GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 57, No. 6, November, pp. 441–452, 1995

Boundaries through Scattered Points of Unknown Density

REMCO C. VELTKAMP*

CWI, Department of Interactive Systems, Amsterdam, The Netherlands; and Eindhoven University of Technology, Department of Computing Science, Eindhoven, The Netherlands

Received March 2, 1993; revised July 13, 1995; accepted July 14, 1995

This paper presents a method for the (re)construction of a simple closed 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 through-passages and inner voids. No a priori knowledge about any topological relation between the points, or additional information such as sample density or normal vectors, is used. The construction technique is based on a parameterized geometric graph, the γ-Neighborhood Graph. The hull of the γ-Neighborhood Graph is iteratively constricted, exploiting geometric information incorporated in the graph. This constriction technique provides a uniform approach for 2D and 3D. © 1995 Academic Press, Inc.

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 boundary. Some boundary of the point set, however, defining an unambiguous object model is often needed for further processing. This paper presents a method to (re)construct such a boundary.

Points on the boundary of an object can be obtained in a variety of ways. If a set of points from a 2D object is given in a sequential order along the boundary, for example obtained by tracing, the points form a contour chain representing the boundary curve. If a set of points from a 3D object is given by a pile of contours, for example obtained from parallel object cross sections, the points on pairs of consecutive contours can be joined so as to generate ribbons of triangles which in turn form a closed boundary surface. There are numerous data sources that do not yield such a clear structural relation between the points in the data set:

• A typical laser range system measures 3D points on the surface of an object by emitting a laser beam at certain x,y-coordinates and inferring the corresponding z-coordinates [21]. Parts of the surface can be hidden and become visible only after rotating the object or moving around it, because of shadowing by self-occlusion [6]. A horizontal scan of points will thus generally not yield points that are successively adjacent on the surface. So, the order in which the points are acquired provides no adjacency relation.

• 3D coordinates of boundary points can be computed from stereographic images, for example, X-ray images, or from a sequence of images of a moving object [22]. In both cases the set of calculated coordinates generally provides no topological relation between the points.

• Image processing techniques for low-level computer vision can extract feature data such as object corners from images [14], which can be converted into coordinates; however, information such as the topological relation between the points are not obtained by this low level processing.

If there is no structural relation between the points, or no such information is available because the data source is not known, or if a single boundary construction method is to be used for data from different sources, no structural relation between the points may be assumed. The only a priori knowledge is that they lie on the boundary of an object. This paper presents a method for the construction of a simple closed 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 through-passages and inner voids, but no assumptions about geometrical relations between the points is made. Additionally, no a priori knowledge such as sample density, curvature, or normal vectors is used.

In the following, N_v will denote the number of vertices, N_e the number of edges, and N_t the number of triangles under consideration.

1.1. *Statement of the Problem*

The problem stated so far, "construct an object boundary through a set of boundary points," is ill-stated. To get a better statement we have to make some restrictions. First of all, we restrict the constructed object to have no through-

* Current address: Department of Computing Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. E-mail: remco.veltkamp@cs.ruu.nl.

passages and inner voids. So, a 2D object must be bounded by a single contour, it has no inner contour; the boundary must be topologically equivalent to a circle. A 3D object must be bounded by a single surface and may not have through-passages (like a torus); the boundary must be topologically equivalent to a sphere. Furthermore, we restrict the boundary to be piecewise linear. So, a 2D boundary must consist of line segments and a 3D boundary of flat polygons. Since a triangle is the only n-gon through n vertices in 3D that is guaranteed to be flat, the class of 3D boundaries is further restricted to consist of triangles.

The 2D boundary construction problem is formally stated as follows:

BOUNDARY CONSTRUCTION 2D. *Given a set of vertices in 2D, find a simple closed polygon through all vertices.*

By definition, a simple polygon is topologically equivalent to a circle, and therefore a valid boundary of an object. In graph-theoretical terms, a simple polygon through all vertices is a Hamilton cycle. I will occasionally call a Hamilton cycle a "Hamilton polygon."

The 3D boundary construction problem is formally stated as follows:

BOUNDARY CONSTRUCTION 3D. *Given a set of vertices in 3D, find a simple closed polyhedron of triangular faces through all vertices.*

By definition, a simple polyhedron is topologically equivalent to a sphere, and thus a valid object boundary. I will call a simple polyhedron through all vertices a "Hamilton polyhedron." Note that this is not a 3D Hamilton cycle of edges.

The boundary construction problems are clearly underconstrained, so that a solution is not unique. Indeed, a set of vertices is an ambiguous boundary representation. In 3D, even a wire frame model (where we are given edges in addition to the vertices) is still ambiguous. Some criterion is needed to select the boundary that is considered the best among all solutions; however, there is no known algorithm that generates all solutions efficiently.

A brute force algorithm for 2D, where a boundary must consist of $N_e = N_v$ edges, takes all combinations of N_v edges out of all $\binom{N_v}{2}$ possible edges and tests whether each of the combinations is a simple polygon through all vertices. This gives a time complexity of

$$\Theta\left(\binom{\binom{N_v}{2}}{N_v}\right),$$

which is at least $\Omega(N_v^{N_v})$. In 3D, where a boundary must consist of $N_t = 2N_v - 4$ triangles, the analogous algorithm gives a time complexity of

$$\Theta\left(\binom{\binom{N_v}{3}}{2N_v - 4}\right),$$

which is at least $\Omega(N_v^{5N_v})$.

Such brute force approaches are clearly infeasible. It is logical to exploit some geometric relation between the vertices. In particular we can construct a graph on the set of vertices such that the existence of edges between the vertices depends on some chosen geometric relation. Such a graph is called a geometric graph. After construction of the geometric graph we may take advantage of the incorporated geometric property. As a first try one can think of a shortest distance property, as in the Nearest Neighbors Graph. Many edges in the Nearest Neighbors Graph can be part of a valid boundary, but the shortest distance does not always suitably correspond with the geodetic metric on the object boundary, specifically at highly curved parts. As a result, nearest neighbors need not be consecutive points on the boundary.

One can also think of other geometric graphs such as the Convex Hull, giving a rough approximation of a boundary, or the ubiquitous Delaunay Triangulation. An advantage of the Delaunay Triangulation is that it is a connected graph that contains not as few joins as, for example, the Euclidean Minimum Spanning Tree, and also not as many as the complete graph. Adding edges (triangles) to a 2D (3D) Delaunay Triangulation introduces overlapping triangles (tetrahedra). However, a brute force search in the Delaunay Triangulation for a collection of edges that forms a valid boundary is not feasible. After all, a Delaunay Triangulation in 2D consists of $3N_v - 6$ edges, giving a time complexity of $\Theta(\binom{3N_v-6}{N_v})$ which is $\Omega(3^{N_v})$. In 3D, the maximum number of triangles in the Delaunay Triangulation is $\Theta(N_v^2)$, giving a worst case complexity of $\Theta(\binom{N_v^2}{2N_v-4})$ which is $\Omega(N_v^{3N_v})$.

All the brute force approaches described require exponential time. In order to achieve a polynomial time complexity some heuristic is used that gives a single boundary. Besides, it is not sufficient to compute all possible simple boundaries: we need to find a good, or likely, one. The heuristic must be chosen so as to yield a boundary that is considered a likely boundary for the given set of vertices, among all possible solutions. This is clearly a subjective notion. Such heuristic approaches and qualitative descriptions of the visual environment receive growing interest in the computer vision community [20].

