What Is the Region Occupied by a Set of Points?

Conference Paper *in* Lecture Notes in Computer Science · September 2006

DOI: 10.1007/11863939_6 · Source: DBLP

CITATIONS
96

READS
939

2 authors:

Antony Galton
University of Exeter
137 PUBLICATIONS **3,703** CITATIONS

SEE PROFILE

Matt Duckham
RMIT University
179 PUBLICATIONS **4,691** CITATIONS

SEE PROFILE

What Is the Region Occupied by a Set of Points?

Antony Galton[1] and Matt Duckham[2]

[1] School of Engineering, Computer Science, and Mathematics
University of Exeter, Exeter EX4 4QF, UK
A.P.Galton@exeter.ac.uk
[2] Department of Geomatics
University of Melbourne, Victoria 3010, Australia
mduckham@unimelb.edu.au

Abstract. There are many situations in GIScience where it would be useful to be able to assign a region to characterize the space occupied by a set of points. Such a region should represent the location or configuration of the points as an aggregate, abstracting away from the individual points themselves. In this paper, we call such a region a 'footprint' for the points. We investigate and compare a number of methods for producing such footprints, with respect to nine general criteria. The discussion identifies a number of potential choices and avenues for further research. Finally, we contrast the related research already conducted in this area, highlighting differences between these existing constructs and our 'footprints'.

1 Introduction

There are many situations in GIScience where it would be useful to be able to assign a region to a set of points, to represent the location or configuration of the points as an aggregate, abstracting away from the individual points themselves.

In map generalisation, for example, what appears at one level of detail as a set of discrete points may be better represented, at a coarser level of detail, as a region. The region indicates where the points are located and can give a general idea of their configuration, but does not indicate how many points there are or their individual locations. In Figure 1, for example, the configuration of points represented at one scale as in the left-hand illustration might appear at a smaller scale as in the right-hand illustration.

One might refer to the 'outline' of a group of trees, or a flock of birds, or generally of any aggregation of discrete point-like objects, but it is important to appreciate that the outline is not by any means uniquely determined. To illustrate this, note that each of the illustrations in Figure 2 represents a possible outline for the same set of points; depending on one's purpose in requiring an outline, some may be 'better' solutions than others, but none is absolutely 'correct'.

Different choices of outline can have an effect on how we understand phrases like 'among the trees' or 'in the wood'. The trees can be considered to form a

M. Raubal et al. (Eds.): GIScience 2006, LNCS 4197, pp. 81–98, 2006.

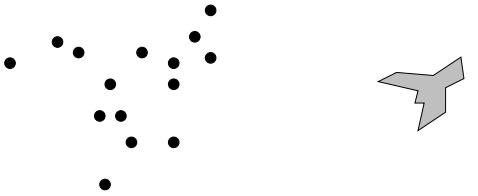


Fig. 1. Generalization and scale for points and regions

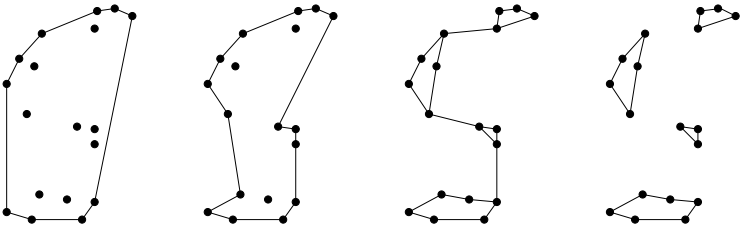


Fig. 2. Possible outlines for the region occupied by a set of points

collection of point-like objects distributed over an area of space. Which other spatial points should we count as 'among the trees'? The obvious answer is 'the points enclosed within the outline of the trees', but as we have seen, this does not provide us with a determinate answer.

Sometimes a configuration of objects is naturally divided into a number of separate clusters. The clustering may be 'obvious', as in Figure 3, where the set of points on the left is shown as falling into three clusters by the fact that the region associated with it (on the right) has three connected components.

Fig. 3. Clustering in regions occupied by a set of points

In other cases, there may be several equally good answers, as for example in Figure 4, where the same set of points is presented as being grouped into either two or four clusters.

One criterion for 'good' clustering may of course be visual salience, but visual salience may in turn serve as a pointer towards some underlying causal mechanisms which lead to the points being clustered in the way that they are. Evaluation of particular solutions will always be subject to the vagaries of human judgment; this is closely tied to the notion of gestalt perception, that faculty of the human perceptual system which enables us to perceive complex configurations as wholes

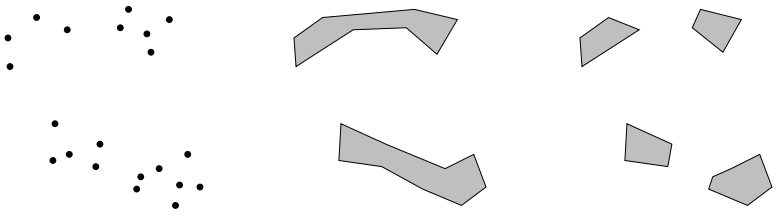


Fig. 4. Ambiguity in clustering

with global properties which cannot be simply derived by aggregating local properties. Our inquiry must therefore go beyond computational geometry to engage with more human-oriented disciplines such as cognitive science.

The criteria for evaluating any particular solution to the problem of associating a region with a set of points, or indeed any general *method* of solution, must in the last instance be determined by the particular application context in which it is being evaluated. We can, however, point to a number of general concerns which will be relevant to a wide range of application contexts and which we can therefore take to represent a basic set of criteria on top of which more refined criteria can be built as the application demands. There follows a list of such concerns; for convenience we shall use S to refer to the given set of points, and $R(S)$ to refer to the region proposed as representing their collective location.

