have all points on the boundary.

The selection rule that captures the criterion value for both the tetrahedra with one and with two boundary triangles, as well as the 2D case then becomes:

> delete the removable simplex with the largest sum of $1 - \gamma$ that correspond to the boundary segments of the simplex (or equivalently, the smallest sum of γ).

Figure 6 shows how a synthetic 3D point set from a candlestick's surface is processed. The left picture shows the result of our constriction algorithm when stopped as soon as all points lie on the boundary. There are still removable tetrahedra, necessarily having two faces on the boundary. Especially with such artificial objects there is no general rule telling how long to proceed removing tetrahedra when already all points lie on the boundary. The right picture shows the resulting object after removing just enough tetrahedra. The faces look quadrilateral because pairs of triangles are coplanar, but are really triangular as is clear from the tetrahedra in the left picture.

Figure 7 illustrates the constriction process performed on points from a real laser-range data set measured from a mask [Rioux and Cournoyer, 88]. The convex hull, two intermediate objects, and the final results are shown. The algorithm is stopped as soon as all points lay on the boundary.

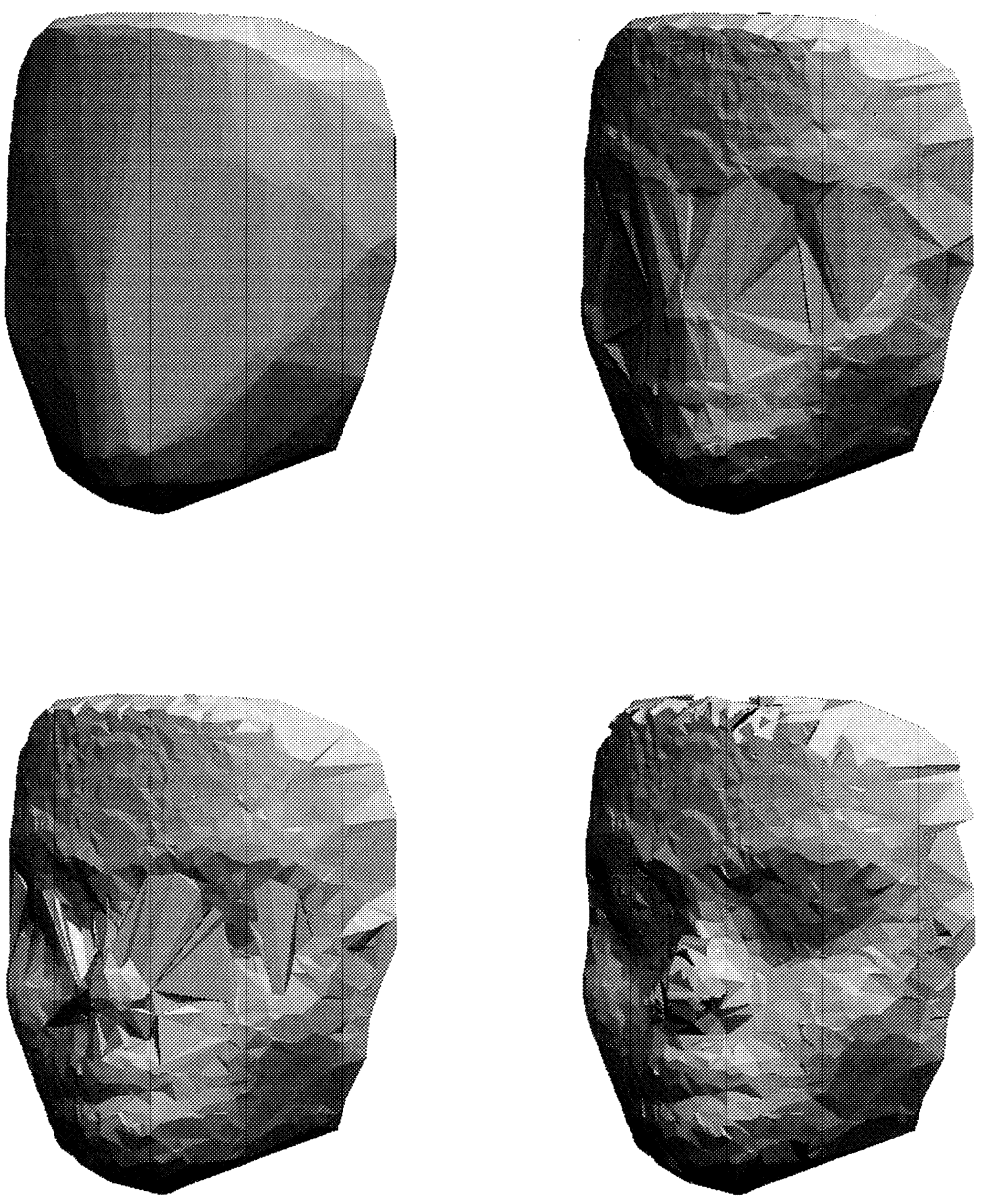

Figure 7: Mask reconstructed from 1468 laser-range data points. Upper left is the initial object consisting of 8883 tetrahedra; 255 points and 504 triangles lie on the convex hull. Upper right is the intermediate object after removing 2500 tetrahedra; 1019 points and 2034 triangles lie on the current boundary. Lower left is the intermediate object after removing 3500 tetrahedra; 1337 points and 2670 triangles lie on the current boundary. Lower right is the final object after removing 4696 tetrahedra; 1468 points and 2930 triangles lie on the boundary.

4.3. Hamiltonicity

Most results from graph-theory on Hamiltonicity apply to planar (embeddings of) graphs, and are closely related to the notion of connectivity. A graph is t-connected if there are at least t paths in the graph between any two vertices. Every planar triangulation is at least 2-connected, and at most 5-connected. Every 4-connected planar graph is Hamiltonian [Tutte, 