1.2. *Related Work*

Several methods expand some initial triangulation until all data points are included in the boundary polyhedron [2, 4, 5]. A drawback of these methods is that additional information is needed, for example the surface normal at the data points, but this information can be estimated from the point data [4] so that only the coordinates of the points are used. A heuristic algorithm that tries to find the polyhedron of minimal surface area is presented in [16]. This heuristic algorithm does not guarantee a surface within a fixed error fraction of the minimal area. Moreover, the minimal area polyhedron can be an unnatural object boundary [4]. The method in [18] tries to find the simple polygon in the Delaunay Triangulation that corresponds to the shortest tree, or skeleton, in the dual Voronoi Diagram. For objects that have no distinct skeleton this method gives strange results, especially for the generalization to 3D. The methods in [3, 4] take the 2D or 3D Delaunay Triangulation and successively delete segments from the hull until it passes through all data points. Not every nondegenerate Delaunay Triangulation contains a simple closed boundary through all data points. Moreover, both in 2D and in 3D this constriction process can get locked in the sense that no more segments can be deleted without yielding an invalid boundary, while not yet all data points lie on the current hull. A method that can handle objects with through-passages and inner voids is presented in [15]; however, the boundary need not exactly pass *through* the data points. The technique is dependent on an a priori known sample density.

Point clustering [11, 12], planar contour construction from rays [1], and surface construction from a pile of contours [13] or multiaxial contours [19] are related but different or simpler problems than we consider here.

The solution presented in this paper is also a constriction method, but will always find a simple closed boundary. The method is not dependent on a known sampling density, normal vectors, or curvature information. The constriction is performed on the γ-Neighborhood Graph, which is presented in Section 2. Section 3 explains the constriction process itself. Section 4 discusses the question of existence of a Hamilton polygon or polyhedron in the graph. Section 5 presents results of our technique and gives a comparison with some other methods. Implementation issues and the algorithmic complexity are treated in Section 6. Section 7 discusses constriction under certain constraints. Finally, Section 8 gives some concluding remarks.

2. THE γ-NEIGHBORHOOD GRAPH

The γ-Neighborhood Graph has been introduced in [25]. Here, we will consider one particular form. For explanatory purposes, let us consider the 2D Delaunay Triangulation on a set of points. The Delaunay Triangulation is a filling of the plane inside the Convex Hull of the point set by triangles with the following properties:

1. the vertices of each triangle are data points;
2. the disc touching the vertices of each triangle contains no other data point in its interior.

A disc that contains no data point in its interior is called empty. If no more than three data points lie on an empty disc, the Delaunay Triangulation is uniquely defined. Otherwise the triangulation on these specific points must be non-overlapping.

The Delaunay Triangulation can equivalently be defined as the collection of all edges that have an empty disc touching its vertices. For all these edges there are two largest possible empty discs touching their vertices, which either touch a third point or have an infinite radius. In this last situation the disc degenerates to an empty half-plane, and the edge lies on the Convex Hull. The radii of the two discs are a scaling of the radius r of the smallest possible disc touching the two vertices. These scaling factors are written as an expression $1/(1 - c)$, with $0 \leq c \leq 1$. The two radii are thus $r/(1 - c_0)$ and $r/(1 - c_1)$, with $0 \leq c_0$, $c_1 \leq 1$.

The Delaunay Triangulation is a particular instance of the γ-Neighborhood Graph, except that the latter incorporates the values of c_0 and c_1. This graph discriminates between the case that the centers of the discs lie at opposite sides of the edge and the case that they lie at the same side. In the latter case the parameter c_0 is taken negative. So, c_0 lies in the range $[-1, 1]$, defining a radius of $r/(1 - |c_0|)$. Parameter c_1 still lies in the range $[0, 1]$. The γ-Neighborhood Graph that coincides with the Delaunay Triangulation is denoted $\gamma([-1, 1], [0, 1])$.

For each edge in the Delaunay Triangulation the *union* of two discs touching its vertices is empty. The γ-Neighborhood Graph also considers the case that only the *intersection* of two discs is empty. In that case the parameter c_1 is taken negative. So, c_1 may lie in the range $[-1, 1]$, defining a radius of $r/(1 - |c_1|)$. The graph $\gamma([-1, 1], [d, 1])$ for some $d \in [-1, 0]$ is the Delaunay Triangulation plus all edges for which there are $c_0 \in [-1, 1]$ and $c_1 \in [d, 0]$ such that the intersection of two discs touching their vertices with radii $r/(1 - |c_1|)$ and $r/(1 - |c_1|)$ is empty. The smaller the value of d, the smaller the area of the intersection, and the more edges are included in the γ-Graph. The $\gamma([-1, 1], [d, 1])$, $d \in [-1, 0]$ will be used to novel fundamental effect: it remedies the fact that the Delaunay Triangulation need not always contain a Hamilton polygon or polyhedron.

For the 3D γ-Graph we consider three data points and two balls touching these points. The radii of the two balls are a scaling of the radius r of the smallest possible ball touching the three vertices: $r/(1 - |c_0|)$ and $r/(1 - |c_1|)$, with $-1 \leq c_0, c_1 \leq 1$. The 3D γ-Graph is a (hyper-) graph connecting three vertices by a triangle if the union or intersection of the two balls is empty.

As in 2D, the γ-Neighborhood Graph that coincides with the 3D Delaunay Triangulation (Tetrahedrization) is $\gamma([-1, 1], [0, 1])$. The graph $\gamma([-1, 1], [d, 1])$ for some $d \in [-1, 0]$ is the Delaunay Tetrahedrization plus all triangles for which there are $c_0 \in [-1, 1]$ and $c_1 \in [d, 0]$ such that the intersection of two balls touching their vertices with radii $r/(1 - |c_0|)$ and $r/(1 - |c_1|)$ is empty. The smaller the value of d, the smaller the volume of the intersection, and the more triangles are included in the γ-Graph. The γ-Graph further incorporates these values of c_0 and c_1.

Each $d \in [-1, 0]$ yields one specific γ-Graph from the whole spectrum of graphs $\gamma([\ldots], [\ldots])$. This spectrum unifies a number of geometric graphs such as the Convex Hull, the Delaunay Triangulation, and in 2D also the Gabriel Graph and the β-Skeleton, into a continuum that ranges from the void to the complete graph. The γ-Graph provides a geometric structure for point pattern analysis and can for example be used for geographics and network analysis. The graph $\gamma([-1, 1], [d, 1])$, $d \in [-1, 0]$ is the γ-Graph that we use for boundary construction.

A formal definition and analysis and many examples of the γ-Neighborhood Graph are given in [25]. The 2D $\gamma([-1, 1], [d, 1])$ can be constructed in $\Theta(N_v \log N_v)$ time for $d = 0$, which is optimal, and $\mathcal{O}(N_v^2)$ time for $d < 0$. The size of the graph is $\Theta(N_v)$ and $\mathcal{O}(N_v^2)$, respectively. In 3D, $\gamma([-1, 1], [d, 1])$ can be constructed in $\Theta(N_v^2)$ time for $d = 0$, which is optimal, and $\mathcal{O}(N_v^4)$ time for $d < 0$. The size is $\mathcal{O}(N_v^2)$ and $\mathcal{O}(N_v^3)$, respectively.

3. BOUNDARY CONSTRUCTION

In this section, we construct a boundary of a set of vertices by constricting a $\gamma([-1, 1], [c, 1])$, $c \in [-1, 0]$ of the input point set. The choice of a suitable value of c is discussed in Section 4. Note that the hull of any $\gamma([-1, 1], [c, 1])$, $c \in [-1, 0]$ is the Convex Hull. Constricting the hull of a graph is the process of deleting a hull segment from the graph in such a way that the boundary is properly defined (see below). A γ-Graph from which segments are deleted is not a γ-Graph any more, but is called a *pruned* γ-Graph. In order to find a boundary through all data points, a $\gamma([-1, 1], [c, 1])$, $c \in [-1, 0]$ is constricted on the basis of the geometric information in the graph, until all data points lie on the pruned graph hull.

3.1. *Boundary Polygon*

We first construct a simple closed polygon through all points of a given planar data set by constricting the 2D $\gamma([-1, 1], [c, 1])$, $-1 \le c \le 0$ on the point set.