1. Should every member of S fall within $R(S)$, as in Figure 5(a,c–g), or are outliers permitted, as in Figure 5(b)?
2. Should any points of S be allowed to fall on the boundary of $R(S)$, as in Figure 5(a–b,d–g), or must they all lie in its interior, as in Figure 5(c)?
3. Should $R(S)$ be topologically regular, as in Figure 5(a–c, e–g), or can it contain exposed point or line elements, as in Figure 5(d)?
4. Should $R(S)$ be connected, as in Figure 5(a–d,f,g), or can it have more than one component, as in Figure 5(e)?
5. Should $R(S)$ be polygonal, as in Figure 5(a–e,g), or can its boundary be curved, as in Figure 5(f)?
6. Should $R(S)$ be simple, i.e., its boundary is a Jordan curve, as in Figure 5(a–c,f), or can it have point connections as in Figure 5(g)?

Note that questions (1) and (2) are concerned with the relationship between the region $R(S)$ and the points in S, whereas (3)–(6) are all concerned with what kinds of regions we are prepared to admit to play the role of $R(S)$.

A further question, which is difficult to frame precisely, let alone answer, is concerned with the amount of 'empty space', unoccupied by points, that is allowed inside a region. Of course, *all* the area of the region apart from the points themselves (which mathematically may be of dimension zero, but in applications, as in our diagrams, might just as well be 'small regions') is empty space; what is meant is rather illustrated in Figure 6: on the left, Figure 5(a) is repeated, with the the largest circle contained in $R(S)$ but disjoint from S highlighted; on

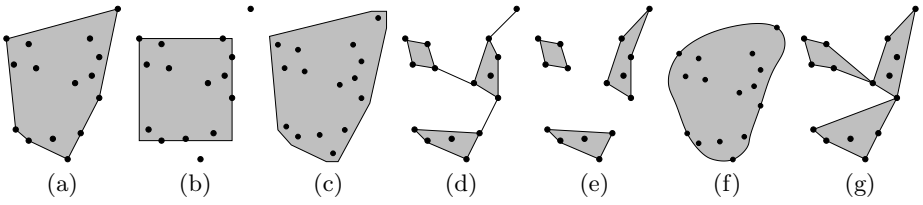


Fig. 5. Examples to illustrate the evaluation criteria

the right we do the same for Figure 5(d). In the latter, $R(S)$ clearly contains less 'empty space', and therefore for some purposes may be regarded as more desirable.

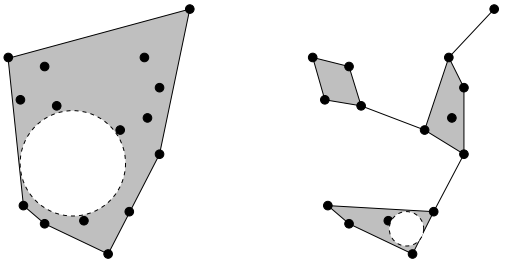


Fig. 6. Empty space as a criterion for region forming

Our question is therefore:

7. How big is the largest circular (or other specified) subregion of $R(S)$ that contains no elements of S?

Further evaluation criteria relate to the *methods* of solution themselves, rather than the results produced by them:

8. How easily can the method used be generalised to three (or more) dimensions?
9. What is the computational complexity of the algorithm?

Finally, it is important to note that we have deliberately chosen objective geometric and computational criteria for comparing $R(S)$ and methods of generating $R(S)$. Our list might also be extended with innumerable further perceptual criteria (e.g., the ability of human visual perception to discern different elements in $R(S)$) and wider cognitive criteria (e.g., the aesthetics of $R(S)$). However, such extensions would make comparison much harder and more laborious.

2 Convex Hulls

Ask a mathematician to define a region associated with a set of points, and there is a fair chance that they will nominate the convex hull. The convex hull

of a set of points S in the plane is the smallest convex polygon that encloses S. The convex hull has the advantage that it is uniquely defined, in a very simple way, for an arbitrary set of points. Moreover, algorithms for computing convex hulls have been thoroughly researched and a good deal is known about their computational properties (e.g., [1,2]).

How do convex hulls fare with respect to our list of evaluation criteria? Note that we are here assuming that the set of points S is finite; if infinite sets are allowed, then the answers to some of the questions will be different.[1]

1. Outliers are not allowed.
2. There are always points of S on the boundary of $R(S)$.
3. The convex hull is topologically regular (unless the points are collinear).
4. The convex hull is connected (a disconnected region is by nature non-convex).
5. The convex hull is polygonal.
6. The boundary of the convex hull is a Jordan curve (unless the points are collinear).
7. The amount of empty space included within the convex hull can be large.
8. The construction readily generalises to n dimensions.
9. There exist convex-hull algorithms of complexity $O(n \log n)$.

Characteristics 4 and 7 in particular mean that the convex hull is unsuitable for many purposes. A good outline—one which represents the shape of a point configuration in a cognitively salient way—must not allow too much empty space. Clustering can also be cognitively salient, and consequently in many cases $R(S)$ must be allowed to be disconnected to provide a satisfactory outline (although clustering might also be treated as a separate problem from outline generation).

These points are illustrated in Figure 7(a) and (b). In (a), we see that the convex hull entirely fails to capture an essential feature of the set of points under consideration, namely that they occupy a roughly C-shaped region. In (b), likewise, the convex hull fails to reveal that the points under consideration fall into two distinct clusters separated by a point-free gap that is comparable in size to the clusters themselves.

For this reason, we have been led to look for methods for generating *non-convex* 'hulls' of various kinds—as shown for the same sets of points in Figure 7(c) and (d). We shall refer to regions of this kind, which are intended to represent the location of a set of points, as *footprints* for that set. A natural way to approach this is by way of extending or generalising existing convex-hull algorithms. We shall consider some examples of this approach first before turning our attention to some alternative methods. Throughout, it should be borne in mind that, unlike the convex hull, footprints are not unique; it never makes sense to ask whether a proposed footprint is 'correct'—certainly not in isolation, but even with respect to any specified application context. For this reason, evaluation of the results is not, and never can be, an exact science, relying as it must to some extent on subtle and unquantifiable elements of human judgment.