77], and thus also also every 5-connected one, and also every 4- and 5-connected Delaunay triangulation. For a long time it has been unknown whether non-degenerate Delaunay triangulations (and thus $\gamma([-1,1],[0,1])$-graphs) exist, that are non-Hamilton [O'Rourke, 86]. A 2-connected example was given in [Dillencourt, 87], see also figure 8, and a 3-connected one in [Dillencourt, 89].

In three-space we are not interested in a Hamilton cycle of edges, but in a Hamilton polyhedron of triangles. It seems as yet not clear whether there are Delaunay tetrahedralizations that contain no Hamilton polyhedron. However, also in 3D the constriction process can get locked, analogous to the situation in figure 2.

Both in 2D and in 3D, the $\gamma([-1,1],[\gamma_0,1])$-graph, for some $\gamma_0 < 0$, connects more points with each other than in the triangulation, forming overlapping simplices. The extra simplices will have smaller interior angles at the vertex opposite to the boundary segment, than overlapping simplices from $\gamma([-1,1],[0,1])$, but they give more choice in the selection of the boundary simplices to delete. When we now delete a boundary simplex during the constriction process, we must also delete all overlapping simplices, in order to keep the boundary valid. For a γ_0 small enough, $\gamma([-1,1],[\gamma_0,1])$ will be Hamiltonian (the complete graph $\gamma([-1,1],[-1,1])$ always contains a Hamilton polygon or polyhedron), and locking of the process will not occur. Figure 8 shows the non-Hamiltonian Delaunay triangulation from [Dillencourt, 87] on the left, and the Hamiltonian $\gamma([-1,1],[-0.2,1])$.

There is no way to tell in advance for which value of γ_0 the graph $\gamma([-1,1],[\gamma_0,1])$ will be Hamiltonian. Moreover, decreasing γ_0 has a somewhat unpredictable effect. Many extra unneeded edges can arise, incident to vertices that cause no problem. Alternatively, in the planar case we can augment $\gamma([-1,1],[0,1])$ to be 4-connected, thereby assuring Hamiltonicity. A sufficient, but not always necessary, procedure is to make every vertex incident to at least four other vertices, that is, making every vertex of degree four. Because in a triangulation each vertex has at least two neighbors, at most two new neighbors need to be found. A vertex having

