Delineating Boundaries for Imprecise Regions

Iris Reinbacher

Marc Benkert

Marc van Kreveld

Joseph S. B. Mitchell

Jack Snoeyink

Alexander Wolff

institute of information and computing sciences, utrecht university

technical report UU-CS-2005-026

www.cs.uu.nl

Delineating Boundaries for Imprecise Regions*

Iris Reinbacher† Marc Benkert‡ Marc van Kreveld†
Joseph S. B. Mitchell§ Jack Snoeyink¶ Alexander Wolff‡

Abstract

In geographic information retrieval, queries often utilize names of geographic regions that do not have a well-defined boundary, such as "Southern France." We provide two classes of algorithms for the problem of computing reasonable boundaries of such regions, based on evidence of given data points that are deemed likely to lie either inside or outside the region. Our problem formulation leads to a number of problems related to red-blue point separation and minimum-perimeter polygons, many of which we solve algorithmically. We give experimental results from our implementation and a comparison of the two approaches.

1 Introduction

Geographic information retrieval is concerned with information retrieval for spatially related data, including Web searching. Certain specialized search engines allow queries that ask for things (hotels, museums) in the (geographic) neighborhood of some named location. These search engines cannot use the standard term matching on a large term index, because the user is not interested in the term "neighborhood" or "near". When a user asks for Web pages on museums near Utrecht, not only Web pages that contain the terms "museum" and "Utrecht" should be found, but also Web pages of museums in Amersfoort, a city about 20 kilometers from Utrecht. Geographic search engines require ontologies—geographic databases—to be able to answer such queries. The ontologies store all geographic information, including coordinates and geographic concepts. Geographic search engines also require a combined spatial and term index to retrieve the relevant Web pages efficiently.

Besides specifying neighborhoods, users of geographic search engines may also use containment and directional concepts in the query. Furthermore, the named location need not be a city name or region with well-specified, administrative boundaries. A typical query could ask for campgrounds in Northern Portugal, or specify locations such as Central Mexico, the Bible Belt, or the British Midlands. The latter two are examples of named regions for which no exact boundaries exist. The extent of such a region is in a sense in the minds of the people. Every country has several such imprecise regions.

Since geographic queries may ask for Web pages about castles in the British Midlands, it is useful to have a reasonable boundary for this imprecise region. This enables us to find Web pages for locations in the British Midlands that mention castles, even if they do not contain the words British Midlands. We need to store a reasonable boundary for the British Midlands in the geographic ontology during preprocessing, and use query time for searching in the spatial part of the combined spatial and term index.

*This research is supported by the EU-IST Project No. IST-2001-35047 (SPIRIT) and by grant WO 758/4-2 of the German Science Foundation (DFG).

†Institute of Information and Computing Sciences, Utrecht University, {iris,marc}@cs.uu.nl

‡Department of Computer Science, Karlsruhe University, i11www.ira.uka.de/algo/group

§Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, jsbm@ams.sunysb.edu

¶Department of Computer Science, University of North Carolina at Chapel Hill, snoeyink@cs.unc.edu

To determine a reasonable boundary for an imprecise region we can use the Web once again. The enormous amount of text on all Web pages can be used as a source of data; the idea of using the Web as a geo-spatial database has appeared before [17, 20, 23]. A possible approach is using so-called *trigger phrases*. For any reasonable-size city in the British Midlands, like Nottingham, it is quite likely that some Web page contains a sentence fragment like "...Nottingham, a city in the British Midlands, ...", or "Nottingham is located in the British Midlands...". Such sentence fragments give a location that is most likely in the British Midlands, while other cities like London or Cardiff, which do not appear in similar sentence fragments, give locations that are not in the British Midlands. Details of using trigger phrases to determine locations inside or outside a region to be delineated can be found in [2]. Obviously the process is not very reliable, and false positives as well as false negatives are likely to occur.

We have arrived at the following computational problem: given a set of "inside" points (red) and a set of "outside" points (blue), determine a reasonable polygon that separates the two sets. Imprecise regions generally are not thought of as having holes or a tentacle-shaped boundary, but rather a compact shape. Therefore, possible criteria for such a polygon are:

- The red points are inside and the blue points are outside the polygon.
- The polygon is simply-connected.
- The polygon has small perimeter.
- The polygon has small absolute angular change.
- The polygon has large compactness ratio, circularity ratio, or form ratio, or it has small elongation ratio. These are shape measures for polygons used in geography [14, 21].

Possibly, a polygon with much better shape can be obtained if a small number of red points are outside and a small number of blue points are inside the polygon. These points may then be misclassified. A suitable balance is needed between the shape measure and the misclassification measure. The field of machine learning is largely devoted to the use of points of known classification (e.g., red and blue) to infer the classification of other points (see, e.g. [10]). In machine learning applications, the feature space is usually high-dimensional and one is generally not constructing explicit region boundaries, as we are motivated to do in our geographical application. In pattern recognition, extraction of shapes from point patterns is also studied [26]. However, since we concentrate on a geographical application, we will only consider the polygon criteria listed above.

In computational geometry, red-blue separation algorithms exist of various sorts; see Seara [24] for a survey. Red-blue separation by a line—if it exists—can be solved by two-dimensional linear programming in $O(n)$ time for n points. Red-blue separation by a line with the minimum number of misclassified points takes $O((n + k^2) \log k)$ expected time, where k is the number of misclassified points [9]. Other fixed separation shapes, such as strips, wedges, and sectors, have also been considered [3, 24]. For polygonal separators, a natural problem is the minimum perimeter polygon that separates the bichromatic point set. This problem is NP-hard (by reduction from Euclidean traveling salesperson [4, 11]); polynomial-time approximation schemes follow from the m-guillotine method of Mitchell [19] and from Arora's method [7]. Minimum-link separation has also received attention [6, 1, 18].

In this paper we present two approaches to determine a reasonable polygon for a set of red and blue points. Based on these approaches we define and solve various algorithmic problems, some of which are of theoretical, some of which are of practical interest. The two global approaches are outlined in Section 2. The first approach takes a red polygon with blue and red points inside, and tries to adapt the polygon to get the blue points outside while keeping the red points inside. We show that for a red polygon with n vertices and only one blue point inside, the minimum perimeter adaptation can be computed in $O(n)$ time. For the case of one blue and $O(n)$ red points inside, an $O(n \log n)$-time algorithm is presented. If there are m blue points but no red points inside, an $O(m^3 n^3)$-time algorithm is given. If there are m red and blue points inside, we give an $O(C^{m \log m} \cdot n)$-time algorithm, for some constant C. These results are given in Section 3. The second approach changes the color of points to obtain a better shape of the polygon. Different recoloring strategies and algorithms are presented in Section 4. The implementation and test results on several data sets for both approaches are given in Section 5.

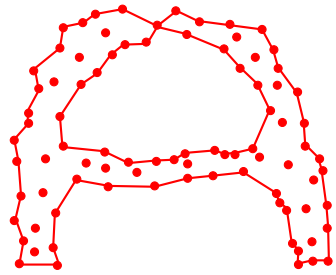

Figure 1: The α-shape of a set of points for a negative value of α.

2 Approaches

This section describes two approaches to obtain a reasonable boundary for a bichromatic set of points that may contain wrongly colored points. Let R be the set of red points and B the set of blue points.

2.1 Adaptation approach

In the adaptation approach we begin by computing on the red points only. A reasonable shape for the red points can for example be based on the α-shape [12, 13]. The concept of α-shapes formalizes the intuitive notion of "shape" for spatial point set data, and they have been studied and used extensively in geometric modeling, grid generation, medical image analysis, and visualizing the structure of earthquake data.

The formal definition of α-hulls, taken from [12], is:

Definition 1 *An* α-disc *for* $\alpha \in \mathbb{R}$ *is*

- *a closed disc of radius* $1/\alpha$ *if* $\alpha > 0$*,*
- *a closed half-plane if* $\alpha = 0$*, and*
- *the closed complement of a disc of radius* $-1/\alpha$ *if* $\alpha < 0$*.*

For a finite set S *of sites in the plane, an* α*-disc is* full *if it contains* S*. The* α-hull *of* S *is the intersection of all full* α*-discs.*

If $\alpha = 0$, the α-hull is the convex hull of the point set S. An *α-shape* is closely related to the α-hull, but it has straight edges. For every pair of points that give an α-disc that defines some part of the boundary of the α-hull, there is an edge between them in the α-shape. Figure 1 gives an example; a formal definition is complex and can be found in [12].

A reasonable choice for a polygon P to delineate an imprecise region enclosing the red points is the outermost set of edges of the connected component of the α-shape that contains the largest number of red points. This polygon is not necessarily simple because vertices may appear more than once on the boundary. We call such a polygon a *degenerate simple polygon*. The value of α has large influence on the shape; it should be chosen depending on the density of cities and towns in the region of interest. Using the component of the α-hull with most red points takes care of red outliers, which are isolated or appear in smaller components. These red points are likely to have been misclassified. This can happen due to ambiguity of place names, or by incorrect listing of a place name in a trigger phrase.

In the adaptation approach, we start with the polygon P based on the red points only. The polygon P may contain blue points, so we will change its shape to get them outside, but without bringing any red points outside. In practice we will only bring blue points outside if this does not change the shape of the polygon too much (for example, the perimeter increase is small). Blue points that remain inside are likely to have been misclassified; they correspond to places that are in the imprecise region, but are not mentioned in any trigger phrase.