[1] For example, if S is an open set, then none of its points lies on the boundary of the convex hull; and if S is a convex curvilinear figure then its convex hull (i.e., S itself) is non-polygonal.

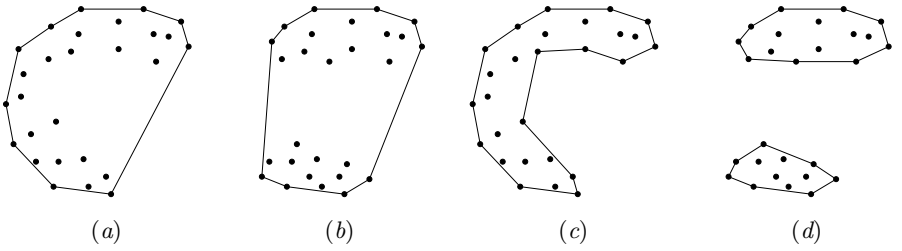


Fig. 7. Two convex hulls and two non-convex 'footprints'

3 Gift-Wrapping and the Swinging Arm

Our first procedure generalises the well-known 'gift-wrap' algorithm for generating convex hulls. In the planar (two-dimensional) version of this algorithm, the hull is generated by a sequence of swings of a half-line L, initially anchored at an extremal point of S. At each step, L is anchored to the last point added to the footprint, and rotated clockwise until it hits another point in S. This is added as the next vertex of the footprint, and the procedure is repeated until we return to the starting point.

The Swinging Arm procedure for generating footprints (see Appendix A) is essentially identical to this except that instead of a half-line we use a line-segment of some constant length r specified as an input to the algorithm. This is the 'swinging arm'. If r is not less than the longest side of the convex hull perimeter, then the procedure will generate the convex hull; but if it is shorter than that, the polygon it generates is non-convex. For sufficiently short arms, the algorithm will initially generate a footprint component which does not include all of S; in this case, we repeat the algorithm starting from a new extremal point selected from the points of S not already included in one of the components. This is repeated until every point of S is included in one of the components. The footprint thus generated will then be disconnected. When the length of the arm is less than the minimal separation of any two points in S, then the swinging arm will never encounter any points at any stage in the process: in this case the footprint is identical to S.

Figure 8 shows the sequence of steps by which the algorithm generates a footprint for a set of nine points, given an arm-length r. At step j, a circle of radius r is inscribed about the latest point ($H_{1,j}$) recruited to the footprint, and the next edge to be added to the footprint ($H_{1,j}H_{1,j+1}$) is indicated by a dashed line, previous edges being shown solid.

The footprint obtained will vary with the length of the swinging arm, as shown in Figure 9 for the same set of points as before. The values of r shown are those at which the footprint changes; for example, the footprint shown for $r = 3$ will result for any arm-length in the range $3 \leq r < \sqrt{10}$. For smaller values of r (e.g., $r = \sqrt{5}$), the polygons constituting the footprint may be degenerate, and for $r < \sqrt{5}$, the footprint is identical to S.

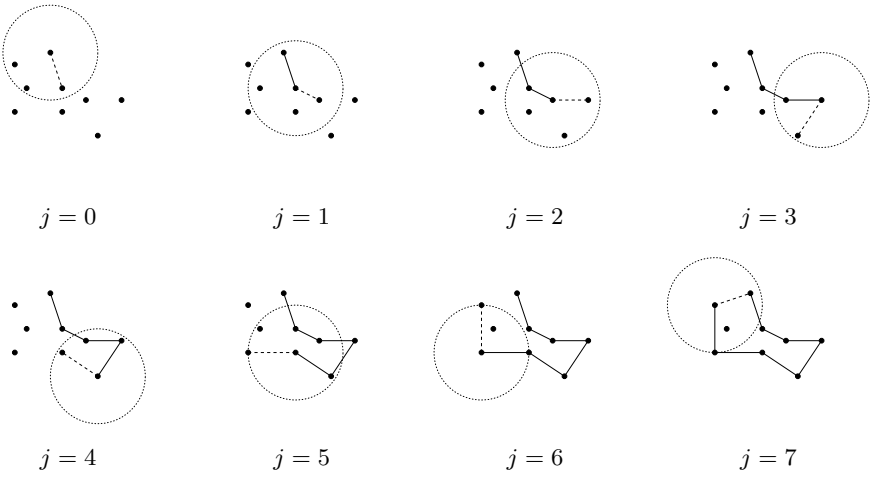


Fig. 8. Construction of a footprint using the 'swinging arm' algorithm

In the swinging arm algorithm presented above, the arm is stipulated to swing clockwise. Does it make any difference if instead we make it swing anticlockwise? In the example considered, there is one case where it does make a difference. The first two illustrations in Figure 10 show the results for arm-length $\sqrt{26}$, swinging clockwise and anticlockwise. Inspection of these figures will reveal the characteristic configuration of points which gives rise to the difference.

Since the direction in which the arm is swung is essentially arbitrary, it is unsatisfying that the choice can lead to different results. For this reason, one might prefer to use the union of the clockwise and anticlockwise swinging-arm footprints for a given set of points, as shown in the third illustration in Figure 10. This idea leads naturally on to the further idea that, in order to obtain this, we could discard the swinging arm altogether, and instead simply join together *all* pairs of points whose separation is less than or equal to the arm length, as shown in the rightmost illustration.[2] To obtain the footprint, we then simply include all points lying within any closed polygon formed out of these joins. This method, which we shall call the *Close Pairs* method, could also be seen as a direct generalisation of a method for generating the convex hull, since if we join together all pairs of points regardless of their separation then we obtain the convex hull.

