3D Computational Morphology

Remco C. Veltkamp

CWI, Dept. Interactive Systems, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
and
Eindhoven University of Technology, Dept. Computer Science, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
remco@win.tue.nl

Abstract

Computational Morphology is the analysis of form by computational means. This discipline typically uses techniques from Computational Geometry and Computer Aided Geometric Design. The present paper is more specifically about the construction and manipulation of closed object boundaries through a set of scattered points in 3D. Original results are developed in three stages of computational morphology:

1. *impose a geometrical structure on the set of points;*
2. *construct a polyhedral boundary surface from this geometrical structure;*
3. *build a hierarchy of polyhedral approximations together with localization information;*

The economic advantage of this approach is that there is no dependency on any specific data source. It can be used for various types of data sources or when the source is unknown.

Keywords: computational morphology, shape analysis, computational geometry, computer aided geometric design.

1 Introduction

The work described here deals with the computational aspects of geometry with respect to form or shape information, that is, morphology. Morphology is closely related to geometry, and thus to Computational Geometry in a broad sense, i.e. discrete and combinatorial geometry, as well as computer aided geometric design. A computational geometric approach to the analysis of form is called *Computational Morphology* [23]. In particular, this paper is about the construction and subsequent manipulation of closed object boundaries from a set of points in 3D. These points are scattered points, i.e. no structural relationship between them is known in advance and they have arbitrary position. We consider three stages of this task:

1. Given a set of scattered points, construct a geometric structure on the points.
2. Given a geometric structure on a set of scattered points, find a closed polyhedral boundary through all points.
3. Given a closed object boundary, construct a hierarchy of approximations and localization information.

In these three steps few assumptions are made about structure and regularity of the input: the data points lie on a closed surface, in contrast to a height surface as in [5] [22], the points do not lie on a regular grid as in [20], and the polyhedral surface typically has no rectangular topology as in [6].

In many applications in geometric modeling, computer graphics, object recognition, distance map image processing, and computer vision, input data is available in the form of a set of 3D coordinates that are points on the boundary of an object. Such points can be synthetic or measured from the boundary of an existing object. A collection of points, however, is an ambiguous representation of an object, and can therefore not be used directly in many applications. It is often essential to have a representation of the whole boundary available, that is unambiguously defining a valid object. The boundary constructed from a set of points can for example be used for the initial design of an artifact, for numerical analysis, or for graphical display.

The way in which the boundary points are acquired may give useful information in order to construct the whole boundary, but can also make the construction method very dependent on the specific data source. If it is not known how the data is obtained or if a single construction method is to be used for data from various types of sources, no structural relation between the input points may be assumed, except that they all lie on the boundary surface of an object. The order of the points in the input then provides no information on their topological relation to each other. In particular, they do not lie on a regular grid, but are scattered points. In this paper we assume no a priori knowledge about any structural relation between the points.

The simplest boundary through a set of points is one that consists of flat triangles, making a 3D simple closed polyhedral surface. A simple closed polyhedron must consist of $2N-4$ triangles, with N the number of data points. A brute force algorithm that tests all combinations of $2N-4$ triangles out of all $\binom{N}{3}$ possible triangles takes

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

time, which is at least $\Omega(N^{5N})$. This is clearly infeasible. We therefore exploit some geometric relation between the data points. Section 2 presents the γ-Neighborhood Graph, which describes the geometric structure on the set of data points. Section 3 describes how the γ-Neighborhood Graph is exploited for the construction of a closed polyhedral object boundary.

In many real applications, a boundary constructed from a set of points consists of thousands of faces. However, an approximation of the object is often sufficient. In animation for example, the motion prohibits the perception of much detail, so that an approximated object is sufficient and is also faster to display. Localization provides bounding volume information. Such information is useful for efficient operations such as collision detection for robot motion planning. Approximation and localization can be combined in a single scheme, and several levels of approximation and localization can be combined in a hierarchical way. Section 4 presents a new hierarchical approximation and localization scheme.

Polyhedral boundaries are only C^0-continuous at the edges; there the normal vectors instantly changes direction. A smoother surface, consisting of curved triangles that interpolate the flat triangles' vertices and are smoothly connected along the edges, is often desired. For example, esthetic demands apply to car body design, physical requirements deduced from aerodynamic or hydrodynamic laws apply to aircraft and ship hull design, and for boundary *re*construction the smoothness demands are determined by the smoothness of the original boundary. A surface that has a continuously changing normal vector is called G^1-continuous. A fourth computational morphology step is presented by [26]. It gives a novel scheme to split curved Bézier triangles and to set their control points so as to make the surface G^1-continuous.

2 Point set analysis

In order to perform any geometric analysis on a set of scattered points, some geometric structure must be imposed on it. Such a structure typically relates points to each other if they satisfy some geometric property, and is represented by a graph. Examples of such geometric graphs are the Nearest Neighbor Graph, the Euclidean Minimal Spanning Tree, the Infinite Strip Graph [8], the Sphere of Influence Graph [24], the Relative Neighborhood Graph [15], the Gabriel Graph [13], the Convex Hull, the Delaunay Triangulation [7] and its dual Voronoi Diagram [29], the α-Shape [11], and the β-Skeleton [14]. Section 2.1 presents the γ-Neighborhood Graph, a parameterized geometric graph which unifies a number of the before-mentioned graphs. The γ-Neighborhood Graph is used in Section 3 for the construction of an object boundary through all data points.

