Polygonal Separators and Digital Shape Recognition

extended abstract

Binay Bhattacharya*
Thomas Shermer*

School of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6

May 14, 1990

Abstract

We present an $O(n \log n)$ algorithm for finding angle-constrained convex separating polygons for two sets of points. In particular, this algorithm allows us to find separating rectangles and equilateral triangles. We also show how our method can be supplemented to find separating squares in $O(n \log^2 n)$ time. These problems find application in pattern analysis, in the problems of recognizing digital polygons. The previous algorithms for digital polygon recognition employ a brute-force approach and have high-order polynomial (at least n^5) running times.

1 Introduction

A problem that has recieved considerable attention in the pattern recognition literature recently is the problem of recognizing digital shapes. Given a plane figure, the digital image of that figure is the set of all points on the unit lattice that are inside the figure (in the sense of the Jordan Curve Theorem). A digital image of a particular type of shape (e.g., a rectangle) is called a *digital shape* (e.g., a digital rectangle). A digital rectangle, and an associated plane rectangle, are shown in figure 1.

In a digital shape recognition problem, one is given a set of (connected) lattice points, and then must determine a plane figure such that the digital image of that figure is the given set of lattice points. Algorithms have been developed for recognizing digital circles [K84] [NA84], digital triangles and rectangles [NA85], digital squares [NA90], and digital convex polygons [K82a].

The computational geometry problem of separation corresponds to the digital shape recognition problem. We define the separation problem for a given class of objects $\mathcal{C}$:

> **$\mathcal{C}$-Separation**
> **Given:** Two sets of points, R ("red points") and B ("blue points").
> **Find:** An element S ("the separator") of $\mathcal{C}$ such that all red points are contained in S, and no blue points are contained in S.

Typical object classes $\mathcal{C}$ are the class of all halfplanes [DR80], all disks [B88], or all triangles.

*Work supported by the Natural Sciences and Engineering Research Council of Canada

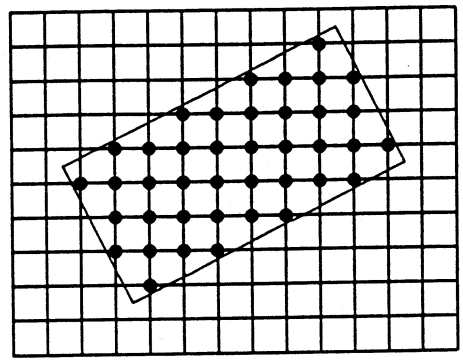

Figure 1: A digital rectangle and a corresponding plane rectangle

A digital shape recognition problem can easily be converted to a separation problem by letting R be the given set of lattice points, and B be all other lattice points. Depending on the shape being recognized, B can be restricted to a certain neighborhood of R (for instance, B may be the lattice points which are not in S but adjacent to members of S). For recognizing digital polygons, the measure of the smallest angle of the polygon to be recognized determines the size of the neighborhood needed. Also, often only the boundary points of R are given.

Let $\mathcal{A}$ be a sequence of angles $(a_1, a_2, \ldots, a_k)$ such that $\mathcal{A}$ is a valid sequence of interior angles for a k-vertex convex polygon. Let $\mathcal{C}_\mathcal{A}$ be the class of all polygons P such that P has angle sequence $\mathcal{A}$. Degenerate polygons (ones with zero-length edges) are allowed. The particular separator problems that we consider in this paper are those that can be expressed as a $\mathcal{C}_\mathcal{A}$-separation problem, for some sequence $\mathcal{A}$. We call these problems polygonal separation problems. For instance, the rectangle-separation problem is the $\mathcal{C}_\mathcal{A}$-separation problem for $\mathcal{A} = (90°, 90°, 90°, 90°)$. We also extend our method to solve the square-separation problem.

2 Angle-constrained polygon algorithm

Our approach to polygonal separation problems is to consider only separators such that each edge of the separator is in contact with the convex hull of the red points. We show that if any separator exists, then a separator so constrained exists. Each such constrained separator can be uniquely identified by the angle that the first edge of the separator makes with the x-axis. We note that if any blue point is in the red convex hull, then no separator exists.

We then consider all potential (constrained) separators by considering all angles in the range $[0°, 360°)$. Such a potential separator is a true separator if no blue points lie inside it. Therefore, for each blue point, we simply mark those angles for which the separator would contain the point. In this manner, each blue point can invalidate up to k intervals of angle. The union of these intervals, for all blue points, gives us the set of invalid separator angles. The complement of this union is the set of valid separator angles.

