2D Computational Morphology

Remco C. Veltkamp
CWI
Department of Interactive Systems
P.O. Box 4079
1009 AB Amsterdam
The Netherlands
remco@cwi.nl

Eindhoven University of Technology
Department of Computer Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands
remco@win.tue.nl

Computational Morphology is the analysis of form by computational means. The present paper is more specifically about the construction and manipulation of closed object boundaries through a set of scattered points in 2D. Results are developed in four successive stages of computational morphology:

1. impose a geometrical graph structure on the set of scattered points;
2. construct a polygonal boundary curve from this geometrical graph structure;
3. build a hierarchy of polygonal approximations together with localization information;
4. construct a geometric continuous object boundary.

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.

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; a computational geometric approach to the analysis of form is called *Computational Morphology* [20]. In particular, this paper is about the construction and subsequent manipulation of closed object

boundaries from a set of points in 2D. These points are scattered points, i.e. no structural relationship between them is known in advance and they have arbitrary position. We consider four 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, construct an object boundary by finding a closed polygon through all points.

3. Given a closed object boundary, construct a hierarchy of approximations and localization information.

4. Given a closed polygonal boundary, construct a geometrically smooth (G^1-continuous) boundary curve.

In many applications in geometric modeling, computer graphics, and object recognition, input data is available in the form of a set of 2D 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, then no structural relation between the input points may be assumed, except that they all lie on the boundary 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 straight segments, making a 2D simple closed polygonal curve. A simple closed polygon must consist of N edges, with N the number of data points. A brute force algorithm that tests all combinations of N edges out of all $\binom{N}{2}$ possible edges takes

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

time, which is at least $\Omega(N^N)$. ($\Theta(f(N))$ can be read as 'order exactly $f(N)$', $\mathcal{O}(f(N))$ as 'order at most $f(N)$', which gives an upper bound, and $\Omega(f(N))$ as 'order at least $f(N)$', which gives a lower bound, all three 'for large N'.) This is

clearly infeasible when N is large. We therefore exploit some geometric relation between the data points. Section presents the γ-Neighborhood Graph, which describes the geometric structure on the set of data points. Section describes how the γ-Neighborhood Graph is exploited for the construction of a closed polygonal object boundary.

In many real applications, a boundary constructed from a set of points consists of thousands of faces. An approximation of the object, however, is often sufficient. Localization provides bounding volume information, e.g. a sequence of bounding rectangles containing pieces of the boundary. 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 presents a new hierarchical approximation and localization scheme.

Polygonal boundaries are only C^0-continuous at the vertices; there the tangent vectors instantly change direction. A smoother curve, consisting of curved segments that interpolate the vertices and are smoothly connected, is often desired. A curve that has a continuously changing tangent vector is called G^1-continuous. Section presents a scheme to make the curve G^1-continuous. Finally, Section presents some concluding remarks.

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 [6], the Sphere of Influence Graph [21], the Relative Neighborhood Graph [14], the Gabriel Graph [11], the Convex Hull, the Delaunay Triangulation [5] and its dual Voronoi Diagram [27], the α-Shape [9], and the β-Skeleton [13]. Section presents the γ-Neighborhood Graph, a parameterized geometric graph which unifies a number of the before-mentioned ones. The γ-Neighborhood Graph is used in Section for the construction of an object boundary through all data points.

2.1. The γ-Neighborhood Graph

The γ-Neighborhood Graph has been introduced in [24]. Here we will consider one particular form. To start with, 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.

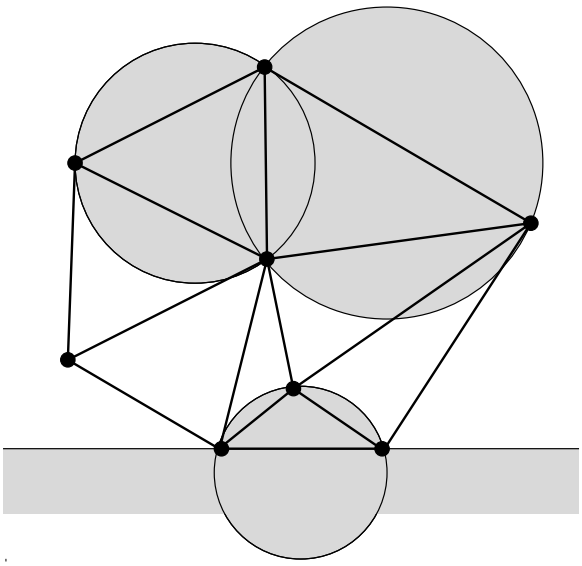


FIGURE 1. Example Delaunay Triangulation and a few 'largest empty discs'.