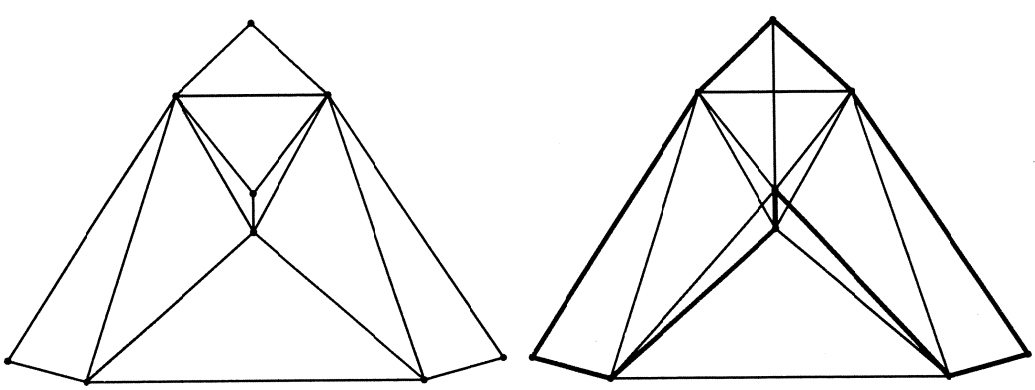

Figure 8: Left: non-Hamiltonian Delaunay triangulation. Right: $\gamma([-1,1],[-0.2,1])$ showing a Hamilton cycle in fat lines.

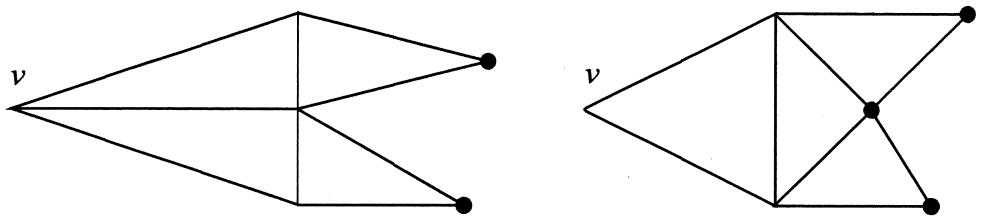

Figure 9: Two partial triangulations showing candidate vertices to make v of degree four.

two or three neighbors is incident to one or two triangles respectively. When two triangles are incident, we consider the two opposite vertices; when only one triangle in incident, we consider the opposite vertex, and the two vertices opposite to that one, see figure 9. Among the candidate vertices we choose the one or two giving the largest value of the γ-parameters defining an empty γ-neighborhood.

5. *Algorithm*

Algorithm 1 shows our constriction algorithm in pseudo-C code. First of all the appropriate γ-graph must have been constructed, so that it can be loaded for processing (line 1). Since the 2D γ-graph is edge-oriented in nature, the main data-structure is edge-based, allowing to address an edge in constant time. The edges are ordered around each vertex. In 3D the γ-graph is triangle-oriented and the main data-structure triangle-based; also the edges are explicitly represented so that both triangles and edges can be addressed in constant time. The triangles are ordered around each edge.

In order to keep track of the boundary simplices and their criterion values, a heap structure stores their boundary segment(s) together with, and sorted on, their criterion value. The heap is initially filled with the removable simplices on the convex hull (line 2). Each point and edge is marked to indicate whether or not it is on the boundary (line 3). While not all points are on the boundary and the heap is not empty (line 4), we try to remove a boundary simplex. We therefore take the simplex in the root of the heap (line 5) and check whether it is removable (line 6). Although each simplex is removable at the time it is inserted into the heap, this check is necessary since it can have become unremovable due to deletion of other simplices. If removal is allowed the simplex is deleted (line 7), involving also the deletion of all overlapping simplices in order to get an unambiguous boundary. The new boundary point and edge are marked and NumBP is incremented when appropriate (line 8). Each simplex neighboring the deleted one (line 9) is inserted into the heap with its criterion value, if its removal is allowed (line 10).

5.1. *Complexity*

For the time complexity analysis of our algorithm, we first consider $\gamma([-1,1],[0,1])$. Its construction is a straightforward adaptation of constructing the Delaunay triangulation, taking the γ-values into account; it can thus be computed in $O(N \log N)$ time in 2D [Lee and Schachter, 80] and $O(N^2 \log N)$ in 3D [Boissonnat, 84a]. The

```
1.  LoadGraph (G);
2.  H = InitHeap (G);
3.  NumBP = MarkBoundary (P, G);
4.  while (NumBP < #P && H ≠ ∅ )
    {
5.    S = Root (H);
6.    if (Removable (S))
      {
7.      Remove (S, G);
8.      NumBP += MarkBoundary (P, G);
9.      for (each neighbor N of S)
10.       if (Removable (N)) Insert (N, H);
      }
    }
```

Algorithm 1: Constriction algorithm

time for loading the graph (line 1) is proportional to the size of the graph, $O(N)$ and $O(N^2)$ respectively.