2.1 The γ-Neighborhood Graph

To start with, let us consider the 3D Delaunay Triangulation on a set of points. The 3D Delaunay Triangulation actually is a tetrahedralization; it is a filling of the space inside the Convex Hull of the point set by tetrahedra with the following properties:

1. the vertices of each tetrahedron are data points,
2. the ball touching the vertices of each tetrahedron contains no other data point in its interior.

A ball or disc that contains no data points in its interior is called empty. If no more than four data points lie on an empty ball and no more than three coplanar data points lie on an empty disc, the Delaunay Tetrahedralization is uniquely defined. Otherwise the tetrahedralization on these specific points must be non-overlapping.

The Delaunay Tetrahedralization can equivalently be defined as the collection of all triangles that have an empty ball touching its vertices. For all these triangles there are two largest possible empty balls touching their vertices, which either touch a fourth point or have infinite radius. In this last situation the ball degenerates to an empty half-space, and the triangle lies on the Convex Hull. The radii of the two balls are a scaling of the radius r of the smallest possible ball touching the three 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 Tetrahedralization is a particular instance of the γ-Neighborhood Graph. This graph discriminates between the case that the centers of the balls lie at opposite sides of the triangles and the case that they lie at the same side. In the latter case the parameter c_0 is taken negative. Then 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 Tetrahedralization is denoted $\gamma([-1,1],[0,1])$. Figure 1 shows a typical $\gamma([-1,1],[0,1])$.

For each triangle in the Delaunay Tetrahedralization the *union* of two balls touching their vertices is empty. The γ-Neighborhood Graph also considers the case that only the *intersection* of two balls is empty. In that case the parameter c_1 is taken negative. Then c_1 lies 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 Tetrahedralization 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 its 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.

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 Tetrahedralization, 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.

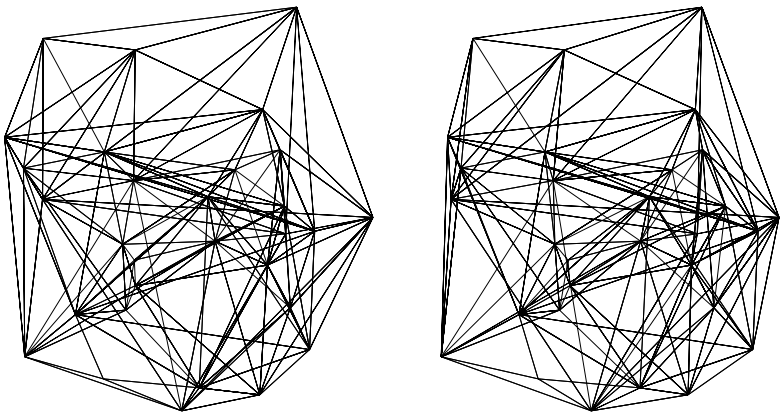

Figure 1: Stereo pair of a perspectively projected $\gamma([-1,1],[0,1])$.

A formal definition and analysis and many examples of the γ-Neighborhood Graph are given in [28]. The 3D $\gamma([-1,1],[d,1])$ can be constructed in $\Theta(N^2)$ time for $d = 0$, which is optimal, and $\mathcal{O}(N^3)$ time for $d < 0$.

3 Boundary surface construction

The boundary construction problem that we deal with is stated as follows: find a simple closed polyhedron of triangular faces through all data points of a given set. This task is clearly underconstrained, so that a solution is not unique. Indeed, a set of points is an ambiguous boundary representation. 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. In order to avoid excessive time complexities some heuristic is needed. The heuristic must be chosen so as to yield a boundary among all possible solutions that is considered a likely boundary for the given point set.

Several methods expand some initial triangulation until all data points are included in the boundary polyhedron [1] [3] [4]. A drawback of these methods is that additional information is needed, for example the surface normal at the data points. 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 [3]. The method in [17] tries to find the simple polyhedron in the Delaunay Tetrahedralization that corresponds to the shortest tree in the dual Voronoi Diagram. For objects that have no distinct skeleton this method gives strange results.

The methods in [2] and [3] take the Delaunay Tetrahedralization and successively delete triangles from the outside until the resulting hull passes through all data points. It is not yet known whether every non-degenerate Delaunay Tetrahedralization contains a simple closed polyhedron through all data points. However, this 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.

3.1 Boundary extraction from the γ-Graph

This section is concerned with the construction of a boundary polyhedron by constricting the graph $\gamma([-1,1],[c,1])$, $c \in [-1,0]$ of the input point set. 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 triangle from the graph in such a way that a hull of the new graph is properly defined (see below). A γ-Graph from which triangles are deleted is not a γ-Graph anymore, 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.