The theoretical question of changing a polygon while increasing its perimeter as little as possible gives rise to various computational problems that we discuss in the next section. In all cases we are interested in a subpolygon $P^* \subseteq P$ that does not contain the blue points in the interior, nor any red points in the exterior, and has smallest perimeter among these or the smallest absolute angular change. The polygon P^* will be simple and possibly degenerate. We will give different algorithms for various instances of the problem, e.g. for convex and simple polygons with one or more blue points inside only and with red and blue points inside, in Section 3. Experimental results are given in Section 5.

2.2 Recoloring approach

In the recoloring approach, we change the colors of points that are likely to have been misclassified. This is the case if many surrounding points have the other color. Let R be the set of red points and B the set of blue points. Compute the Delaunay triangulation of $R \cup B$. For each point, consider its color and the colors of its neighbors. If the point is largely surrounded by points of the other color, then we change its color, and continue. To formalize "largely surrounded", we consider for a point the maximum angle of consecutive differently colored neighbors. If this angle is greater than some pre-specified value, say, 230°, then we recolor. A more formal definition of the angle is given in Section 4. If the critical angle is smaller than 180°, then the method may not terminate so we always assume that it is at least 180°.

In general there are several points that can be recolored at some moment, and it is important in which order the recoloring is performed. Different orders can give different final colorings. We examine different recoloring schemes and prove bounds on the number of recolorings that can occur in certain schemes in Section 4. Experimental results of the recoloring approach are given in Section 5.

3 Adaptation method

In the adaptation method, we start with a polygon P and adapt it until all blue points inside P are no longer inside, or the shape has to be changed too dramatically. By choosing P initially as an α-shape, with α chosen such that e.g. 90% of the red points lie inside P, we can determine an appropriate initial shape and remove red outliers (red points outside P) in the same step. The parameter α can also be chosen based on "jumps" in the function that maps α to the perimeter of the initial polygon that we would choose. Once P is computed, the remaining problem is to change P so that the blue points are no longer inside. The resulting polygon P^* should be contained in P and its perimeter should be minimum. In this section we discuss the algorithmic side of this problem. In practice it may be better for the final shape to allow some blue points inside, which then would be considered misclassified. Some of our algorithms can handle this extension.

3.1 One blue point inside P

First, we assume that there is only one blue point b inside P. The optimal, minimum-perimeter polygon P^* inside P has the following structure:

Lemma 1 *An optimal polygon P^* is a – possibly degenerate – simple polygon that (i) has b on its boundary, and (ii) contains all edges of P, with the exception of one edge.*

We consider two versions of the problem: the special case where P contains only one point (namely b), and the general case where P contains b and a number of red points.

3.1.1 One blue and no red points inside P

Let P be a red polygon with only one blue point b inside. Let $e = \overline{v_1 v_2}$ be the edge of P that is not an edge of P^*. The endpoints v_1 and v_2 are connected by a path F via b inside the boundary

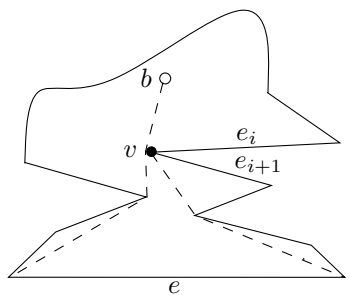

Figure 2: Illustration of the proof of Lemma 2.

of P. We denote the path that leads to the minimum-perimeter polygon P^* by F^*; it consists of a shortest geodesic path between b and v_1, and between b and v_2. Note that these paths may have vertices at red points other than v_1 and v_2; such vertices are concave vertices of P, as shown in Figure 3 (left). In the optimal solution P^*, the edge e and the shortest path F^* have the following additional properties:

Lemma 2 *(i) The path F^* is a simple funnel. (ii) A funnel F with root b and base e can only be minimal if e is partially visible from b.*

Proof: We prove the first claim by contradiction (see Figure 2). Let P be a simple polygon with a point b inside. Let e and F^* be the edge and path that lead to the minimum perimeter polygon P^*, and assume that F^* has some edge on both shortest geodesic paths between b and v_1 and v_2. Then the edges incident to b must also be the same. Let this shared segment be $\overline{bv}$. Vertex v is the endpoint of two edges e_i, e_{i+1} of P^*. The solution using e and F^* has perimeter strictly greater than the perimeter of P plus twice the length of $\overline{bv}$. However, using e_i instead of e and the geodesics to e_i's endpoints yields a solution of length at most the perimeter of P plus twice $\overline{bv}$, a contradiction.

The second claim follows from the first claim. As the geodesics do not share any edges, there must be some opening angle $0 < \phi < \pi$ at b, and F^* is a funnel. The angle ϕ is determined by the two edges of the funnel that are incident to b. As the interior of F^* does not contain polygon edges, the base edge e is visible from b through ϕ. □

We use the algorithm of Guibas et al. [15] to find the shortest path from the point b to every vertex v of the polygon. For every two adjacent vertices v_i and v_{i+1} of the polygon, we compute the shortest paths connecting them to b. The algorithm of Guibas et al. [15] can find all these paths in $O(n)$ time. For each possible base edge and corresponding funnel, we add the length of the two paths and subtract the length of the edge between v_i and v_{i+1} to get the added length of this choice. We obtain the following result.

Theorem 1 *For a simple polygon P with n vertices and with a single point b inside, we can compute in $O(n)$ time the minimum-perimeter polygon $P^* \subseteq P$, that contains all vertices of P, and has b on its boundary.*

3.1.2 One blue and several red points inside P

In the general case we may also have red points inside P. Let R be the set of these red points, and assume that its size is $O(n)$. We need to adapt the algorithm given before to take the red points into account. We first triangulate the polygon P. Ignoring the red points R, we compute all funnels F from b to every edge e of the polygon. We get a partitioning of P into $O(n)$ funnels with disjoint interiors. In every funnel we do the following: If there are no red points inside F, we just store the length of the funnel without its base edge. Otherwise, we need to find a shortest

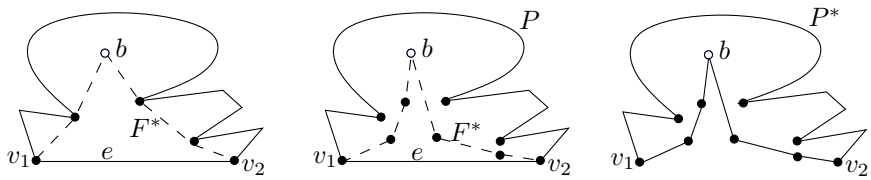

Figure 3: Left: the path F^* if $R = \emptyset$. Middle: the case $R \neq \emptyset$. Right: P^* for $R \neq \emptyset$.

path $\pi_{\min}$ from one endpoint of the base edge to b and back to the other endpoint of the base edge, such that all red points in R still lie inside the resulting polygon P^*.

The shortest path $\pi_{\min}$ inside some funnel F with respect to a set $R \cap F$ of red points consists of two chains which, together with the base edge e, again form a funnel F^*, see Figure 3 (middle). This funnel is not allowed to contain points of $R \cap F$ in its interior. We need to consider all possible ways of making such funnels, which involves partitioning the points of $R \cap F$ into two subsets. The red points of $R \cap F$ can only appear as reflex points on the funnel F^*, and therefore we can use an order on these points. For ease of description we let e be horizontal. Then F has a left chain and a right chain. The optimal funnel F^* also has a left chain and a right chain, such that all points of $R \cap F$ lie between the left chains of F and F^* and between the right chains of F^* and F. We extend the two edges of F incident to b, so that they end on the base edge e. This partitions F into three parts: a left part, a middle triangle, and a right part. In the same way as the first claim of Lemma 2, we can show that all points of $R \cap F$ in the left part must be between the left chains of F and F^*, and all points of $R \cap F$ in the right part must be between the right chains of F and F^*. The points of $R \cap F$ in the middle triangle are sorted by angle around b. Let the order be $r_1, \ldots r_h$, counterclockwise.

Lemma 3 *There is an index i such that the points $r_1, \ldots, r_i$ lie between the left chains of F and F^*, and $r_{i+1}, \ldots, r_h$ lie between the right chains of F^* and F.*

We iterate through the $h + 1$ possible partitions that can lead to an optimal funnel F^*, and maintain the two chains using a dynamic convex-hull algorithm [8]. Every next pair of chains requires a deletion of a point on one chain and an insertion of the same point on the other chain. We maintain the length of the path during these updates to find the optimal one.

As to the efficiency, finding all shortest paths from b to all vertices of P takes linear time for a polygon. Assigning the red points of R to the funnels takes $O(n \log n)$ time using either plane sweep or planar point location. Sorting the h red points inside F takes $O(h \log h)$ time, and the same amount of time is taken for the dynamic convex hull part. Since each red point of R appears in only one funnel, the overall running time is $O(n \log n)$.