How do the Swinging Arm (SA) and Close Pairs (CP) methods stand with respect to our list of general evaluation criteria? Except where otherwise stated, the comments below apply equally to both methods.

1. Every member of S falls within $R(S)$.
2. There must be points of S on the boundary of $R(S)$.
3. $R(S)$ is in many cases topologically non-regular.

[2] This the same as the '$r_{\min}$-circle graph' of [3].

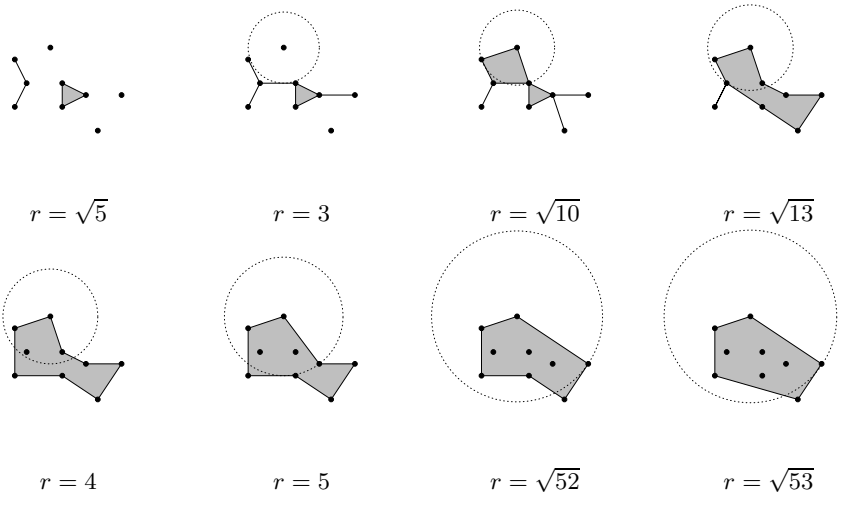


Fig. 9. Effect of varying arm-length on the footprint

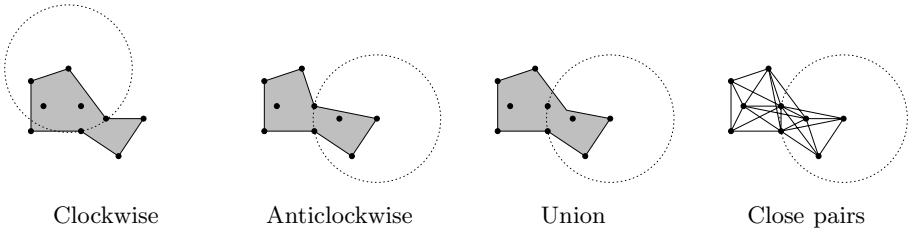


Fig. 10. Clockwise *vs* anticlockwise swinging arm ($r = \sqrt{26}$)

4. $R(S)$ is in many cases disconnected.
5. $R(S)$ is always polygonal, so long as this term is understood to include degenerate polygons such as polylines or isolated points.
6. In many cases, a large area of empty space included within the convex hull will be excluded from the footprint when r is made short enough (for example the large concavity of a 'C'-shaped distribution of points). This is not always the case, however, for example if the points of S are, say, the vertices of a many-sided regular polygon; if r is at least as long as the side-length of the polygon, the SA and CP algorithms will both give the convex hull, including the large empty interior.
7. $R(S)$ may have point connections (i.e., non-Jordan boundary).
8. Generalising SA to three dimensions is possible, but not straightforward—we need to use a 'swinging flap', and whereas in two dimensions the boundary of a region is linear, and hence can be constructed sequentially, in three-dimensions there is the added complication that we have always to decide which edge to swing the flap about next.

Generalising CP seems more straightforward: first we must join together all pairs of points whose separation is less than r; next include the interior of any planar polygon formed from the edges thus obtained; and finally include the interior of any polyhedron bounded by those polygons. A complication here is that one might also wish to include the interiors of non-planar polygons, but then it is not necessarily obvious how the interior of such a polygon is to be defined.

9. For both SA and CP the computational complexity is larger than one might imagine at first sight.

   With SA, in the course of generating a footprint-component, the swinging arm might encounter a point which already lies in the interior of that component; such points should not be included in the perimeter of the footprint. In the algorithm this is accomplished at step 3(e)(ii) in which such interior points are marked as unavailable; this adds significantly to the complexity. However, even with this complication, the worst-case complexity is $O(n^3)$, and in practice it is usually much better than this ($O(n^2)$ or less). Note that the complexity depends on r as well as the size of the input set—in general, the complexity decreases as r increases, up to the point at which the footprint becomes convex.

   With CP, having constructed the edges one must then identify the closed polygons. Note that the diagrams in Figure 12 were produced simply by selecting the edges of length less than r, for a range of suitable values of r. This is sufficient for the human eye to perceive the relevant footprint, but a CP algorithm needs to identify that footprint as an explicit polygon. Although we have not implemented this, initial investigations suggest that such an algorithm would be computationally quite expensive. Overall, a time complexity of $O(n^2)$ seems the best that could possibly be hoped for, with the likelihood that a more thorough investigation of a CP algorithm would reveal a worse complexity.

Regarding item 6, a simple refinement of the CP algorithm allows one to eliminate large areas of empty space. Instead of including the interiors of *all* polygons formed by the edges of length less than r, we can select only those polygons with less than some specified number of sides. For example. we could restrict it to just the triangles. A triangle whose sides are of length at most r has area at most $0.433r^2$ (i.e., $\frac{\sqrt{3}}{4}r^2$)—but the largest *circle* that can be inscribed in such a triangle has area at most $0.262r^2$ (i.e., $\frac{1}{12}\pi r^2$).