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 boundary, 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 1 (γ-indicator) *Let $v_iv_jv_k$ be a triangle of an intermediate hull, let $v_iv_jv_\ell$, $v_jv_kv_\ell$, and $v_iv_kv_\ell$ be triangles in the (pruned) γ-Graph, and let c_0 be defined by $R(v_i,v_j,v_k,v_\ell) = r(v_i,v_j,v_k)/(1-|c_0|)$. The γ-indicator of $v_iv_jv_k$ with respect to v_ℓ is $|c_0|$ 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 $-|c_0|$ if the center lies at the other side; and is zero if $c_0 = 0$.*

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_k, and v_ℓ have the same orientation as v_i, v_j, v_k, and the sphere center C, that is, if

$$sign([v_i - v_\ell, v_j - v_\ell, v_k - v_\ell]) = sign([v_i - C, v_j - C, v_k - C]), \tag{1}$$

where '[ ]' denotes the determinant.

The solid angle φ at v_ℓ 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 the value of the γ-indicator $c_0^i \leq 0$, then $r(v_i,v_j,v_k)/R(v_i,v_j,v_k,v_\ell) = 1 - c_0^i$, and if $c_0^i \geq 0$, then $2 - r(v_i,v_j,v_k)/R(v_i,v_j,v_k,v_\ell) = 1 - c_0^i$, provided that $v_iv_jv_k$ is fixed. So, the larger $1 - c_0^i$, the larger φ, i.e. the wider the solid angle at v_ℓ. On the other hand, if $1 - c_0^i$ 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 Figure 2. Since A/R^2 is independent of the size of the tetrahedron, it seems obvious to use a selection criterion based on both $1 - c_0^i$ and A/R^2. However, it appears that using $1 - c_0^i$ alone gives better results. Indeed, a typical Convex Hull contains many triangles of small area (see for example Figure 3), 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. However, it can result in an extra boundary 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 non-overlapping parts of space, it is obvious to sum the values $1 - c_0^i$ of both hull triangles in the selection rule.

Deletion of a hull triangle from the graph must leave the hull of the graph properly defined. 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

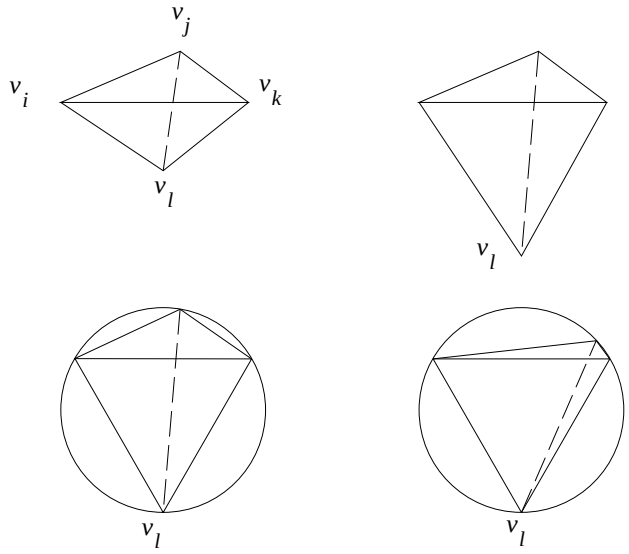

Figure 2: The solid angle at v_ℓ in the left column is larger than in the right column. Top row: fixed triangle $v_i v_j v_k$. Bottom row: fixed γ-indicator.

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 $1 - c_0^i$ is equivalent to a small value of c_0^i, 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.*

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

Figure 3 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 [19].

In [26] results are compared with the results of constricting the Delaunay Tetrahedralization [2] and the boundary corresponding to the shortest tree in the Voronoi Diagram [17]; our results turn out to be better.

Once the γ-Neighborhood Graph has been built the boundary extraction algorithm takes $\Theta(T \log T)$ time, with T the number of triangles in the γ-Graph.

4 Hierarchical approximation and localization

Two facilities are often used for efficient manipulation of complex polyhedral objects consisting of many faces: approximation and localization, which can both be performed hierarchically. The purpose of both techniques is to avoid unnecessary processing of much detail.

If the vertices of the polygon or polyhedron lie on a regular grid, finding an approximation or localization is a simpler task than for an arbitrary vertex connectivity structure, or topology. For example [20] provide an efficient approximation scheme for triangular polyhedra whose vertices lie on a regular grid. In the following we will only consider approximation and localization schemes for arbitrary topologies.