Both in 2D and in 3D the convex hull consists of $O(N)$ boundary segments. It can be extracted from the γ-graph, in $O(N)$ time, given an initial convex hull segment. Our γ-graph construction algorithm provides one such segment implicitly as the first one in the output, without extra cost in time. Building a heap of k elements takes $O(k \log k)$ time, so line 2 costs $O(N \log N)$. Marking all points in line 3 takes $O(N)$.

The while-loop is executed $O(N)$ times to include all points into the boundary, but as much as $O(N^2)$ times (in 3D only) to remove tetrahedra without adding an extra point to the boundary. Taking the root of the heap and rearranging takes $O(\log k)$ time for a heap size of k. Since only boundary simplices are in the heap, this amounts to $O(\log N)$ for line 5. Checking whether a boundary simplex is removable (line 6) takes constant time; in 3D this can only be done when edges are represented explicitly, as we do. Deletion of a simplex (line 7) and marking a point and edge (line 8) takes constant time. For the constant number of neighboring simplices, checking whether they are removable takes constant time, but insertion costs $O(\log N)$. The while-loop thus costs $O(N \log N)$ in 2D, and $O(N^2 \log N)$ in 3D, which is also the total complexity of the algorithm.

For $\gamma([-1,1],[\gamma_0,1])$, $\gamma_0 < 0$, there is as yet no output sensitive algorithm, so that we must resort to a naive algorithm giving a time complexity of $O(N^2)$ in 2D and $O(N^3)$ in 3D [Veltkamp, 90b]. To keep the heap of size $O(N)$ we store for each boundary segment only the boundary simplex with the largest criterion value among the overlapping ones. The resulting total complexity is then $O(N^2 \log N)$ for 2D and $O(N^3 \log N)$ for 3D.

These are all worst case complexities, combining the worst possible situations. For practical situations we are also interested in the expected behavior of the al-

	2D	3D
worst case $\gamma_0 = 0$	$O(N \log N)$	$O(N^2 \log N)$
worst case $\gamma_0 < 0$	$O(N^2 \log N)$	$O(N^3 \log N)$
expected case	$O(N \log N)$	$O(N \log N)$

Table 1: Overview of the worst case and expected case time complexities.

gorithm, for which we leave $\gamma([-1,1],[\gamma_0,1])$, $\gamma_0 < 0$ out of consideration. The expected complexity and computation time of $\gamma([-1,1],[0,1])$ is linear in N, both in 2D and in 3D. This yields expected time complexities of $O(N \log N)$. The various time complexities are summarized in table 1.

All storage complexities are dominated by the size of the γ-graph: $O(N)$ and $O(N^2)$ for $\gamma([-1,1],[0,1])$ in 2D and 3D respectively, for $\gamma([-1,1],[\gamma_0,1])$, $\gamma_0 < 0$ $O(N^2)$ and $O(N^3)$.

5.2. Implementation

The algorithm has been implemented in C. In 3D only constriction of $\gamma([-1,1],[0,1])$ is actually implemented, since we encountered no situations where the constriction process gets locked, or where there is no Hamilton polyhedron. In 2D also the constriction of $\gamma([-1,1],[\gamma_0,1])$, $\gamma_0 < 0$ was implemented because (usually for small input point sets) constriction can actually get locked, and there are even known $\gamma([-1,1],[0,1])$ graphs that contain no Hamilton polygon, see figure reff-dillencourt. The constriction process takes about four seconds on a Sun SparcStation 1+ for the mask data set, starting with 8883 tetrahedra.

The pictures in figure 6, 7, 11 and 12 have been generated using flat-shading (as opposed to more advanced shading methods), in order to clearly show the triangles.

6. Comparison

We have also implemented two other boundary reconstruction methods based on the Delaunay triangulation and the dual Voronoi diagram: the minimal area change constriction of the Delaunay triangulation, and the shortest skeleton growth in the Voronoi diagram, see section 2.

