Efficient generation of simple polygons for characterizing the shape of a set of points in the plane

Matt Duckham[1], Lars Kulik[2], Mike Worboys[3], Antony Galton[4]

1. Department of Geomatics
University of Melbourne, Victoria, 3010, Australia
2. Department of Computer Science and Software Engineering
University of Melbourne, Victoria, 3010, Australia
3. National Center for Geographic Information and Analysis
University of Maine, Orono, ME 04469, USA
4. Department of Computer Science
University of Exeter, Exeter EX4 4QF, UK

Abstract

This paper presents a simple, flexible, and efficient algorithm for constructing a possibly non-convex, simple polygon that characterizes the shape of a set of input points in the plane, termed a *characteristic shape*. The algorithm is based on the Delaunay triangulation of the points. The shape produced by the algorithm is controlled by a single normalized parameter, which can be used to generate a finite, totally ordered family of related characteristic shapes, varying between the convex hull at one extreme and a uniquely defined shape with minimum area. An optimal $O(n \log n)$ algorithm for computing the shapes is presented. Characteristic shapes possess a number of desirable properties, and the paper includes an empirical investigation of the shapes produced by the algorithm. This investigation provides experimental evidence that with appropriate parameterization the algorithm is able to accurately characterize the shape of a wide range of different point distributions and densities. The experiments detail the effects of changing parameter values and provide an indication of some "good" parameter values to use in certain circumstances.

Email address: `mduckham@unimelb.edu.au` (Matt Duckham[1], Lars Kulik[2], Mike Worboys[3], Antony Galton[4]).
URL: `www.duckham.org` (Matt Duckham[1], Lars Kulik[2], Mike Worboys[3], Antony Galton[4]).

1 Introduction

The construction of convex hulls is a fundamental operation in computational geometry. In the Cartesian plane, the convex hull of a set of points S is the smallest convex polygon which contains all points in S. However, for sets of points with a pronounced non-convex distribution the convex hull can never provide good characterization of that distribution.

In this paper we present an algorithm for building "non-convex hulls." The algorithm is as efficient as an optimal convex hull algorithm, $O(n \log n)$ computation time for n points. For a finite set of input points P, the algorithm produces a simple, possibly non-convex polygon that contains all the points in P and is contained within and possibly equal to the convex hull. We refer to the polygons produced by the algorithm as "characteristic shapes" or simply χ (chi) shapes.

Two features of our characteristic shapes are worth highlighting at this point. First, while there exists only one convex hull for a set of points there can be many different characteristic shapes. There is no "correct" characteristic shape. We argue that in many cases the algorithm yields a *better* characterization of distribution of a set of points than the convex hull. To illustrate, figure 1 shows a gallery of convex and characteristic shapes for some example point sets with clearly non-convex distributions. However, deciding precisely what constitutes a "better characterization" of the distribution of a set of points is as much a matter for human cognition and preference as for computational geometry. Despite this inherent underspecification in the problem statement, our contention is that the characteristic shapes produced by our algorithm are useful. Further, this paper explores experimentally some of the attributes of a shape which may constitute "better" or "worse" characterizations of the distribution of a set of points, and proposes some natural choices for parameterizing the characteristic shape algorithm in a way that generates a uniquely defined result.

Second, characteristic shapes are simple (Jordan) polygons, homeomorphic to the closed unit disk. Thus, characteristic shapes are simply connected (all of one piece containing no holes nor islands) and regular. In some cases, however, the distribution of a set of points may be best characterized by multiple (possibly non-convex) polygons enclosing disconnected regions of space (e.g., an "i" or "="shape). In this paper we do not consider directly such cases, and are primarily concerned with cases where the distribution of points can be adequately characterized as a single simple polygon. However, it is possible to deal with such cases indirectly by first preprocessing the input point set to partition it into subsets, each of which may be adequately characterized by a single simple polygon, explored briefly in section 6.4. In other cases where the

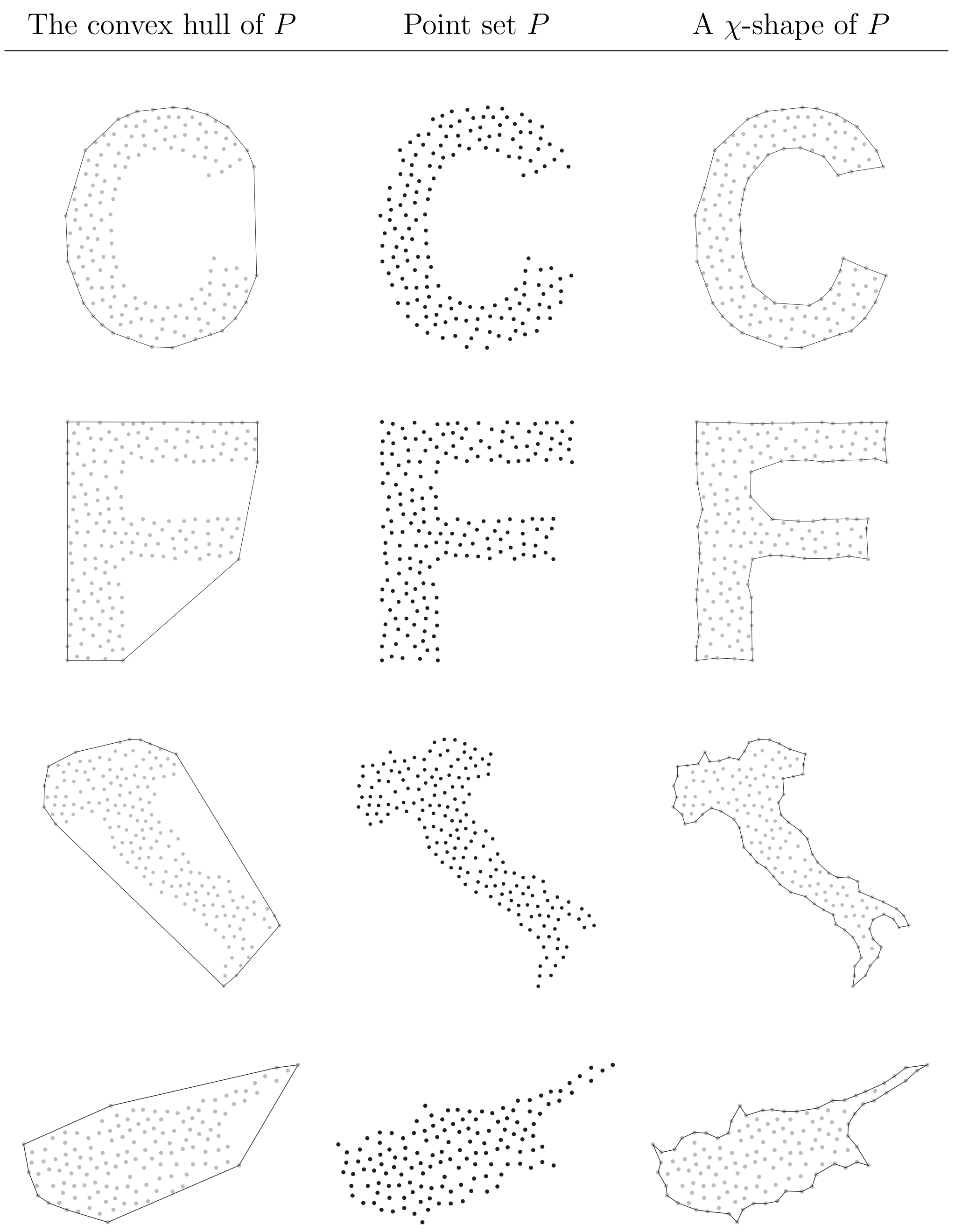


Fig. 1. Gallery of convex hulls and χ-shapes for several point sets in the plane

distribution of points is best characterized using a polygon containing one or more holes (e.g., an "8" shape), the characteristic shape algorithm presented in this paper will not be able to generate these holes. It will, however, still successfully generate a characterization of the external edge of such a region.

2 Related work

An early, and influential, attempt to characterize the shape of a set of points is due to [1], which introduced a construction known as "α-shape" as a generalization of the convex hull. For a finite set P of points in the plane, the "α-hull" for $\alpha \neq 0$ is the intersection of all closed discs of radius $1/\alpha$ containing all the points of P (where for negative values of α a closed disk of radius, $1/\alpha$ is interpreted as the complement of an open disk of radius $-1/\alpha$). As α approaches 0, the α-hull approaches the ordinary convex hull, and therefore the 0-hull is stipulated to *be* the convex hull. The α-shape is a straight-line graph (usually a polygon) derived in a straightforward manner from the α-hull. When $\alpha = 0$, this is the convex hull, and for large negative values of α it is P itself.

A related notion, $\mathcal{A}$-shape, was introduced in [2]. Given a finite set of points P, and a set $\mathcal{A}$ (which evidently needs to be disjoint from P, although the authors do not specify this), we can define the $\mathcal{A}$-shape of P by first constructing the Voronoi diagram for $\mathcal{A} \cup P$ and then joining together any pair of points $p, q \in P$ whose Voronoi cells both border each other and border some common Voronoi cell containing a point of $\mathcal{A}$. The edges pq belong to the Delaunay triangulation of $\mathcal{A} \cup P$: they are the "$\mathcal{A}$-exposed" edges of the triangulation. An important issue discussed in the paper is how to choose $\mathcal{A}$ so that the $\mathcal{A}$-shape of P is "adequate." In a later paper [3], the $\mathcal{A}$-shape is used as the basis for an "onion-peeling" method, by analogy with the popular convex onion-peeling method for organizing a set of points and extracting a "central" embedded convex shape from them [4].