If three edges in the graph implicitly form a triangle and one of the edges is a hull edge, the triangle is called a hull triangle. Our constriction process selects in each iteration the hull edge that is the best candidate for deletion. Note that deletion of an edge of a hull of a (pruned) γ-Graph must keep the boundary a single manifold, i.e., a simple polygon. Therefore, deletion is only allowed if the hull triangle vertex v_k opposite to a hull edge v_iv_j is not already

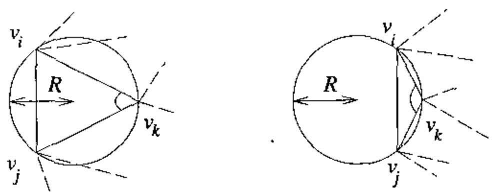

FIG. 1. Left: γ-indicator > 0. Right: γ-indicator < 0.

in the current boundary. A hull edge v_iv_j that satisfies this condition is called *removable* (with respect to v_k).

The selection of the next removable edge v_iv_j to be deleted is based on the observation that the interior vertex v_k of the hull triangle $v_iv_jv_k$ that has the largest angle $\angle(v_i, v_k, v_j)$ has the largest possibility to be seen or sensed from outside the hull. Additionally, the change of shape of the hull is small, relative to the size of the triangle.

The radius of the disc through v_i, v_j, v_k is denoted by $R(v_i, v_j, v_k)$, and is equal to $r(v_i, v_j)/(1 - |d|)$ for some $d \in [-1, 1]$. This value is used in the following definition.

DEFINITION 1 (γ-Indicator). *Let v_iv_j be an edge of an intermediate hull, let v_iv_k and v_jv_k be edges in a (pruned) γ-Graph, and let d be defined by $R(v_i, v_j, v_k) = r(v_i, v_j)/(1 - |d|)$. The γ-indicator of v_iv_j with respect to v_k is $|d|$ if the center of the circle through v_i, v_j, v_k lies at the same side of v_iv_j as v_k; is $-|d|$ if the center lies at the other side; and is zero if $d = 0$.*

The magnitude of the γ-indicator is calculated during construction of the γ-Graph, and is stored in it. The sign of the γ-indicator is positive if v_i, v_j, and v_k have the same orientation as v_i, v_j and the circle center C, that is, if

$$sign([v_i - v_k, v_j - v_k]) = sign([v_i - C, v_j - C]),$$

where "[ ]" denotes the determinant.

The more negative the γ-indicator, the closer v_k lies to v_iv_j, and the larger is angle $\angle(v_i, v_k, v_j)$; see Fig. 1. Note that the γ-indicator is independent of the size of the triangle. The selection rule based on the γ-indicator is the following:

SELECTION RULE. *Delete the removable hull edge that has the smallest γ-indicator.*

This selection criterion combines a local measure (the γ-indicator) and global information (the smallest value), and is orientation and scale independent.

Let us investigate the exact relation between the γ-indicator γ_{ind} and angle $\angle(v_i, v_k, v_j)$. According to the sine rule $\sin(\angle(v_i, v_k, v_j)) = r(v_i, v_j)/R(v_i, v_j, v_k)$. By definition of γ_{ind}, we have $r(v_i, v_j)/R(v_i, v_j, v_k) = 1 - |\gamma_{\text{ind}}|$. If $\gamma_{\text{ind}} \ge 0$, then $1 - \gamma_{\text{ind}} = r(v_i, v_j)/R(v_i, v_j, v_k)$, and if $\gamma_{\text{ind}} \le 0$, then $1 - \gamma_{\text{ind}} = 2 - r(v_i, v_j)/R(v_i, v_j, v_k)$. So, if

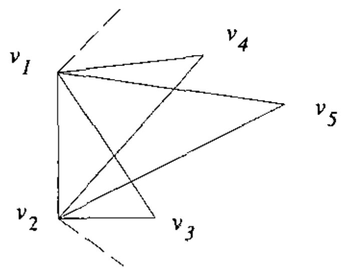

FIG. 2. Boundary edge in a 2D $\gamma([-1, 1], [c, 1])$, $c \in [-1, 0)$.

$\gamma_{\text{ind}} \geq 0$, then $\angle(v_i, v_k, v_j)$ increases when $r(v_i, v_j)$ $R(v_i, v_j, v_k)$ increases; if $\gamma_{\text{ind}} \leq 0$, then $\angle(v_i, v_k, v_j)$ increases when $2 - r(v_i, v_j)/R(v_i, v_j, v_k)$ increases. The largest value of $1 - \gamma_{\text{ind}}$ is obtained for the smallest value of γ_{ind}, leading to the selection criterion stated above.

Consider a hull edge v_1v_2 as in Fig. 2, with hull triangles $v_1v_2v_3$, $v_1v_2v_4$, and $v_1v_2v_5$. In this example, the γ-indicator with respect to v_3 is smaller than those with respect to v_4 and v_5. If v_1v_2 is selected for deletion because it has the smallest γ-indicator of all removable hull edges, the edges v_2v_4 and v_2v_5 must also be deleted in order to maintain a single manifold boundary. If v_1v_2 is not removable with respect to v_3 but is removable with respect to v_4, and v_1v_2 is selected for deletion due to the γ-indicator with respect to v_4, then v_1v_3 and v_2v_5 must be deleted. In general, if hull edge v_iv_j is deleted due to the γ-indicator with respect to v_k, any edge crossing $v_iv_jv_k$ must also be deleted.

Selection and deletion of a hull edge is repeated until all vertices are part of the boundary polygon. The time complexity of the entire constriction algorithm depends on the way we keep track of the removable hull edges and their value of the γ-indicator. Implementational and complexity issues are discussed in Section 6.

Figure 3 illustrates the constriction algorithm on a set of vertices from the silhouette of Ucello's chalice, which serves as the cover picture of the journal *Computer Aided Geometric Design* [23].

3.2. *Boundary Polyhedron*

In this section, we construct a simple closed polyhedron through all points of a given 3D data set by constricting the 3D $\gamma([-1, 1], [c, 1])$, $-1 \leq c \leq 0$ on the point set. If four triangles implicitly form a tetrahedron, and one of the triangles is a hull triangle, the tetrahedron is called a hull tetrahedron. A hull tetrahedron can have one, two, or three hull triangles.

Let us first consider tetrahedra with exactly one hull triangle. The selection of the next triangle $v_iv_jv_k$ to be deleted is based on the observation that the opposite vertex v_ℓ of the tetrahedron $v_iv_jv_kv_\ell$ that has the largest solid angle φ has the largest probability to be sensed from outside the hull, for example by a laser beam. Additionally, the change of shape of the hull is then small, relative to the size of the tetrahedron.

Let a hull triangle $v_iv_jv_k$ be incident to hull tetrahedron $v_iv_jv_kv_\ell$. The radius of the ball touching v_i, v_j, v_k, and v_ℓ is denoted by $R(v_i, v_j, v_k, v_\ell)$. The selection rule is based on the notion of γ-indicator:

DEFINITION 2 (γ-Indicator). *Let $v_iv_jv_k$ be a triangle in a (pruned) γ-Graph hull, let $v_iv_jv_\ell$, $v_jv_kv_\ell$, and $v_iv_kv_\ell$ be triangles in the graph, and let d be defined by $R(v_i, v_j, v_k, v_\ell) = r(v_i, v_j, v_k)/(1 - |d|)$. The γ-indicator of $v_iv_jv_k$ with respect to v_ℓ is $|d|$ if the center of the sphere through v_i, v_j, v_k, v_ℓ lies at the same side of $v_iv_jv_k$ as v_ℓ; is $-|d|$ if the center lies at the other side; and is zero if $d = 0$.*

The magnitude of the γ-indicators are calculated during construction of the γ-Graph and are stored in it. The sign of the γ-indicator is positive if v_i, v_j, v_{nk}, and v_ℓ have the same orientation as v_i, v_j, v_k, and the sphere center C, that is, if