As long as the points are sampled fine enough along the boundary, our algorithm behaves well. Point sets that are not sampled fine enough suggest gaps where there

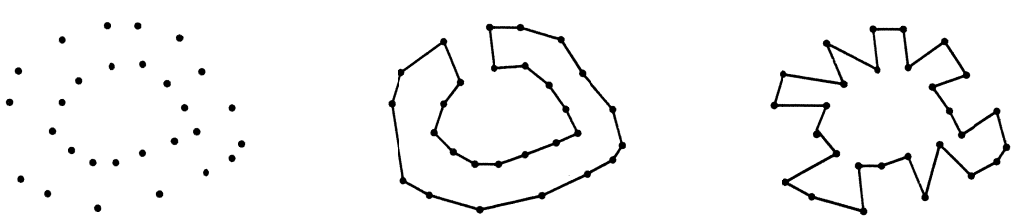

Figure 10: From left to right: input point set, Voronoi skeleton result, and γ-based constriction result.

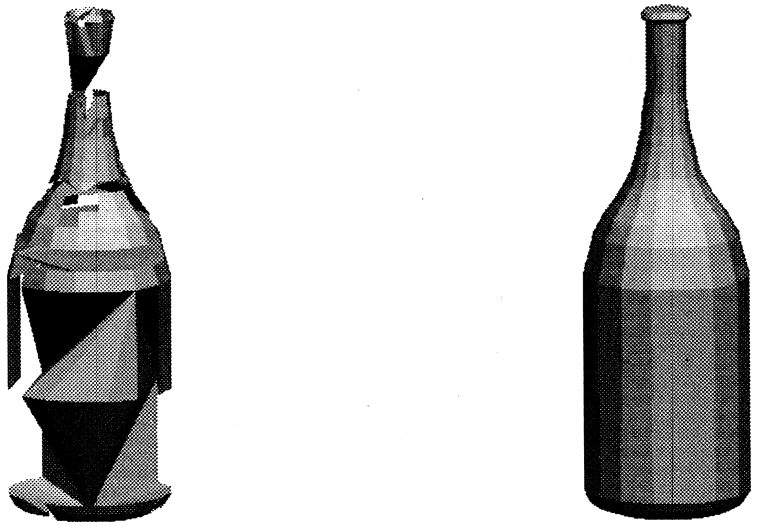

Figure 11: Left: result of the Voronoi skeleton growth; right: result of the γ-based constriction.

actually are none. Another condition is that the object is not too much folded, such that the points are not visible from outside the convex hull of the object. A 2D example from [O'Rourke et al., 87] is shown in figure 10. The point set on the left suggests a circularly formed object like the picture in the middle, corresponding to the shortest Voronoi skeleton. Our constriction algorithm produces the serrated boundary on the right, which is right if all points are actually visible from outside the (convex hull of the) object. All other example point sets from [O'Rourke et al., 87] give intuitively expected boundaries.

In 3D the shortest Voronoi skeleton does not give good results for our test point sets. An example is shown on the left side of figure 11. Although all points lie on the resulting boundary, the body of the object is not filled properly, because the object does not precisely correspond to a skeleton. This is the case for most 3D objects.

In 2D the minimal area change method seems to work as good as the γ-based criterion. In 3D however, the minimal area change algorithm is more often inclined to eat its way into the object because the tetrahedra are removed in the wrong order. Another difference is shown in figure 12. The γ-based selection criterion gives a smoother surface than the minimal area change criterion. In [Boissonnat, 84a] another criterion is mentioned, which seems to be effectively the same as our γ-based criterion, except that we take the sum of the values of both triangles when the tetrahedron has two triangles on the boundary.

We have compared the running time performance of the three methods, when applied to the mask data set, leaving all disk i/o out of consideration. The minimal area change constriction algorithm spends much time on calculating the criterion value. Since the γ-values are already computed while constructing the γ-graph, the only thing our algorithm has to compute for the criterion value, is the sign of the γ-value according to the rule as given in section 4. This leads to a speed-up of approximately 17%. Both constriction algorithms take about four seconds on a Sun SparcStation 1+, while the Voronoi skeleton growth algorithm takes about two and a half hours.