Two rather different constructs, *r-shape* and *s-shape*, were defined in [5] as follows. The initial set of points P is assumed to be a *dot pattern*, that is, a planar point set whose elements are "clearly visible as well as fairly densely and more or less evenly distributed." To obtain the s-shape, the plane is partitioned into a lattice of square cells of side-length s. The s-shape is simply the union of lattice cells containing points of P. The authors suggest a procedure for optimizing the value of s so that the s-shape best approximates the perceived shape of the dot pattern. For the r-shape, they first construct the union of all disks of radius r centered on points of P. For points $p, q \in P$, the edge pq is selected if and only if the boundaries of the disks centered on p and q intersect in a point which lies on the boundary of the union of all the disks. The r-shape of P is the union of the selected edges, and the authors show that this can be computed in time $O(n)$, where n is the cardinality of P. They note that the r-shape is a subgraph of the α-shape in the sense of [1]. Regarding the selection of r, they note that "to get a perceptually acceptable shape, a suitable value of r should be chosen, and there is no closed form solution to this problem," and that moreover "'perceptual structure' of P ... will vary from one person to another to a small extent."

An alternative method, also designed to be applied to dot patterns, was proposed by [6]. This procedure starts by constructing the convex hull of the points, and then uses a “split and merge” procedure to successively insert extra edges or smooth over zigzags. The splitting procedure results in a highly jagged outline, which is then made smoother by the merging procedure. The resulting outline gives an approximation to the perceived shape of the dot pattern. The complexity of the procedure is limited by the complexity of finding the initial convex hull, $O(n \log n)$.

The use of Voronoi diagrams for constructing regions from point-sets has also been advocated in the context of GIS [7]. In this context, the set P consists of points known to be in a certain region, for which an approximation to the boundary is required. It is assumed that in addition to P another point-set P' is given, consisting of points known to lie *outside* the region to be approximated. From the Voronoi diagram for $P \cup P'$, the method simply selects the union of the Voronoi cells containing points of P. The resulting shape differs from the characteristic shapes constructed in this paper in that the original point-set lies entirely in its interior. Depending on one’s purposes, this feature may either be desirable or undesirable.

A similar method [8] is based on Delaunay triangulations. Given sets P and P' as before, the Delaunay triangulation of $P \cup P'$ is constructed, and then the midpoint of every edge which joins a point in P to a point in P' is selected. The final region is produced by joining all pairs of selected midpoints belonging to edges of the same triangle.

In all these cases, as with the method we describe in this paper, the goal is to generate a region which in some sense “covers” the given set of points, some of which may end up on the boundary of the region, others in its interior. A somewhat different, though related problem, is to generate a region such that *all* of the points lie on its boundary. A typical application, in three dimensions, works with points which are sampled from the surface of some three-dimensional object, the intention being to reconstruct the entire surface from the samples. Methods which have been used for this problem include the Power Crust method in [9, 10], which first generates a finite union of balls as an approximation to the medial axis transform of the object, and then derives from this a piecewise-linear approximation to the object’s surface—the power crust. The balls chosen are a subset of the Voronoi balls for the set of samples. An alternative approach to the same problem, based on the Delaunay tessellation rather than the Voronoi, is given in [11].

Given that a considerable amount of research has been done on finding regions corresponding to point-sets, and much of this research takes convex hulls, Voronoi diagrams, or Delaunay triangulations as its starting point, it is perhaps surprising that our Delaunay-based method, though extremely simple in conception, does not appear to have been proposed before. With so many different methods in existence, all giving different results, there is a clear need for some systematic comparison of the methods and evaluation of their relative merits in different application contexts. Some initial work suggesting criteria upon which to base such a systematic comparison is given in [12]. However, in the remainder of this paper we concentrate primarily on the presentation of our algorithm and its properties and the empirical evaluation of the algorithm's performance.

3 The χ (chi) algorithm

For a finite set of at least three points in the Cartesian plane $P \subset \mathbb{R}^2$, the characteristic shape algorithm yields a possibly non-convex area with a shape that "characterizes" the distribution of the input point set. All the sets under consideration in this paper are sets of points in the Cartesian plane $\mathbb{R}^2$, and these sets are assumed to be finite. The χ-shape produced by the algorithm has the properties that:

(1) it is a simple polygon;
(2) it contains all the points of P; and
(3) it bounds an area contained within and possibly equal to the convex hull of the points of P.

The χ-shape algorithm is based on "shaving" exterior edges (edges that bound only one triangle) from a triangulation of the input point set in order of the length of edges and subject to a regularity constraint. The algorithm itself has a time complexity of $O(n \log n)$, where n is the number of input points. Although the algorithm is presented in detail in the following section, it can be summarized as comprising the following steps for an input point set P and a length parameter l:

(1) Generate the Delaunay triangulation of the set of input points P;
(2) Remove the longest exterior edge from the triangulation such that:
    (a) the edge to be removed is longer than the length parameter l; and
    (b) the exterior edges of the resulting triangulation form the boundary of a simple polygon;
(3) Repeat 2. as long as there are more edges to be removed
(4) Return the polygon formed by the exterior edges of the triangulation

In exploring the algorithm more carefully, we begin with some preliminary material on the underlying structure for the triangulation, a combinatorial map (3.1). Then we present the algorithm itself (3.2). In the next section (4) we discuss the properties of the algorithm and the χ-shapes, introduced above, in more detail.

3.1 Combinatorial maps

The χ algorithm is based on an explicit orientation of the edges in the triangulation around a given vertex. The orientation of edges in a graph can be represented by an oriented *combinatorial map*. Introduced in [13], combinatorial maps are well-known in computational geometry, and are the formal basis of several common data structures, such as the winged-edge and half-edge data structures [14,15]. The following definitions build on the functional specification of combinatorial maps given in [16].

Definition 1 *A* (2-dimensional) oriented combinatorial map*, or just* map*, $\mathcal{M}$, is a triple $\langle D, \Theta_0, \Theta_1 \rangle$, where D is a finite set of elements, called* darts*, Θ_0 is an involutory bijection*[1] *on D, and Θ_1 is a bijection on D. We may also assume that Θ_0 has no fixed points.*

Θ_0 partitions the set of darts into sets of pairs of darts, and each such pair is called an *edge* of map $\mathcal{M}$. Each of the cycles of Θ_1 represents a *vertex* of $\mathcal{M}$. It is straightforward to use Θ_0 and Θ_1 to calculate the ordering of edges round faces in a combinatorial map. The cycles of the composition $\Theta_0\Theta_1$ gives the ordering of darts, and converting the darts to their (unique) associated edges gives the ordering of edges. In general, a *face* of map $\mathcal{M}$ is a cycle of edges associated with a cycle of darts in $\Theta_0\Theta_1$. Alternatively, focusing on vertices rather than edges, we can consider the cycle of vertices (uniquely) associated with the edges to be the face.

To illustrate, figure 2 provides an example of a triangulation where:

- $D = \{1, 2, 3, 4, 5, ..., 28\}$;
- $\Theta_0 = (1\ 13)(2\ 3)(4\ 12)(5\ 16)(6\ 22)(7\ 8)...$ (in cyclic notation); and
- $\Theta_1 = (1\ 2)(3\ 4\ 5\ 6\ 7)(8\ 9\ 10)(11\ 12\ 13)...$ (in cyclic notation).

Let $E = E(\mathcal{M})$ be the set of edges, $F = F(\mathcal{M})$ be the set of faces, and $V = V(\mathcal{M})$ be the set of vertices of $\mathcal{M}$. The surjective functions $edge : D \to E$ and $vertex : D \to V$ provide the edge and vertex which contains a dart, respectively (i.e., $edge : d \mapsto \{d, \Theta_0 d\}$ and $vertex : d \mapsto v \in V$ such that d is a dart in v).

Definition 2 *Let $\mathcal{M}$ be a given combinatorial map. A* triangle *in $\mathcal{M}$ is a face in $F = F(\mathcal{M})$ that is a 3-cycle of edges associated with a 3-cycle of darts*

[1] A bijection is a function that is both injective (one-to-one) and surjective (onto). An involution is a function that is its own inverse, e.g. $\Theta_0(\Theta_0(x)) = x$.

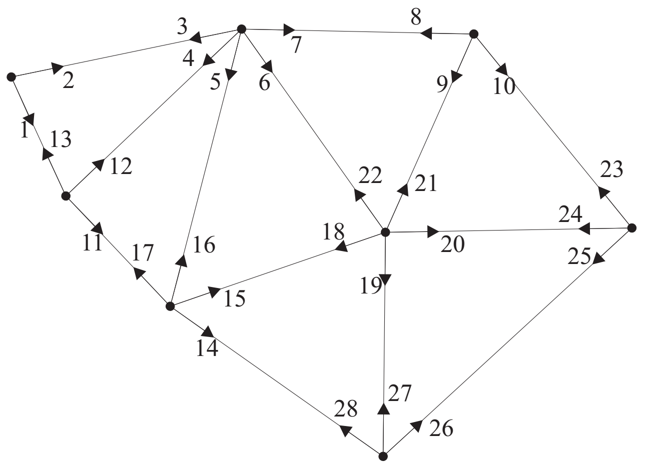


Fig. 2. Example triangulation structured as a combinatorial map

in $\Theta_0\Theta_1$. Alternatively and equivalently, a triangle is a 3-cycle of vertices associated with a 3-cycle of darts in $\Theta_0\Theta_1$.

Definition 3 *A* triangulation Δ *is a combinatorial map which has the property that every edge in E belongs to either one or two triangles.*

From now on we will work with triangulations rather than more general combinatorial maps. Suppose from now on that our underlying triangulation is Δ.

Definition 4 *An* interior edge *of Δ is an edge that belongs to two triangles in Δ. A* boundary edge *of Δ is an edge that belongs to exactly one triangle in Δ. The* edge-interior *of Δ is the collection of its interior edges. The* edge-boundary *of Δ is the collection of its boundary edges.*

Definition 5 *An* interior vertex *of Δ is a vertex containing no boundary edges. A* boundary vertex *of region Δ is a vertex containing boundary edges. The* vertex-interior *of Δ is the collection of its interior vertices. The* vertex-boundary *of Δ is the collection of its boundary vertices.*

Definition 6 *A triangle is an* interior triangle *of Δ if all its edges are interior edges of Δ. A triangle is a* boundary triangle *of Δ if at least one of its edges is a boundary edge of Δ. The* triangle-interior *of Δ is the collection of interior triangles of Δ. The* triangle-boundary *of Δ is the collection of its boundary triangles.*

Definition 7 *A triangulation Δ is* regular *if each boundary vertex of Δ contains exactly two boundary edges of R.*

Definition 8 *A* planar embedding *of Δ is a function $f : V(\Delta) \to \mathbb{R}^2$ from the set of vertices in Δ to points in the plane. The length of an edge $||e||$ is the Euclidean distance $\delta(a, b)$ where $a = vertex(d)$, $b = vertex(\Theta_0(d))$, and $d \in e$*

is a dart of e.