(A disc is a filled circle.) A region that contains no data point in its interior is called empty. If no more than three data points lie on any 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. Both situations are illustrated in Figure 1. 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, which 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, i.e contains no data points. 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 $[-1,1]$,

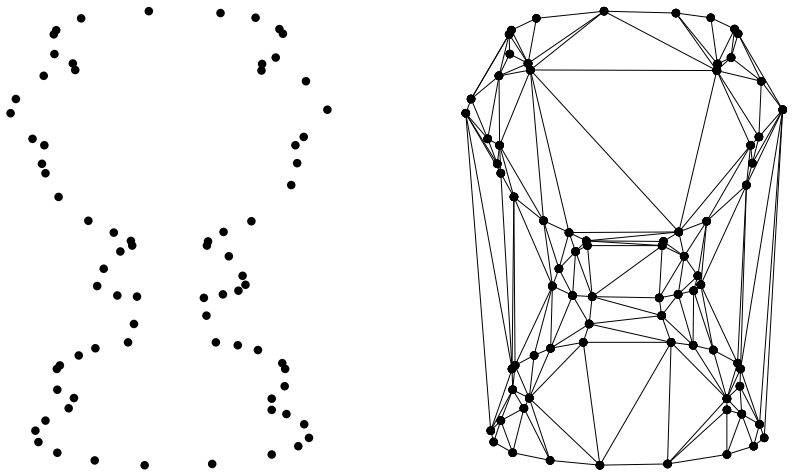


FIGURE 2. Left: 77 vertices. Right: the corresponding $\gamma([-1,1],[0,1])$ (Delaunay Triangulation).

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 $r/(1-|c_0|)$ 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.

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, 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 [24]. The 2D $\gamma([-1,1],[d,1])$ can be constructed in time $\Theta(N \log N)$ for $d = 0$, which is optimal, and $\mathcal{O}(N^2)$ for $d < 0$. Figure 2 gives an example of a point set and the $\gamma([-1,1],[0,1])$ on that set. The point set is taken from the silhouette of Ucello's chalice, which serves as the cover picture of the journal 'Computer Aided Geometric Design' [19].

3. BOUNDARY CONSTRUCTION

The boundary construction problem that we deal with is stated as follows: find a simple closed polygon 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.

One possibility is to find the boundary of minimal length, but this is NP-hard. A heuristic algorithm for the 3D case is presented in [16] but may result in an unnatural object boundary [4]. The method in [17] tries to find the simple polygon in the Delaunay Triangulation 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 [3] and [4] take the Delaunay Triangulation and successively delete edges from the outside until the resulting boundary passes through all data points. This constriction process can get locked in the sense that no more edges can be deleted without yielding an invalid boundary, while not yet all data points lie on the current boundary. Moreover, not every non-degenerate Delaunay Triangulation contains a simple closed polygon through all data points.

3.1. Boundary extraction from the γ-Graph

This section is concerned with the construction of a boundary polygon by constricting the $\gamma([-1,1],[c,1])$, $c \in [-1,0]$ 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 edge from the graph in such a way that the boundary of the new graph is properly defined (see below). A γ-Graph from which edges 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 three edges in the graph 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 that hull edge which 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 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 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

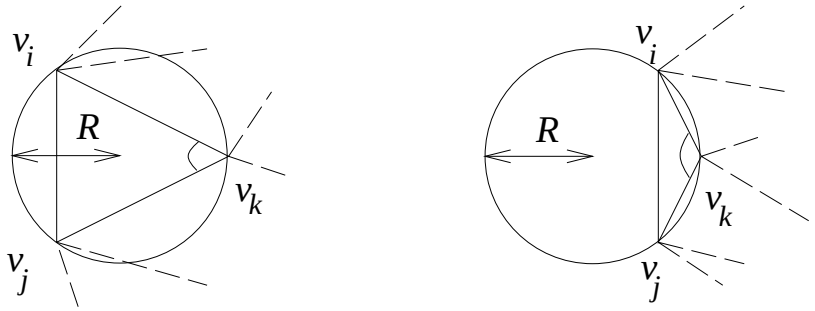


FIGURE 3. Left: γ-indicator > 0. Right: γ-indicator < 0.