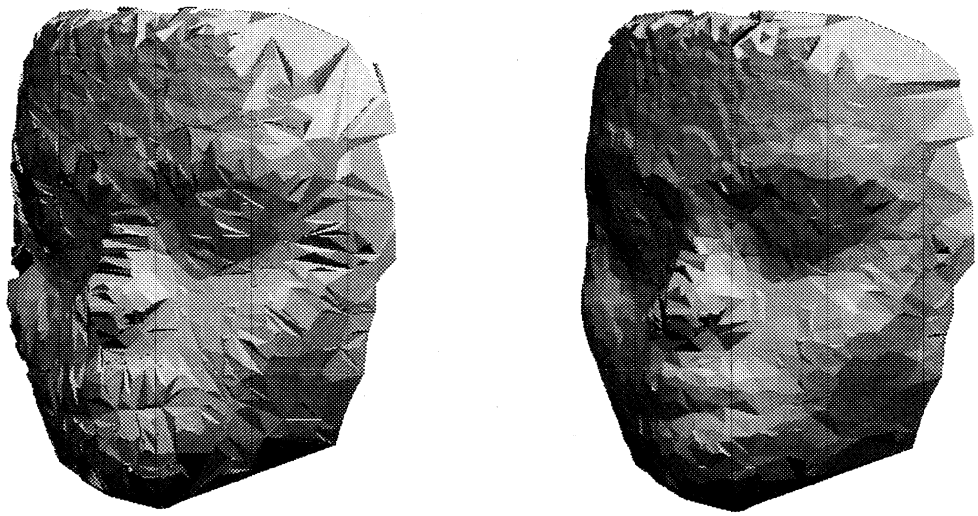

Figure 12: Left: result of the minimal area-change criterion; right: result of the γ-based criterion.

7. *Concluding remarks*

The reconstruction method presented results in a volumetric object description (consisting of tetrahedra), which defines an unambiguous boundary. The triangulation of the body of the object allows easy calculation of properties such as volume and mass. The triangulation also corresponds to a skeleton, see subsection 2.7. In the case that a $\gamma([-1,1],[\gamma_0,1])$, $\gamma_0 < 0$ is to be used because the $\gamma([-1,1],[0,1])$ does not suffice, redundant overlapping internal simplices can be removed to obtain a triangulation.

The input points are assumed to lie on the boundary of an object without holes, but an inner contour or surface can be handled separately. The triangulation of the outer boundary does not correspond anymore with the body of the object, but for the calculation of some properties the value for the inner boundary can be subtracted from the value for the outer boundary. A subject of further research is the case of objects with handles.

One of the attractions of our constriction method is that it is essentially the same for 2D as for 3D. It does also not depend on data such as Gaussian curvature, which is typically not available directly, but must be estimated. Informally speaking, the objects reconstructed from the γ-neighborhood graph have correct and smooth boundaries when compared to other methods, as has been shown by several examples.

References

[Ahuja, 82] Ahuja, N. Dot pattern processing using Voronoi neighborhoods. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-4(3), 1982, 336 – 343.

[Alevizos et al., 87] Alevizos, P. D., Boissonnat, J. D., and Yvinec, M. An optimal $o(n \log n)$ algorithm for contour reconstruction from rays. In *Poceedings of the 3rd ACM Symposium on Computational Geometry*, 1987, New York. ACM Press, 162 – 170.

[Boissonnat, 82] Boissonnat, J.-D. Representation of objects by triangulating points in 3-D space. In *Proceedings 6th International Conference on Pattern Recognition*, 1982, 830 – 832.

[Boissonnat, 84a] Boissonnat, J.-D. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4), 1984, 266 – 286.

[Boissonnat, 84b] Boissonnat, J. D. Representing 2D and 3D shapes with the Delaunay triangulation. In *Proceedings 7th International Conference on Pattern Recognition*, 1984, 745 – 748.

[Boissonnat, 88] Boissonnat, J.-D. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, 44, 1988, 1 – 29.

[Choi et al., 88] Choi, B. K., Shin, H. Y., Yoon, Y. I., and Lee, J. W. Triangulation of scattered data in 3D space. *Computer Aided Design*, 20(5), 1988, 239 – 248.

[Christiansen and Sederberg, 78] Christiansen, H. N. and Sederberg, T. W. Conversion of complex contour line definitions into polygonal element mosaics. *Computer Graphics*, 13(2), 1978, 187 – 192.

[Dillencourt, 87] Dillencourt, M. B. A non-Hamiltonian, nondegenerate Delaunay triangulation. *Information Processing Letters*, 25(3), 1987, 149 – 151.