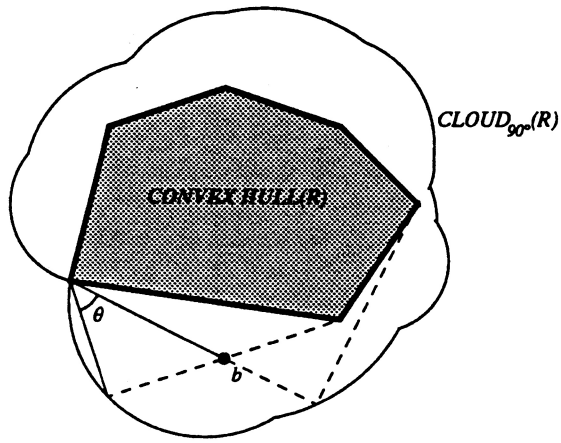

Figure 2: Computing invalid angles

Each blue point can invalidate one subinterval of $[0°, 360°)$ for each angle in $\mathcal{A}$. For some angle a_i, and some blue point b, we compute the invalid subinterval for a_i and b as follows. First, let $CLOUD_{a_i}(R)$ be the trajectory of the vertex of angle a_i in the potential separators as the reference angle is swept through the interval $[0°, 360°)$ (see figure 2). $CLOUD_{a_i}(R)$ is a circular-arc polygon, and can be computed in linear time [T88]. If b is outside of $CLOUD_{a_i}(R)$, then b does not invalidate any angle for a_i. If b is inside (or on the boundary) of $CLOUD_{a_i}(R)$, then b invalidates the interval of angle determined by its two tangents, with one reflected off of $CLOUD_{a_i}(R)$. In figure 2, b invalidates the section of angle shown as θ.

The entire algorithm runs in $O(kn \log kn)$ time, and returns a list of valid intervals for separators.

3 Square algorithm

We need consider only those squares that have two opposite sides supporting the red convex hull. We begin by using the angle-constrained polygon separator algorithm to find all valid angles for rectangle separators. We consider these valid angles, in order.

We can determine if a valid angle α admits a square separator in the following manner. For any angle β, let $d(\beta)$ be the distance between the two support lines (in direction β) of the red convex hull. First compute $d(\alpha)$ by finding the supporting lines of the red convex hull in direction α. Next, let B_1 and B_2 be those subsets of B which are between these supporting lines, and on opposite sides of R (see figure 3). Compute $d(\alpha + 90°)$. Let $e(\alpha + 90°)$ be the distance between the closest support lines, in direction $\alpha + 90°$, of the convex hulls of B_1 and B_2. The ratios $d(\alpha + 90°)/d(\alpha)$ and $e(\alpha + 90°)/d(\alpha)$ are the extremes of the set of aspect ratios possible for constrained rectangle separators in direction α; if the number 1 lies between these two ratios, then angle α admits a square separator.

We extend this method in a straighforward manner to check for the existence of separating squares in any interval of angle where B_1, B_2, and the vertices of support for the six support lines remain the same; there are at most $O(n)$ such intervals. Each such interval can be checked in

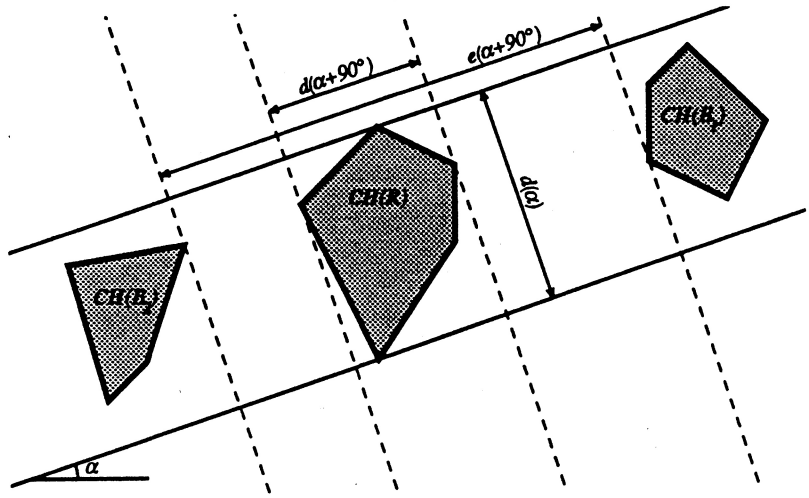

Figure 3: Checking for a square separator

$O(\log n)$ time, given the convex hulls of R, B_1, and B_2.

The remaining details of the algorithm are therefore determining when intervals end ($O(n \log n)$ time, using a heap), and maintaining the convex hulls of B_1 and B_2 ($O(n \log^2 n)$ time, using the algorithm of [OV81]). Thus, our square-separator algorithm runs in $O(n \log^2 n)$ time.

References

[B88] B. Bhattacharya, Circular Separability of Planar Point Sets, in *Computational Morphology*, G. Toussaint, ed., North-Holland, 1988, 25-39.

[DR80] D. Dobkin and S. Reiss, The Complexity of Linear Programming, *Theoretical Computer Science* **11**, 1980, 1-18.

[K82a] C. Kim, Digital Convexity, Straightness, and Convex Polygons, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **4**, 1982, 618-626.

[K84] C. Kim, Digital Disks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 1984, 372-374.

[NA84] A. Nakamura and K. Aizawa, Digital Circles, *Computer Vision, Graphics, and Image Processing* **26**, 1984, 242-255.

[NA85] A. Nakamura and K. Aizawa, Digital Images of Geometric Pictures, *Computer Vision, Graphics, and Image Processing* **30**, 1985, 107-120.

[NA90] A. Nakamura and K. Aizawa, Digital Squares, *Computer Vision, Graphics, and Image Processing* **49**, 1990, 357-368.

[OV81] M. Overmars and J. van Leeuwen, Maintenance of Configurations in the Plane, *Journal of Computer and Systems Science* **23**, 1981, 166-204.

[T88] M. Teichmann, Shoving a Table into a Corner, in *Snapshots of Computational and Discrete Geometry*, G. Toussaint, ed., McGill University TR SOCS-88.11, 1988, 99-118.