4 A Delaunay-Based Method

Work by Duckham et al. [4] has used Delaunay triangulations as the basis for defining a footprint for a set of points. The method, here designated DT, begins by building the Delaunay triangulation of the points. Then the algorithm 'shaves off' boundary edges in decreasing order of length, subject to constraints that ensure that the final shape contains all the input points and is both regular and

has a Jordan curve boundary (topologically isomorphic to the unit cell). Figure 11 provides an illustration of the process.

The DT algorithm terminates at a uniquely defined final shape. In Figure 6, this terminal shape is shown in step 8, where no further edges can be removed without violating the regularity and Jordan constraints. However, in general, running the algorithm to its terminal shape can produce very sinuous shapes with low visual salience. Therefore, Duckham et al. suggest a number of ways of parameterizing the algorithm based on a minimum edge-length which leads to shapes that have higher visual salience. The final algorithm presented by Duckham et al. is highly efficient in terms of time complexity.

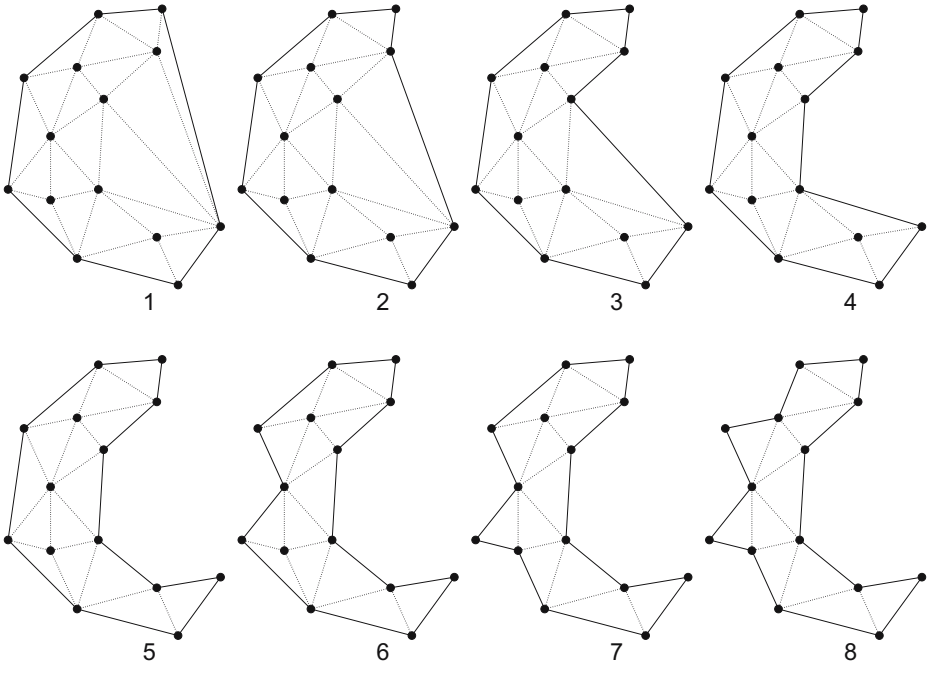


Fig. 11. Some footprints produced by edge-removal from a Delaunay triangulation

A comparison of the DT method (Figure 11) with the CP method (Figure 12) reveals that while for some values of r, the CP-footprint coincides with the corresponding DT footprint, the CP method also produces footprints that are inaccessible via the DT method. Figure 12 shows a selection of CP-footprints generated for the same set of points used in Figure 11. Note in particular that footprint 3 cannot be a DT footprint since two of the vertices arise from the crossing of two edges. Note further that as r becomes small (as in footprints 7 and 8), non-regularity sets in. Figure 12.6 has a point-connection—i.e., two triangles joined only at a shared vertex; topologically, this is regular (the footprint

is the closure of its interior), but this feature is also disallowed by the DT method. Footprints 7 and 8 are not only non-regular, they are also disconnected, another feature not possible for a DT-footprint.

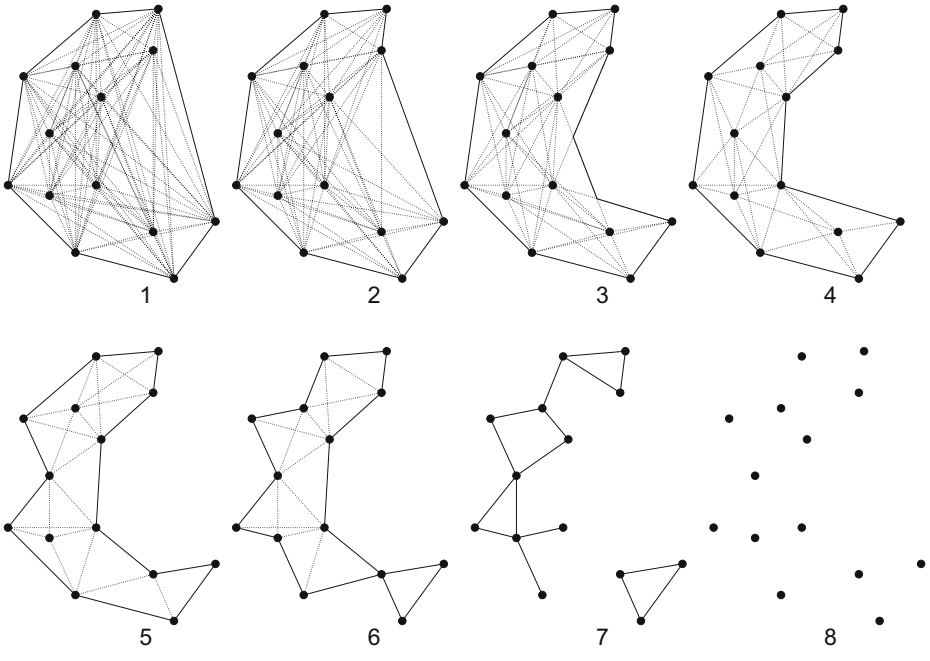


Fig. 12. Some footprints produced by the 'close pairs' method (cf. Figure 11)