Theorem 2 *For a simple polygon P with n vertices, a point b in P, and a set R of $O(n)$ red points inside P, we can compute in $O(n \log n)$ time the minimum-perimeter polygon $P^* \subseteq P$, that contains all vertices of P and all red points of R, and has b on its boundary.*

3.2 Several blue points inside P

When there are more blue points inside P, we use other algorithmic techniques. We first address the case of only blue points inside P and give a dynamic-programming algorithm. Then we assume that there are a constant number of blue and red points inside P, and give a fixed-parameter tractable algorithm.

3.2.1 Several blue and no red points inside P

Let P be a simple red polygon with n vertices, given in clockwise order. Let B be a set of m blue points inside P. In this section, we present a polynomial-time algorithm to compute a (possibly

degenerate) simple polygon, $P^* \subset P$, of minimum perimeter such that (1) no point of B is interior to P^*, (2) no point of P is exterior to P^*, and (3) $P^* \subseteq P$.

We begin by stating a structure lemma, which forms the basis for our algorithm (see Figure 4):

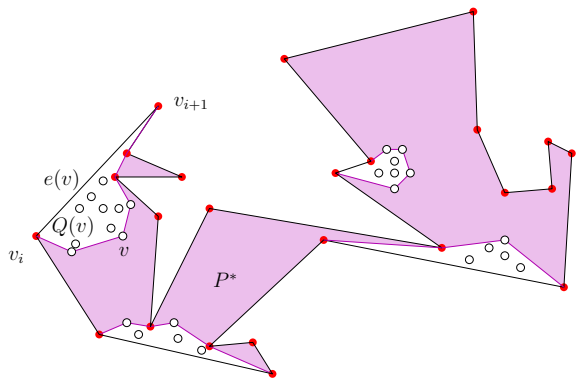

Figure 4: The (degenerate) simple polygon P^* is shaded and lies inside the black boundary of the input polygon P. The red points are vertices of P and of P^*, and the blue points (hollow circles) that were inside P are excluded from P^*, lying within pockets, each of which is formed by the relative convex hull of the blue points within it.

Lemma 4 *The polygon P^* is a (possibly degenerate) simple polygon whose vertices are either red points (of P) or blue points (of B). Each red point of P appears at least once (and at most twice) as a vertex of P^*. Each blue vertex, v, of P^* is necessarily a reflex vertex of P^* and is associated with an edge of P, $e(v) = a_v b_v$, which is the* lid *of the* pocket*, $Q(v)$, associated with v. $Q(v)$ is a simple polygon and is one of the connected components of the difference $P \setminus P^*$. The boundary of $Q(v)$ consists of the lid edge $e(v) = a_v b_v$, together with a polygonal chain of red and blue vertices; $Q(v)$ is the relative convex hull of the set $\{a_v, b_v\} \cup (B \cap Q(v))$.*

In general, P^* is a degenerate simple polygon, with "pinch points" at red vertices that arise along the boundary of a relative convex hull of blue points within a pocket. Such pinch points appear twice in a boundary traversal of the vertices of P^*.

Let $\pi_P(u, v)$ denote the geodesic path within P from $u \in P$ to $v \in P$. We note that, by the defining property of relative convex hulls, each blue vertex v that shows up on the boundary of a pocket $Q = Q(v)$ is *geodesically visible* to the endpoints, a_v and b_v, of the lid of the pocket Q: $\pi_P(v, a_v) \subset Q$ and $\pi_P(v, b_v) \subset Q$.

Since P^* is a simple polygon, it has a triangulation, $\mathcal{T}$, whose diagonals join pairs of vertices of P^*. (The existence of a triangulation holds even though P^* is potentially degenerate.) Our algorithm is a dynamic programming algorithm that searches for an optimal polygon P^*, together with a triangulation of it.

Our dynamic programming algorithm defines a value function, $f(\cdot)$, which assigns to each *subproblem* the cost of an optimal solution of the subproblem. In order to describe what goes into defining a subproblem, consider a diagonal, uv, of the triangulation $\mathcal{T}$ of P^*. We consider uv to partition P^* and $\mathcal{T}$ into the "done" and the "yet-to-be-done" portion of the triangulated optimal solution. The endpoints of uv are vertices of P^*, which may be red or blue. In the case that one or both are blue, we will also need to specify the identity of the lid of the pocket associated with the blue vertex; this permits us to partition the region P, within which the blue points are located that need to be excluded in the optimal enclosure P^*.

For a blue vertex u of P^*, we let $a_u b_u = e(u)$ be the lid of the pocket $Q(u)$, with a_u preceding b_u in clockwise order around P; for a red vertex u, we define $a_u = b_u = u$, and $Q(u) = u$, for convenience. Then, the "done" and the "yet-to-be-done" portions of P are separated by a polygonal path, $\pi_P(b_u, u), uv, \pi_P(v, a_v)$, which is the concatenation of the geodesic path $\pi_P(b_u, u) \subset Q(u)$,

the diagonal $uv \subset P^*$, and the geodesic path $\pi_P(v, a_v) \subset Q(v)$. The subproblem associated with the "state" (b_u, u, v, a_v) is to compute a minimum-length red-blue separator (polygonal path) that starts at u, ends at v, lies inside P, encloses all red vertices of P going clockwise from b_u to a_v, and excludes the set B_{b_u,u,v,a_v} of blue points within P that lie to the left of the polygonal path $\pi_P(b_u, u), uv, \pi_P(v, a_v)$. We let $f(b_u, u, v, a_v)$ be the length of such a shortest separator; i.e., $f(b_u, u, v, a_v)$ is the *value* (or *cost*) associated with the subproblem.

For any diagonal, uv, of P^* in the triangulation $\mathcal{T}$, there is a triangle, Δuvw, to the left of the oriented diagonal uv. We can view this triangle (i.e., the choice of vertex w) as the optimal "action" in the dynamic program. By optimizing over all choices of w, we get a recurrence relation (the Bellman equations) that must be satisfied by the value function f.

We distinguish several cases, depending on the color of the endpoints u and v:

(1) u and v are both blue. (Necessarily, u and v lie on different pockets, by the relative convexity of pockets, since a diagonal uv is, by definition, internal to P^*.) There are several subcases according to the choice of w:

(a) w is a blue vertex on a different pocket, with lid $e(w) = a_w b_w$; see Figure 5. Then, there are two new subproblems, (b_u, u, w, a_w) and (b_w, w, v, a_v). We define

$$f^{(1a)}(b_u, u, v, a_v) = \min_{w \in W^{(1a)}, a_w \in [b_u, a_v)} \left[f(b_u, u, w, a_w) + f(b_w, w, v, a_v) \right],$$

where $[b_u, a_v)$ indicates the half-closed interval of vertices of P in the clockwise listing from b_u to a_v, and $W^{(1a)}$ is the subset of B_{b_u,u,v,a_v} for which no blue point lies inside or interior to an edge of the triangle Δuvw. (We can tabulate in advance the set of all such triples of blue points.)

(b) w is a blue vertex on the same pocket as u; see Figure 6. Then, there is one new subproblem, $(b_w = b_u, w, v, a_v)$. We define

$$f^{(1b)}(b_u, u, v, a_v) = \min_{w \in W^{(1b)}} \left[|uw| + f(b_u, w, v, a_v) \right],$$

where $W^{(1b)}$ is the subset of B_{b_u,u,v,a_v} for which no blue point lies inside or interior to an edge of the triangle Δuvw, and uw lies inside P.

(c) w is a blue vertex on the same pocket as v. This case is analogous to case 1b and leads to a similarly defined function $f^{(1c)}$.

(d) w is a red vertex, with both uw and wv being (internal) diagonals of P^*; see Figure 7. Then, there are two new subproblems, (b_u, u, w, a_w) and (b_w, w, v, a_v). We define

$$f^{(1d)}(b_u, u, v, a_v) = \min_{w \in (b_u, a_v) \cap W^{(1d)}} \left[f(b_u, u, w, w) + f(w, w, v, a_v) \right],$$

where (b_u, a_v) indicates the open interval of vertices of P in the clockwise listing from b_u to a_v, and $W^{(1d)}$ is the set of red points for which no blue point lies inside or interior to an edge of the triangle Δuvw.

(e) w is a red vertex, with uw an edge of P^*, namely the first link in the geodesic path $\pi_P(u, b_u)$; see Figure 8. Let w' be the (red) successor of w in the geodesic path $\pi_P(u, b_u)$; possibly, $w' = b_u$. Then, there are two new subproblems, (b_u, w', w, w) and (w, w, v, a_v). We define

$$f^{(1e)}(b_u, u, v, a_v) = \min_{w \in [b_u, a_v] \cap W^{(1e)}} \left[|uw| + f(b_u, w', w, w) + f(w, w, v, a_v) \right],$$

where $[b_u, a_v]$ indicates the closed interval of vertices of P in the clockwise listing from b_u to a_v, and $W^{(1e)}$ is the set of red points for which no blue point lies inside or interior to an edge of the triangle Δuvw.

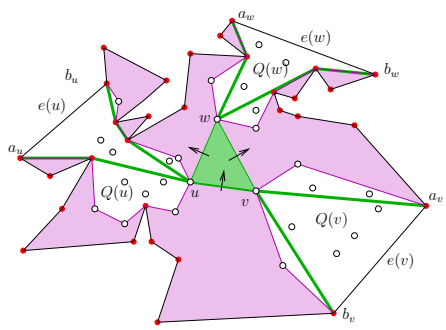

Figure 5: A step of the dynamic program: u and v are blue (hollow circles), and w is blue and lies on a different pocket.

