Finding Monochromatic L-Shapes in Bichromatic Point Sets

Farnaz Sheikhi* Mark de Berg† Ali Mohades* Mansoor Davoodi*

Abstract

Given a set R of red points and a set B of blue points in the plane of total size n, we study the problem of determining all angles for which there exists an L-shape containing all points from B without containing any points from R. We propose an algorithm to solve the problem in $O(n^2 \log n)$ time and $O(n)$ storage. We also describe an output-sensitive algorithm that reports all angles in $O(n^{5/3+\varepsilon} + k \log k)$ time and $O(n^{5/3+\varepsilon})$ storage, where k is the number of reported angular intervals.

1 Introduction

In a separability problem one is given two colored point sets R and B—the *red* and the *blue* point set, respectively—of total size n, and a geometric shape called a *separator*. The goal is to decide whether one can place the separator such that it separates sets R and B completely. If such a placement is possible, one often also wants to compute all such placements or the placement minimizing some cost function. Geometric separability arises in applications where classification is required, such as machine learning and image processing. There has been a fair amount of work on different kinds of separators, both in the plane and in higher dimensions. For separability in the plane, which is the topic of our paper, the following results are known. The problem of deciding whether the two point sets can be separated by a single line was solved by Megiddo [9] in linear time. O'Rourke *et al.* [11] presented a linear-time algorithm for deciding whether the two point sets can be separated by a circle. The problem of finding a convex polygon with minimum number of edges separating the two point sets, if it exists, was solved by Edelsbrunner and Preparata [5]. Fekete [6] showed that the problem of determining a simple polygon with minimum number of edges separating the two point sets is NP-complete, and a polynomial-time approximation algorithm was provided by Mitchell [10]. Separability problems have also been studied for separators in the form of strips and wedges [7]. A thorough study is presented by Seara [12].

Motivated by geometric model reconstruction from LIDAR data, Van Kreveld *et al.* [8] recently studied the separability problem in the plane for the case where the separator is a (not necessarily axis-aligned) rectangle. They proposed an $O(n \log n)$ time algorithm to compute all angles for which a rectangular separator exists. They mentioned the case of a non-convex separator, namely an L-shape, as an open problem; this is the topic of our paper. Next we define the problem more precisely and we state our results.

We define an axis-aligned L-shape to be the set-theoretic difference $r_1 \setminus r_2$ of two axis-aligned rectangles r_1 and r_2 such that $r_2 \subset r_1$ and the top-right corners of r_1 and r_2 coincide. An *L-shape with orientation* θ is then defined as an axis-aligned L-shape that has been rotated in counterclockwise direction over an angle of θ. Thus, given the point sets B and R, we wish to find all angles $\theta \in [0, 2\pi)$ for which there exists an L-shape L with orientation θ such that $B \subset L$ and $R \cap L = \emptyset$. From now on, we call such an L-shape a *blue L-shape*. The orientations for which a blue L-shape exists form a collection of subintervals of $[0, 2\pi)$. We present two algorithms for computing this collection of intervals.

The first algorithm is a simple algorithm running in $O(n^2 \log n)$ time and using only $O(n)$ storage. The second algorithm is a more complicated output-sensitive algorithm which uses $O(n^{5/3+\varepsilon} + k \log k)$ time and $O(n^{5/3+\varepsilon})$ storage, where k is the number of reported intervals and $\varepsilon > 0$ is any fixed constant.

2 The algorithms

Terminology and notation. We start by defining some terminology and notation.

Our global strategy will be to do a rotational sweep: we increase θ from 0 to 2π and we report the angular intervals for which there is a blue L-shape while we sweep. It will be convenient to think about the sweep as rotating the coordinate frame. Thus, we define the x_θ-axis and the y_θ-axis as the coordinate axes after the coordinate frame has been rotated over an angle θ in counterclockwise direction. We denote the coordinates of a point p in the rotated coordinate frame by $x_\theta(p)$ and $y_\theta(p)$. Whenever we talk about the top-right corner of a rectangle, we mean the top-right corner with respect to the current coordinate frame.