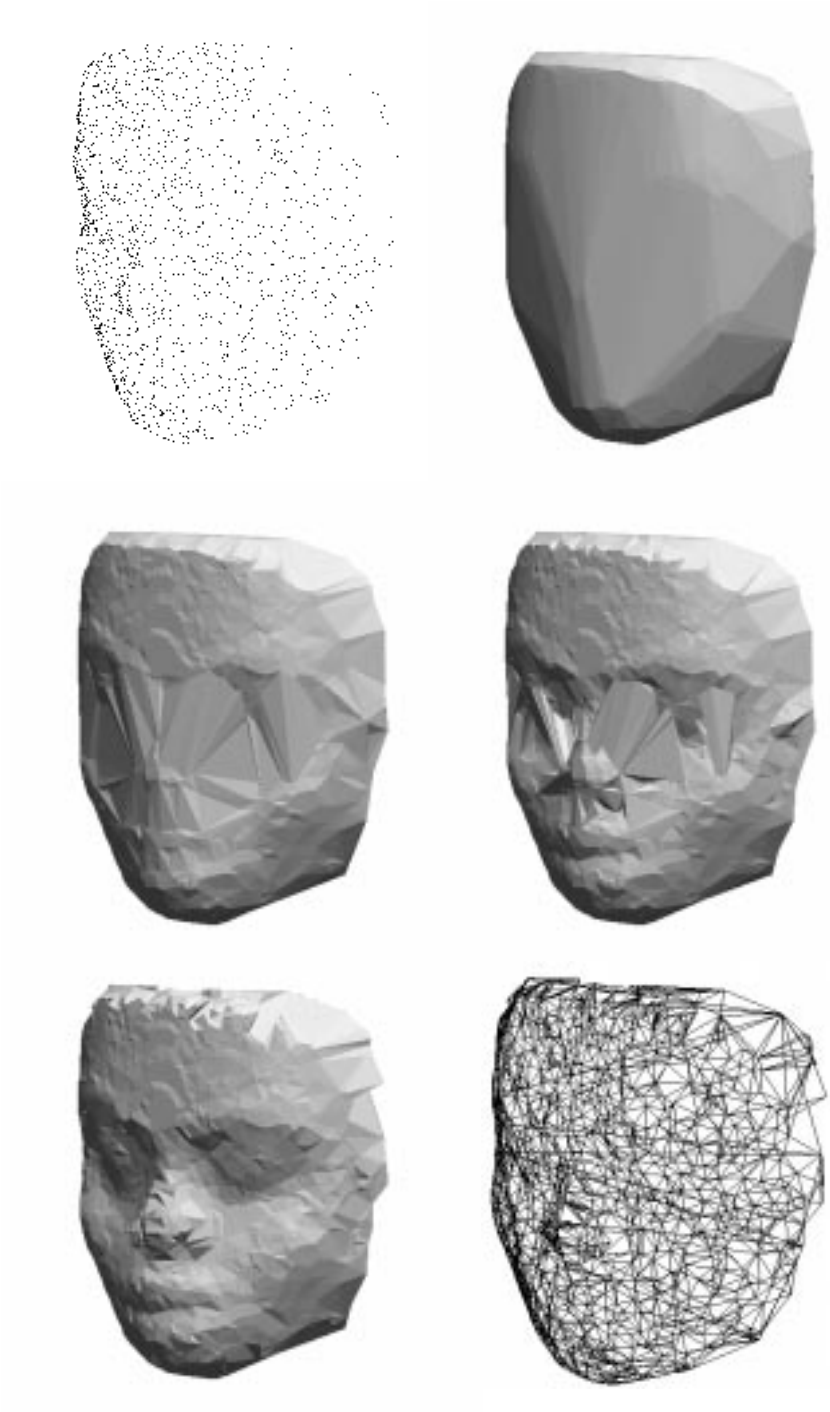

Figure 3: Indian mask reconstructed from laser-range data points. Top left: 1468 scattered vertices. Top right: Convex Hull consisting of 255 vertices and 504 triangles. Middle left: intermediate hull, 1019 vertices and 2034 triangles. Middle right: intermediate hull, 1337 vertices and 2670 triangles.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

Approximation and localization of polyhedra of arbitrary topology have not received much attention. The Delaunay Pyramid [5] can only be used for terrain surfaces, which is too limited for our application of closed object boundaries. A 3D analogue of the 2D iterative end point fit approximation [10] or Douglas–Peucker algorithm [9] is presented in [12]. I will call this the 3D Iterative Extreme Point Fit algorithm. This scheme provides no localization information. The Prism Tree [18] is a localization scheme that stores bounding volumes of the open polyhedra approximated by the 3D Iterative Extreme Point Fit algorithm. The bounding volumes are truncated pyramids which are called prisms. A level in the tree corresponds to a collection of truncated pyramids enclosing the boundary, rather than to an approximation polyhedron.

In the next section I will present a hierarchical approximation and localization scheme which is computationally efficient for hierarchical operations such as intersection and point-in-object tests.

4.1 The Flintstone scheme

The approximation algorithm is based on the way the localization of a part of the closed polyhedron, which is an open polyhedron, is performed. The bounding volume of such an open polyhedron is called a Flintstone. Note that the bounding volumes enclose parts of the object's surface, not the object's volume.

The ball through points a, b, c, and d is denoted $B(a,b,c,d)$. The half-space containing d and whose boundary passes through a, b, and c, is denoted by $H(a,b,c;d)$. A half-space can be considered as a ball with a radius of $-\infty$. Before presenting the approximation algorithm, the basic bounding volume, Flintstone, must be defined:

Definition 2 (Flintstone) *Let P be an open polyhedron, and v_p, v_s, and v_t three distinct vertices on the closed boundary of P. Let v_q be a vertex of P such that $B(v_p,v_s,v_t,v_q)$ contains all vertices lying in $H(v_p,v_s,v_t;v_q)$. If $P \subset H(v_p,v_s,v_t;v_q)$, then the Flintstone F of P is defined as $F(P) = B(v_p,v_s,v_t,v_q) \cap H(v_p,v_s,v_t;v_q)$. Otherwise $F(P)$ is the smallest of $B(v_p,v_s,v_t,v_q) \cap B(v_p,v_s,v_t,v_r)$ and $B(v_p,v_s,v_t,v_q) \cup B(v_p,v_s,v_t,v_r)$, where v_r is a vertex of P not in $H(v_p,v_s,v_t;v_q)$ such that $B(v_p,v_s,v_t,v_r)$ contains all vertices in $H(v_p,v_s,v_t;v_r)$.*