3.2 Algorithm

The χ algorithm has two components. The main component (Algorithm 1) takes a set of points and a non-negative length parameter l as input. Algorithm 1 constructs the Delaunay triangulation of the input point set (line 1) and the list of boundary edges B sorted in descending order of edge length (lines 1–1). Determining whether a particular edge is a boundary edge can be achieved in constant time by checking for 3-cycles of darts in the combinatorial map, as shown by the "e-boundary" function $e\text{-}\partial : E(\Delta) \to \{\text{true}, \text{false}\}$ defined as follows:

$$e\text{-}\partial : \{d_1, d_2\} \mapsto \begin{cases} \text{false} & \text{if } \Theta_0\Theta_1\Theta_0\Theta_1\Theta_0\Theta_1 d_1 = d_1 \text{ and} \\ & \quad \Theta_0\Theta_1\Theta_0\Theta_1\Theta_0\Theta_1 d_2 = d_2 \\ \text{true} & \text{otherwise} \end{cases} \tag{1}$$

Determining whether a particular vertex is a boundary vertex could be achieved in a similar way, by checking whether any of the edges incident with that vertex are boundary edges. However, because a vertex may have any number of incident edges, using this approach can increase the computational complexity of the χ algorithm. Instead, lines 1–1 in Algorithm 1 pre-process the set of edges to initialize a "v-boundary" function $v\text{-}\partial : V(\Delta) \to \{\text{true}, \text{false}\}$, which determines whether a vertex is a boundary vertex or not.

With all the preprocessing completed, the algorithm then cycles through each boundary edge in order (longest first, lines 1–1). At each iteration the longest boundary edge is removed (line 1) from B. Additionally, this edge will be removed from the triangulation if:

(1) the resulting triangulation is regular; and
(2) the edge length is at least l (line 1).

When an edge e is removed, the two new boundary edges that are revealed by the removal of e are added to the list of boundary edges B, respecting the edge-length ordering of B (line 1). Additionally, the $v\text{-}\partial$ function is updated to store the boundary vertex revealed by the removed edge (1). The boundary edges (and so vertex) that are revealed by the removal of an edge can be found using the combinatorial map. For this purpose we define the function

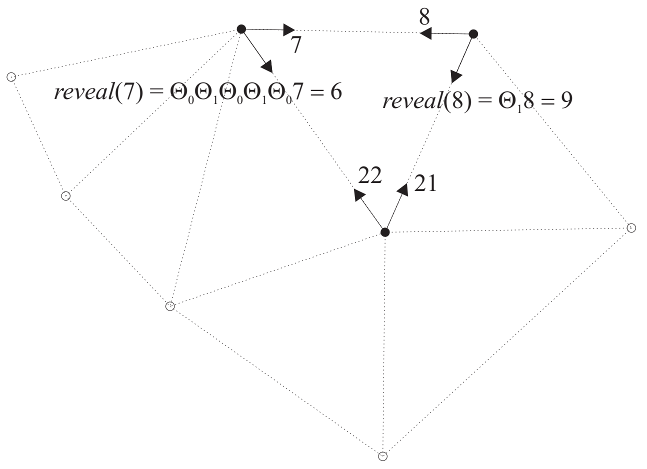


Fig. 3. Darts belonging to the edge-interior of a boundary triangle accessed using the *reveal* function

reveal : $D \rightarrow D$ as follows.

$$reveal : d \mapsto \begin{cases} \Theta_1 d & \text{if } \Theta_0\Theta_1\Theta_0\Theta_1\Theta_0\Theta_1 d = d \\ \Theta_0\Theta_1\Theta_0\Theta_1\Theta_0 d & \text{otherwise} \end{cases} \tag{2}$$

Figure 3 helps to explain the idea behind Equation 2. The *reveal* function applied to dart d maps to the dart $d' \in vertex(d)$ such that d' is a dart of the edge which will be revealed at the boundary if e were removed from the triangulation. The algorithm terminates when B is empty.

Algorithm 2 presents an efficient test to decide whether or not the regularity constraint in line 1 is satisfied. Originally applied in a completely different context, algorithm 2 is derived from an idea first developed in [17] as part of their algorithm for detecting topological changes in regions monitored by geosensor networks.

Since at each iteration only one edge is removed, the effects on the regularity of the triangulation of removing this edge can be checked by examining the third vertex of the triangle containing this edge. For example, figure 4 shows the same regular triangulation as figure 2. Removing edge *ab* will result in a regular triangulation, because of the interior vertex *d* of the triangle *abd*. Conversely removing edge *bc* will not result in a regular triangulation, because of the boundary vertex *e* of the triangle *bcd*.

Given that the input triangulation, the Delaunay triangulation, is regular (the boundary of the Delaunay triangulation is the convex hull of the input point set), we can infer that the output triangulation is also regular, as long as the single edge removal does not introduce any local irregularities. Algorithm 2 describes the procedure for checking regularity, requiring a regular triangulation and an edge of that triangulation as input. The algorithm returns “true” if the triangulation resulting from removing that edge is regular, “false” otherwise.

Algorithm 1: Characteristic shape algorithm: $\chi(P, l)$

```
Data: Set of points $P \subset \mathbb{R} \times \mathbb{R}$; length $l \in \mathbb{R}$
Result: Characteristic shape $\chi(P, l)$
1.1  Construct the Delaunay triangulation $\Delta$ of $P$;
1.2  Construct the list $B$ of boundary edges, containing the set
     $\{e \in E(\Delta) | e\text{-}\partial(e) = \text{true}\}$;
1.3  Sort the list $B$ in descending order of edge length;
1.4  Initialize the function $v\text{-}\partial : V(\Delta) \rightarrow \{\text{true}, \text{false}\}$, $v\text{-}\partial : v \mapsto \text{false}$;
1.5  foreach $e = (d_1, d_2) \in E(\Delta)$ do
1.6      if $e\text{-}\partial(e)$ then
1.7          Set $v\text{-}\partial : vertex(d_1) \mapsto \text{true}$;
1.8          Set $v\text{-}\partial : vertex(d_2) \mapsto \text{true}$;
1.9  while $B$ is not empty do
1.10     Set $e \leftarrow \text{head}(B)$;
1.11     Remove $e$ from $B$;
1.12     if $||e|| > l$ and $Regular(\Delta, e)$ then
1.13         Remove edge $e$ from triangulation $\Delta$;
1.14         Insert the two edges $edge(reveal(d_1))$ and $edge(reveal(d_2))$ into $B$ in
             order of edge length, where $d_1$ and $d_2$ are the two darts in $e$;
1.15         Set $v\text{-}\partial : vertex(reveal(d_1)) \mapsto \text{true}$;
1.16 return the polygon formed by the set of boundary edges of triangulation $\Delta$;
```

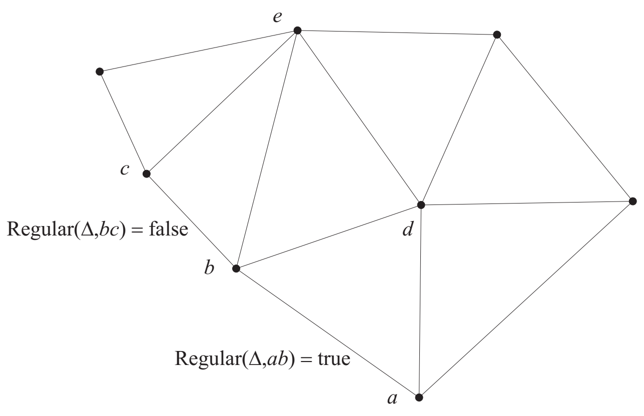


Fig. 4. Regularity constraint on removing triangulation edges

4 Properties

The properties of the χ algorithm and the characteristic shape have been introduced at the beginning of section 3. In this section we explore these properties in more detail.

Algorithm 2: Regularity algorithm: *Regular*(Δ, e)

```
Data: Regular triangulation $\Delta$, edge $e$ of $\Delta$
Result: True if $\Delta - e$ is regular, false otherwise
2.1 if $e$-$\partial(e)$ = true then
2.2     Set $v$ to be the (unique) vertex $v = vertex(\Theta_0(reveal(d)))$ for an arbitrary
        dart $d \in e$;
2.3     if $v$-$\partial(v)$ = false then
2.4         return true;
2.5 return false;
```

4.1 Algorithmic properties

In this section we show that the time complexity of Algorithm 1 is $O(n \log n)$, where n is the cardinality of the input point set. The two preprocessing steps of creating the Delaunay triangulation (line 1) and sorting the list of boundary edges (line 1) each require $O(n \log n)$ time:

- It is a standard result in computational geometry that the Delaunay triangulation (line 1) can be computed in $O(n \log n)$ time (see [18]).
- By Euler's formula, the total number of edges in a planar triangulation Δ is linearly related to the number of vertices (if E is the number of edges, V_B is the number of boundary vertices and V_I is the number of interior vertices, then $E = 2V_B + 3V_I - 3$). Thus, the number of boundary edges in the sorted list B is also linearly proportional to the number of vertices. Using any standard sorting algorithm results in a sorting step of $O(n \log n)$.

Finding the set of boundary edges (line 1) and initializing the v-∂ function (lines 1–1) each require a single pass through the entire list of edges E, which as discussed above is linearly proportional to the number of vertices. Hence, these two preprocessing steps each have time complexity $O(n)$.