*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology, {f.sheikhi,Mohades,mdmonfared}@aut.ac.ir

†Department of Computer Science, TU Eindhoven, PO Box 513, 5600 MB, Eindhoven, the Netherlands. mdberg@win.tue.nl

For an angle θ, we denote the minimum bounding rectangle of the blue point set B with orientation θ by $M_b(\theta)$; we call $M_b(\theta)$ the *blue rectangle*. Let $R_\theta := R \cap M_b(\theta)$, and define $M_r(\theta)$ to be the smallest rectangle with orientation θ that contains R_θ and shares its top-right corner with $M_b(\theta)$; we call $M_r(\theta)$ the *red rectangle*. Note that $L_\theta := M_b(\theta) \setminus M_r(\theta)$ is an L-shape. Moreover, $B \cup R$ admits a blue L-shape with orientation θ if and only if L_θ is a blue L-shape. The L-shape L_θ excludes all points from R by definition. Hence, L_θ is blue if and only if it contains all points from B.

To determine whether L_θ contains all points from B, we will define a so-called *blue step-shape*. We say that a point q *dominates* another point p (at orientation θ) if $x_\theta(q) > x_\theta(p)$ and $y_\theta(q) > y_\theta(p)$, and we say that a point $p \in B$ is *maximal* (at orientation θ) if there is no point $q \in B$ that dominates p. Connecting the maximal points in B, we can get a *blue staircase*, and we define the blue step-shape as the region left of the blue staircase and bounded by the boundary of $M_b(\theta)$.

Using the blue step-shape, we can characterize when L_θ contains all blue points. To this end, we define the *red witness* as the lower-left corner of $M_r(\theta)$; note that the red witness is the reflex corner of L_θ. See Fig. 1.

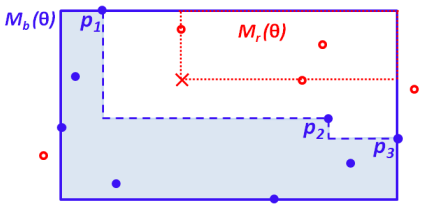

Figure 1: Points p_1, p_2 and p_3 are maximal points. The blue staircase is shown dashed and the blue region shows the blue step-shape. The red witness is shown by a cross.

Observation 1 *The L-shape L_θ is blue if and only if the red witness lies outside the blue step-shape.*

The global strategy. As remarked earlier, our global strategy is to perform a rotational sweep. While we sweep, we will maintain the following information:

- The blue rectangle, $M_b(\theta)$. More precisely, we maintain the set of blue points on the boundary of $M_b(\theta)$. (Except at events, there are at most four such points.)
- The blue step-shape or, more precisely, the blue staircase bounding the blue step-shape from the right. (The other part of the boundary of the blue step-shape is formed by parts of the boundary of $M_b(\theta)$.)
- The red witness or, more precisely, the two red points defining the red witness.

While we do the sweep, we are interested in the angles where the red witness crosses the blue staircase; these angles define the angular intervals we have to report. Next we discuss the various events that arise during the sweep.

Blue-rect event: this event occurs when the set of points defining $M_b(\theta)$ changes. This happens when the blue point with the maximum or minimum x_θ-coordinate (or the maximum or minimum y_θ-coordinate) changes.

Blue-sc event: this event occurs when the blue staircase changes, that is, when the set of maximal blue points changes.

Red-set event: this event occurs when R_θ, the set of red points inside $M_b(\theta)$, changes.

Witness event: this event occurs when the points defining the red witness change. This happens when the red point in R_θ with the minimum x_θ-coordinate changes, and when the red point in R_θ with the minimum y_θ-coordinate changes. It can be triggered by a red-set event, or by two red points that were already in R_θ swapping order along the x_θ-axis or y_θ-axis.

Crossing event: this event occurs when the red witness crosses the blue staircase.