Paraphrasing, $F(P)$ is the smallest bounding volume of P defined by (the intersection or union of) two balls touching v_p, v_s, and v_t. Note that such a v_q in the definition always exists, and also a v_r if not all vertices of P lie in $H(v_p,v_s,v_t;v_q)$. The definition is illustrated in Figure 4 for the case that the

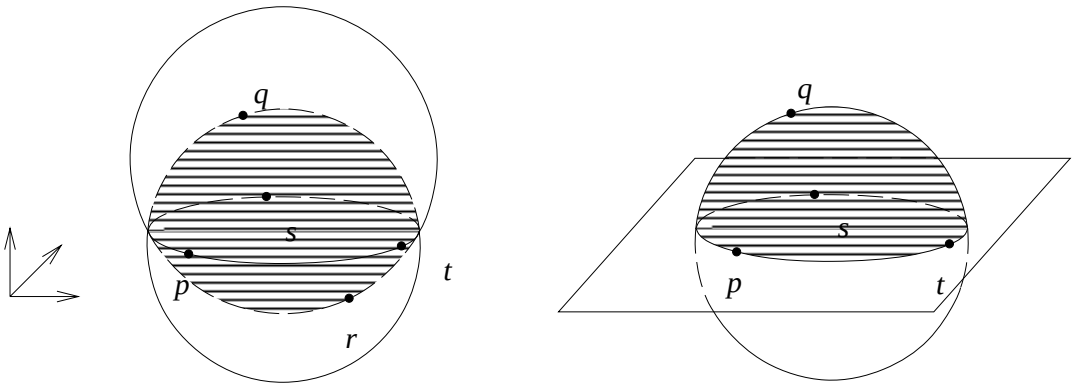

Figure 4: 3D Flintstones.

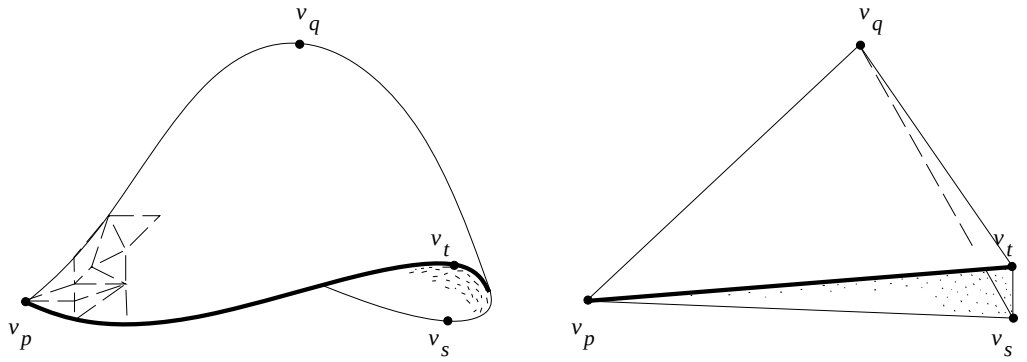

Figure 5: Triangle $v_p v_s v_t$ is split into three at a point inside the polyhedral surface.

Flintstone is an intersection of two balls or a ball and a half-space. It is easily verified that $P \subset F(P)$, so $F(P)$ is indeed a bounding volume for P.

The hierarchical approximation of a closed polyhedron starts with the calculation of the smallest ball that touches at least three vertices, say v_i, v_j, and v_k, and contains all vertices. Such a ball always exists, but need not be the smallest bounding ball, which may touch only two vertices.

The three shortest paths of edges in the polyhedron running between v_i and v_j, v_j and v_k, and between v_k and v_i, divide the closed polyhedron into two open polyhedra. Triangle $v_i v_j v_k$ is the zeroth order approximation of the closed polyhedron.

At all next levels of the hierarchical approximation we consider an open polyhedron P of more than three vertices, and three distinct vertices v_p, v_s, and v_t on the boundary of P. If $F(P) = B(v_p, v_s, v_t, v_q) \cap H(v_p, v_s, v_t; v_q)$, then P is approximated by the three triangles $v_p v_q v_s$, $v_q v_s v_t$, and $v_p v_q v_t$. If $F(P) = B(v_p, v_s, v_t, v_q) \cap B(v_p, v_s, v_t, v_r)$ and if the radius of $B(v_p, v_s, v_t, v_q)$ is smaller than the radius of $B(v_p, v_s, v_t, v_r)$, P is approximated by $v_p v_q v_s$, $v_q v_s v_t$, and $v_p v_q v_t$, as illustrated in Figure 5; if the radius of $B(v_p, v_s, v_t, v_q)$ is larger, then P is approximated by $v_p v_r v_s$, $v_r v_s v_t$, and $v_p v_r v_t$. If instead $F(P) = B(v_p, v_s, v_t, v_q) \cup B(v_p, v_s, v_t, v_r)$ and if the radius of $B(v_p, v_s, v_t, v_q)$ is larger than the radius of $B(v_p, v_s, v_t, v_r)$, P is approximated by $v_p v_q v_s$, $v_q v_s v_t$, and $v_p v_q v_t$; if the radius of $B(v_p, v_s, v_t, v_q)$ is smaller, then P is approximated by $v_p v_r v_s$, $v_r v_s v_t$, and $v_p v_r v_t$. In other words, the new triangles are made with either v_q or v_r, whichever lies on the ball that contributes most to the 'width' of the Flintstone.