The complexity of the core algorithm loop (lines 1–1) is linear, $O(n)$. The critical observations in understanding this result are to note that:

(1) at any iteration, a boundary edge found to belong to a triangle with no interior vertices (i.e., one resulting in an irregular triangulation if removed) can never subsequently become a candidate for removal; and
(2) every time a boundary edge is removed from the triangulation, two new edges must be added to the list of boundary edges.

As a consequence, at each iteration one edge is discarded from B, with possibly two new edges being added to B. Either the edge will be removed, and so by 2 above two new edges added to the list B; or its removal would result in an irregular triangulation, and so by 1 above it need not be checked again; or its length is less than l, in which case it, and all remaining (shorter) edges in B, need not be checked again. The maximum number of new boundary edges that could possibly be added to B in the course of the algorithm is clearly fewer than the total number of interior edges. So in the worst case the algorithm must iterate fewer than $|E|$ times. As we have already seen, in a planar triangulation the number of edges $|E|$ is linearly related to the number of input vertices n.

Note also that checking whether removing an edge will result in a regular triangulation (line 1 and Algorithm 2) can be achieved in constant time. For the boundary edge in question, it is only necessary to look up whether the third vertex of the boundary triangle containing that edge is an interior vertex. This third vertex can be found in constant time from the combinatorial map. Consequently, the overall time complexity of the χ algorithm is dominated by the preprocessing steps, and is $O(n \log n)$.

Finally, if the length parameter l is set to zero, then the algorithm will run through every possible χ-shape for a given point set P. Thus, by modifying the algorithm slightly to store new χ-shapes at each iteration allows the entire family of χ-shapes for P to be generated in $O(n \log n)$ time.

4.2 Characteristic shape properties

A polygon X is a closed planar path composed of a finite number of sequential line segments. The straight line segments that make up X are called its edges and the points where the sides meet are the vertices. Polygon X is said to be *simple* if the only points of the plane belonging to two polygon edges of X are the polygon vertices of X. Clearly, so long as the points are not all collinear, the initial triangulation is regular, and hence yields a shape that is simple (the convex hull). Each iteration of the algorithm preserves regularity. A regular triangulation must have a simple polygon boundary, by the definition of regularity in section 3.1. Thus, the χ-shape must also be simple.

The initial triangulation contains all the elements of initial point set as vertices, thus initially all elements of the point set must be incident with at least two edges. Since the algorithm removes at most one edge from the triangulation at each iteration, an element of the input point set can only lie outside the characteristic shape if first at some iteration it was a vertex incident with only one edge. Such a situation is prohibited by the regularity constraint. Thus, we infer that the entire input point set must be vertices of the final triangulation, and so contained within the characteristic shape.

Finally, the area bounded by the characteristic shape must be contained within and possibly equal to the convex hull. In the extreme case where no edges are removed, then the algorithm returns the polygon boundary of the convex hull. Every iteration of the algorithm that removes an edge from the triangulation will exclude those parts of the convex hull that were contained within the triangle bounded by the deleted edge.

5 Parameterization

The shape of the characteristic shape produced by the algorithm described above is parameterized using the length l. Because the algorithm runs through boundary edges in descending order, any edge that is removed for a parameter l will also be removed for a smaller parameter $l' < l$. Thus, for any set of input points P and length parameters $l' \leq l$, it follows that the characteristic shape of P with parameter l' is contained within the characteristic shape of P with parameter l, i.e., $l' \leq l \leftrightarrow \chi(P, l') \subseteq \chi(P, l)$.

5.1 Normalized length parameters

The parameter l can potentially take the value of any non-negative real number. However, it is more convenient to normalize the parameter with respect to a particular set of points P by using the maximum and minimum edge lengths of the Delaunay triangulation of P. Increasing l beyond the maximum edge length of the Delaunay triangulation cannot reduce the number of edges that will be removed (which will be zero anyway). Decreasing l beyond the minimum edge length of the Delaunay triangulation cannot increase the number of edges that will be removed. Thus, for a set of points P we define two lengths $\max_P$ and $\min_P$ as follows:

$$\max_P \equiv \max(\{||e|| \,|\, e \in E(\Delta_P)\})$$

$$\min_P \equiv \min(\{||e|| \,|\, e \in E(\Delta_P)\})$$

Given these two lengths, we can now define a normalized length parameter $\lambda_P \in [0, 1]$ as follows:

$$\lambda_P = \begin{cases} 1 & \textit{if } l \geq \max_P \\ \frac{l - \min_P}{\max_P - \min_P} & \textit{if } \min_P \leq l < \max_P \\ 0 & \textit{if } l < \min_P \end{cases}$$

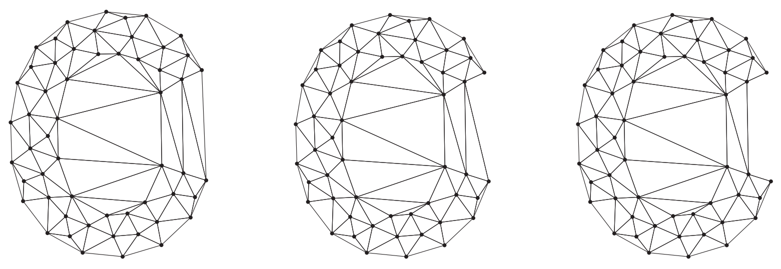
Figure 5 shows an example of all the different characteristic shapes produced by different normalized λ_P parameters for a sparse set of points P roughly in the shape of the letter "C". To help illustrate the effects of the λ_P parameter, figure 5 shows the full triangulation associated with each λ_P value. However, it should be noted that the χ algorithm only returns the polygonal boundary for the triangulation.

5.2 Choices of λ_P

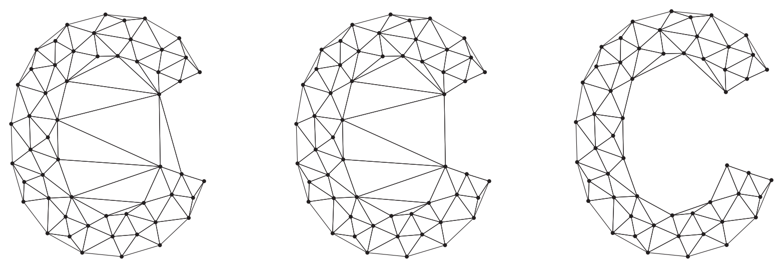
As shown above, the choice of λ_P has a determining effect on the precise shape obtained from the characteristic shape algorithm. One way of choosing a value for λ_P, then, is to try a range of different values and then *a posteriori* select the value that produces a shape that best fits some desired criteria (such as area-perimeter ratio). However, there are a range of possible *a priori* choices for values of λ_P.

Two natural choices are to set λ_P to an extreme value, zero or one. Setting $\lambda_P = 1$ means that no edges will be removed from the Delaunay triangulation, so the resulting polygon will be the convex hull (Figure 5.a). It is desirable that the χ-shape algorithm degrades gracefully to yield the convex hull at one extreme, but clearly the aim of the χ-shape algorithm is to provide a better characterization of shape than the convex hull. Setting $\lambda_P = 0$ means that all edges that can be removed subject to the regularity constraint will be removed (Figure 5.l). However, running the χ algorithm to its conclusion in this way often creates polygons that are eroded beyond the point where they provide a desirable characterization of the shape.

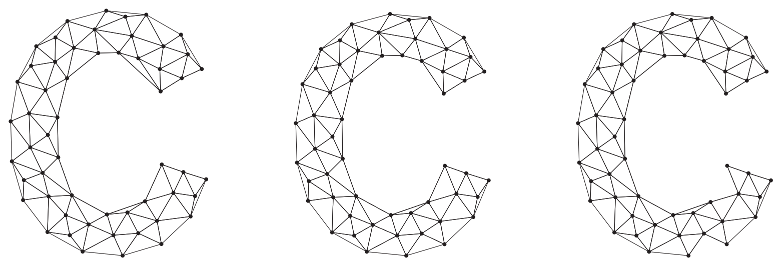
Given that extreme values of λ_P tend to lead to unsatisfactory χ-shapes, it would be useful to be able to define *a priori* an intermediate value for the parameter, $0 < \lambda_P < 1$, that could adapt to a range of different point sets to produce acceptable shape characterizations. For example, one possibility is to use the length of the longest edge in the minimum spanning tree of the Delaunay triangulation (which we coined the "max-MST" edge length). The minimum spanning tree is the subgraph of the Delaunay triangulation with the smallest total edge length that connects all the vertices of the triangulation. In the case of the point distribution in figure 5 the max-MST edge length corresponded to a λ_P value of 0.1, yielding the shape in figure 5.l. Another possibility is to find the shortest edge for each triangle in the Delaunay triangulation, and use the maximum length of all these shortest edges (which we termed the "max-min Δ" edge length). For the point distribution in figure 5, the max-min Δ edge length corresponded to a λ_P value of 0.56, yielding the shape in figure 5.e.



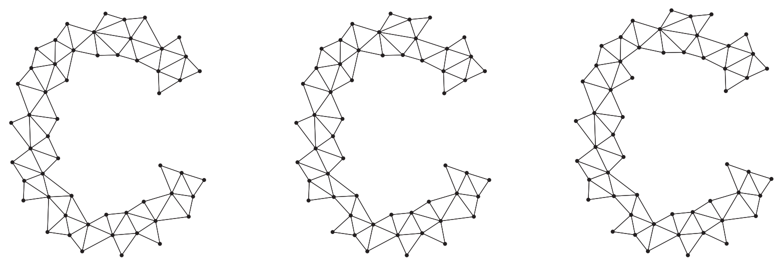
a. $0.77 < \lambda_P \leq 1.00$ b. $0.73 < \lambda_P \leq 0.77$ c. $0.67 < \lambda_P \leq 0.73$



d. $0.60 < \lambda_P \leq 0.67$ e. $0.51 < \lambda_P \leq 0.60$ f. $0.39 < \lambda_P \leq 0.51$



g. $0.38 < \lambda_P \leq 0.39$ h. $0.29 < \lambda_P \leq 0.38$ i. $0.27 < \lambda_P \leq 0.29$



j. $0.23 < \lambda_P \leq 0.27$ k. $0.20 < \lambda_P \leq 0.23$ l. $0.00 < \lambda_P \leq 0.20$

Fig. 5. Examples of varying λ_P parameter for characteristic shape algorithm

Initial investigations using these two possibilities revealed that while one or other sometimes provided a satisfactory result, neither could be be relied upon to consistently provide a "good" characterization of shape (as illustrated by Figure 5, where neither parameter yields a shape that closely approximates the

"C" shape of the original point distribution). Potentially, there innumerable other possible *a priori* choices of λ_P that might be defined. For example, an intermediate value of λ_P half-way between the max-MST and max-min Δ values often, but not always, yielded satisfactory results. Ultimately, no *a priori* method for choosing λ_P can be expected always to provide a "good" characterization of the shape of a set of points.