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 the (possibly pruned) γ-Graph, and let d be such that $R(v_i, v_j, v_k) = r(v_i, v_j)/(1 - |d|)$. 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 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 Figure 3. 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 $r(v_i, v_j)/R(v_i, v_j, v_k) = 1 - |\gamma_{ind}|$. If $\gamma_{ind} \geq 0$, then $1 - \gamma_{ind} = r(v_i, v_j)/R(v_i, v_j, v_k)$, and if $\gamma_{ind} \leq 0$, then $1 - \gamma_{ind} = 2 - r(v_i, v_j)/R(v_i, v_j, v_k)$. So, if $\gamma_{ind} \geq 0$, then $\angle(v_i, v_k, v_j)$ increases when $r(v_i, v_j)/R(v_i, v_j, v_k)$ increases; if $\gamma_{ind} \leq 0$, 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 $1 - \gamma_{ind}$ 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 Figure 4, 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 have a properly defined boundary. If v_1v_2

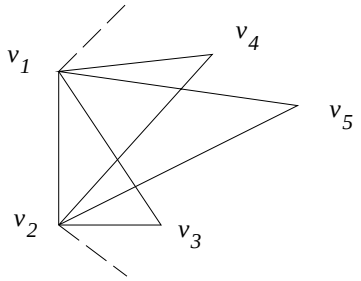


FIGURE 4. Boundary edge in a 2D $\gamma([-1,1],[c,1])$, $c \in [-1,0)$.

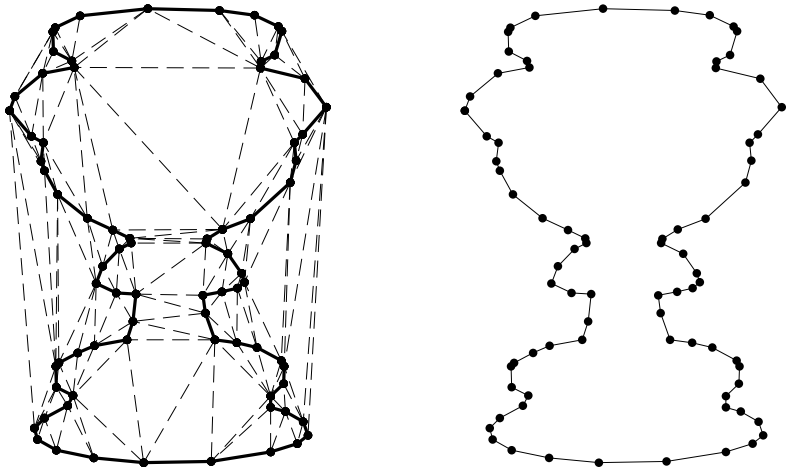


FIGURE 5. Left: $\gamma([-1,1],[0,1])$ with the constructed boundary. Right: the constructed boundary.

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. For most practical cases the constriction process on a $\gamma([-1,1],[0,1])$, i.e. the Delaunay Triangulation, successfully results in a

Hamilton polygon. The constriction process can get locked however: no more edges can be deleted without causing the boundary to be invalid, while not yet all vertices lie on the hull. Also, not all Delaunay Triangulations contain a Hamilton polygon. In both cases the constriction process can be performed on a $\gamma([-1,1],[c,1])$, $c < 0$, which contains more edges than the Delaunay Triangulation, thus providing more degrees of freedom in the constriction.

The time complexity of the entire constriction algorithm is $\mathcal{O}(E \log E)$, with E the number of edges in the γ-Graph [26]. Figure 5 illustrates the constriction algorithm on the $\gamma([-1,1],[0,1])$ of Figure 2.