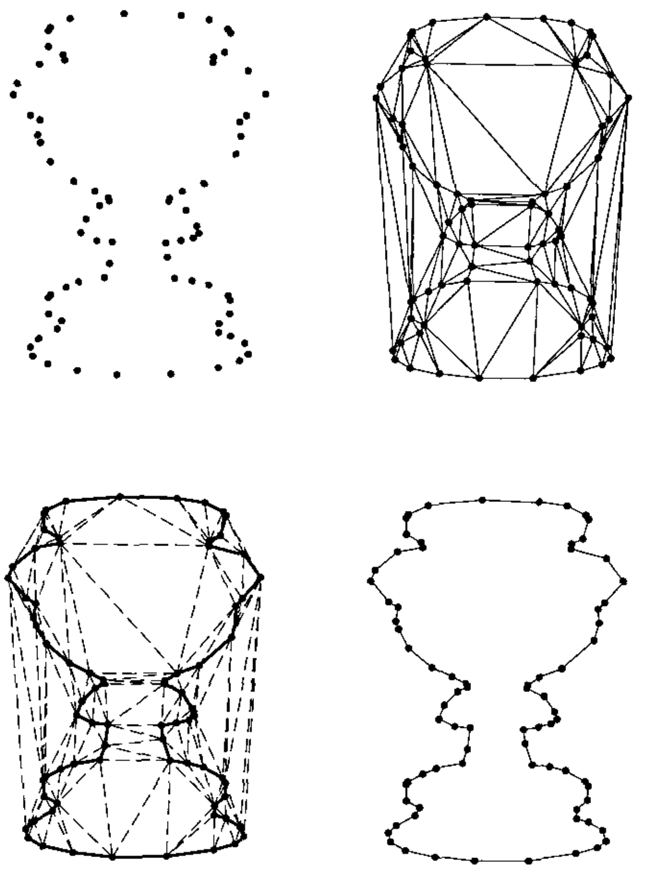

FIG. 3. Boundary construction example in 2D. Top left: 77 vertices. Top right: the corresponding $\gamma([-1, 1], [0, 1])$. Bottom left: $\gamma([-1, 1], [0, 1])$ with the constructed boundary. Bottom right: the constructed boundary.

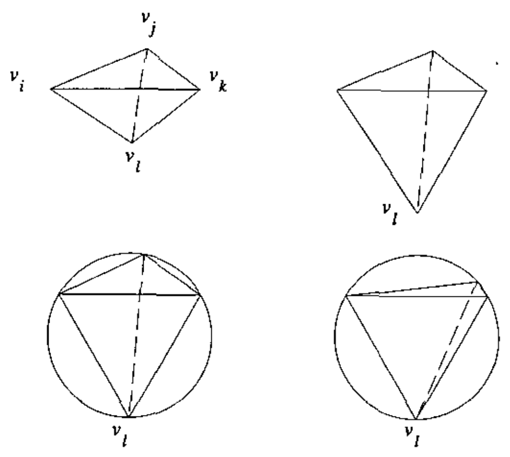

FIG. 4. The solid angle at v_ℓ in the left column is larger than that in the right column. Top row: fixed triangle $v_iv_jv_k$. Bottom row: fixed γ-indicator.

$$\begin{aligned} sign([v_i - v_\ell, v_j - v_\ell, v_k - v_\ell]) \\ = sign([v_i - C, v_j - C, v_k - C]), \end{aligned}$$

where "[ ]" denotes the determinant.

There is no "3D sine rule" relating the solid angle φ at v_ℓ to $R(v_i, v_j, v_k, v_\ell)$ and $r(v_i, v_j, v_k)$. We observe, however, that φ depends on how close v_ℓ lies to $v_iv_jv_k$ relative to the size of the tetrahedron, and on the shape of $v_iv_jv_k$. Observe that $r(v_i, v_j, v_k)/R(v_i, v_j, v_k, v_\ell)$ is independent of the size of the tetrahedron. If the value of the γ-indicator $\gamma_{\text{ind}} \leq 0$, then $r(v_i, v_j, v_k)/R(v_i, v_j, v_k, v_\ell) = 1 - \gamma_{\text{ind}}$, and if $\gamma_{\text{ind}} \geq 0$, then $2 - r(v_i, v_j, v_k)/R(v_i, v_j, v_k, v_\ell) = 1 - \gamma_{\text{ind}}$, provided that $v_iv_jv_k$ is fixed. So, the larger $1 - \gamma_{\text{ind}}$, the larger φ; i.e., the wider the solid angle at v_ℓ. On the other hand, if $1 - \gamma_{\text{ind}}$ is fixed and the shape of $v_iv_jv_k$ varies, then φ increases when the area A of $v_iv_jv_k$ increases; see Fig. 4. Since A/R^2 is independent of the size of the tetrahedron, it seems obvious to use a selection criterion based on both $1 - \gamma_{\text{ind}}$ and A/R^2. However, it appears that using $1 - \gamma_{\text{ind}}$ alone gives better results. Indeed, a typical Convex Hull contains many triangles of small area (see, for example, Figs. 5 and 7), and they should be deleted to obtain a good boundary polyhedron.

Let us now consider a tetrahedron with exactly two hull triangles. All four vertices now lie on the hull, so deletion of the two triangles does not add a new vertex to the boundary. This can result in an extra hull tetrahedron, and moreover, deletion of the tetrahedron gives the two vertices opposite to the hull triangles "more air," enlarging the probability that they are sensed from these directions. Because the solid angles at the two vertices bound two nonoverlapping parts of space, it is obvious to sum the values $1 - \gamma_{\text{ind}}$ of both hull triangles in the selection rule.

Deletion of a hull triangle from the graph must leave the boundary properly defined, i.e., a single manifold boundary. Therefore, if a hull triangle $v_iv_jv_k$ is removed due to the γ-indicator with respect to a vertex v_ℓ, any triangle crossing $v_iv_jv_kv_\ell$ must also be removed. The boundary polyhedron must also remain simple. Thus, deletion of three hull triangles of a hull tetrahedron is never allowed, deletion of two hull triangles $v_iv_jv_k$, $v_jv_kv_\ell$ is only allowed if the edge v_iv_ℓ is not already in the current hull (in which case they are called removable with respect to v_iv_ℓ), and deletion of one hull triangle $v_iv_jv_k$ is only allowed if the opposite vertex v_ℓ is not already in the current hull (in which case it is called removable with respect to v_ℓ).

Since a large value of $1 - \gamma_{\text{ind}}$ is equivalent to a small value of γ_{ind}, the selection rule that captures both the tetrahedra with exactly one and with two hull triangles then becomes:

SELECTION RULE. *Delete those removable triangles (of a single tetrahedron) that have the smallest sum of γ-indicators.*

Again, this selection criterion combines a local measure (the γ-indicator) and global information (the smallest value), and is orientation and scale independent.

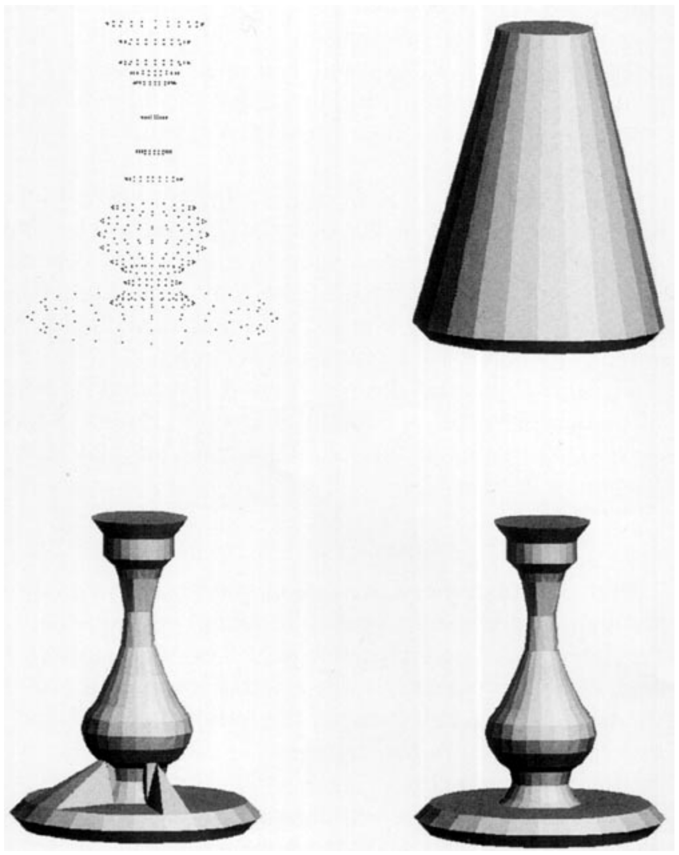

FIG. 5. Synthetic candlestick object. Top left: 481 vertices in 3D. Top right: Convex Hull, consisting of 73 vertices and 142 triangles. Bottom row: both boundaries consist of 481 vertices and 958 triangles.