6 Experimentation

In this section we investigate some of the empirical properties of the characteristic shape algorithm. However, as asserted in section 1, in general the question of what constitutes a "better" characterization of the shape of a set of points is an underspecified problem to which there can be no single "correct" answer. Therefore, in the following experiments we generate randomized point distributions with a well-defined shape (such as a letter of the alphabet or a country of the world) and compare the χ-shape with that original shape.

The experiments that follow fall into three distinct categories. First, the experiments examine the effects of varying the normalized length parameter λ_P upon χ-shapes (6.1). Second, the effects of varying point densities upon the optimal normalized length parameter are analyzed (6.2). Third, the effects of increasing inhomogeneity in point distributions are tested (6.3). All the experiments were conducted using a version of the χ algorithm implemented in Java. This software utilizes the half-edge data structure to store and query the triangulation efficiently. As highlighted above, this commonly-used data structure is derived from the combinatorial map.

6.1 Parameterization

Section 5.2 suggested some natural choices for parameterizing the characteristic shape algorithm using the normalized length λ_P. In this section we examine more carefully the response of the algorithm to changes in normalized length.

To evaluate objectively the performance of the characteristic shape algorithm, a series of experiments were conducted with point distributions of known shapes. The χ-shapes generated using different normalized length parameters were compared with the shapes of these input point distributions. Initial experiments compared the ratio of the area of the characteristic shape to the original shape of the point distribution. Using area is simple but does not provide a particularly good measure of closeness of the two shapes, since two very different shapes can still have the same area. For this reason it is preferable to use the area of the region enclosed between the boundaries of the original shape and the corresponding characteristic shape, termed the L^2 *error norm*. The L^2 error norm can be computed by finding area of the symmetric difference between and original region O and a χ-shape C as a proportion of the total area of the χ-shape C (i.e., $\frac{area((O-C)\cup(C-O))}{area(C)}$). An L^2 error norm of zero means that not only are the areas of the two shapes equal, but also that their boundaries are in complete agreement.

Figure 6 shows the variation in the L^2 error norm for characteristic shapes produced using a range of normalized length parameters for a number of known point distributions. To compensate for differences in the absolute areas of the different shapes, the figure shows the L^2 error norm values as a proportion of the total area of the original shape. The four different distributions used are based on the shapes of the uppercase letters "C," "F," "G," and "S." These letters were chosen for the figure because they exhibit a range of different levels of sinuosity and angularity. However, the results are representative of all the letter shapes tested (i.e., those can be represented as a simple polygon, unlike lowercase "i" or uppercase "A").

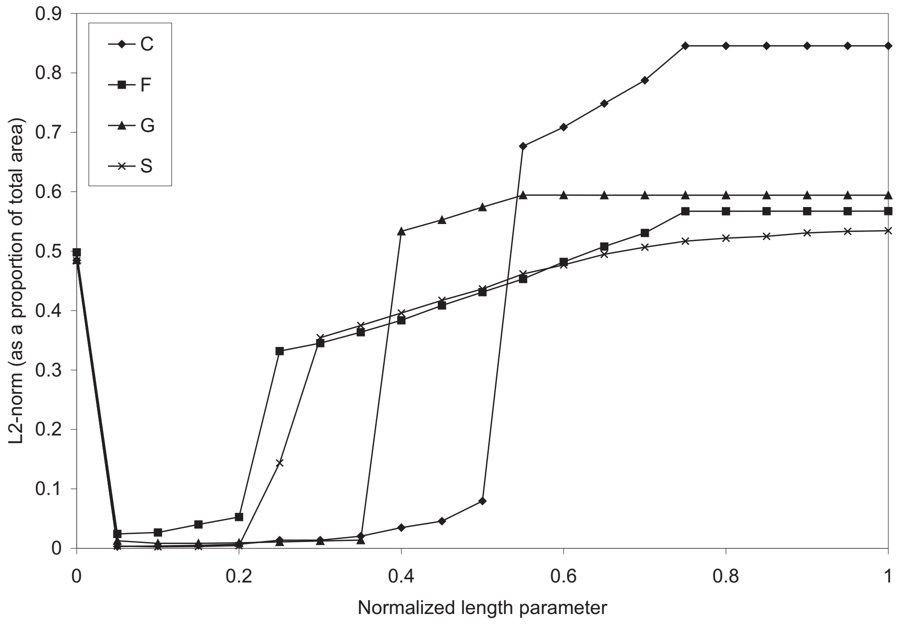


Fig. 6. Variation in characteristic shape accuracy with normalized length parameter λ (letter shapes)

The letter shapes were generated using a sans serif font (Arial). The boundary of each shape was approximated as a polygon using a number of evenly spaced vertices connected by straight-line segments. Each shape was then filled with a semi-random distribution of internal points, where each point must be greater than a certain threshold distance d from any other points, but otherwise is randomly positioned. Truly random distributions of points can have strongly inhomogeneous densities, leading to the formation of clusters and holes which mask the true shape of the letter itself. Hence, the semi-random distribution was used for these initial experiments.

Together the polygon vertices and the internal points compose the input point set. For each shape, 20 semi-random internal point sets were generated, ensuring randomized, but reasonably evenly spaced input point set distributions. Figure 6 shows the average area of these 20 distributions for each shape at each of 21 normalized length parameters (0.0, 0.05, 0.1, ..., 1.0). Thus, figure 6 summarizes the properties of a total of $4 \times 21 \times 20 = 1680$ different characteristic shapes.

The curves in figure 6 exhibit progressive improvements the χ-shape's approximation of shape of the input point set, indicated by decreasing L^2 error norm value, as the normalized length parameter decreases from 1.0 (i.e., the convex hull). Below a certain normalized length parameter, the algorithm begins to "eat in" to the body of the shape, leading to a rapid increase in L^2 error norms as the normalized length parameter decreases from values around 0.05. The response curves for the different figures also exhibit a number of pronounced "steps." These steps correspond to the removal of a small number of triangles with relatively large areas from the triangulation (for example those that make up the interior of the triangulated "C" shape, as in Figure 5).

All the shapes in figure 6 have response curves that reach a minimum L^2 error norm ratio of less than 0.03 (i.e., the total area of disagreement between the characteristic shape and original shape is on average less than 3% of the total area of the shape). However, even in the very worst cases (recall that each data point in figure 6 represents an average of the characteristic shapes of 20 different randomized point distributions) all randomized point distributions achieved a minimum L^2 error norm ratio of less than 0.08 (8% of the total shape area).

Figure 7 shows the same experiment as in figure 6, but repeated with rather different shapes: the boundary shapes of four countries of the world (France, Germany, Italy, Vietnam). Again, these shapes were chosen as providing a range of sinuosity and elongation from amongst those countries with borders that can be described as a simple polygon. The performance of the algorithm for these country shapes is similar to the performance for the letter shapes. In general there are fewer step-changes in figure 7 than 6. This is to be expected, since basic geographical principles tend to favor roughly convex country shapes without large cavities.

The minimum L^2 error norm ratio achieved for each country shape was again relatively low. The algorithm performed worst (higher L^2 error norm) with the shape of Vietnam. The boundary of Vietnam is the most elongated of the

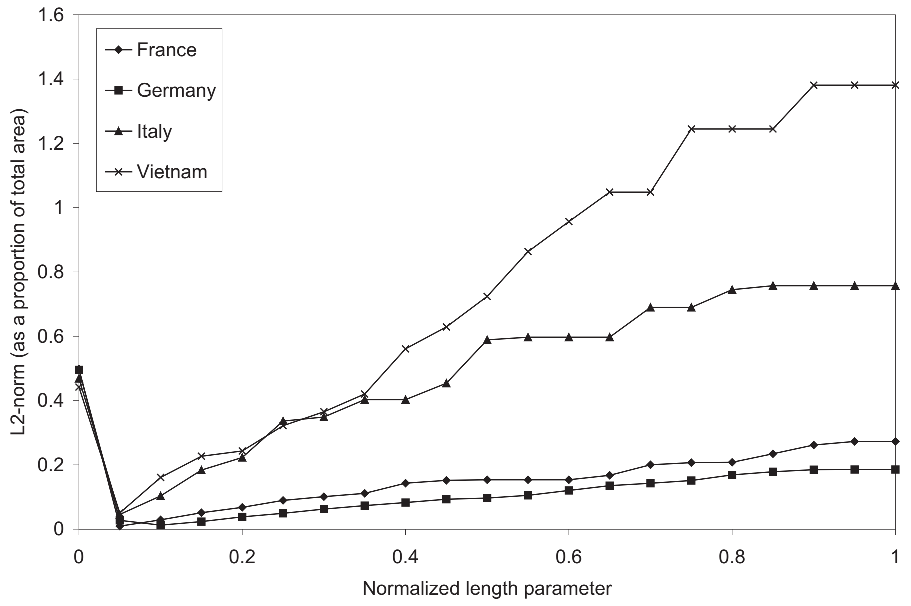


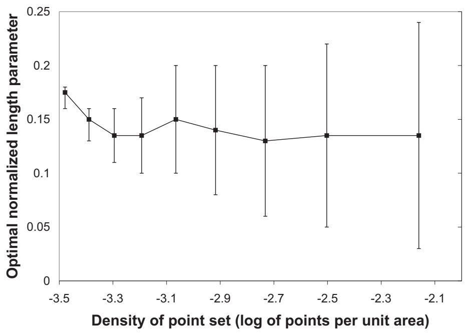
Fig. 7. Variation in characteristic shape accuracy with normalized length parameter λ (country shapes)

countries tested, with a relatively small area to perimeter ratio. As a consequence, the chance of boundary errors having a greater effect on the area error is also greater. However, the minimum L^2 error norm of 0.05 still represents a relatively low figure when considering that the point sets themselves are semi-random.