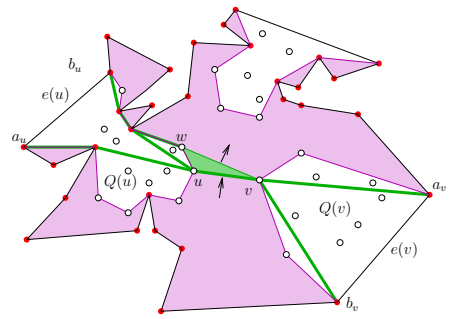

Figure 6: A step of the dynamic program: u and v are blue, and w is blue and lies on the same pocket as u.

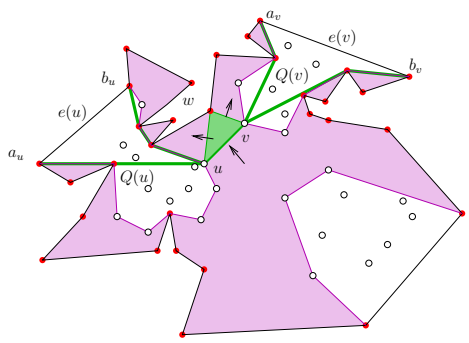

Figure 7: A step of the dynamic program: u and v are blue, and w is red, with uw and wv being (internal) diagonals of P^*.

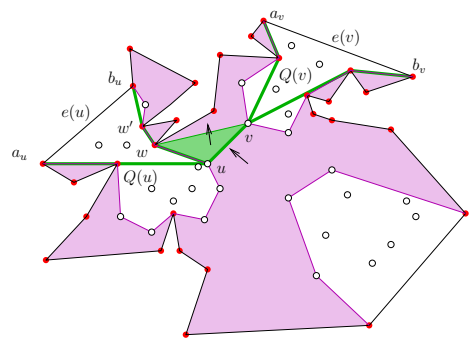

Figure 8: A step of the dynamic program: u and v are blue, and w is red, with uw an edge of P^* (the first link in the geodesic path $\pi_P(u, b_u)$).

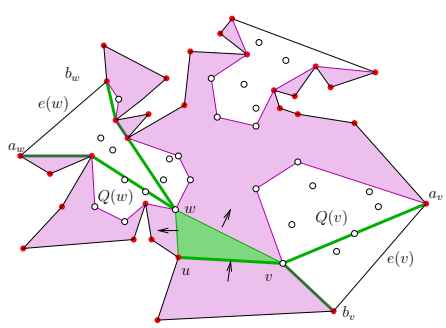

Figure 9: A step of the dynamic program: u is red, v is blue, and w is blue and lies on a different pocket.

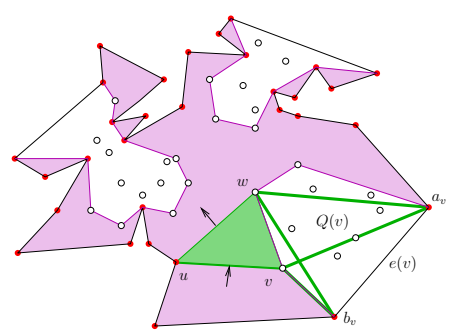

Figure 10: A step of the dynamic program: u is red, v is blue, and w is blue and lies on the same pocket as v.

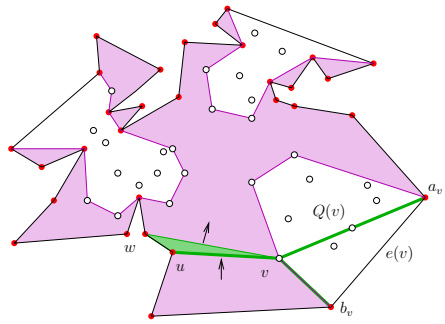

Figure 11: A step of the dynamic program: u is red, v is blue, and w is red. In this particular case, w is the clockwise successor of u around P, but it could be that uw is a diagonal of P^*, in which case a nonvacuous subproblem is specified by (u, u, w, w).

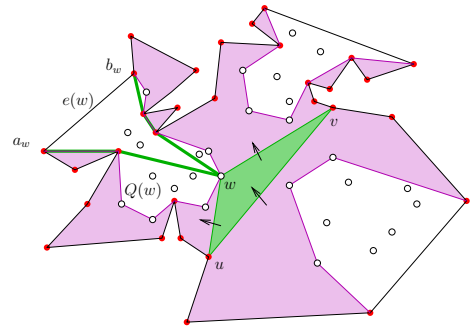

Figure 12: A step of the dynamic program: u and v are red, and w is blue.

(f) w is a red vertex, with wv an edge of P^*, namely the first link in the geodesic path $\pi_P(v, a_v)$. This case is analogous to case 1e and leads to a similarly defined function $f^{(1f)}$.

(2) u and v have different colors (say, u is red and v is blue). There are subcases depending on the choice of w:

(a) w is blue, lying on a different pocket as v; see Figure 9. As in case 1a, we obtain a value function $f^{(2a)}$ by optimizing over choices of w and lid $a_w b_w$, requiring triangle Δuvw to be blue-free (except at its vertices).

(b) w is blue, lying on the same pocket as v; see Figure 10. As in cases 1b and 1c, we obtain a value function $f^{(2b)}$ by optimizing over choices of w, requiring triangle Δuvw to be blue-free (except at its vertices).

(c) w is red, lying in the interval $(u, a_v]$; see Figure 11. As in cases 1b and 1c, we obtain a value function $f^{(2c)}$ by optimizing over choices of w, requiring triangle Δuvw to be blue-free (except at its vertices). If w is the clockwise successor of u around P (as it is in the figure), then the value of the corresponding vacuous subproblem, (u, u, w, w), is defined to be the Euclidean length, $|uw|$, of edge uw.

(3) u and v are both red vertices; see Figure 12, where one of the two possible subcases (that in which w is a blue vertex) is shown. The corresponding subcases, 3a and 3b, are handled analogously to above, resulting in value functions $f^{(3a)}$ and $f^{(3b)}$, corresponding to whether w is blue or red.

Then, the overall optimization to compute f optimizes over all of the above subcases, 1a-1f, 2a-2c, and 3a-3b.

The correctness of the dynamic programming algorithm follows by induction, using the usual Principle of Optimality from the theory of dynamic programming.

The running time of the algorithm is $O(m^3 n^3)$, since there are $O(n^2 m^2)$ choices of state (b_u, u, v, a_v) and there are at most $O(mn)$ choices of actions (when we optimize over choices of w and a_w).

We summarize our result in the following theorem:

Theorem 3 *For a polygon P with n vertices and m blue points inside it, we can compute a minimum-perimeter polygon P^* in time $O(m^3 n^3)$.*

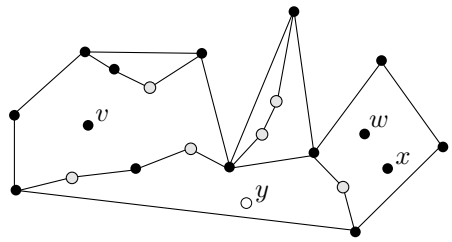

Figure 13: Structure of an optimal subpolygon when both blue (hollow) and red (solid) points are inside the original red polygon P.

Finally, we remark that, at a cost of increasing the complexity of the state space by a factor of K, and increasing the complexity of the action space also by a factor of K (in order to specify how to partition the "budget" K), we can extend our results to optimize over red-blue separators that allow up to K misclassified blue points, resulting in

Theorem 4 *For a polygon P with n vertices and $m \geq 1$ blue points inside it, we can compute a minimum-perimeter polygon P^*, allowing up to K misclassified blue points, in time $O((K+1)^2 m^3 n^3)$.*

3.2.2 Several blue and red points inside P

We next give an algorithm that can handle k red and blue points inside a red polygon P with n vertices. The algorithm is fixed-parameter tractable: it takes $O(C^{k \log k} \cdot n)$ time, where C is some constant. Hence, if k is constant, this solution is more efficient than by dynamic programming.

Let R and B be the sets of red and blue points inside P, respectively, and let $k = |R| + |B|$. The structure of the solution is determined by a partitioning of $R \cup B$ into groups; see Figure 13. One group contains points of $R \cup B$ that do not appear on the boundary of P^*. In Figure 13, this group contains v, w, x, and y. For the other groups, the points are in the same group if they lie on the same chain that forms a pocket, together with some lid. On this chain, points from B must be convex and points from R must be concave for the pocket. Besides points from $R \cup B$, the chain consists of geodesics between consecutive points from $R \cup B$. For all points in the group not used on chains, all points that come from B must be in pockets, and all points that come from R may not be in pockets (they must lie inside P^*).

For a fixed-parameter tractable algorithm, we try all options for $R \cup B$. To this end, we generate all permutations of $R \cup B$, and all splits of each permutation into groups. The first group can be seen as the points that do not contribute to any chain. For any other group, we get a sequence of points (ordered) that lie in this order on a chain. We compute the full chain by determining geodesics between consecutive points, and determine the best edge of P to replace by this chain (we need one more geodesic to connect to the first point and one to connect to the last point of the chain). The best edge of P is the one with the smallest length increase. We repeat this for every (ordered) group in the split permutation, resulting in a candidate polygon $P^{(*)}$. Then we test if $P^{(*)}$ is (degenerate) simple, if the blue points of the first group are outside $P^{(*)}$, and the red points of the first group are inside $P^{(*)}$. If so, $P^{(*)}$ is one of the real candidates from which we want to find one with minimum perimeter.