Our list of evaluation criteria can be applied to the DT method of [4], with results as follows:

1. Every member of S falls within $R(S)$.
2. There are always points of S on the boundary of $R(S)$.
3. $R(S)$ will always be topologically regular.
4. $R(S)$ will always be connected.
5. $R(S)$ will always be polygonal.
6. The boundary of $R(S)$ will always be a Jordan curve.
7. It is possible for $R(S)$ to contain a large amount of empty space, like a convex hull—although since $R(S)$ must be a subset of the convex hull, it cannot contain *more* empty space. The requirements for regularity and simplicity mean, for example, that it is not possible to generate polygons with holes using the DT algorithm. Thus, a 'ring' of points generates a circular polygon hull rather than an annulus.
8. Extensions to three dimensions are problematic (see below).
9. Duckham et al. report that their algorithm achieves a computational complexity of $O(n \log n)$, although this efficiency depends on the regularity and simplicity constraints, as well as only holding in two dimensions.

Duckham et al. argue that the regularity constraints are useful both from the perspective of visual salience, and from the perpective of algorithm efficiency. It is conceivable that a variety of different algorithms might be developed based on Delaunay triangulations and without regularity constraints. These would allow, for example, the generation of disconnected regions and regions with holes (hence, the criteria 3, 4, 6 and 7 listed for the DT method above do not in general hold for all Delaunay triangulation-based methods). However, we can observe a number of features of any Delaunay-based algorithm, regardless of the specific algorithm used.

1. Starting with a Delaunay triangulation immediately constrains the solution space for the boundary of the resulting region, much more so than the SA and CP methods. Some edges that might be added using SA and CP methods are not present in the Delaunay triangulation, and so can never be present in the final shape; the converse is not true.
2. Developing an algorithm that must compute the Delaunay triangulation immediately places a lower bound of $\Omega(n \log n)$ on the optimal time complexity for the entire algorithm (in itself not a negative feature, since this clearly still allows for the specification of efficient, scalable Delaunay-based algorithms).
3. Although Delaunay triangulations can be extended beyond two dimensions, there are implications for extending Delaunay-based algorithms to higher-dimensional spaces. For example, the possibility of unshellable triangulations in more than two dimensions presents problems for extending Duckham et al.'s method to three or more dimensions.

5 Extended Footprints

Why should we insist on finding a polygon (or similar) region whose vertices are points of the original set? To represent the region occupied by the points at a certain granularity, it is natural to suppose that each point has an 'area of influence', and to represent the region by the aggregation of all the areas of influence of the points in the set. The points themselves would then all lie in the interior of the region, with none on the perimeter, as in Figure 5(c): this leads to solutions which differ from all the ones considered up to now with respect to our second criterion. Let us call such a region an *extended footprint* for the set of points.

An obvious way of generating an extended footprint is as follows. For each point in the set, construct a closed circular disc of radius $\frac{1}{2}r$ centred on that point, and let the region be the union of these discs. Let us call this the Covering Discs (CD) method. An example is shown in Figure 13. To produce a smoother outline, we could draw in the shared tangents of any pair of discs which overlap, and include also the area between the tangents. This is the Covering Discs with Tangents (CDT) method; an example is shown in the right-hand illustration of Figure 13.

The CD/CDT method is closely related to the dilation and erosion operations common in computer graphics and mathematical morphology (e.g., [5]). Simple and obvious though this method is, the footprints produced do not seem very

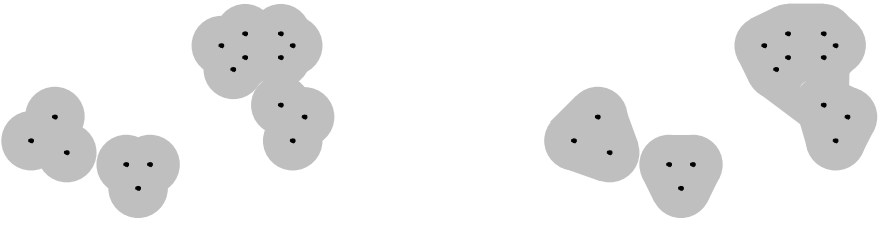


Fig. 13. An extended footprint without and with added tangents

natural. An important question concerns how far the footprint should extend into the 'outer space'; the CD method assumes that this distance should be the same all round the periphery of the footprint, and this may indeed be the most natural default position. However, one situation where we may be able to make a more informed choice about how far the footprint should extend beyond the points is when we have additional information about points which are to be *excluded* from the region. This allows one to make use of the methods of [6] and [7] which are briefly discussed in the next section.

6 Related Work

In terms of motivation, the closest work we are aware of is that of Edelsbrunner *et al.* [8], in which the concept of 'α-shape' is developed as a generalisation from a certain definition of convex hull. The α-shape of a point-set S is a straight-line graph derived from the α-hull, which is defined to be the intersection of all closed discs of radius $1/\alpha$ that contain all the points of S. (If $\alpha < 0$, the closed disc of radius $1/\alpha$ is interpreted as the complement of the open disc of radius $-1/\alpha$.) For $\alpha = 0$, we have the convex hull; for sufficiently large negative values of α, we have the set S itself. Unlike with our footprints, some of the points of S may lie outside the α-shape (cf. criterion 1).

Traka and Tziritas [9] give an algorithm for constructing a non-convex hull $R(S)$ by starting with the convex hull of S and successively adding extra points from S to the perimeter; but for their purposes it is required that *all* points of S lie on the perimeter of $R(S)$ (which is therefore a Hamiltonian circuit of the complete graph on those points), whereas for our more general purposes this condition is unnecessarily stringent.