Note that all these events take place at angles defined by pairs of points (more precisely, a pair p, q defines two angles, one given by the line $\ell(p, q)$ through p and q, the other given by a line orthogonal to $\ell(p, q)$): blue-rect events and blue-sc events take place at angles defined by two blue points, red-set events take place at angles defined by a red and a blue point, witness events occur at angles defined by a red and a blue or two red points, and crossing events occur at angles defined by a red and a blue point.

A simple algorithm. Our first algorithm considers all angles defined by a pair of points in $B \cup R$ (even though not all such angles actually define an event). The simplest way of doing this is as follows.

1. Set $\theta := 0$, and initialize $M_b(\theta)$, and the blue staircase, and $M_r(\theta)$. The most time consuming part of the initialization is the computation of the maximal blue points (which define the blue staircase), which takes $O(n \log n)$ time.
2. Compute all $\Theta(n^2)$ angles defined by a pair of points in $B \cup R$, and sort these angles.
3. Go through the angles in order. At each angle, check whether the points defining the angle induce a (blue-rect, blue-sc, red-set, witness, or crossing)

event. With the information available, this can be done in $O(1)$ time. If a blue-rect (or blue-sc, or witness) event occurs then update $M_b(\theta)$ (or the blue staircase, or the red witness). If a crossing event occurs, then check whether the red witness enters the blue step-shape or leaves the blue step-shape. In the first case, L_θ stops being blue, and an angular interval ending at the event angle must be reported. In the second case, a new angular interval must be started; this interval will be reported later, when its endpoint is found.

The algorithm above clearly runs in $O(n^2 \log n)$ time. However, it uses $O(n^2)$ space to store all the angles defined by pairs of points in $B \cup R$. We can reduce the storage as follows. We dualize [4] the points in $B \cup R$, obtaining a set $(B \cup R)^*$ of n lines. Generating the angles defined by pairs of points in $B \cup R$ in order now translates to generating the x-coordinates of the intersection points of the lines in $(B \cup R)^*$ in order. This can be done using only $O(n)$ storage by using a standard[1] line-segment intersection algorithm [4].

Theorem 1 *Let B be a set of blue points and let R be a set of red points in the plane, with $n := |B| + |R|$. Then, we can compute all angles for which there is a blue L-shape in $O(n^2 \log n)$ time and using $O(n)$ space.*

An output-sensitive algorithm. We now describe our main result, which is an output-sensitive algorithm to compute all angular intervals for which there is a blue L-shape. The main idea is to use the fact that the number of blue-rect, blue-sc, red-set, and witness events is small, and that the number of crossing events is proportional to the output size. Thus, we will pre-compute the blue-rect, blue-sc, red-set, and witness events, and then compute all crossing events by using some data-structuring techniques. Next we explain how to do this. We start by showing how to pre-compute the blue-rect, blue-sc, red-set, and witness events.

Blue-rect events. The blue-rect events are easy: after computing the convex hull of the blue points, we can find the blue-rect events by using rotating calipers [14]. Thus, there are $O(n)$ events in total, and computing them takes $O(n \log n)$ time and $O(n)$ storage.

Blue-sc events. Recall that these events are events where the set of maximal blue points changes, as we rotate the coordinate frame. Bae *et al.* [3] have proved that the set of maximal points changes at most $O(n)$ times during a full rotation from $\theta = 0$ to $\theta = 2\pi$; they also presented an algorithm to compute all these changes in $O(n^2)$ time, using $O(n)$ space. However, we can also use an algorithm by Avis *et al.* [2], who essentially studied the same problem; their algorithm runs in $O(n \log n)$ time and $O(n)$ storage. In summary, there are $O(n)$ blue-sc events, and they can be computed in $O(n \log n)$ time and $O(n)$ storage.

Red-set events. These are the events where a red point q enters or exists the blue rectangle $M_b(\theta)$. These angles are determined by the tangent lines from q to the convex hull of the blue points (denoted by CH_B) [8]. After computing CH_B, we can find the tangent lines from q in $O(\log n)$ time. (Note that if q lies inside CH_B, it will always be in R_θ.) Hence, there are $O(n)$ red-set events, and they can be computed in $O(n \log n)$ time and $O(n)$ storage.