[Dillencourt, 89] Dillencourt, M. B. An upper bound on the shortest exponent of inscribable polytopes. *Journal of Combinatorial Theory, Series B*, 46(1), 1989, 66 – 83.

[Edelsbrunner et al., 83] Edelsbrunner, H., Kirkpatrick, D. G., and Seidel, R. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, IT-29(4), 1983, 551 – 559.

[Fuchs et al., 77] Fuchs, H., Kedem, Z. M., and Uselton, S. P. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10), 1977, 683 – 702.

[Gabriel and Sokal, 69] Gabriel, K. R. and Sokal, R. R. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18, 1969, 259 – 278.

[Ganapathy and Dennehy, 82] Ganapathy, S. and Dennehy, T. G. A new general triangulation method for planar contours. *Computer Graphics*, 16(3), 1982, 69 – 75.

[Glassner, 89] Glassner, A. S., editor. *An Introduction to Ray-Tracing.* 1989, Academic Press.

[Herron, 85] Herron, G. Smooth closed surfaces with discrete triangular interpolants. *Computer Aided Geometric Design*, 2(4), 1985, 297 – 306.

[Huijsmans, 83] Huijsmans, D. P. Closed 2D contour algorithms for 3D reconstruction. In ten Hagen, P. J. W., editor, *proceedings Eurographics '83*, 1983, 157 – 168.

[Jain and Jain, 90] Jain, R. C. and Jain, A. K., editors. *Analysis and Interpretation of Range Images.* 1990, Springer-Verlag.

[Keppel, 75] Keppel, E. Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development*, 19(1), 1975, 2 – 11.

[Kirkpatrick and Radke, 85] Kirkpatrick, D. G. and Radke, J. D. A framework for computational morphology. In Toussaint, G. T., editor, *Computational Geometry*, 1985. Elsevier Science Publishers, 217 – 248.

[Lee and Schachter, 80] Lee, D. T. and Schachter, B. J. Two algorithms for constructing the delaunay triangulation. *International Journal of Computers and Information Science*, 9(3), 1980, 219 – 242.

[Medek, 81] Medek, V. On the boundary of a finite set of points in the plane. *Computer Vision, Graphics, and Image Processing*, 15, 1981, 93 – 99.

[O'Rourke, 81] O'Rourke, J. Polyhedra of minimal area as 3D object model. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981, 664 – 666.

[O'Rourke, 86] O'Rourke, J. The computational geometry column. *Computer Graphics*, 20(5), 1986, 232 – 234.

[O'Rourke et al., 87] O'Rourke, J., Booth, H., and Washington, R. Connect-the-dots: A new heuristic. *Computer Vision, Graphics, and Image Processing*, 39, 1987, 258 – 266.

[QuaVis, 90] QuaVis. AAAI-90 workshop on qualitative vision. 1990.

[Rioux and Cournoyer, 88] Rioux, M. and Cournoyer, L. The NRCC three-dimensional image data files. Technical report, 1988, National Research Council Canada.

[Toussaint, 80] Toussaint, G. T. Pattern recognition and geometrical complexity. In *Proceedings of the 5th International Conference on Pattern Recognition*, 1980, 1324 – 1347.

[Toussaint, 88] Toussaint, G. T., editor. *Computational Morphology - A Computational Geometric Approach to the Analysis of Form*, volume 6 of *Machine Intelligence and Pattern Recognition.* 1988, North-Holland, Amsterdam.

[Tutte, 77] Tutte, W. T. Bridges and Hamiltonian circuits in planar graphs. *Aequationes Mathematicae*, 15, 1977, 1 – 33.

[Veltkamp, 88] Veltkamp, R. C. The γ-neighbourhood graph for computational morphology. In *Computing Science in the Netherlands '88*, 1988, 451 – 462.

[Veltkamp, 89] Veltkamp, R. C. 2D and 3D computational morphology on the γ-neighborhood graph. *Acta Stereologica*, 8(2/2), 1989, 595 – 600.

[Veltkamp, 90a] Veltkamp, R. C. The Flintstones representation and approximation scheme. In *Computing Science in the Netherlands '90*, 1990, 485 – 498.

[Veltkamp, 90b] Veltkamp, R. C. The γ-neighborhood graph. Technical Report CS-R9034, 1990, CWI.