4. Hierarchical approximation and localization

Two facilities are often used for efficient manipulation of complex polygonal 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 morphological detail.

If the vertices of the polygon lie on a regular grid, finding an approximation or localization is a simpler task than for an arbitrary vertex connectivity structure, or topology. In the following we will only consider approximation and localization schemes for arbitrary topologies.

The min-# approximation problem for a polygonal curve is the problem of finding an approximation polygon with the minimal number of vertices and with the error within a given bound. The min-ϵ problem is the problem of finding an approximation polygon with the minimal error and a given number of vertices. Algorithms for both optimality problems are discussed by [12]. A sequence of successively more detailed approximations requires the iterative application of these algorithms. However, in general this yields no hierarchy of approximations, and it is computationally expensive. The iterative end point fit method for approximating plane polylines is described in [8]. It starts with connecting two initial end points. If the error is larger than a chosen bound, the vertex that determines the error is connected with the two end points, yielding two new line segments. The process is repeated for the new line segments. This method is also known, especially in the cartography community, as the *Douglas–Peucker* algorithm, after [7]. This scheme provides no localization information. The strip tree [1] is a localization scheme that stores bounding areas of the open polylines approximated by the iterative end point fit method. The bounding areas are enclosing rectangles which are called strips. A level in the tree corresponds to a collection of strips enclosing the boundary, rather than to an approximation polygon. A variant of the strip tree is the Binary Line Generalization (BLG) tree [15], used for Geographic Information System applications.

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

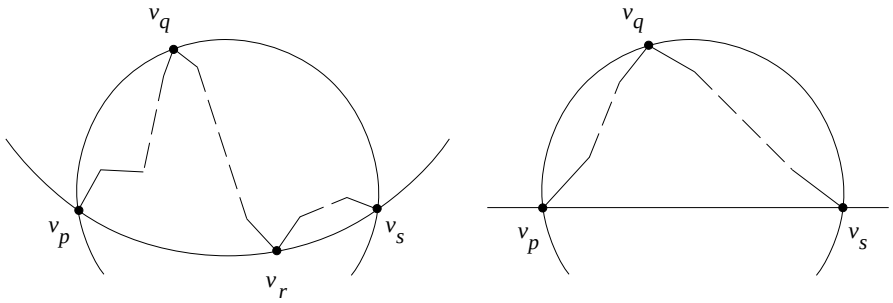


FIGURE 6. 2D Flintstones. Left: $F(P) = D(v_p, v_s; v_q) \cap D(v_p, v_s; v_r)$. Right: $F(P) = D(v_p, v_s; v_q) \cap H(v_p, v_s; v_q)$.

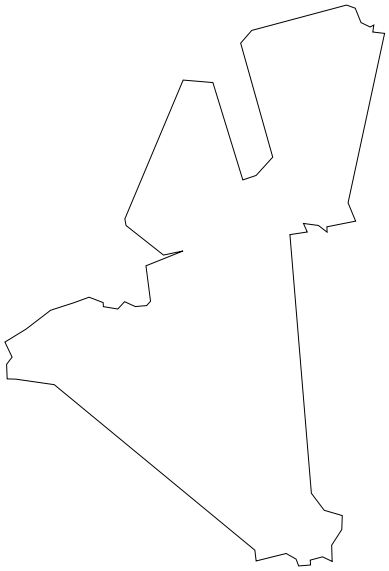
The approximation algorithm is based on the way the localization of a part of the closed polygon, which is an open polyline, is performed. The bounding area of such an open polyline is called a Flintstone. Note that the bounding areas enclose parts of the object's boundary, not the object's body.