Figure 5 shows how a synthetic set of 3D vertices modeling a candlestick is processed. Facets that look rectangular are actually two triangles. The bottom left picture shows the result of the constriction algorithm when stopped as soon as all points lie on the boundary. In this case, there are still removable triangles. Especially for such artificial objects there is no general rule telling how long to continue the same constriction procedure when already all vertices lie on the boundary. That decision is typically made interactively by the user. The right picture shows the resulting object after removing just enough triangles.

4. HAMILTONICITY

The constriction process on a $\gamma([-1, 1], [c, 1])$ for some fixed $c \in [-1, 0]$ may stop without finding a Hamilton polygon or polyhedron. Moreover, not every graph contains one. Results from graph theory on Hamiltonicity apply to Hamilton cycles, not to Hamilton polyhedra. For example, graph theory surveys the conditions a graph should satisfy to contain a Hamilton cycle, or how efficiently a Hamilton cycle can be found. A very general result is that a graph contains a Hamilton cycle if every vertex has at least $N_v/2$ neighbors [10]. More strict conditions often apply to planar graphs. For example, every four-connected, and thus five-connected, planar graph is Hamiltonian [24]. (A graph is n-connected if there are n different paths between any two distinct vertices, or equivalently, if the removal of any $n - 1$ vertices leaves the graph connected.) A special planar graph is a planar triangulation, which is at least two-connected and at most five-connected. A particular planar triangulation is the 2D Delaunay Triangulation, so that we conclude that every four- and five-connected 2D Delaunay Triangulation is Hamiltonian. For a long time it has been unknown whether or not two- and three-connected nondegenerate Delaunay Triangulations (and thus $\gamma([-1, 1], [0, 1])$s) are always Hamiltonian [17]. A two-connected counterexample was given in [7], which is shown in Fig. 6, and a three-connected one in [8]. So, a 2D nondegenerate $\gamma([-1, 1], [0, 1])$ need not be Hamiltonian, and in any case, the constriction process can be unsuccessful.

It is not yet known whether every nondegenerate Delaunay Tetrahedrization contains a simple closed polyhedron through all data points. However, the constriction process can get locked in the sense that no more triangles can be deleted without yielding an invalid boundary, while not yet all data points lie on the current hull.

If we vary $c \in [-1, 0]$, then $\gamma([-1, 1], [c, 1])$ gives a whole spectrum of γ-Graphs. For a c sufficiently smaller than $d < 0$, graph $\gamma([-1, 1], [c, 1])$ contains more edges (2D) or triangles (3D) than $\gamma([-1, 1], [d, 1])$, implicitly forming more overlapping triangles (see Fig. 6) or tetrahedra. So, for a c sufficiently small, $\gamma([-1, 1], [c, 1])$ is Hamiltonian. After all, $\gamma([-1, 1], [-1, 1])$ is the complete graph. There is no way of telling in advance for which value of c the graph $\gamma([-1, 1], [c, 1])$ will be Hamiltonian, or constriction will not get stuck. The value should be guessed.

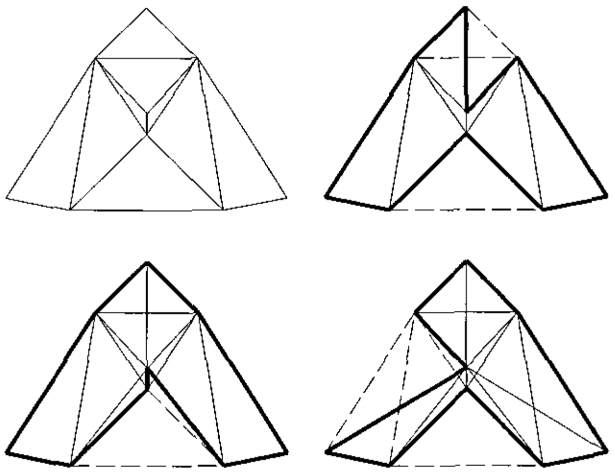
FIG. 6. Top left: a 2D two-connected non-Hamiltonian Delaunay Triangulation. Top right: the corresponding $\gamma([-1, 1], [-0.1, 1]$. Bottom left: $\gamma([-1, 1], [-0.2, 1])$. Bottom right: $\gamma([-1, 1], [-0.3, 1])$. The three γ-Graphs show the Hamilton cycle found by our constriction algorithm in fat lines, and the deleted edges in dashed lines.

The smallest γ-indicator of hull faces in a $\gamma([-1, 1], [c, 1])$ can be smaller than in a $\gamma([-1, 1], [d, 1])$ for $-1 \leq c < d < 0$, providing better choices for deletion. This is illustrated in Fig. 6, where some of the extra boundary triangles have larger angles at the interior vertex. Indeed, constriction of $\gamma([-1, 1], [-0.3, 1])$ gives a different Hamilton polygon than $\gamma([-1, 1], [-0.2, 1])$ and $\gamma([-1, 1], [-0.1, 1])$, although $\gamma([-1, 1], [-0.1, 1])$ is already Hamiltonian.

It turns out, however, that for practical situations constriction of $\gamma([-1, 1], [0, 1])$ works fine, and results in a Hamiltonian polygon or polyhedron. Indeed, the examples in the next section are based on $\gamma([-1, 1], [0, 1])$.

5. RESULTS

Figure 3 already showed a 2D example, and Fig. 5 a 3D one. The fact that this synthetic object is an object of revolution does not help the algorithm, because the algorithm is completely local and greedy. Figure 7 illustrates the constriction process performed on points from the surface of an Indian mask from the Man Museum in Ottawa, Canada, measured by a laser-range system [21]. The mask is truly 3D, because it has a few points added at the back (which nicely illustrates that the points may be distributed with nonuniform density).

Once we have the γ-Graph constriction algorithm, the "Delaunay Triangulation constriction method" [3] and the

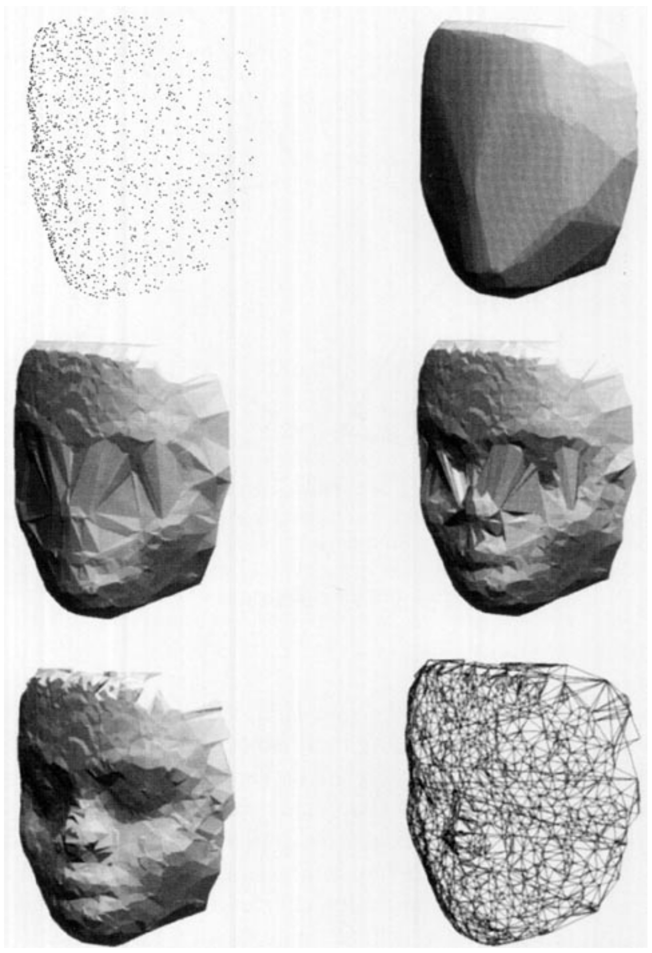

FIG. 7. Indian mask reconstructed from 3D laser-range data points. Top left: 1468 scattered vertices. Top right: Convex Hull consisting of 255 vertices and 504 triangles. Middle left: intermediate hull, 1019 vertices and 2034 triangles. Middle right: intermediate hull, 1337 vertices and 2670 triangles. Bottom left: final boundary, 1468 vertices and 2930 triangles. Bottom right: wire frame of the final boundary.