Theorem 5 *For a polygon P with n vertices, and k red and blue points inside it, we can compute a minimum-perimeter subpolygon P^* in $O(C^{k \log k} \cdot n)$ time, for some constant C.*

Proof: The number of permutations and splits of $R \cup B$ is $O(C_0^{k \log k})$. For each permutation and split we can compute P^* in $O(kn)$ time. Clearly, $O(C_0^{k \log k} \cdot kn) = O(C^{k \log k} \cdot n)$ for $C > C_0$. $\square$

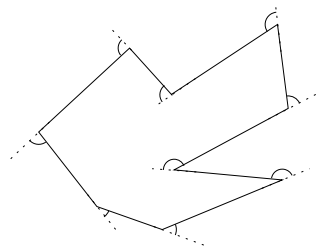

Figure 14: The absolute angular change of the simple polygon shown is the sum of the angles shown highlighted.

Remark: The algorithm can easily be adapted to deal with up to K misclassified red and blue points in R and B within the same time bound.

Remark: If P is convex and there are only blue points inside P, we can improve the running time to $O(C^k \cdot n)$ by considering crossing-free partitions of B only. We can prove that in an optimal solution, two chains cannot intersect, and hence we need only consider partitionings into groups whose convex hulls do not intersect. There are only $O(C^k)$ crossing-free partitions [25].

3.3 Minimizing absolute angular change

Getting all blue points out of P and at the same time minimizing the perimeter length of the red region may in some cases lead to a fjord-like boundary, which is not very desirable. One way to overcome this problem is, as already stated, to allow a number of up to K blue points to remain inside the red region and consider them as misclassified. Another way is to minimize not the perimeter length, but the total amount of "twists and turns" the boundary is allowed to take. In order to model this, we define the "absolute angular change" (or *total turn*, as it has been called in the study of bicriteria path optimization [5, 22]); see Figure 14.

Definition 2 *The* absolute angular change *of a polygon is the sum of the absolute value of the angle between the extension of an edge and its successor edge, when walking counterclockwise along the polygon.*

It is easy to see that the absolute angular change equals 2π in case of a convex polygon and is strictly larger than 2π in case of any other simple polygon.

Now the problem statement is as follows. We wish to determine the subpolygon $P^* \subseteq P$ that does not contain any blue points in the interior, nor any red points in the exterior and has smallest absolute angular change.

After an examination of the four possible cases (one or more blue points inside P with none or several red points inside P) we find that all the methods that were presented in Sections 3.1 and 3.2 can easily be adapted to find the solution with the smallest absolute angular change. The running times for all methods described above remain the same.

4 Recoloring methods

In the adaptation method, we change the boundary of the red region to bring blue points to the outside. However, if a blue point p is surrounded by red points, it may have been classified wrongly and recoloring it to red may lead to a more natural boundary of the red region. Similarly, red points surrounded by blue points may have been classified wrongly and we can recolor them to blue.

In this section we present methods for recoloring the given points, that is, assigning a new inside-outside classification. The starting point for our methods is as follows: We are given a set

P of n points, each of which is either red or blue. We first compute the Delaunay Triangulation DT(P) of P. In DT(P), we color edges red if they connect two red points, blue if they connect two blue points, and green otherwise. A red point is incident only to red and green edges, and a blue point is incident only to blue and green edges. To formalize that a point is surrounded by points of the other color, we define:

Definition 3 *Let the edges of DT(P) be colored as above. Then the* green angle ϕ *of* $p \in P$ *is*

- $360°$*, if* p *is only incident to green edges,*
- $0°$*, if* p *has at most one radially consecutive incident green edge,*
- *the maximum turning angle between two or more radially consecutive incident green edges otherwise.*

We recolor points only if their green angle ϕ is at least some threshold value Φ. Note that if Φ has any value less than $180°$, then it may be that a point is recolored red and blue indefinitely, with no termination. (A simple example consists of red points at $(-\epsilon, 1)$ and $(-\epsilon, -1)$, blue points at $(\epsilon, 1)$ and $(\epsilon, -1)$, for some small, positive ϵ, and a point at (0,0), which repeatedly changes color if $\Phi < 180°$.) So we assume in any case that $\Phi \geq 180°$; a suitable value for the application can be found empirically. After the algorithm has terminated, we define the regions as follows. Let M be the set of midpoints of the green edges. Then, each Delaunay triangle contains either no point or two points of M. In each triangle that contains two points of M, we connect the points by a straight line segment. These segments define the boundary between the red and the blue region. Note that each point of M on the convex hull of the point set is incident to one boundary segment while the other points of M are incident to exactly two boundary segments. Thus, the set of boundary segments consists of connected components that are either cycles, or chains that connect two points on the convex hull. An alternative is to choose the separating polygon based on relative convex hulls. We define the perimeter of the separation to be the total length of the boundary cycles and chains. The intuition behind this approach is that the perimeter decreases with most recolorings and that we end up with a compact shape that separates the red and blue points. To get an even more compact shape in our experiments we defined the boundary segments via a point at a constant small distance from the red endpoint of each green edge. For green edges that were shorter than the fixed distance, we used the midpoint.

Before we present different recoloring schemes, we make the following basic observation.

Observation 1 *If we can recolor a blue point, then we do not destroy this option if we first recolor other blue points. We also cannot create possibilities for recoloring a blue point if we have only recolored red points before.*

We can now describe our first recoloring method, the preferential scheme. We first recolor all blue points with green angle $\phi \geq \Phi$ red, and afterwards, all red points with green angle $\phi \geq \Phi$ blue. It can occur that points that are initially blue become red, and later blue again. However, with our observation we can see that no more points can be recolored. Hence, this scheme leads to at most a linear number of recolorings. As this scheme gives preference of one color over the other, it is not fair and therefore not satisfactory. It would for example be better to recolor by decreasing green angle, since we then recolor points first that are most likely to be misclassified. Note that a red preferential scheme and a blue preferential scheme exist. They give the recolorings with the maximum number of red and blue points, respectively.

Another scheme is a true adversary scheme, in which an adversary may recolor any point with green angle $\phi \geq 180°$. First, we show that the adversary scheme terminates after at most an exponential number of steps.

We observe that points on the boundary of T can be recolored at most once: such a point p is on the convex hull, and any edge sequence with angle greater than π must include both boundary edges incident to p. Thus, p can be recolored only to the color its two neighbors on the hull already have. Therefore, each point on the convex hull is recolored at most once.

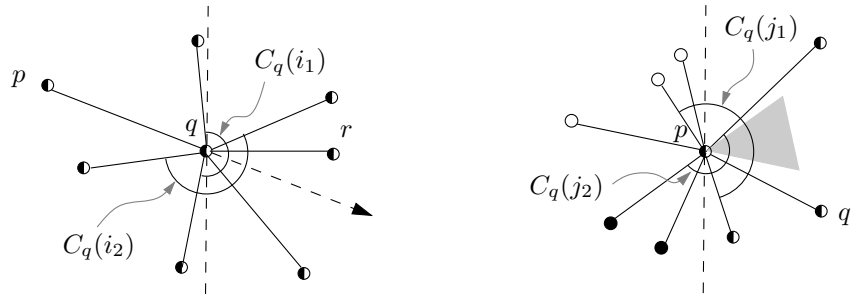

Figure 15: Illustrating Lemmas 5 and 6. Left: Either an edge or its opposite is in any coloring set. Right: If vertices to the left of p do not change color, but p does, then its color depends upon q. Points are shown solid (blue) or hollow (red) if they do not change color anymore; half-solid points indicate points whose color may still change.

Consider an edge $\overline{pq}$ with $p.x < q.x$. Unless q is the rightmost point on the convex hull, we can choose an *opposite edge* $\overline{qr}$ such that there is no edge between $\overline{qr}$ and the ray from q that goes in the direction opposite p, which is drawn dashed in Figure 15 (left). Fix a particular sequence of recoloring steps. If the color of point p is changed at step j, let $C_v(j)$ denote the *coloring set*, which is the maximal sequence of angularly consecutive green edges incident to q. Each coloring set for q must contain an edge or its opposite:

Lemma 5 *For any pair of an edge $\overline{pq}$ with its opposite $\overline{qr}$, if point q changes color, then q receives either the color of p or of r.*

Proof: By the definition of an opposite edge, any coloring set $C_q(j)$ that does not contain $\overline{pq}$ must include the opposite edge $\overline{qr}$ for its angle to be at least π. □

To show that any sequence of recoloring steps converges, we study the colors for a monotone chain: Starting with any edge $\overline{p_0p_1}$, choose a *chain* of vertices $p_0, \ldots, p_m$, ordered by x-coordinate, such that $\overline{p_ip_{i+1}}$ is opposite $\overline{p_{i-1}p_i}$, for all $0 < i < m$. This chain will end with p_m on the convex hull. Define the *color-change number* of this chain to be the number of integers $0 < i \leq m$ for which vertices p_{i-1} and p_i have different colors. Lemma 5 implies that recoloring of the interior vertices of this chain (i.e. vertices $p_1, \ldots, p_{m-1}$) never causes the color-change number to increase.