The disc touching points a, b, and c is denoted $D(a, b, c)$. The half-plane containing c and whose boundary passes through a and b is denoted by $H(a, b; c)$. A half-plane can be considered as a disc with a radius of ∞. Before presenting the approximation algorithm, the basic bounding area, Flintstone, must be defined:

DEFINITION 2 (FLINTSTONE). *Let P be an open polyline with endpoints v_p and v_s. Let v_q be a vertex of P such that $D(v_p, v_s, v_q)$ contains all vertices lying in $H(v_p, v_s; v_q)$. If $P \subset H(v_p, v_s; v_q)$, then the Flintstone F of P is defined as $F(P) = D(v_p, v_s, v_q) \cap H(v_p, v_s; v_q)$. Otherwise $F(P)$ is the smallest of $D(v_p, v_s, v_q) \cap D(v_p, v_s, v_r)$ and $D(v_p, v_s, v_q) \cup D(v_p, v_s, v_r)$, where v_r is a vertex of P not in $H(v_p, v_s; v_q)$ such that $D(v_p, v_s, v_r)$ contains all vertices in $H(v_p, v_s; v_r)$.*

Paraphrasing, $F(P)$ is the smallest intersection or union of two discs touching v_p and v_s that contains $v_p \ldots v_s$. Note that such a vertex v_q in the definition always exists. The definition is illustrated in Figure 6 for the case that the flintstone is an intersection of two discs or a disc and a half-plane. It is easily verified that $P \subset F(P)$, so that $F(P)$ is indeed a bounding area for P.

The hierarchical approximation of a closed polygon of N vertices starts with the calculation of the smallest bounding disc (SBD), that is, the smallest disc that contains all vertices. This disc touches at least two vertices, say v_i and v_j, $i < j$. If more than two vertices lie on the boundary of the smallest bounding disc, we take two vertices that are farthest apart. Edge $v_i v_j$ is the zeroth order approximation of the polygon, dividing it into two polylines $v_i v_{i+1} \ldots v_j$ and $v_j v_{j+1} \ldots v_i$ (here and in the rest of this section the indices are taken modulo N).



For the next level in the hierarchy of approximations, let us consider a polyline $P = v_p \ldots v_s$. The approximation depends on the flintstone $F(P)$. If $F(P) = D(v_p, v_s, v_q) \cap H(v_p, v_s; v_q)$, then P is approximated by the edges $v_p v_q$ and $v_q v_s$. If $F(P) = D(v_p, v_s, v_q) \cap D(v_p, v_s, v_r)$ and if the radius of $D(v_p, v_s, v_q)$ is smaller than the radius of $D(v_p, v_s, v_r)$, then P is approximated by the edges $v_p v_q$ and $v_q v_s$, otherwise by $v_p v_r$ and $v_r v_s$. If instead $F(P) = D(v_p, v_s, v_q) \cup D(v_p, v_s, v_r)$ and if the radius of $D(v_p, v_s, v_q)$ is larger than the radius of $D(v_p, v_s, v_r)$, then P is approximated by the edges $v_p v_q$ and $v_q v_s$, otherwise by $v_p v_r$ and $v_r v_s$. In other words, the new edges are made with either v_q or v_r, whichever lies on the disc that contributes most to the 'width' of the Flintstone.

An edge $v_p v_s$ is not subdivided if $p + 1 = s$, in which case this edge is in the original polygon. In order to construct the complete hierarchy, the iteration is performed until all vertices are contained in the approximation polygon, which then coincides with the original one. By definition, an edge $v_p v_{p+1}$ has no flintstone.

If there exists a disc touching v_p and v_s that contains $v_{p+1} \ldots v_{s-1}$, then the flintstone bounding $v_{p+1} \ldots v_{s-1}$ is the intersection of two discs or a disc and a half-plane. By construction of the approximation, each approximated polyline is contained in a disc. So, *all* flintstones are the *intersection* of two discs or a disc and a half-plane. In particular the degenerate case that a flintstone is the union of a half-plane and a disc simply cannot occur. That would be the case if a vertex v_i, $p < i < s$, lies on the line through v_p and v_s and outside the line segment $v_p v_s$, but by construction of the approximation, this never happens. The shape of the intersection of two discs gave rise to the name 'flintstone'.

A binary tree is a natural data structure to store the hierarchical approximation. The root, level zero of the tree, contains vertices v_i and v_j. The left subtree stores vertices $v_{i+1}, \ldots, v_{j-1}$ such that the symmetric or infix order traversal yields the successive vertices of the polygon. The right subtree stores vertices $v_{j+1}, \ldots, v_{i-1}$ analogously. A level-ℓ approximation simply corresponds to the levels $0, \ldots, \ell$ of the tree. The bounding areas are stored as follows. The root stores the smallest bounding disc. A node containing vertex v_q at level ℓ, $\ell \geq 1$, of the tree, contains $D(v_p, v_s, v_q)$ and $D(v_p, v_s, v_r)$, where v_p is the predecessor and v_s the successor of v_q at approximation level ℓ.