In the context of GIS, a method based on Voronoi diagrams has been suggested by [6]. This is applicable where the points S are representative localities within some region R, and in addition we are given a set S' of points known to lie outside R. In this Dynamic Spatial Approximation Method (DSAM), the Voronoi diagram of $S \cup S'$ is constructed, and an approximation to region R is derived as the union of the Voronoi cells containing members of S. This approximation to R is a kind of footprint, typically non-convex, of the set S. It is, of course, an extended footprint. An example is shown in Figure 14(a), where the initial set of points S (shown by the black circles) is supplemented by additional

points (shown by the white circles) which are supposed to be definitely outside the target region R. The dotted lines give the Voronoi tessellation for the full set of points, and the solid line shows the boundary suggested by the DSAM method for region R. The resulting approximation to R is shown shaded. It consists, in fact, of all those points for which the nearest classified point is in S.

A drawback is that this method can only be used if a suitable set S' is given, which is often not the case; however, one might consider modifying the method to cover cases where all we are given is the set S, by introducing some arbitrary set of points lying outside the convex hull and using these as S'. The method could be iterated: once a footprint has been formed by the DSAM method, we create a new S' from points selected to lie outside that footprint, and then generate a new footprint. Successive iterations will lead to footprints that cling more tightly to the original set S.[3]

A related method, involving Delaunay triangulations, is suggested in [7]. This is illustrated in Figure 14(b). Here the Delaunay triangulation of the same set S is shown; each edge of the triangulation is either dotted (if it joins two 'white' points), dashed (if it joins two 'black' points), or solid (if it joins a 'white' and a 'black' point). Region R (shown shaded) is obtained by joining up the midpoints of the solid edges in the triangulation. It will be seen that while it resembles the Voronoi-based approximation, it has the advantage of giving a smoother outline.

In three-dimensions, the Power Crust method of [10,11], which is based on Medial Axis Transforms, provides good 'footprints'[4] very much in the spirit of our own requirements. However, the application context of this work is surface reconstruction from data points captured from real objects. As a consequence, the initial set of points are required to lie on the surface of the output region, which means that the problem under consideration is somewhat different—albeit related—to ours.

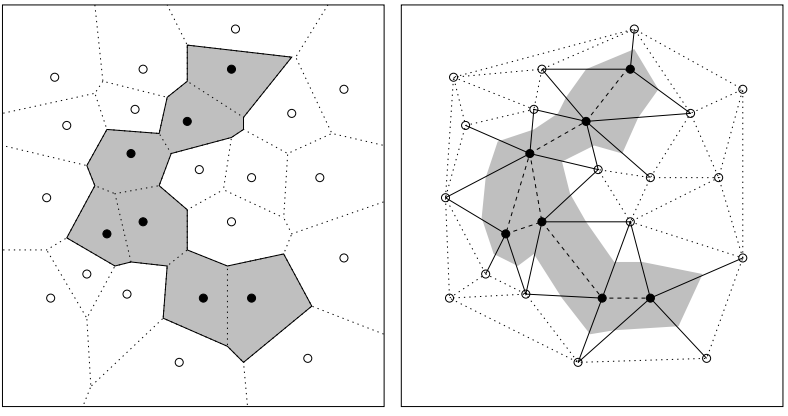
Our notion of a footprint is in some ways superficially reminiscent of, though in essence quite different from, the *relative convex hull* of [12]. The relative convex hull of a region $R \subseteq R'$ is the figure with the minimum perimeter whose perimeter lies entirely in the closure of $R' \setminus R$.

The fact that our footprints are parametrised by r, the swinging arm-length (or disc diameter), recalls multiresolution approaches to shape such as [13] and indeed reflects the same underlying motivation of revealing different kinds of structure existing at different granularities. However, these approaches are essentially concerned with the shapes of regions, whereas we are concerned with constructing a region to represent a discrete set of points.

Having obtained such a region, we still need to be able to describe the resulting shape—what is it, for instance, about Figure 7(c) which makes it a C-shape: where does the *linearity* of the 'C' come from? Features such as this might be revealed by some form of *skeletonisation* procedure. Skeletonisation techniques

[3] This method has been investigated by Nicholas Talbot in an unpublished final-year undergraduate project supervised by Antony Galton at Exeter.

[4] The inverted commas here merely reflect the slight oddity of referring to a three-dimensional region as a footprint.



(a) (b)

Fig. 14. Approximations to the region defined by a set of points, using (a) a Voronoi tessellation (after Alani *et al.* [6]) and (b) a Delaunay triangulation (after Arampatzis *et al.* [7]).

are being extensively investigated by a number of researchers, e.g., [14], but not in connection with our footprint problem.

7 Conclusions and Further Work

In this paper we have only scratched the surface of what is potentially a rich field for investigation. We have described a number of methods for generating footprints for sets of points, and have hinted at some possible applications for such techniques. However, for the further evaluation of these different methods, it will be necessary to adopt a sharper focus in this respect: ultimately, it will only be in specific contexts of application that a final evaluation of any given method can be given, and this must serve as a pointer to further work.

Other topics for investigation include the mathematical properties of the footprints derived by the various methods, a closer examination of the computational properties of the algorithms (and in particular a more detailed analysis of their complexity), and, of course, a generalisation of the methods to three dimensions.

In relation to the first of these topics, it would be useful to investigate the relationship between our various kinds of footprint and the general geometrical notion of a *hull*. Klette and Rosenfeld [15] give the conditions for a hull operator H as

H1. $S \subseteq H(S)$
H2. $S_1 \subseteq S_2 \rightarrow H(S_1) \subseteq H(S_2)$
H3. $H(H(S)) \subseteq H(S)$.

If H2 is replaced by

H4. $S_1 \subseteq S_2 \rightarrow area(H(S_1)) \leq area(H(S_2))$

then we have what is called a 'near-hull'. If neither H2 nor H4 is stipulated then we have a 'pseudo-hull'. Our footprints are clearly related to these various kinds of hull but for the most part do not exactly correspond to any of them. In particular, note that property H3 will not be applicable to those footprint-generating methods which require a finite set of points as input, since in that case the footprint operator R cannot be applied to a second time, so there is no such region as '$R(R(S))$'.