At all levels of approximation there is an open polyhedral part of the original polyhedron associated with each approximation triangle $v_p v_s v_t$. The boundary polygon of that open polyhedron consists of

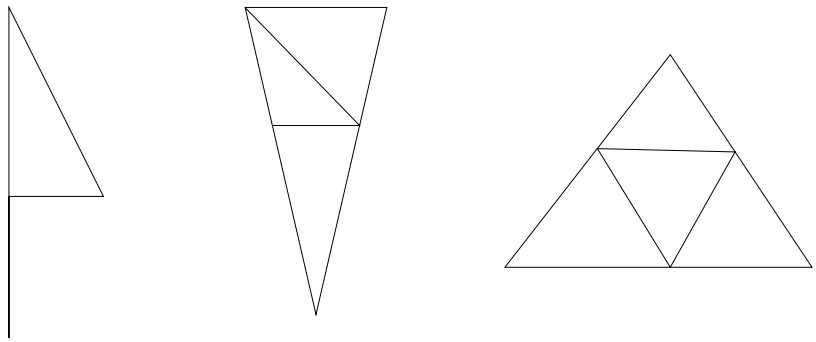

Figure 6: Schematic splitting at sides.

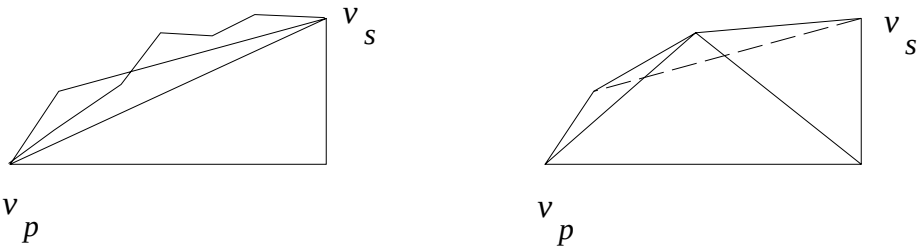

Figure 7: Two neighboring triangles are split at their common side.

the three shortest paths between v_p, v_s, and v_t. Note that each pair of these paths may partly coincide, especially near v_p, v_s, or v_t.

If the triangles are always split into three new ones as described above, the new triangles become more and more elongated, because each time the angles at the vertices are divided. Moreover, the edges are retained in all next levels, which will generally not lead to a good approximation. To avoid too thin triangles, the angle is not split if it is less than some chosen value. In that case the triangle is split at two sides, as in Figure 6 (middle). A triangle can also have two angles that are too small to be split, in which case the triangle is split at one side, as in Figure 6 (left). If a triangle is split at a side, its neighboring triangle sharing that side should also be split at the same point, in order to prevent cracks in the approximation polyhedron. A triangle can thus be forced to split, even if it has no small angles. Therefore, a triangle can also split at all three sides into four new triangles, as illustrated in Figure 6 (right).

Because a triangle can be forced to split at a side by its neighbors, the effect of splitting at a side propagates through the current approximation polyhedron. Therefore, all edges in the current approximation polyhedron that must be split should be determined first. Only then it is known how the splitting of all triangles should be done.

The position where a side should be split is at a vertex on the shortest path between the end points, which is part of the boundary polygon of the open polyhedron approximated by the triangle. The vertex where the split is performed is the one that gives the smallest difference between the lengths of the new sides. If the shortest path consists of a single edge, there is no such vertex, and the split is not performed. So, no new vertices are introduced. Such a split of a triangle into two new ones, together with its neighboring triangle, is shown in Figure 7. As stated before, two paths may partly coincide, and this may affect the result of the splitting. For if the vertex where a side is split is part of two paths, then degenerate triangles will result. This, however, causes no problem, since the degenerate triangles can simply be recognized by their coalescing vertices.

The refinement iteration is repeated until no approximated open polyhedron contains interior vertices, and the shortest paths between the vertices of all approximation triangles consist of single edges. This is the most detailed level of approximation. By definition, the approximation triangles at the lowest level have no Flintstone.

Because no new vertices are introduced when splitting a triangle at a side, the shortest paths consist of edges of the original polyhedron. And because triangles that do not approximate a part of the polyhedron are discarded, all the final approximating triangles coincide with the original triangles. So, at the most detailed level of approximation the original polyhedron is recovered.

The error of approximation of a polyhedron by m Flintstones is $\max_{1 \leq i \leq m} w_i$, with w_i informally defined as the width of the i-th Flintstone. The approximation algorithm can be made adaptive by only refining a triangle if the approximation error of *that* triangle is larger than a given bound. Figure 8 shows an example of a polyhedral object and two adaptive Flintstone approximations using this error criterion.