The worst case time complexity to build the complete Flintstone tree is $\Theta(N^2)$ and the storage complexity is linear [23]. Figure 7 shows a few approximations of an arbitrary polygon (which happens to be the outer border of the mainland of The Netherlands, Belgium and Luxembourg). The lower right approximation is obtained by only refining an edge if the approximation error of *that edge* (rather than the whole polygon) is larger than a given bound. Such an approximation is called adaptive.

5. Smooth boundary construction

Smooth boundaries can easily be constructed by piecewise polynomials. The Bézier formulation is a convenient method to describe other polynomial schemes

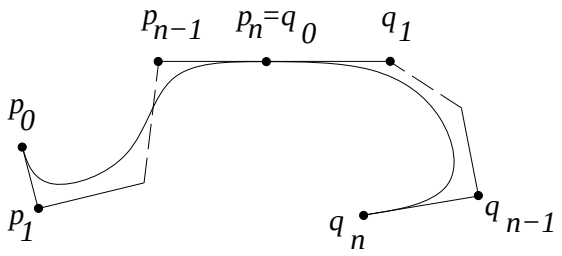


FIGURE 8. G^1-continuity conditions for two Bézier segments.

as well as to develop new schemes. (Bézier curves and surfaces were independently developed by de Casteljau at the Citroën and by Bézier at the Renault automobile company, but de Casteljau's development was never published, so that this curve and surface scheme was named after Bézier.) The Bézier formulation is often used because of the geometrical significance of the coefficients.

A Bézier curve of degree n is defined as

$$P(t) = \sum_{i=0}^{n} p_i B_i^n(t),$$

with B_i^n the Bernstein polynomials

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}, \ 0 \leq u \leq 1,$$

which are often called blending functions and the p_i are called weights or control points.

5.1. G^1-continuous boundary curve

In this section we construct a smooth boundary as a G^1-continuous closed Bézier curve. It is readily verified that $P(0) = p_0$ and $P(1) = p_n$, so that the curve interpolates the first and last control points. The derivative has the following form:

$$P^{(1)}(u) = n \sum_{i=0}^{n-1} (p_{i+1} - p_i) B_i^{n-1}(u).$$

In particular $P^{(1)}(0) = n(p_1 - p_0)$ and $P^{(1)}(1) = n(p_n - p_{n-1})$, so that the tangent vector at $P(0)$ lies on the line through p_0 and p_1 and the tangent vector at $P(1)$ lies on the line through p_{n-1} and p_n.

Let us consider two Bézier segments P with control points p_i and Q with control points q_i, and a common end point, say $P(1) = Q(0)$, so that $p_n = q_0$. As is just shown, the tangent vector at $P(1)$ has direction $p_n - p_{n-1}$ and the tangent vector at $Q(0)$ has direction $q_1 - q_0$. So, for the two tangent vectors to have the same direction, p_{n-1}, $p_n = q_0$ and q_1 should be collinear, see Figure 8.

Note that a common tangent line is not sufficient for G^1-continuity, since the tangent vectors must additionally have the same direction. In other words, the two curves must have the same orientation, otherwise they join with a sharp cusp.

First we estimate the tangent vector at each vertex, then we construct a G^1 curve that interpolates vertices and tangent vectors. Let us denote the tangent vector at data point v_i with T_i. This tangent line can be estimated by weighting the vectors $(v_i - v_{i-1})$ and $(v_{i+1} - v_i)$ and normalize the weighted sum:

1. Weight by length: take $(v_i - v_{i-1}) + (v_{i+1} - v_i)$, giving $v_{i+1} - v_{i-1}$, and normalize to unit length. The reasoning behind this method is that the larger of the segments $v_{i-1}v_i$ and v_iv_{i+1} corresponds to a larger part of the curve, and should affect the tangent vector most.