We can start considering the sequence after the convex hull vertices have changed color. By starting a little inside the hull, we can show that the color-change number for some monotone chain must decrease.

Lemma 6 *Fix an integer $j > 0$ and let p be the point with smallest x-coordinate of all the points that are recolored in steps $\geq j$; point p is recolored a finite number of times.*

Proof: If p is recolored once, then the lemma is trivially true, so assume that p is recolored at least twice. Note that p cannot be on the convex hull.

We claim that p always receives the color of a point q to its right. But this would imply that the color-change number of a chain beginning with edge $\overline{pq}$ decreases with each recoloring, so the maximum number of recolorings of p is the number of edges in such a monotone chain. Thus, it is sufficient to prove this claim.

Let j_1 and j_2, with $j \leq j_1 < j_2$, denote the next two recoloring steps for p. The coloring sets $C_p(j_1)$ and $C_p(j_2)$ contain edges to neighbors to the left of p that have different colors, as illustrated in Figure 15 (right), since points to the left of p cannot change color by assumption. Thus, there is a wedge left of p between two different colors that cannot be contained in any coloring set $C_p(k)$ for steps $k \geq j$. All such coloring sets for p, therefore, span the reflection of this wedge through point p, which is shown shaded in Figure 15 (right). Any edge $\overline{pq}$ that goes to the

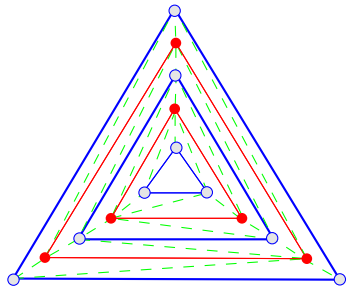

Figure 16: The construction for the quadratic lower bound of recolorings.

right inside this shaded wedge, or is the next edge clockwise or counterclockwise of this wedge, is contained in all coloring sets for p after step j, which means that p receives the color of q when recolored. □

For the upper bound, observe that there are 2^n possible color assignments to points. As there is no cycling in the adversary scheme, that means it is not possible to return to a color assignment that has already occurred before, so the adversary scheme is finite. Therefore, the upper bound on the number of recolorings is $2^n - 1$.

Second we show that there exists a point set such that the number of recolorings is at least $\Omega(n^2)$. We arrange the n points as $n/3$ equilateral triangles that have the same center and orientation, but different sizes, so they contain each other. Each triangle initially only has points of one color, and the triangles have alternating colors from smallest to largest; see Figure 16. We call these equilateral triangles *layers*, to avoid confusion with the Delaunay triangles for this point set. The layers are numbered 1 and up from the inside out.

The point set is degenerate, and several Delaunay triangulations are possible. The freedom is only how to triangulate between two consecutive layers. We do so asymmetrically, as shown in the figure. For any layer i, one of its points has edges to all three points of the enclosing layer $i+1$, and this point has green angle $> 180°$. We can choose the relative sizes of the layers such that points on layers more to the inside always have the point with the largest green angle. Furthermore, we can make sure that after recoloring a point of a layer, the other two points will be recolored next since they will then have the largest green angle.

We choose the first (innermost) layer to be blue. By construction its points will be recolored first to be red (first the point with degree 5, then the point with degree 4, and then the point with degree 3), resulting in a first and second layer that are red. Now a point of the second layer has the largest green angle (the point that has edges to all three blue points of the third layer), and consequently the points of the second layer will be colored blue. Then the first layer will be colored blue again. In the next phase the first three layers will be colored red. Repeating this construction gives the lower bound of $\Omega(n^2)$. We can summarize:

Theorem 6 *Given any triangulation T of a finite number of points, each colored in one of two colors, then any sequence of recolorings is finite and the number of recolorings is at most $2^n - 1$. There exists a triangulated colored point set such that the number of recolorings is $\Omega(n^2)$.*

Finding a polynomial upper bound on the number of recolorings is difficult. The lower bound example shows that the total green angle need not decrease during every recoloring, so we cannot use this argument to bound the number of recolorings. Similarly, the number of green edges need not decrease. There are examples of recolorings that increase the number of points with green angle $\geq 180°$. Recoloring a point with green angle $\geq 180°$ can also increase the separation perimeter.

To select a practical recoloring scheme for implementation, we choose one that makes a reasonable choice on which point to recolor next, namely the one with the largest green angle. Further-

more, we only want to recolor a point if this leads to a decrease in the perimeter of the separating polygon. We call this the angle-and-perimeter scheme. Even with this choice, the best known upper bound on the number of recolorings is still $2^n - 1$.

To implement the algorithm efficiently, we maintain the subset of points that can be recolored, sorted by decreasing green angle, in a balanced binary search tree. We extract the point p with largest green angle, recolor it, and recolor the incident edges. This can be done in time linear in the degree of p. We must also examine the neighbors of p. They may get a different green angle, which we must recompute. We must also test if recoloring a neighbor still decreases the perimeter length. This can be done for each neighbor in time linear in its degree. We summarize:

Theorem 7 *The running time for the angle-and-perimeter recoloring algorithm is $O(n + Z \cdot n \log n)$, where Z denotes the actual number of recolorings.*

4.1 The Angle-and-Degree Scheme

In the angle-and-degree scheme we use the same angle condition as before, complemented by requiring that the number of green edges decreases. For any red (blue) point p, we define $\delta(p)$ to be the difference between the number of green edges and the number of red (blue) edges incident to p. We recolor a point if its green angle ϕ is at least some threshold Φ and its δ-value is larger than some threshold $\delta_0 \geq 1$. We always choose the point with largest δ-value, and among these, the largest green angle. In every recoloring step the number of green edges in $DT(P)$ decreases by $\delta(p)$, so we get a linear number of recolorings.

Theorem 8 *The running time for the angle-and-degree recoloring algorithm is $O(n^2 \log n)$.*

Remark: In practice, vertices in Delaunay triangulations have constant degree. In this case we obtain running times of the two schemes of $O((n + Z) \log n)$ and $O(n \log n)$.

5 Experiments

In cooperation with our partners in the SPIRIT project [16] we received four data sets, namely Eastanglia (14, 57), Midlands (56, 52), Southeast (51, 49), and Wales (72, 54). The numbers in parentheses refer to the numbers of red and blue points in each data set, respectively. The red points were determined by Web searches using www.google.uk in September 2004 and trigger phrases such as "located in the Midlands" and then extracting the names of the corresponding towns and cities in the search results. The coordinates of the towns were looked up in the SPIRIT ontology. Finally the same ontology was queried with an axis-parallel rectangle 20% larger than the bounding box of the red points. The blue points were defined to be those points in the query result that were not red. The exact procedure is described in [2]. Notice that Wales is not an imprecise region, but our delineation methods can be used for evaluation.

We have implemented both the adaptation and the recoloring method using C++, and the libraries LEDA, CGAL and Qt. Although we made no attempt to minimize computation time, all of our tests took only a few seconds on an Intel-Xeon with a 2.80-GHz CPU and 2 GB memory under Linux-2.6. We show and discuss a few screen shots. Figure 17 features the adaptation method for two different values of α. The corresponding radius-α disk can be found in the lower right corner of each subfigure. Regarding the recoloring method we give an example of the angle scheme and of the angle-and-degree scheme; see Figure 18. In each figure blue points are marked by hollow circles and red points by black circles. Due to the α-shape that is used as initial region in the adaptation method, the area of the resulting region increases with increasing α; see Figure 17. We found that good values of α have to be determined by hand. For smaller values of α, the α-shape is likely to consist of several connected components, which leads to strange results; see Figure 17 (left). Here, the largest component captures Wales quite well, but the other components seem to make little sense. For larger values of α the results tend to change very little, because

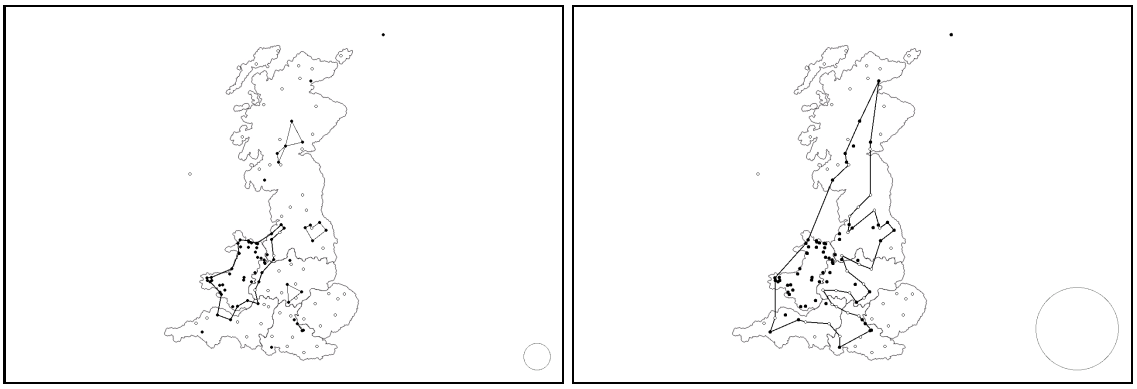

Figure 17: Regions for Wales computed by the adaptation method. The α-values are shown as radius-α disks in the lower right corner.