"Voronoi skeleton method" [18] are easily implemented, because both methods are based on the Delaunay Triangulation, that is, the $\gamma([-1, 1], [0, 1])$. It turns out that in 2D all three methods often give the same result. A set of vertices that gives different results, taken from [18], is shown in Fig. 8. We see that if the original object is very curled, it is not likely that the vertices are sensed from

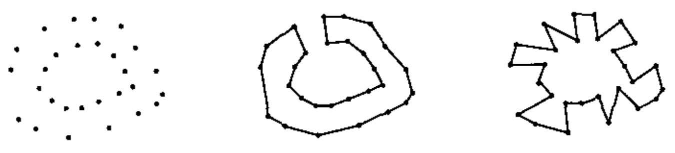

FIG. 8. Left: set of 2D vertices from [18]. Middle: Voronoi skeleton result. Right: constriction result.

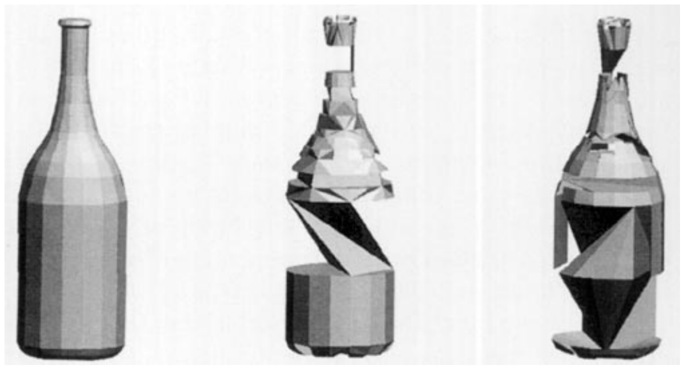

FIG. 9. Comparing three methods on a set of vertices from a bottle's surface. Left: γ-indicator method. Middle: minimal area change method. Right: Voronoi skeleton method.

some distance of the object, and the γ-indicator provides no proper heuristic. Conversely, if the original object does not have a clear skeleton, the Voronoi skeleton method used the wrong heuristic. For all other example sets of vertices from [18], the constriction algorithms give intuitively expected boundaries.

In 3D, the three methods often give different results. The Delaunay Triangulation constriction is sometimes inclined to sculpture its way into the object because the hull triangles are deleted in the wrong order, see Fig. 9 (middle), although for many other data sets [3, 4] the results are similar to those of the γ-Graph constriction method. The mask data set suggests that constriction based on the γ-indicator may yield a smoother boundary (see Fig. 10, right column).

The Voronoi skeleton method is not suitable for many 3D objects, because there often is no clear 3D object skeleton, or at least not one that corresponds to a Voronoi skeleton. Figures 9 and 10 show examples of this phenomenon. Although all vertices lie on the resulting boundary, the body of the object is not filled properly.

However, since the boundary reconstruction problem is underconstrained, there cannot be a single best algorithm for all cases. Small-scale experiments are not sufficient to draw definite conclusions.

Recall that Fig. 6 gives an example of more theoretical interest. The example shows that the graph $\gamma([-1, 1], [c, 1])$, $c \in [-1, 0]$, can be used to find a boundary even if the Delaunay Triangulation contains no Hamilton polygon.

6. IMPLEMENTATION AND COMPLEXITY

The complexity of the constriction algorithm depends on the implementation, and in particular on the data structures. In 2D, the γ-Graph is edge-based, so the edges are stored explicitly. The edges incident to a vertex are ordered around that vertex. In 3D, the γ-Graph is triangle-oriented

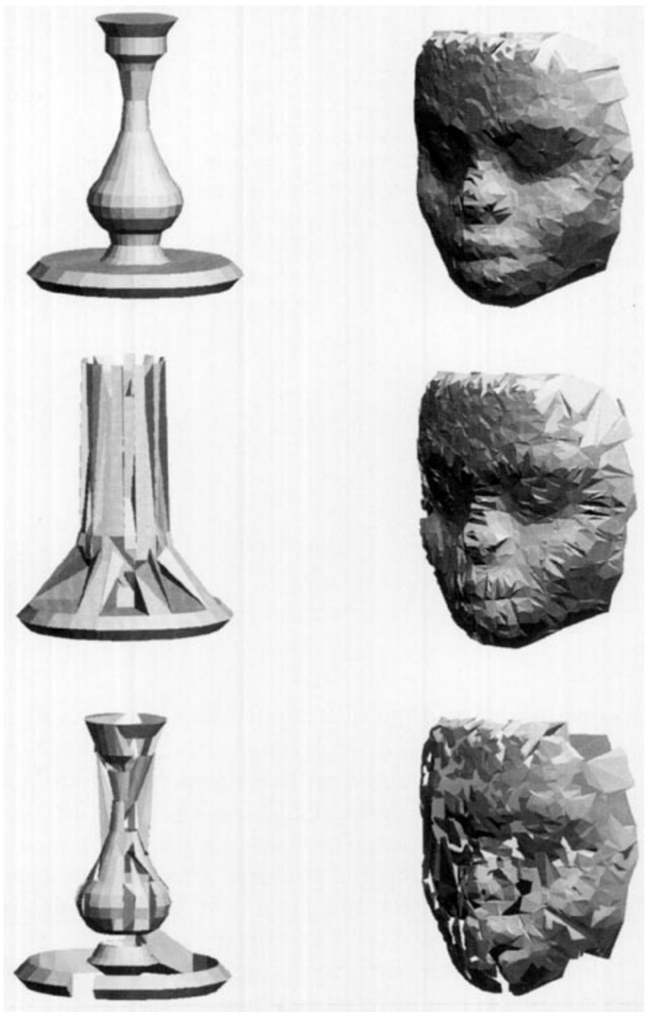

FIG. 10. Comparing three methods on the candlestick (left column) and mask (right column) data set. Top: γ-indicator method. Middle: Delaunay Triangulation constriction method. Bottom: Voronoi skeleton method.

and so the triangles are stored explicitly. Edges are also stored explicitly, and the triangles incident to an edge are ordered around that edge.

With these data structures, both triangles and edges can be addressed in constant time, in particular to check whether they lie on an intermediate hull. Given a hull face in a pruned $\gamma([-1, 1], [0, 1])$, the γ-indicator can be computed in constant time. Given a hull face in a pruned $\gamma([-1, 1], [c, 1])$, $c < 0$, the smallest γ-indicator can be calculated in $\mathcal{O}(m)$ time, where m is the number of faces incident to the hull face, which is $\mathcal{O}(N_v)$.

In order to keep track of the hull faces and their γ-indicators, they are stored in a heap structure sorted on increasing γ-indicator value. In 2D, the root of the heap contains the hull edge that has the smallest γ-indicator, in 3D it contains the hull triangles of a single tetrahedron that have the smallest sum of γ-indicators. Fetching the hull face and revalidating the heap takes $\mathcal{O}(n \log n)$ time for a heap of n elements. In order to keep the heap of size $\mathcal{O}(N_v)$ in the case of a $\gamma([-1, 1], [c, 1])$, $c < 0$, only the smallest γ-indicator or sum of γ-indicators of each hull face is stored, not the values of the overlapping edges or triangles.