2. Weight uniformly: take $(v_i - v_{i-1})/\|v_i - v_{i-1}\| + (v_{i+1} - v_i)/\|v_{i+1} - v_i\|$, and normalize to unit length.

3. Weight by inverse length: take $(v_i - v_{i-1})/\|v_i - v_{i-1}\|^2 + (v_{i+1} - v_i)/\|v_{i+1} - v_i\|^2$, and normalize to unit length. The idea of this method is that a close neighboring vertex knows more about the local curve tangent and should have a larger weight than the far vertex.

All these methods only take into account the neighboring vertices v_{i-1} and v_{i+1}. Other methods could be applied that use more vertices and for example estimate the tangent line by a least-squares fit. The analogue of method 1 for surface normal estimation has been reported to work well [18]; therefore method 1 is used here.

In order to get a closed G^1 curve we construct a Bézier segment between each pair of consecutive vertices. Let us consider the degree n Bézier segments P between v_{i-1} and v_i, and Q between v_i and v_{i+1} (again the indices are considered modulo N). In order to interpolate the vertices, we must set $p_0 = v_{i-1}$, $p_n = q_0 = v_i$, and $q_n = v_{i+1}$. For the derivative vectors $P^{(1)}(1)$ and $Q^{(1)}(0)$ to be collinear, p_{n-1} and q_1 must lie on the line through T_i, denoted by $Tline_i$. If $Tline_{i-1}$ intersects $Tline_i$, and $Tline_i$ intersects $Tline_{i+1}$, we can set $n = 2$, p_1 to $Tline_{i-1} \cap Tline_i$, and q_1 to $Tline_i \cap Tline_{i+1}$. The resulting Bézier segments are then completely defined, and have collinear tangent vectors. All other segments are defined in the same way, so that a closed quadratic curve is constructed. However, with only tangent *line* continuity the tangent *vectors* can still have opposite directions, which gives sharp cusps, i.e. not G^1-continuity.

In order to ensure G^1-continuity we need more degrees of freedom. This is obtained by using one more control point, so that we get cubic Bézier segments. Now we must set $p_0 = v_{i-1}$, $p_3 = q_0 = v_i$, and $q_3 = v_{i+1}$, to interpolate the vertices. The control points p_1 and p_2 are determined by the following heuristic method, which gives good results in many practical cases. Vertex p_3 is orthogonally projected onto $Tline_{i-1}$, giving p_3^*. Control point p_1 is then set to $p_0 + (p_3^* - p_0)/3$. Analogously, $p_2 = p_3 + (p_0^* - p_3)/3$, where p_0^* is the

FIGURE 9. Vertices and constructed linear interpolation (above), and the constructed Bézier control polygon and G^1 curve.

orthogonal projection of p_0 onto $Tline_i$. All other segments are defined in the same way, so that a closed cubic curve is constructed. The curve is G^1-continuous at v_i if $P^{(1)}(1)$ and $Q^{(1)}(0)$ have the same direction, which is the case when p_3^* and q_0^* lie at opposite sides of v_i on $Tline_i$. This condition is satisfied for many sets of vertices whose successive tangent lines do not change direction wildly. In other cases, the tangent line at a vertex should be changed in order for the projection method above to work properly.

The above algorithm clearly has time complexity $\mathcal{O}(N)$. Figure 9 shows an example of this algorithm to construct a G^1-continuous curve, using method 1 to estimate the tangent line.

Cubic curves are necessary and sufficient to achieve even G^2-continuity, see [10] and [2]. In 2D the construction of a G^1-continuous boundary in Bézier form is a straightforward procedure. The 3D case is much harder. A new solution to the 3D problem has been presented in [22].

6. Concluding remarks

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

1. 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-Skeleton, and the Delaunay Triangulation. The γ-Graph unifies these graphs into a continuous spectrum.

2. 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, 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.

3. The flintstone scheme is both an approximation and a localization scheme, and is hierarchical. This scheme can be applied to the constructed polygonal boundaries. Its definition is based on discs, which makes the representation storage efficient, and hierarchical operations, for example intersections, computationally cheap.