Witness events. The red witness point is defined by the red point in R_θ with the minimum x_θ-coordinate and the red point in R_θ with the minimum y_θ-coordinate. To find witness events we proceed as follows. For each red point q, define the function $f_q(\theta)$ as follows:

$$f_q(\theta) = \begin{cases} x_\theta(q) & \text{if } q \in R_\theta \\ \text{undefined} & \text{otherwise} \end{cases}$$

Recall that the angles at which a red point q may enter or leave the blue rectangle (and, hence, become or cease to be a point of R_θ) correspond to the tangents from q to CH_B. Hence, $f_q(\theta)$ has at most two pieces. So, over all points $q \in R$ we have $O(n)$ pieces of functions. The point of R_θ with the minimum x_θ-coordinate is given by the lower envelope of these functions. Note that any two pieces intersect in at most one point. Namely, we can only have $f_q(\theta) = f_{q'}(\theta)$ if θ is the orientation orthogonal to the line through q and q'.[2] It is known that the complexity of the lower envelope of $O(n)$ curves such that any two curves intersect in at most one point is $O(n\alpha(n))$ [13]. Hence, the minimum x_θ-coordinate changes $O(n\alpha(n))$ times. We can find these events by computing the lower envelope in $O(n\alpha(n) \log n)$ time. Similarly, the minimum y_θ-coordinate changes $O(n\alpha(n))$ times, and we can find the changes in $O(n\alpha(n) \log n)$ time.

The algorithm. Our output-sensitive algorithm now works as follows. After pre-computing all blue-rect,

[1] In fact, we are sweeping some details under the rug here. One is that the vertical direction needs a special treatment, since points with the same x-coordinate dualize to parallel lines. A slightly more cumbersome issue is that we do not only need the angles defined by the lines through pairs of points, but also the angles defined by the lines orthogonal to them. This can be handled by performing two sweeps "in parallel": one starting at $x = 0$ in the dual plane, and the other starting at $x = \pi/2$. Also note that both sweeps should "wrap around at infinity".

[2] In fact, there are two opposite orientations orthogonal to this line. Thus the pieces can intersect twice. However, by treating the angles $[0, \pi)$ and $[\pi, 2\pi)$ separately, we can reduce this to the case of a single intersection.

blue-sc, red-set, and witness events, as explained above, we start our rotational sweep. During the rotational sweep, we handle each of the blue-rect, blue-sc, red-set, and witness events in the normal way. However, we also need to detect the crossing events, as follows.

Consider two consecutive (non-crossing) events, and let θ_i and θ_{i+1} denote the angles at which these events take place. Then we need to report all the crossing events taking place at angles θ with $\theta_i \leqslant \theta < \theta_{i+1}$. To this end, we store the blue staircase in a suitable data structure, as explained next. The blue staircase consists of horizontal edges (parallel to the x_θ-axis) and vertical edges (parallel to the y_θ-axis). We explain how to store the horizontal edges; the vertical edges can be handled similarly. Let p and p' be two consecutive points along the staircase, where p' has larger y_θ-coordinate, and consider the horizontal staircase edge e incident to p. Thus, the right endpoint of e is p, and the left endpoint has the same y_θ-coordinate as p and the same x_θ-coordinate as p'.

Now suppose the red witness is defined by red points q and q', with q' having larger y_θ-coordinate. Thus, the red witness is the point $(x_\theta(q'), y_\theta(q))$.