Having mentioned the basics, Algorithm 1 shows the constriction algorithm in pseudo C-language code (the nature of the algorithm is the same as the one in [4]). The heap is initially filled with the removable faces on the Convex Hull and their γ-indicator value (line 2); the hull vertices, edges, and (in 3D) triangles are marked to lie on the boundary, to facilitate the test whether a face is removable. As long as not all the vertices are on the boundary and the heap is not empty (line 3), the face in the root of the heap is taken (line 4), involving revalidating the heap. Although each face is removable at the time it is inserted into the heap, the check in line 5 is necessary since a face can have become unremovable due to deletion of other faces. If deletion is allowed, then the face is deleted (line 6), involving the deletion of overlapping faces in the case of a $\gamma([-1, 1], [c, 1])$, $c < 0$ in order to get an unambiguous boundary. If necessary, the new boundary vertex, edges, and triangles (in 3D), are marked and N_{hv} (the number of vertices on the hull) is incremented when appropriate. Each new hull face (line 7) is inserted into the heap with its γ-indicator value, if its removal is allowed (line 8). The final boundary can be extracted from the graph (line 9), if desired.

Let us analyze the time complexity of the algorithm for five different cases: the worst case in 2D and 3D for both $\gamma([-1, 1], [0, 1])$ and $\gamma([-1, 1], [c, 1])$, $c < 0$, and the expected case for $\gamma([-1, 1], [0, 1])$ (with points uniformly distributed within a ball [9]). Note that the while-loop is executed $\mathcal{O}(N_v)$ times in 2D to include all vertices into the boundary, but $\mathcal{O}(N^2)$ times in 3D, because hull faces can be removed without adding vertices to the boundary. The results are listed in Table 1, where for line 3 and 7 the number of iterations is given.

All storage complexities are dominated by the size of the γ-Graph. For practical cases, the use of $\gamma([-1, 1], [0, 1])$ is predominant, resulting in a worst-case time complexity of $\mathcal{O}(N_v \log N_v)$ for 2D and $\mathcal{O}(N_v^2)$ for 3D. Note, however, that the latter complexity stems from the worst possible situation; i.e., the number of triangles in the γ-Graph is $\mathcal{O}(N_v^2)$. In terms of the number of triangles N_t, line 1 takes $\mathcal{O}(N_t \log N_t)$, and the iteration over line 3 is performed $\mathcal{O}(N_t)$ times, resulting in a total of $\mathcal{O}(N_t \log N_t)$ worst case time complexity.

Algorithm 1 has been implemented in C. The constriction process takes about one second on a contemporary high-end workstation type of machine for the mask data set of Fig. 7. The $\gamma([-1, 1], [0, 1])$ on that set of

ALGORITHM 1
Constriction Algorithm

```
    Constrict ( )
    { graph Graph;                                              // (pruned) $\gamma$-Graph
      face Face, NewFace;                                       // 2D: edge; 3D: triangle
      int $N_{hv}$;                                             // number of hull vertices
      heap Heap;
1.    Load-$\gamma$-Graph (Graph);
2.    $N_{hv}$ = InitialBoundary (Graph, Heap);
3.    while ($N_{hv} < N_v$ && Heap != $\emptyset$)
      {
4.      Face $\doteq$ Root (Heap);
5.      if (Removable (Face))
        {
6.        $N_{hv}$ += Delete (Face, Graph);
7.        for (each NewFace on the hull)
8.          if (Removable (NewFace))  Insert (NewFace, Heap);
        }
      }
9.    ReportBoundary (Graph);
    }
```

vertices consists of 18,274 triangles, forming 8633 tetrahedra.

7. CONSTRAINED CONSTRICTION

We have assumed that no relations between the vertices, such as connectivity along the boundary, is known in advance. If such relations are known for all vertices, a more powerful boundary construction method could be used, exploiting the additional information. In the case that only a small number of connectivity relations are known, however, we would like the constriction algorithm to respect these connections. For example, an expert might know that certain connections must exist in experimental data, or a designer may demand specific connections when specifying vertices of an artifact.

To be more precise, assume that a collection F of faces are known to be part of the boundary. The faces of F that are not part of the γ-Graph used for constriction are added to the graph; this augmented γ-Graph is used in the following constrained constriction algorithm, which is not allowed to delete any of the faces from F.

The constrained constriction works as follows. The faces from F are never removable and therefore never put into the heap of removable hull faces. A face from F is not only unremovable when it lies on an intermediate hull, but may also not be deleted from the graph if it crosses a hull triangle (2D) or hull tetrahedron (3D). Consider first the

TABLE 1
Complexity of the Algorithm per Step

	Worst case				
	2D		3D		Exp. case 2D and 3D
Line	$\gamma([-1, 1], [0, 1])$	$\gamma([-1, 1], [<0, 1])$	$\gamma([-1, 1], [0, 1])$	$\gamma([-1, 1], [<0, 1])$	$\gamma([-1, 1], [0, 1])$
1	$\Theta(N_v)$	$\mathcal{O}(N_v^2)$	$\mathcal{O}(N_v^2)$	$\mathcal{O}(N_v^3)$	$\mathcal{O}(N_v)$
2	$\Theta(N_v \log N_v)$	$\mathcal{O}(N_v^2)$	$\Theta(N_v \log N_v)$	$\mathcal{O}(N_v^2)$	$\Theta(N_v \log N_v)$
3	$\mathcal{O}(N_v) \times$	$\mathcal{O}(N_v)\times$	$\mathcal{O}(N_v^2)\times$	$\mathcal{O}(N_v^2)\times$	$\mathcal{O}(N_v)\times$
4	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$
5	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
6	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
7	$\Theta(1)\times$	$\mathcal{O}(N_v)\times$	$\Theta(1)\times$	$\mathcal{O}(N_v)\times$	$\Theta(1)\times$
8	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$	$\mathcal{O}(\log N_v)$
9	$\Theta(N_v)$	$\mathcal{O}(N_v)$	$\mathcal{O}(N_v)$	$\mathcal{O}(N_v)$	$\mathcal{O}(N_v)$
Total	$\mathcal{O}(N_v \log N_v)$	$\mathcal{O}(N_v^2 \log N_v)$	$\mathcal{O}(N_v^2 \log N_v)$	$\mathcal{O}(N_v^3 \log N_v)$	$\mathcal{O}(N_v \log N_v)$

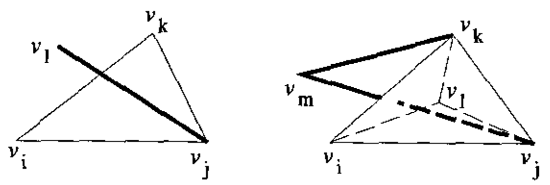

FIG. 11. Constrained constriction in 3D: the bold edge (left) and the bold triangle (right) may not be deleted.

2D case that an edge $v_j v_\ell$ from F crosses a hull triangle $v_i v_j v_k$ whose hull edge $v_i v_j$ is selected for deletion; see Fig. 11 (left). Normally, the deletion of $v_i v_j$ involves the deletion of $v_j v_\ell$ from the graph, but $v_j v_\ell$ must remain in the graph. Instead, $v_i v_j$ and the γ-indicator with respect to v_k is deleted from the heap, and to prevent reinsertion, $v_i v_k$ is deleted from the graph. If $v_i v_\ell$ is not already present, it is inserted into the graph, so that $v_i v_j v_\ell$ is a hull triangle. If $v_i v_j$ is removable with respect to v_ℓ, it is inserted into the heap together with the γ-indicator with respect to v_ℓ. In one of the next iterations, $v_i v_j$ may be deleted from the graph and the heap, and bring $v_j v_\ell$ into the hull. A constrained constriction may get locked if $v_j v_\ell$ is not yet in the boundary and $v_i v_j$ has become unremovable. Note that apart from $v_j v_\ell$, other edges may cross $v_i v_j v_k$ as well; however, that does not effect the constrained constriction presented above.