Another issue that needs to be investigated is the relationship between footprint formation and clustering. Algorithms which can generate footprints with multiple components do, by that very fact, function as clustering algorithms; but they are not necessarily very good clustering algorithms, and it is arguable that since clustering is a very different topic from footprint formation, it should not be attempted to accomplish both using a single algorithm. A better solution might be to apply some dedicated clustering algorithm first, and then derive footprints from each cluster individually. One obvious possibility would be the nearest neighbor (single link) cluster algorithm, which generates clusters based on a sub-graph of the Delaunay triangulation [16]. Although not clustering algorithms, the Gabriel graph [17] and the relative neighborhood graph [18] are two other well-studied sub-graphs of the Delaunay triangulation that would be potentially useful in characterizing the region occupied by a set of points.

Acknowledgments

Antony Galton's work has benefited from discussions with Pier Frisco, Joviša Žunic, Nick Talbot, and Max Dupenois. The discussion of the Delaunay triangulation footprint method summarises some collaborative work between the authors and Lars Kulik, Mike Worboys, and Glenn Hudson. Matt Duckham is partially supported by Australian Research Council (ARC) Discovery grant no. DP0662906. Finally, both authors would like to acknowledge the constructive comments of the three anonymous reviewers.

References

1. de Berg, M., Schwarzkopf, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and applications. 2nd edn. Springer, Berlin (2000)
2. O'Rourke, J.: Computational Geometry in C. 2nd edn. Cambridge University Press, Cambridge, UK (1998)
3. van Kreveld, M.: Finding the wood by the trees. CG Tribune (1998) http://www.inria.fr/prisme/personnel/bronnimann/cgt/cgt10.ps.
4. Duckham, M., Kulik, L., Worboys, M., Galton, A.: Efficient characteristic hulls. (2006, in preparation)
5. Serra, J.: Image Analysis and Mathematical Morphology. Academic Press, New York (1982)
6. Alani, H., Jones, C.B., Tudhope, D.: Voronoi-based region approximation for geographical information retrieval with gazetteers. International Journal of Geographical Information Science **15** (2001) 287–306

7. Arampatzis, A., van Kreveld, M., Reinbacher, I., Jones, C.B., Vaid, S., Clough, P., Joho, H., Sanderson, M., Benkert, M., Wolff, A.: Web-based delineation of imprecise regions. In: Workshop on Geographic Information Retrieval (SIGIR 2004). (2004) (http://www.geo.unizh.ch/~rsp/gir/abstracts/arampatzis.pdf, accessed 8/7/05).
8. Edelsbrunner, H., Kirkpatrick, D.G., Seidel, R.: On the shape of a set of points in the plane. IEEE Transactions on Information Theory **IT-29** (1983) 551–559
9. Traka, M., Tziritas, G.: Panoramic view construction. Signal Processing: Image Communication **18** (2003) 465–481
10. Amenta, N., Choi, S., Kolluri, R.: The power crust. In: Sixth ACM Symposium on Solid Modeling and Applications. (2001) 249–260
11. Amenta, N., Choi, S., Kolluri, R.: The power crust, unions of balls, and the medial axis transform. Computational Geometry: Theory and Applications **19** (2001) 127–153
12. Sklansky, J., Kibler, D.: A theory of nonuniformly digitized binary pictures. IEEE Transactions on Systems, Man, and Cybernetics **6** (1976) 637–647
13. Cinque, L., Lombardi, L.: Shape description and recognition by a multiresolution approach. Image and Vision Computing **13** (1995) 599–607
14. Borgefors, G., Nyström, I., Sanniti di Baja, G.: Computing skeletons in three dimensions. Pattern Recognition **32** (1999) 1225–1236
15. Klette, R., Rosenfeld, A.: Digital Geometry: Geometric Methods for Digital Picture Analysis. Morgan Kaufmann (2004)
16. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks **16** (2005) 3
17. Gabriel, K.R., Sokal, R.R.: A new statistical approach to geographic variation analysis. Systematic Zoology **18** (1969) 259–278
18. Toussaint, G.T.: The relative neighborhood graph of a finite planar set. Pattern Recognition **12** (1980) 261–268

A The Swinging Arm Algorithm

Input: A finite set S of points in the plane, and a positive real number r.
Output: A footprint of S.

1. Mark all points of S as 'available' and 'unvisited'.
2. Let $i = 0$.
3. Repeat
   (a) Increase i by 1.
   (b) Let $H_{i,0}$ be the available point in S with maximal y-coordinate (if there is more than one such, choose the one with minimal x-coordinate).
   (c) Let L be a line-segment of length r anchored at $H_{i,0}$ parallel to the positive y-axis.
   (d) Let $j = 0$.
   (e) Repeat
      i. Rotate L clockwise about $H_{i,j}$ until either it meets another available point in S or it returns to its starting position. In the former case, let $H_{i,j+1}$ be the point found (if more than one, choose the one closest to $H_{i,j}$); in the latter, let $H_{i,j+1} = H_{i,j}$.

ii. If $H_{i,j+1}$ is marked as visited, then identify the greatest $k \leq j$ such $H_{i,j+1} = H_{i,k}$, and mark as unavailable all points of S in the interior of the polygon with vertices $H_{i,k}, H_{i,k+1}, \ldots, H_{i,j}$.
iii. Mark $H_{i,j+1}$ as visited.
iv. Let L be a line-segment of length r anchored at $H_{i,j+1}$ and passing through $H_{i,j}$.
v. Increase j by 1.

until $H_{i,j} = H_{i,0}$.

(f) Add the polygon with vertices $H_{i,0}, H_{i,1}, \ldots, H_{i,j-1}$ as component C_i of the footprint, and mark all those vertices as unavailable.

until all points in S are unavailable.

4. Return components $C_1, \ldots, C_i$.