Figure 18: Region for Wales computed by the angle scheme with $\Phi = 185°$ (left) and for Eastanglia computed by the angle-and-degree scheme with $\Phi = 200°$ and $\delta_0 = 4$ (right).

then the alpha shape becomes similar to the convex hull of the red points. However, the value of α may not be too large since then outliers are joined in the α-shape and cause strange effects. For example, in Figure 17 (right) Wales has an enormous extent. When the initial value of α was well chosen, the results matched the region quite well for all our data sets, as far as this can be judged at all for imprecise regions.

For the angle scheme we found the best results for values of Φ that were slightly larger than 180°, say in the range 185°–210°. Larger values of Φ severely restrict color changes. This often results in a large perimeter of the red region, which is not very desirable. We compared the results of the angle scheme and of the preferential-blue and preferential-red scheme for $\Phi = 185°$. The preferential-red scheme recolors all eligible points from blue to red first and then all eligible points from red to blue. Only slight differences occurred on all four data sets.

The results strongly depend on the quality of the input data. Figure 18 shows this effect: Although Wales is contained in the resulting region, the region is too large. Too many points were classified falsely positive. The quality of the Eastanglia data was better, and the resulting region nearly matches the extent of Eastanglia (as generally considered).

For a small degree threshold, say $\delta_0 \leq 4$, the angle-and-degree scheme yielded nearly the same results as the angle scheme. This occurs because points having a green angle larger than 180° are likely to have a positive δ-value. For increasing values of δ_0 the results depend less and less on the angle threshold Φ. If a point has a δ-value above 4, then its green angle is usually large anyway. However, if the main goal is not the compactness of the region, or if the input is fairly reliable, larger values of δ_0 can also yield good results; see Figure 18 (right), where only the two light-shaded points were recolored.

Comparing the adaptation method and the angle scheme shows that the two schemes behave

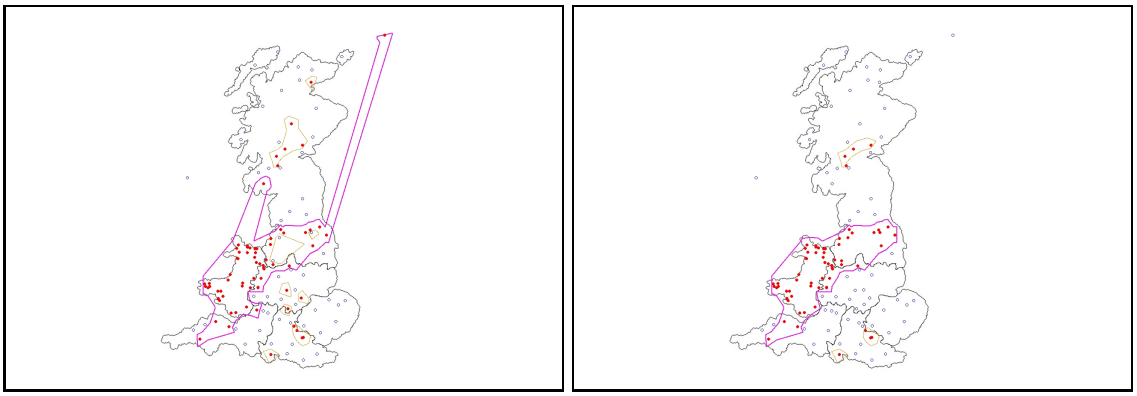

Figure 19: The red regions of the input data of Wales (left) and the angle scheme with preference 'largest green angle' for $\Phi = 215°$ (right).

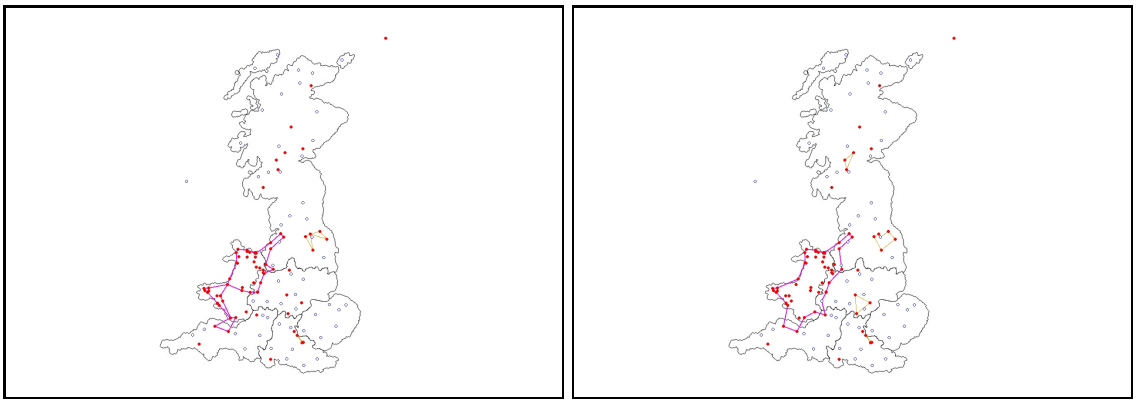

Figure 20: The red regions of Wales of the adaptation method for $\alpha = 0.0035$ (left), and $\alpha = 0.0068$ (right).

similarly on the inner part, the "core", of a point set. The main differences occur along the boundaries. The adaptation method may produce more fjord-like boundaries than the angle scheme. For examples, compare Figure 17 (right) and Figure 18 (left).

In the case of Wales, which is a region with known administrative boundaries, we can compare it to the extent of the polygons we derived using our algorithms. In Table 1 we contrast the following data (related to the area of Wales): Wales correctly identified, Wales not identified, and regions incorrectly identified as Wales. On first glance the recoloring method performs better, as it identifies almost all of Wales correctly. However, this comes at the high cost of identifying regions twice as big as Wales itself incorrectly as Wales. The adaptation method yields the best result for $\alpha = 0.0068$, where the correctly identified part of Wales lies at 80% and the incorrectly as Wales identified region is only half the size of Wales. For smaller values of α both correctly and incorrectly identified regions decrease, whereas for larger values of α mainly the regions incorrectly identified as Wales increase. See Figures 19 and 20. Note, however, that a considerable part of the boundary of Wales is coast, and we do not have blue points in the water by default. This influences the results.

We now investigate how the parameters α and Φ of the adaptation and recoloring approach, respectively, influence shape measures for polygons that are commonly used in geography [14, 21]. Such measures are compactness (A/r^2), circularity ratio (A/p^2), and form ratio (A/d^2), where A is the area of a given polygon P, r is the radius of the smallest enclosing circle of P, p is the perimeter of P, and d is the diameter of P. It turned out that the results of our experiments did not vary much between the different measures, so we only show how the compactness of the red

Table 1: Comparison of derived regions of Wales with administrative borders of Wales. Percentages are given with respect to the actual size of Wales.

	% of Wales identified	% of Wales not identified	% of incorrectly identified as Wales
Adaptation method			
$\alpha = 0.0035$	52.2	47.8	23.6
$\alpha = 0.0045$	62.3	37.7	33.1
$\alpha = 0.0068$	81.6	18.4	41.1
$\alpha = 0.008$	81.6	18.4	46.7
$\alpha = 0.0114$	83.5	16.5	103.1
Recoloring method			
$\Phi = 185$	97.1	2.9	243.7
$\Phi = 215$	96.3	3.7	190.1
$\Phi = 260$	97.2	2.8	240.8

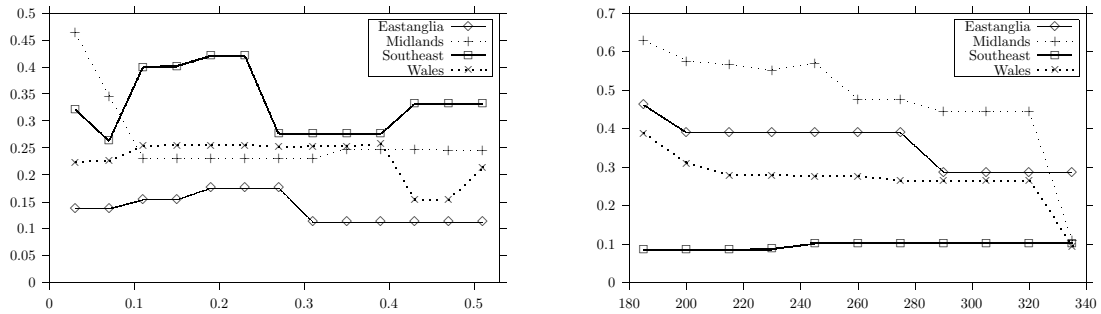

Figure 21: Compactness for various values of α (left) and Φ (right).

region in our four data sets depends on α and Φ; see Figure 21. In the left figure, the sudden jumps show where the number of points that are classified as outliers changes. Since the recoloring method can give rise to several red regions, we only counted the largest one in Figure 21 (right).

Finally, we investigate the performance of the angle scheme for the recoloring approach on random data. We are especially interested in the number of recolorings under the angle scheme, since we have not been able to show a polynomial upper bound on the number of recolorings. In order to imitate real instances we draw the point sets from the following distribution: from a given number n of points we draw $n/2$ blue points uniformly distributed from the unit square centered at the origin O, $n/8$ red points from a square of side length $7/8$ centered at O, and the remaining $3n/8$ red points from a circle of radius $1/4$ also centered at O. For an example with $n = 200$, see Figure 23 (left). The result of the angle scheme recoloring for $\Phi = 210°$ is depicted in Figure 23 (right).