The analogous 3D situation is depicted at the right in Fig. 11, where $v_i v_j v_k$ is a hull triangle and $v_j v_k v_m$ a triangle from F. Normally, the deletion of $v_i v_j v_k$ involves the deletion of $v_j v_k v_m$, but $v_j v_k m_k$ must remain in the graph. Instead, $v_i v_j v_k$ and the γ-indicator with respect to v_ℓ is deleted from the heap, and to prevent reinsertion into the heap, $v_i v_j v_k$ is also deleted from the graph. If $v_i v_j v_m$ and $v_i v_k v_m$ are not already present, they are inserted into the graph, so that $v_i v_j v_k v_m$ is a hull tetrahedron. If $v_i v_j v_k$ is now removable with respect to v_m, it is inserted into the heap together with the γ-indicator with respect to v_m. As in 2D, a constrained constriction may get locked when $v_j v_k v_m$ is not yet in the boundary but has become unremovable. Apart from $v_j v_k v_m$ other triangles may cross $v_i v_j v_k v_\ell$ as well, but that does not effect the constrained constriction.

8. CONCLUDING REMARKS

We have presented a method for the construction of a polygonal or polyhedral boundary passing through all the points of a given set. The points are assumed to lie on the boundary of a closed object without through-passages and inner voids, but no assumptions on the relative order of the boundary points are made, and no other information such as sample density or normal vectors is available. The construction technique is based on a parameterized geometric graph, the γ-Neighborhood Graph. The γ-Graph on the set of points is successively constricted until the hull of the pruned γ-Graph is a proper object boundary, passing through all vertices. For specific choices of the γ-parameters, the method remains the same, but the underlying graph is different. The results shown demonstrate that the γ-Graph based method is more powerful in cases where the Delaunay Triangulation contains no Hamilton cycle/polyhedron or constriction gets locked, and that it gives good results in a number of experimental cases. Sufficient algorithmic and implementational information is provided so that an experienced practitioner can reproduce the results.

While constriction of the Delaunay Triangulation may stop without having found a Hamilton polygon or polyhedron, the parameters of the γ-Graph provide the flexibility to find a proper boundary. The selection of the hull faces to remove is based on the combination of a local measure (the γ-indicator), and global information (the minimum value). This criterion yields good looking boundaries in comparison with related methods. The constriction algorithm is easily extended so as to prevent the deletion of a priori known boundary faces.

The approach used in this work in not completely novel (see, in particular, [3, 4]), but the γ-Neighborhood Graph and the selection criterion on which the approach is based are novel and they are used to novel fundamental effect: it remedies the fact that the Delaunay Triangulation need not always contain a Hamilton polygon or polyhedron.

It is not widely acknowledged that construction methods based on a geometric graph (such as the Delaunay Triangulation, Voronoi Diagram, or γ-Graph) are generally not consequent. That is, if the constructed boundary of a vertex set V is B and a new vertex that lies on B is added to V, the boundary resulting from the new input data set need not be B. This results from the fact that in general the geometric graph of the new vertex set does not contain all the edges or triangles of the graph on V, in particular, those of the boundary segments of B.

Another interesting case is the behavior of the algorithm when the vertices are distributed arbitrarily dense over a known object boundary. The object boundary segments then become arbitrarily small. When the limit is reached, the γ-indicator of each segment is 1, but the γ-indicator of intermediate hull segments is smaller than 1. So, the object boundary segments are never removed, and therefore the constructed boundary is the known object boundary.

The result of our constriction process is a pruned graph. The hull of this graph is the boundary of the object. The boundary segments can easily be oriented consistently, such that the normal points always inward or always outward. Apart from the hull, the pruned $\gamma([-1, 1], [0, 1])$ implicitly defines a triangulation of the interior of the object. A pruned $\gamma([-1, 1], [c, 1])$, $c < 0$ can easily be pruned further to obtain a triangulation (or tetrahedrization) of the interior. Such a triangulation of the interior can be used to calculate properties such as the volume and mass of the object. We have seen that an object skeleton need not be a good tool to construct a boundary. Conversely, the triangulation or tetrahedrization of the pruned γ-Graph always provides *some* skeleton of the object.

The vertices are assumed to lie on the boundary of an object without inner voids, but an inner contour or surface can be handled separately. The triangulation of the interior of the outer boundary does not correspond to the body of the object anymore, but for the calculation of some properties the value corresponding to the inner boundary can be subtracted from the result corresponding to outer boundary. The case of objects with handles is considered to be difficult. A solution which is dependent on a priori known sample density is presented in [15]. By contrast, our solution does not depend on a known sample density because it exploits a local measure in the construction process (the γ-indicator). In addition, our boundary will pass *through* all the data points.

REFERENCES

1. P. D. Alevizos, J. D. Boissonnat, and M. Yvinec, An optimal $o(n \log n)$ algorithm for contour reconstruction from rays, in *Proceedings of the 3rd ACM Symposium on Computational Geometry*, pp. 162–170, ACM Press, New York 1987.
2. J.-D. Boissonnat, Representation of objects by triangulating points in 3-D space, in *Proceedings of the 6th International Conference on Pattern Recognition,* 1982, pp. 830–832.
3. J.-D. Boissonnat, Representing 2D and 3D shapes with the Delaunay triangulation, in *Proceedings of the 7th International Conference on Pattern Recognition,* 1984, pp. 745–748.
4. J.-D. Boissonnat, Geometric structures for three-dimensional shape representation, *ACM Trans. Graphics* **3,** 1984, 266–286.
5. B. K. Choi, H. Y. Shin, Y. I. Yoon, and J. W. Lee, Triangulation of scattered data in 3D space, *Comput. Aided Design* **5,** 1988, 239–248.
6. N. R. Corby and J. L. Mundy, Applications of range image sensing and processing, in *Analysis and Interpretation of Range Images* (R. C. Jain and A. K. Jain Eds.), pp. 255–272, Springer-Verlag, Berlin/New York, 1990.
7. M. B. Dillencourt, A non-Hamiltonian, nondegenerate Delaunay triangulation, *Inform. Process. Lett.* **25,** 1987, 149–151.
8. M. B. Dillencourt, An upper bound on the shortest exponent of inscribable polytopes, *J. Combin. Theory Ser. B* **46,** 1989, 66–83.
9. R. A. Dwyer, Higher-dimensional Voronoi diagrams in linear expected time, in *Proceedings of the 5th ACM Symposium on Computational Geometry,* pp. 326–333, ACM Press, New York, 1989.
10. G. A. Dirac, On Hamiltonian circuits and Hamiltonian paths, *Math. Ann.* **197,** 1972, 57–70.
11. H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, On the shape of a set of points in the plane, *IEEE Trans. Inform. Theory* **IT-29,** 1983, 551–559.
12. H. Edelsbrunner and E. P. Mücke, Three-dimensional alpha shapes, *ACM Trans. Graphics* **13,** 1994, 43–72.
13. A. B. Ekoule, F. C. Peyrin, and L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours, *ACM Trans. Graphics* **10,** 1991, 182–199.
14. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision,* Vol. 1, Addison-Wesley, Reading, MA, 1992.
15. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, *Comput. Graphics* **26,** 1992, 71–78.
16. J. O'Rourke, Polyhedra of minimal area as 3D object models, in *Proceedings of the International Joint Conference on Artificial Intelligence,* 1981, pp. 664–666.
17. J. O'Rourke, The computational geometry column, *Computer Graphics* **20,** 1986, 232–234.
18. J. O'Rourke, H. Booth, and R. Washington, Connect-the-dots: A new heuristic, *Comput. Vision Graphics Image Process.* **39,** 1987, 258–266.
19. B. A. Payne and A. W. Toga, Surface reconstruction by multiaxial triangulation, *IEEE Comput. Graphics Appl.* **14,** 1994, 28–35.
20. AAAI-90 Workshop on Qualitative Vision, 1990.
21. M. Rioux and L. Cournoyer, *The NRCC Three-Dimensional Image Data Files,* Technical Report CNRC 29077, National Research Council Canada, 1988.
22. Y. Shirai, *Three-Dimensional Computer Vision,* Springer-Verlag, Berlin/New York, 1987.
23. C. Thoenes, Uccello's chalice, *Comput. Aided Geom. Design* **1,** 1984, 97–99.
24. W. T. Tutte, Bridges and Hamiltonian circuits in planar graphs, *Aequationes Math.* **15,** 1977, 1–33.
25. R. C. Veltkamp, The γ-neighborhood graph, *Comput. Geom. Theory Appl.* **1,** 1992, 227–246.