There is no simple representation scheme that implicitly represents the hierarchy of triangles and

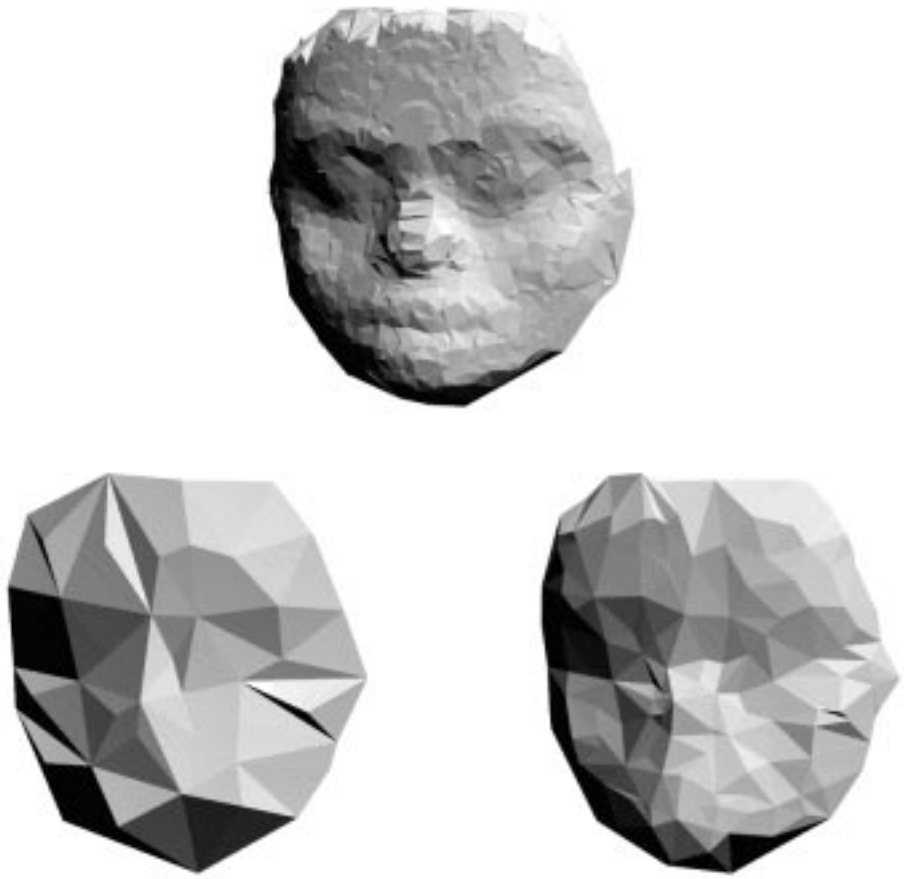

Figure 8: Top: polyhedral object of 2930 triangles. Bottom left: adaptive Flintstone approximation of 89 triangles. Bottom right: adaptive Flintstone approximation of 277 triangles.

Flintstones. They are therefore stored explicitly. The root contains the center and radius of the ball bounding the whole polyhedron. The two sons of the root contain v_i, v_j and v_k, approximating the open polyhedra P_1 and P_2, and the centers and signed radii of the balls that define the Flintstones. All nodes except the root can have up to four sons, which are constructed as described above.

The worst case time complexity to build the complete Flintstone tree is $\Theta(N^2 \log N)$ and the storage complexity is linear [27].

5 Conclusions

The main results of the research described in this paper are summarized below.

Section 2. The γ-Neighborhood Graph is a new parameterized geometric graph. By its two parameters, a whole family of geometric graphs is defined, ranging from the empty to the complete graph. For particular choices of the parameters, the γ-Graph reduces to known graphs such as the Convex Hull, the Gabriel Graph, the β_c-Skeleton, and the Delaunay Triangulation. The γ-Graph unifies these graphs into a continuous spectrum.

Section 3. The geometric information contained in the γ-Graph is used to construct a closed piecewise linear object boundary from scattered points. The γ-Graph on the set of points is successively constricted until the hull of the pruned γ-Graph is a proper object boundary surface, passing through all the vertices. While constriction of the Delaunay Triangulation may stop unsuccessfully, the parameters of the γ-Graph provide the flexibility to find a boundary through all the vertices. The use of the geometric information in the γ-Graph by means of the γ-indicator results in good looking boundaries.

Section 4. The Flintstone scheme is both an approximation and a localization scheme, and is hierarchical. This new scheme can be applied to the constructed closed polyhedral surfaces. Its definition is based on balls, which makes the representation storage efficient, and hierarchical operations, for example intersections, computationally cheap.

In [26] a fourth computational morphology step is presented: given a closed polyhedral surface with (estimated) normal vectors at the vertices, a G^1-continuous piecewise cubic Bézier boundary is constructed in a local way. A six-split algorithm to subdivide a Bézier triangle into six micro triangles avoids thin triangles, and provides sufficient degrees of freedom to apply an optimization in order to prevent severe oscillations of the G^1-continuous surface.