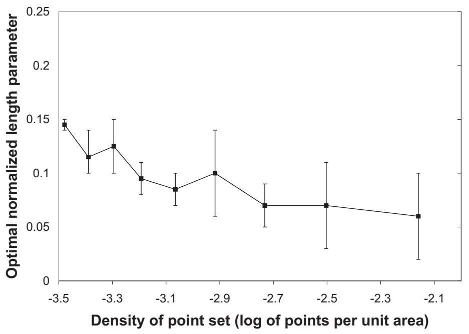
6.2 Effects of point density

The results in the previous section suggest that normalized length parameters of around 0.05–0.2 often provide good characteristic shapes, since the L^2 error norm often reaches its minimum at around these normalized length parameter values. However, all the shapes tested in the previous section used similar densities of points: approximately 0.003 points per unit area. The unit area for the experiments was a single screen pixel: in other words, all the point sets used for experiments in the previous section filled their shapes using on average 1 point occupying a region of approximately 18 × 18 pixels. We might expect the optimal normalized length parameter (the parameter value that corresponds to the lowest L^2 error norm) to depend on the density of points used, especially at lower point densities where the number of points used to define the same shape is much lower.

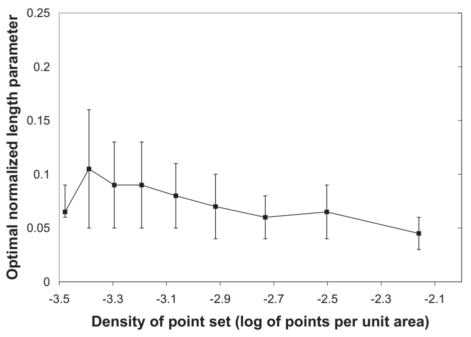
To investigate this potential relationship, each of the four graphs in Figure 8 shows the average changes in optimal normalized length parameter across a range of point densities for random point sets derived from the four shapes we have already encountered (letters “S” and “F” and the countries Germany and Vietnam). For each graph, the log of the density of the input point set (i.e., with lowest point density on the left-hand side and highest point density on the right-hand side) is plotted against the optimal normalized length parameter, averaged (using the median to minimize the impact of outliers) across five randomized point sets. The point densities tested range from 0.0003 per unit area (1 point occupying a region of approximately 57 × 57 pixels) to 0.007 (1 point occupying a region of approximately 12 × 12 pixels). The most important feature to note across all the graphs is that while the optimal length parameter varies from shape to shape, it is relatively stable across all point densities.



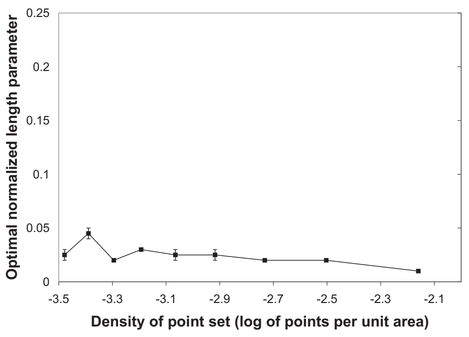
a. Letter “S”



b. Letter “F”



c. Germany



d. Vietnam

Fig. 8. Effects of changing density of input point set on optimal normalized length parameter

In some cases, the optimal value of the normalized length parameter is quite sensitive to changes in density, and slight increases or decreases in the normalized length parameter substantially increase the errors in shape. However, in many cases, the optimal value is remarkably stable, and a range of normalized length parameters provide near-optimal errors in shape. To represent this, the “error” bar on each plotted point in figure 8 shows the range of normalized length parameters that lead to an overall L^2 error norm within 1% of the lowest possible L^2 error norm. Again, the graph shows the modal maximum and modal minimum normalized length parameter averaged across each set of 5 repetitions. Those data points with narrower "error" bars are more sensitive to changes in the precise normalized length parameter value; those with wider "error" bars are less sensitive to changes in normalized length parameter (thus, the term "error bars," conventionally used to describe these features on graphs, is a misnomer in this case, since we are representing stability rather than error). For instance, for the highest densities of the "S" shape, normalized length parameters from 0.02 to 0.24 yielded on average an L^2 error norm within 1% of the lowest possible error norm. Conversely, the complex, elongated shape of Vietnam was much more sensitive to changes in normalized length parameter, and a change in normalized length parameter of 0.01 or less from the optimal value tended to result in a substantial (i.e., greater than 1%) decrease in the resulting L^2 error norm.

6.3 Effects of point distribution

Finally, as discussed at the beginning of section 6, all the point distributions so far were semi-randomly distributed (position is random, subject to a minimum distance between any two pairs of points). Thus, while the point distributions used were randomized, the distribution of points was homogeneous, as illustrated by the point distributions in figure 1. They are "dot patterns" in the sense of [5]. A truly random distribution of points will exhibit clusters that are expected to mask the desired shape of the distribution. The less homogeneous the distribution of points, the greater the expected deviation between the characteristic shape and the desired shape. In fact, the χ algorithm seemed surprisingly tolerant to increasing randomness in point distribution. Figure 9 shows one example of a set of 250 points randomly distributed throughout the letter "F" with the corresponding (optimal) characteristic shape.

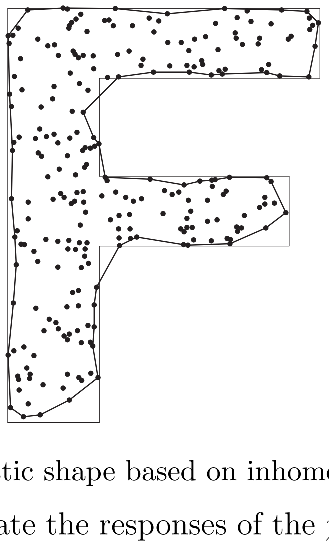


Fig. 9. Example characteristic shape based on inhomogeneous point distributions

To systematically investigate the responses of the χ algorithm to increasingly inhomogeneous point distributions, one further experiment was executed. Figure 10 shows the effects on shape error (in terms of optimal L^2 error norm ratio) of varying the homogeneity of the point distributions, for each of the four shapes used in the experiments in section 6.2. In a semi-random point distribution, each point must be greater than a certain threshold distance d from any other points. The larger the distance d, the fewer points in total that will be able to fit inside a given shape. Thus, there is a direct relationship between the number of points that can fit inside a given shape and the threshold distance d.

For n points, max_d denotes the largest threshold distance such that all n points can still fit inside the shape (such as in the point distributions in figure 1). Point distribution homogeneity can then be represented using a normalized measure $h = \frac{d}{max_d}$, such that $h \in [0, 1]$. Setting $h = 0$ results in wholly inhomogeneous, truly random point distributions. Setting $h = 1$ results in wholly homogeneous point distributions (where the entire point set just fits inside the shape). To enable comparison between experiments, figure 10 uses the normalized parameter h for the abscissa values. The average density of points remains constant across all experiments (approximately 0.001 points per unit area, 1 point occupying a region of approximately 30×30 pixels).

The experiment summarized in figure 10 again used 5 randomized shapes to generate each data point. One important difference with previous experiments is that no boundary points were used in generating the point distributions, in order to provide truly random point distributions. Instead, the L^2 error norm is calculated with respect to the original shape, which may include smooth curves (for example the letter "S"). As a consequence of this difference, the absolute levels of shape errors in figure 10 are higher than for previous experiments (e.g., figure 8). However, it is the relative change in shape error, rather than the absolute shape error, which is of primary interest in this experiment.

As expected, figure 10 does show an increase in errors with increasingly random point distributions across all shapes tested. However, the magnitude of increasing errors is relatively low, with truly random distributions typically increasing the error rates by about 50% when compared with homogeneous point distributions. In effect, these results indicate that the χ algorithm degrades gracefully in the presence of inhomogeneous point distributions. To provide some context for this statement, we note that the *convex* hulls of the four shapes in figure 10 are associated with L^2 error norm ratios ranging from approximately 17% (for Germany shape) through approximately 54% (for the letters "F" and "S") to more than 150% (for Vietnam shape). Even using truly random point distributions, the best characteristic shapes are associated with error norm ratios of between 12% (for Germany shape) to 21% (for Vietnam shape).

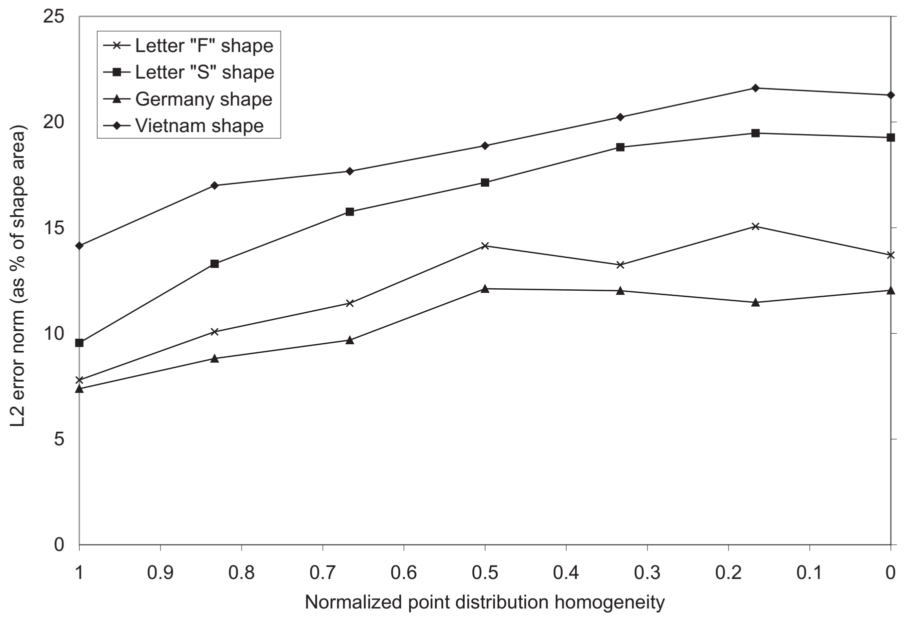


Fig. 10. Variation in shape error with changing point distribution homogeneity

6.4 Discussion of experimental results

In summary, the experimental evaluation of the χ algorithm yielded the following key results for those shapes tested:

- The χ algorithm is able to accurately characterize the shape of point sets derived from a range of different shape types, given appropriate parameterization.
- Although the optimal parameter value varies for different shapes and point distributions, normalized parameter values of between 0.05–0.2 typically produce optimal or near-optimal shape characterization across a wide range of point distributions.
- The optimal parameter value algorithm is reasonably tolerant to changes in point density and point distribution homogeneity (from semi-random to truly random) for those shapes tested.