4. Given a polygonal boundary with (estimated) tangent vectors at the vertices, a G^1-continuous piecewise cubic Bézier boundary is constructed in a local way.

Care has been taken to introduce new concepts that naturally generalize from 2D to 3D [25]. This is exhibited by the definition of the γ-graph, the deletion rule in the constriction algorithm, and the definition of the flintstone scheme.

References

1. Dana H. Ballard (1981). Strip trees: A hierarchical representation for curves. *Communications of the ACM*, 24(5), pp. 310 – 321.
2. sc Wolfgang Böhm (1985). Curvature continuous curves and surfaces. *Computer Aided Geometric Design*, 2, pp. 313 – 323.
3. J.-D. Boissonnat (1984). Representing 2D and 3D shapes with the Delaunay triangulation. In *Proceedings of the 7th International Conference on Pattern Recognition*, pp. 745 – 748.
4. Jean-Daniel Boissonnat (1984). Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4), pp. 266 – 286.
5. B. Delaunay (1928). Sur la sphère vide. In *Proceedings of the International Congress on Mathematics (Toronto 1924)*, volume 1, pp. 695 – 700. University of Toronto Press.
6. Luc Devroye (1988). The expected size of some graphs in computational geometry. *Computers & Mathematics with Applications*, 15(1), pp. 53 –64.
7. David H. Douglas and Thomas K. Peucker (1973). 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.
8. Richard O. Duda and Peter E. Hart (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.
9. Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, IT-29(4), pp. 551 – 559.
10. Gerald Farin (1982). Visually C^2 cubic splines. *Computer Aided Design*, 14(3), pp. 137 – 139.
11. K. R. Gabriel and R. R. Sokal (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18, pp. 259 – 278.
12. H. Imai and M. Iri (1988). Polygonal approximations of a curve – formulations and algorithms. In Toussaint [20], pp. 71 – 86.
13. David G. Kirkpatrick and John D. Radke (1985). A framework for computational morphology. In G. T. Toussaint, editor, *Computational Geometry*, pp. 217 – 248. Elsevier Science Publishers.
14. P. M. Lankford (1969). Regionalization: Theory and alternative algorithms. *Geographical Analysis*, 1(2), pp. 169 – 212.
15. Peter van Oosterom and Jan van den Bos (1989). An object-oriented approach to the design of geographic information systems. *Computers & Graphics*, 13(4), pp. 409 – 418.
16. Joseph O'Rourke (1981). Polyhedra of minimal area as 3D object models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 664 – 666.
17. Joseph O'Rourke, Heather Booth and Richard Washington (1987). Connect-the-dots: A new heuristic. *Computer Vision, Graphics, and Image Processing*, 39, pp. 258 – 266.
18. Kenneth Sloan (1991). Surface normal (summary). *Usenet comp.graphics*

article, (September).
19. Christof Thoenes (1984). Uccello's chalice. *Computer Aided Geometric Design*, 1, pp. 97 – 99.
20. G. T. Toussaint, editor (1988). *Computational Morphology – A Computational Geometric Approach to the Analysis of Form*. North-Holland.
21. G. T. Toussaint (1988). A graph theoretical primal sketch. In *Computational Morphology – A Computational Geometric Approach to the Analysis of Form* [20], pp. 229 – 260.
22. Remco C. Veltkamp (1992). Closed G^1-continuous cubic Bézier surfaces. Technical Report CS-R9226, CWI.
23. Remco C. Veltkamp (1992). The Flintstones: Hierarchical approximation and localization (extended abstract). In *Abstracts of the 8th European Workshop on Computational Geometry (CG'92), Technical Report RUU-CS-92-10, Utrecht University, The Netherlands*, pp. 69 – 72.
24. Remco C. Veltkamp (1992). The γ-neighborhood graph. *Computational Geometry, Theory and Applications*, 1(4), pp. 227 – 246.
25. Remco C. Veltkamp (1993). 3D computational morphology. *Computer Graphics Forum*, 12(4), Proceedings Eurographics '93).
26. Remco C. Veltkamp (1993). Boundaries through scattered points of unknown density. Submitted to *CVGIP: Graphics Models and Image Processing*.
27. Georges Voronoï (1908). 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.