All the algorithms themselves are fast enough for near real time response, even for large data sets. It is the data transfer from or to the computer's hard-disc or the graphics board that may be the bottleneck. Although the steering of the algorithms can be done interactively, the whole boundary construction process is far less interactive than the approach taken by [25]. There, 3D boundaries are constructed by interactively deforming primitive objects, such as tubes and sphere-like objects, so as to fit the given data. These deformable objects are represented by B-spline surfaces and are deformed by high level operators such as 'squeeze' and 'bend', which internally operate on the B-spline control points.

References

1. J.-D. Boissonnat. Representation of objects by triangulating points in 3-D space. In *Proceedings of the 6th International Conference on Pattern Recognition*, pp. 830 – 832, 1982.
2. J.-D. Boissonnat. Representing 2D and 3D shapes with the Delaunay triangulation. In *Proceedings of the 7th International Conference on Pattern Recognition*, pp. 745 – 748, July 1984.
3. J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, **3**(4), pp. 266 – 286, 1984.
4. B. K. Choi, H. Y. Shin, Y. I. Yoon, and J. W. Lee. Triangulation of scattered data in 3D space. *Computer Aided Design*, **20**(5), pp. 239 – 248, 1988.
5. L. DeFloriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics & Applications*, **9**(2), pp. 67 – 80, 1989.
6. M. J. DeHaemer Jr. and M. J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers & Graphics*, **15**(2), pp. 175 – 184, 1991.
7. B. Delaunay. Sur la sphère vide. In *Proceedings of the International Congress on Mathematics (Toronto 1924)*, volume 1, pp. 695 – 700. University of Toronto Press, 1928.
8. L. Devroye. The expected size of some graphs in computational geometry. *Computers & Mathematics with Applications*, **15**(1), pp. 53 –64, 1988.
9. D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, **10**(2), pp. 112 – 122, 1973.
10. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.
11. H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *Proceedings of the Boston Volume Visualization Workshop '92*, 1992.

12. O. D. Faugeras, M. Hebert, P. Mussi, and J. D. Boissonnat. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics, and Image Processing*, **25**, pp. 169 – 183, 1984.
13. K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, **18**, pp. 259 – 278, 1969.
14. D. G. Kirkpatrick and J. D. Radke. A framework for computational morphology. In In G. T. Toussaint, editor, *Computational Geometry*, Elsevier Science Publishers, pp. 217 – 248, 1985.
15. P. M. Lankford. Regionalization: Theory and alternative algorithms. *Geographical Analysis*, **1**(2), pp. 169 – 212, 1969.
16. J. O'Rourke. Polyhedra of minimal area as 3D object models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 664 – 666, 1981.
17. J. O'Rourke, Heather Booth, and Richard Washington. Connect-the-dots: A new heuristic. *Computer Vision, Graphics, and Image Processing*, **39**, pp. 258 – 266, 1987.
18. J. Ponce and O. Faugeras. An object centered hierarchical representation for 3D objects: the prism tree. *Computer Vision, Graphics, and Image Processing*, **38**(1), pp. 1 – 28, 1987.
19. M. Rioux and L. Cournoyer. The NRCC three-dimensional image data files. Technical Report CNRC 29077, National Research Council Canada, 1988.
20. F. Schmitt and B. Gholizadeh. Adaptive polyhedral approximation of digitized surfaces. *Proceedings of the SPIE – The International Society for Optical Engineering (proceedings of the SPIE conference Computer Vision for Robots, Cannes, France, December 1985)*, **595**, pp. 101 – 108, 1986.
21. F. J. M. Schmitt, B. Barsky, and W.-H. Du. An adaptive subdivision method for surfaces from sampled data. *Proceedings SIGGRAPH '86, Computer Graphics*, **20**(4), pp. 179 – 188, 1986.
22. F. J. M. Schmitt, X. Chen, and W.-H. Du. Geometric modeling from range image data. In F. H. Post and W. Barth, editors, *EUROGRAPHICS'91*, pp. 317 – 328, 1991.
23. G. T. Toussaint, editor. *Computational Morphology – A Computational Geometric Approach to the Analysis of Form*. North-Holland, 1988.
24. G. T. Toussaint. A graph theoretical primal sketch. In [23], pp. 229 – 260, 1988
25. R. C. Veltkamp. An interactive solid modeling approach to three-dimensional reconstruction. Master's thesis, Leiden University, Leiden, The Netherlands/IBM Scientific Center, Paris, France, September 1985.
26. R. C. Veltkamp. *Closed Object Boundaries from Scattered Points*. PhD thesis, Erasmus University Rotterdam, Rotterdam, The Netherlands, November 1992.
27. R. C. Veltkamp. The Flintstones: Hierarchical approximation and localization (extended abstract). In *Abstracts of the 8th European Workshop on Computational Geometry (CG'92), 12/13 March 1992, Utrecht, The Netherlands, Technical Report RUU-CS-92-10, Utrecht University, The Netherlands*, pp. 69 – 72, 1992.
28. R. C. Veltkamp. The γ-neighborhood graph. *Computational Geometry, Theory and Applications*, **1**(4), pp. 227 – 246, 1992.
29. Georges Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire — Recherche sur les parralléloèdres primitifs, Introduction et première partie. *Journal für die reine und angewandte Mathematik*, **134**, pp. 198 – 287, 1908.