Although the results presented here summarize the behavior of χ algorithm across many hundreds of different randomized point distributions, to aid comparison these point distributions have necessarily been derived from a relatively small number of basic shapes (letters and countries). Further experimental work would be needed to examine the algorithm's response to a wider range of shapes[2]. However, since the shapes were chosen to exhibit a range

[2] To facilitate further investigation, Java software to generate characteristic shapes is available online from the corresponding author at `http://www.duckham.org`.

of different levels of sinuosity and angularity, there is no a priori reason to believe that different shapes would yield substantially different results.

7 Discussion

Before concluding, we reflect on three broader issues pertinent to χ-shapes: the relationship between χ-shapes and other shape characterization algorithms; the performance of the algorithm in the presence of outliers and the generation of non-simple polygons; and potential applications of χ-shapes.

7.1 Relationship to other shape characterization algorithms

As identified at the end of section 2, a systematic comparison of the many shape characterization algorithms is conspicuously absent from the existing literature. Such a comparison would in itself represent a valuable contribution, but faces a variety of substantial obstacles.

The primary obstacles relate to the task of comparing the shapes generated by different algorithms. First, as originally stated in section 1, there can exist no "correct" shape for a set of points in the plane. As a consequence, all non-convex hull shape algorithms must rely on at least one parameter in order to generate a family of shapes for a single input point set. The shapes within a single algorithm's family can be quite varied (for example, see figure 5). Thus any comparison of the shapes generated by two different algorithms must account for the different parameterization, either by finding a way to link the parameters for the two algorithms being compared, or by comparing the entire family of algorithms at once. However, there exists no obvious mechanism or experimental design for realizing either of these possibilities. This problem is compounded by the need to compare each algorithm's performance for not just one, but across a range of input point sets. Further, as already argued, the decision as to whether a particular shape is "better" or "worse" is as much a matter for human cognition and preference as computational geometry. Consequently, choosing an objective metric for comparison is also problematic. In the light of these difficulties, it becomes more understandable why no research has yet attempted such a systematic comparison of the different algorithms, and why such a comparison is regarded as beyond the scope of this paper.

Comparing the computational characteristics of the different algorithms is more straightforward. However, most algorithms, including our characteristic shape algorithm, have the same computational complexity, $O(n \log n)$. The characteristic shape algorithm is arguably more efficient than other algorithms, such as α-shapes, when it is necessary to generate simple polygons, but only marginally so at best. The characteristic shape algorithm can generate a family of simple polygons in $O(n \log n)$ time. The α-shape algorithm, for example, can generate a family of (possibly non-simple) polygons in similarly $O(n \log n)$ time. [19] has shown that it is possible to test for polygon simplicity in $O(n')$ time, where n' is the number of polygon vertices, but the complexity of this algorithm means $O(n' \log n')$ algorithms are more often used in practice. Thus, checking each of m polygons in the family of α-shapes for simplicity might be expected to lead to a polynomial time α-shape-based algorithm for generating simple polygons. However, in practice it would most likely be possible to optimize such an algorithm to $O(n \log n)$ time (for example using binary search with knowledge of the total ordering of α-shapes in the family).

χ-shapes are not simply a special case of some other non-convex shapes. For example, figure 11 shows a χ-shape (solid hairline) and an α-shape (thick dashed line) for a set of points (in the shape of Cyprus). The shapes overlap in a complex manner, some parts of the χ-shape lying outside of the α-shape, and vice versa. Since the families of both χ- and α-shapes are linearly ordered by spatial containment, this figure provides a proof by example that χ-shapes are not a special case of α-shapes (nor vice versa). Note that the α-shape is not simple, and in this case even includes a topological irregularity (a linear component in the far right-hand side of the figure). However, aside from the fact that the α-shape is not simple (simplicity being a desirable property for the shapes of many geographic regions, like most countries) there is little about one shape that is "better" than the other: both seem to provide "reasonable" representations of this shape in their own way.

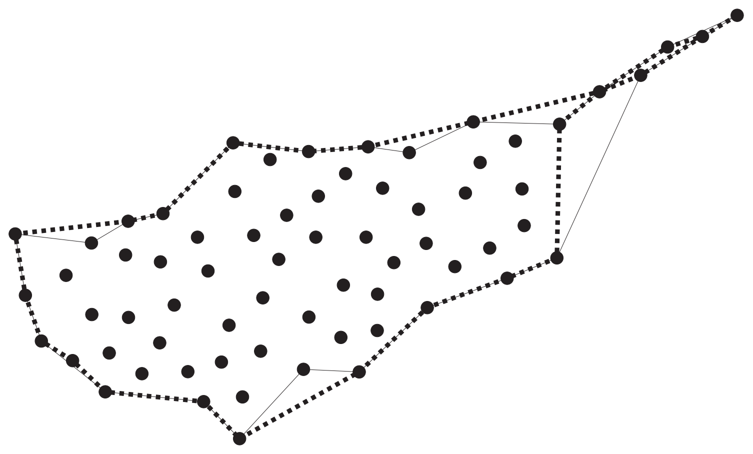


Fig. 11. Example differences between χ-shape (solid hairline) and α-shape (thick dashed line)

As already highlighted, the characteristic shape algorithm generates a simple polygon that contains *all* the points in the input data set. Hence, the algorithm does not deal with outliers especially well. Figure 12 shows a C-shaped point set with an outlier in the mouth of the C, along with two examples of the resulting a χ-shape and α-shape (both generated using appropriate manually chosen parameterization). The χ-shape algorithm includes the outlier. Although the α-shape algorithm also includes the outlier, it does so as an isolated, disconnected point component. The other, polygonal component of the α-shape is free of the outlier.

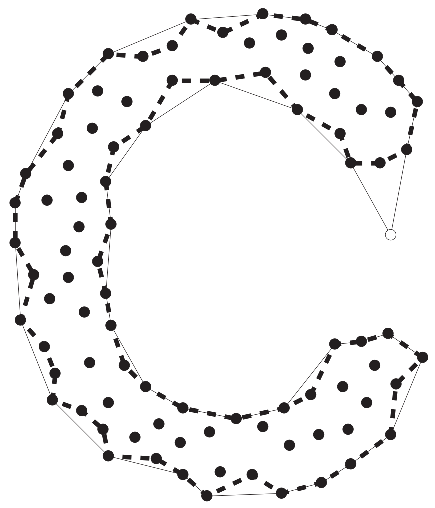


Fig. 12. Example χ-shape (solid hairline) and α-shape (thick dashed line) for a "C" shaped point set containing an outlier (white point).

However, outliers can be eliminated by preprocessing. As suggested earlier, a clustering algorithm can potentially be used to identify and eliminate outliers from the input point set prior to applying the χ-shape algorithm. In the case of figure 12, the DBSCAN clustering algorithm (density-based scan spatial clustering algorithm, [20]) easily clusters the figure into exactly the two clusters required: one singleton cluster containing the outlier and a second cluster containing all the other points in the C shape. Thus, the outlier could be removed before running the χ-shape algorithm simply by preprocessing the input point set with a spatial clustering algorithm such as DBSCAN.

The disadvantage of preprocessing the data set to detect and remove outliers in this way is that any clustering algorithm will require additional parameterization. In the case of DBSCAN, two parameters are required, which essentially define what constitutes a "neighborhood" in the desired result (the parameters reflect the maximum radius of a neighborhood and the minimum of points in a neighborhood). The α-shape algorithm requires no such additional parameterization, and manages the entire shape generation process in one step with the one parameter. However, the advantage of preprocessing is much greater flexibility. There are a wide range of algorithms that can be brought to bear on spatial clustering (see [21]), and these can be used to allow independent manipulation of the different components of the resulting disconnected shape. By contrast, using α-shapes it would not be possible to achieve a less eroded C shape while still eliminating the outlier from figure 12 (since increasing the alpha value to decrease the erosion of the C shape also leads to the outlier becoming a connected part of the polygonal shape again).

The problem of disconnected regions can be dealt with similarly to outliers. Preprocessing using clustering enables distinct groups of points to be distinguished. These distinct groups can then be processed using separate passes of the χ-shape algorithm. The union of of all generated χ-shapes will result in an aggregate, disconnected χ-shape. Figure 13 illustrates the idea showing an aggregate χ-shape generated for a similar point set to figure 12, following preprocessing using DBSCAN spatial clustering algorithm. As for outliers, above, this approach has the advantage of flexibility, but the disadvantage of requiring additional parameterization.