Figure 22 (left) shows how the number of recolorings (y-axis) depends on the angle threshold Φ (x-axis) for random data sets of size $n = 800$. Figure 22 (right) shows how the number of recolorings depends on the number of points (x-axis) for a fixed angle threshold $\Phi = 210°$. Both figures show how often blue nodes were colored red, how often red nodes were colored blue, and the sum of these two numbers. All numbers were averaged over 30 random point sets of a given size or for a given threshold. The error bars show the minimum and the maximum total number of recolorings that occurred among those 30 point sets. It is interesting to see that the minimum, maximum, and average number of recolorings do not vary much, and that the number of recolorings

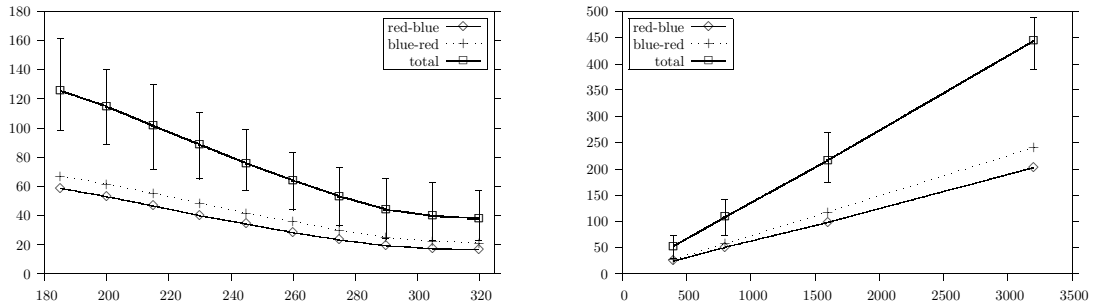

Figure 22: Number of recolorings as a function of Φ (left) and as a function of the number of points (right) resp. for random data.

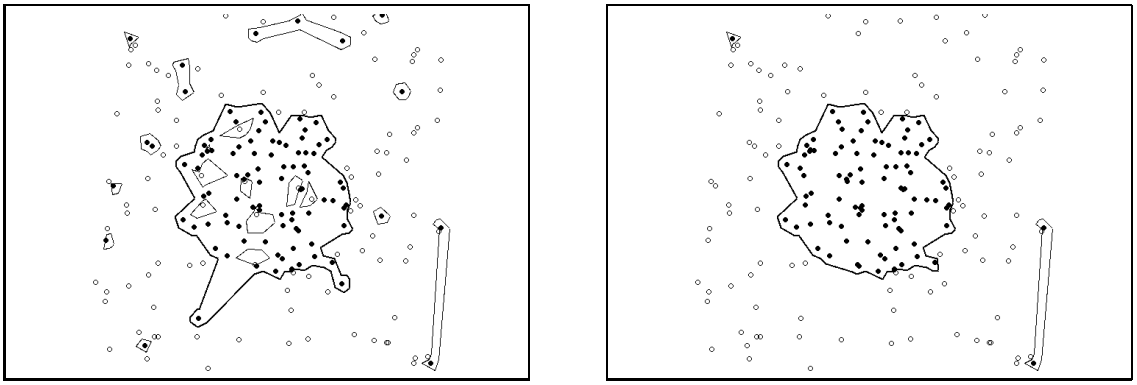

Figure 23: The red regions of a random point set before (left) and after (right) applying the angle scheme for $\Phi = 210°$.

seems to scale perfectly with the number of points.

The strength of the recoloring method is its ability to eliminate false positives provided that they are not too close to the target region. Since the differences between the various schemes we investigated seem to be small, a scheme that is easy to implement and terminates quickly can be chosen, e.g. the preferential-red or preferential-blue scheme.

6 Conclusions

This paper discussed the problem of computing a reasonable boundary for an imprecise geographic region based on noisy data. Using the Web as a large database, it is possible to find cities and towns that are likely to be inside and cities and towns that are likely to be outside the imprecise region. We presented two basic approaches to determine a boundary. The first was to formulate the problem as a minimum perimeter polygon computation, based on an initial red polygon and additional points that must be outside (blue) or stay inside (red). For the case of one blue point inside the red polygon we presented a linear time algorithm, and an $O(n \log n)$ time algorithm if there are also red points that must stay inside. If there are m blue points and no red points inside, we gave an $O(m^3 n^3)$ time algorithm, and a variation that allows misclassified points. For the case of k red and blue points inside, we presented a fixed-parameter tractable algorithm running in $O(C^{k \log k} \cdot n)$ time. This algorithm can also be adapted to deal with misclassified points.

The second approach involved changing the color, or inside-outside classification of points if they are surrounded by points of the other color. We proved a few lower and upper bounds on the number of recolorings for different criteria of recoloring. An interesting open problem is whether the most general version of this recoloring method has a polynomial upper bound on the number

of recolorings. In tests we always had less than n recolorings.

We also presented test results of our algorithms, based on real-world data and random data. The real-world data included places obtained from trigger phrases for several British regions. Indeed the data appeared to be noisy, which is one reason why the boundaries determined were not always acceptable. Using post-processing, it is possible to improve the shape of the boundaries in various ways. Also, with the expanding of the Web, more reliable data may present itself automatically, when smaller towns also make their appearance on the Web and can be found using trigger phrases.

A natural extension is to assign weights or confidence values with red and blue points. Cities that appear many times in a trigger phrase are surely inside, and cities mentioned often on the Web, but never in a trigger phrase, are almost surely outside. For small towns not mentioned, the situation is less clear.

Acknowledgements: Iris Reinbacher was partially supported by a travel grant of the Netherlands Organization for Scientific Research (NWO). J. Mitchell is partially supported by the National Science Foundation (CCR-0098172, ACI-0328930, CCF-0431030), NASA (NAG2-1620), Metron Aviation, and the US-Israel Binational Science Foundation (2000160).

We thank Subodh Vaid, Hui Ma, and Markus Völker for implementing the algorithms, and Hideo Joho and Paul Clough for providing the data for the experiments.

References

[1] P. K. Agarwal and S. Suri. Surface approximation and geometric partitions. *SIAM J. Comput.*, 27:1016–1035, 1998.

[2] 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. Technical Report UU–CS–2005–036, Utrecht University, 2005.

[3] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some separability problems in the plane. In *Abstracts 16th European Workshop Comput. Geom.*, pages 51–54. Ben-Gurion University of the Negev, 2000.

[4] E. M. Arkin, S. Khuller, and J. S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10:399–427, 1993.

[5] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 153–156, 1991.

[6] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Minimum-link watchman tours. *Inform. Process. Lett.*, 86(4):203–207, May 2003.

[7] S. Arora and K. Chang. Approximation schemes for degree-restricted MST and red-blue separation problem. *Algorithmica*, 40(3):189–210, 2004.

[8] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, pages 617–626, 2002.

[9] T. M. Chan. Low-dimensional linear programming with violations. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 570–579, 2002.

[10] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[11] P. Eades and D. Rappaport. The complexity of computing minimum separating polygons. *Pattern Recogn. Lett.*, 14:715–718, 1993.

[12] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[13] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Inform. Theory*, IT-29:551–559, 1983.

[14] G. D. Garson and R. S. Biggs. *Analytic Mapping and Geographic Databases*. Sage Publications, Newbury Park, 1992.

[15] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[16] C. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies – an overview of the SPIRIT project. In *Proc. 25th Annu. Int. Conf. on Research and Development in Information Retrieval (SIGIR 2002)*, pages 387–388, 2002.

[17] A. Markowetz, T. Brinkhoff, and B. Seeger. Exploiting the internet as a geospatial database. In *Workshop on Next Generation Geospatial Information*, 2003.

[18] J. S. B. Mitchell. Approximation algorithms for geometric separation problems. Technical report, Department of Applied Mathematics, SUNY Stony Brook, NY, July 1993.

[19] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28:1298–1309, 1999.

[20] Y. Morimoto, M. Aono, M. Houle, and K. McCurley. Extracting spatial knowledge from the Web. In *Proc. IEEE Sympos. on Applications and the Internet (SAINT'03)*, pages 326–333, 2003.

[21] D. O'Sullivan and D. J. Unwin. *Geographic Information Analysis*. John Wiley & Sons Ltd, 2003.

[22] C. D. Piatko. *Geometric Bicriteria Optimal Path Problems*. Ph.D. thesis, Cornell University, 1993.

[23] R. S. Purves, P. Clough, and H. Joho. Identifying imprecise regions for geographic information retrieval using the web. In *Proceedings of GISRUK 2005, Glasgow*, pages 313–318, 2005.

[24] C. Seara. *On Geometric Separability*. PhD thesis, Universitat Polytècnica de Catalunya, Barcelona, 2002.

[25] M. Sharir and E. Welzl. On the number of crossing-free matchings, cycles, and partitions. Manuscript, 2005.

[26] G. Toussaint. Pattern recognition pages, `http://cgm.cs.mcgill.ca/~godfried/teaching/pr-web.html`.