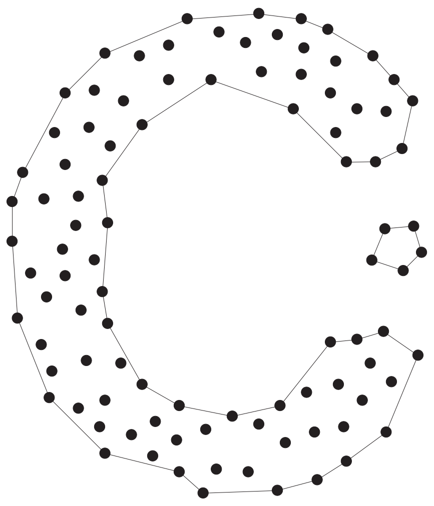


Fig. 13. Example aggregate χ-shape based on characteristic shapes of individual clusters identified using DBSCAN

Finally, an important feature of some shape characterization algorithms, like α-shapes, compared with characteristic shapes is the ability to generate shapes with holes. It is not possible to generate such shapes with the χ-shape algorithm, although the algorithm degrades gracefully in the sense that it will still successfully generate the external edge of the region. It is not especially difficult to modify the χ-shape algorithm to deal with holes. However, doing so increases in the computational complexity of the algorithm (since the efficient test for regularity in algorithm 2 becomes more computationally intensive) and so we only present here the efficient $O(n \log n)$ algorithm.

7.3 Applications of characteristic shapes

There are a wide range of potential applications for χ-shapes and related shape generation algorithms, especially in the geographical domain. Two examples of such applications are geographic information retrieval (GIR) and the generation of geographic "footprints" for vague and imprecise spatial concepts, like "South East England" [8]; and the characterization of dynamic collectives, such as a flock or crowd [22]. Another emerging application is in the domain of geosensor networks, where the shape of salient regions (such as "hot spots") need to be generated from point-based sensor nodes (for example, measuring temperature, [23]).

Figure 14 illustrates a simple example of using χ-shapes in a GIR-related application. The figure shows a map of the Mornington Peninsula, an important wine growing region in Victoria, Australia, along with 36 of the best-known wineries in the region (actual POI data from `http://gps-data-team.com/`). The "Mornington Peninsula wine region" is an example of a vague geographical concept as it has no crisp boundary: while there are places that are definitely in the Mornington Peninsula wine region, and places that are definitely not, no crisp boundary exists that separates the two.

To represent the indeterminacy at the boundary, Figure 14 shows the entire family of characteristic shapes super-imposed on top of each other. The darkest shading fills those parts of the region that are present in all of the characteristic shapes (i.e., the "core": those places that are definitely part of the "Mornington Peninsula wine region"). Lighter shading fill those parts of the region that are present in fewer of the characteristic shapes (i.e., the lighter the shading, the worse the candidate for being part of the "Mornington Peninsula wine region"). Characteristic shapes are a good choice for this task as we can stipulate a priori that the Mornington Peninsula wine region is a simple polygon and all of one piece (i.e., no holes or disconnected components); the algorithm guarantees that all wineries are part of the region, with outliers being unlikely in such a data set; and the required entire family of χ-shapes can be computed in $O(n \log n)$ time. Such a representation of vagueness would for example be useful in a GIR application for responding to user queries about the "Mornington Peninsula wine region."

8 Conclusions

In this paper we have presented a new algorithm for generating a simple, connected, possibly non-convex polygon that characterizes the shape of a set of points in the plane. The algorithm, based on the Delaunay triangulation

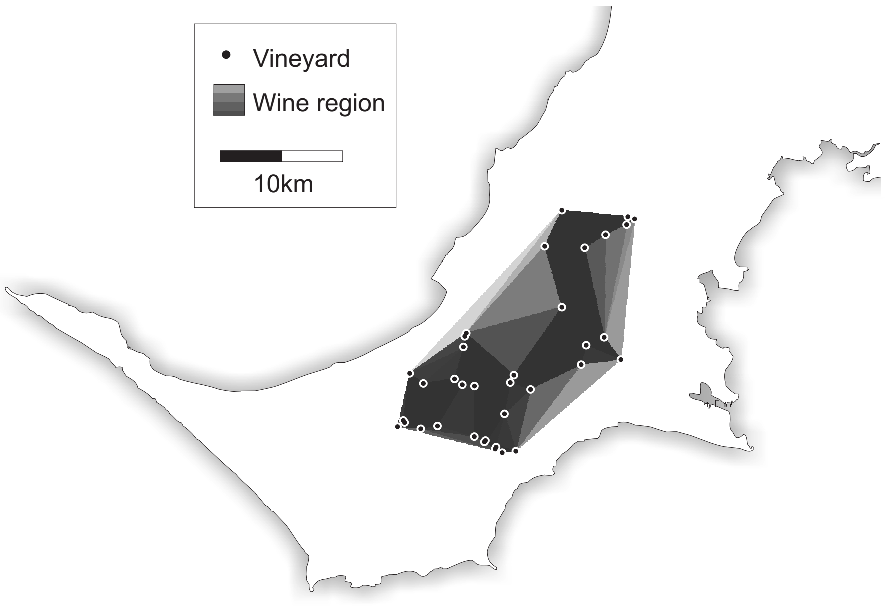


Fig. 14. Example application of χ-shapes: the shape of the Mornington Peninsula wine region

of the point set, is optimal, requiring $O(n \log n)$ time to execute. The shape produced by our algorithm is parameterized by means of a single normalized length parameter. Changing the length parameter produces one of a finite family of totally ordered characteristic shapes, ranging from the convex hull at one extreme to a uniquely defined simple polygon with minimal area at the other extreme.

No one parameter value can ever yield a "correct" answer; instead different parameter values are expected to be required for different applications. However, some guidelines for good lengths are suggested by experiments using the algorithm. Experimental results demonstrate the algorithm's stability and graceful degradation across a wide range of input point sets. A range of further experimental work is suggested by this research, including:

- experiments to evaluate the performance of the χ algorithm across a wider range of shapes; and
- more extensive experimental work to directly compare the performance of this and the other shape characterization algorithms reviewed in section 2.

Extensions of the algorithm to higher dimensions are possible, but problematic. A direct extension to three-dimensional space does present efficiency problems. In particular, since in three dimensions any number of exterior faces of a regular three-dimensional triangulation could meet at a single vertex, a three-dimensional regularity check algorithm would be expected to require substantially more computation.

Acknowledgments

Glenn Hudson developed the Java software used to run the experiments described in this paper and provided helpful feedback and comments on the algorithm itself. Matt Duckham's research is supported by the Australian Research Council under ARC Discovery Grant DP0662906.

References

[1] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. IT–29, no. 4, pp. 551–558, 1983.

[2] M. Melkemi and M. Djebali, "Computing the shape of a planar points set," *Pattern Recognition*, vol. 33, pp. 1423–1436, 2000.

[3] M. J. Fadili, M. Melkemi, and A. ElMoataz, "Non-convex onion-peeling using a shape hull algorithm," *Pattern Recognition Letters*, vol. 25, pp. 1577–1585, 2004.

[4] B. Chazelle, "On the convex layers of a planar set," *IEEE Transactions on Information Theory*, vol. 31, pp. 509–517, 1985.

[5] A. R. Chaudhuri, B. B. Chaudhuri, and S. K. Parui, "A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border," *Computer Vision and Image Understanding*, vol. 68, no. 3, pp. 257–275, 1997.

[6] G. Garai and B. B. Chaudhuri, "A split and merge procedure for polygonal border detection of dot pattern," *Image and Vision Computing*, vol. 17, pp. 75–82, 1999.

[7] H. Alani, C. B. Jones, and D. Tudhope, "Voronoi-based region approximation for geographical information retrieval with gazetteers," *International Journal of Geographical Information Science*, vol. 15, no. 4, pp. 287–306, 2001.

[8] A. Arampatzis, M. van Kreveld, I. Reinbacher, C. B. Jones, S. Vaid, P. Clough, H. Joho, and M. Sanderson, "Web-based delineation of imprecise regions," *Computers, Environment, and Urban Systems*, vol. 30, no. 4, pp. 436–459, 2006.

[9] N. Amenta, S. Choi, and R. Kolluri, "The power crust," in *Sixth ACM Symposium on Solid Modeling and Applications*, 2001, pp. 249–260.

[10] ——, "The power crust, unions of balls, and the medial axis transform," *Computational Geometry: Theory and Applications*, vol. 19, no. 2–3, pp. 127–153, 2001.

[11] D. Attali, "r-regular shape reconstruction from unorganised points," *Computational Geometry*, vol. 10, pp. 239–47, 1998.

[12] A. Galton and M. Duckham, "What is the region occupied by a set of points?" in *GIScience*, ser. Lecture Notes in Computer Science. Springer, 2006, vol. 4197, pp. 81–98.

[13] J. Edmonds, "A combinatorial representation for polyhedral surfaces," *Notices of the American Mathematical Society*, vol. 7, p. 646, 1960.

[14] B. Baumgart, "A polyhedron representation for computer vision," in *Proc. AFIPS National Computer Conference*, vol. 44, 1975, pp. 589–596.

[15] K. Weiler, "Edge-based data structures for solid modeling in curved-surface environments," *Computer Graphics and Applications*, vol. 5, no. 1, pp. 21–40, 1985.

[16] J.-F. Dufourd and F. Puitg, "Functional specification and prototyping with oriented combinatorial maps," *Computation Geometry—Theory and Applications*, vol. 16, no. 2, pp. 129–156, 2000.

[17] M. F. Worboys and M. Duckham, "Monitoring qualitative spatiotemporal change for geosensor networks," *International Journal of Geographic Information Science*, vol. 20, no. 10, pp. 1087–1108, 2006.

[18] J. O'Rourke, *Computational geometry in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1998.

[19] B. Chazelle, "Triangulating a simple polygon in linear time," *Discrete Computational Geometry*, vol. 6, pp. 485–524, 1991.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, 1996, pp. 226–231.

[21] H. J. Miller and J. Han, Eds., *Geographic Data Mining and Knowledge Discovery.* CRC Press, 2001.

[22] A. Galton, "Dynamic collectives and their collective dynamics," in *COSIT*, ser. Lecture Notes in Computer Science. Springer, 2005, vol. 3693, pp. 300–315.

[23] M. Duckham, S. Nittel, and M. Worboys, "Monitoring dynamic spatial fields using responsive geosensor networks," in *ACM GIS 2005*, C. Shahabi and O. Boucelma, Eds. New York: ACM Press, 2005, pp. 51–60.