This witness point crosses e for some $\theta \in [\theta_i, \theta_{i+1})$ if and only if the following conditions are met: (i) the angle that $\ell(p, q)$, the line through p and q, makes with the (original) positive x-axis lies in the range $[\theta_i, \theta_{i+1})$, and (ii) $x_\theta(p') \leqslant x_\theta(q')$. We now map the edge e to the point $(x_0(p), y_0(p), x_0(p'), y_0(p'))$ in $\mathbb{R}^4$. Now, for any given red points q and q' defining the red witness and angles θ_i and θ_{i+1}, there is a region $Q(q, q', \theta_i, \theta_{i+1})$ in $\mathbb{R}^4$ with the property that the red witness crosses a horizontal staircase edge e if and only $(x_0(p), y_0(p), x_0(p'), y_0(p')) \in Q(q, q', \theta_i, \theta_{i+1})$. Thus, we store the points of $Q(q, q', \theta_i, \theta_{i+1})$ in a dynamic data structure $\mathcal{D}$ so that we can perform a range query with the range $Q(q, q', \theta_i, \theta_{i+1})$. Note that $\mathcal{D}$ must be updated at each blue sc-event.

The conditions (i) and (ii) above can be expressed as a Boolean formula whose terms are polynomials in the coordinates $(x_0(p), y_0(p), x_0(p'), y_0(p'))$. Hence, $Q(q, q', \theta_i, \theta_{i+1})$ is a semi-algebraic set of constant complexity. Thus, the data structure $\mathcal{D}$ is a data structure for range searching with semi-algebraic sets in $\mathbb{R}^d$ for $d = 4$, and we can get the following performance [1]: For any $n \leqslant m \leqslant n^b$ and any fixed $\varepsilon > 0$, we can obtain $O(n^{1+\varepsilon}/m^{1/b} + t)$ query time (where t is the number of answers) with a structure using $O(m^{1+\varepsilon})$ storage and with $O(m^{1+\varepsilon}/n)$ update time, where $b = 2d - 3 = 5$. The number of queries we have to do is equal to the total number of blue-rect, blue-sc, red-set, and witness events, and so the number of queries is $O(n\alpha(n))$. We therefore set $m := n^{5/3}$. This way each query takes $O(n^{2/3+\varepsilon} + t)$ time and each update takes $O(n^{2/3+\varepsilon})$ time. After performing a query and reporting t crossing events, we need to sort the crossing events in $O(t \log t)$ time to find the angular intervals we have to report. This gives an overall time for our algorithm of $O(n^{5/3+\varepsilon} + k \log k)$, where k is the number of reported angular intervals. We obtain the following theorem.

Theorem 2 *Let B be a set of blue points and let R be a set of red points in the plane, with $n := |B| + |R|$. Then, for any fixed $\varepsilon > 0$, we can compute all k angular intervals for which there is a blue L-shape in $O(n^{5/3+\varepsilon} + k \log k)$ time and $O(n^{5/3+\varepsilon})$ storage.*

References

[1] P.K. Agarwal and J. Matoušek. On range searching with semi-algebraic sets. *Discr. Comput. Geom.*, 11:393–418, 1994.

[2] D. Avis, B. Beresford-Smith, L. Devroye, H. Elgindy, E. Guévremont, F. Hurtado, and B. Zhu. Unoriented Θ-maxima in the plane: complexity and algorithms. *SIAM J. Comput.*, 28:278–296, 1999.

[3] S.W. Bae, C. Lee, H.-K. Ahn, S. Choi, and K.-Y. Chwa. Maintaining extremal points and its applications to deciding optimal orientations. *In Proc. 18th Int. Sympos. Alg. Comput. (ISAAC)*, 788–799, 2007.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications (3rd edition)*. Springer-Verlag, 2008.

[5] H. Edelsbrunner and F. P. Preparata. Minimum polygonal separation. *Inform. Comput.*, 77:218-232, 1988.

[6] S. Fekete. On the complexity of min-link red-blue separation. Manuscript, 1992.

[7] F. Hurtado, M. Noy, P. A. Ramos, and C. Seara. Separating objects in the plane with wedges and strips. *Discr. Appl. Math.*, 109:109-138, 2001.

[8] M. van Kreveld, T. van Lankveld, and R. Veltkamp. Identifying well-covered minimal bounding rectangles in 2D point data. *In 25th European Workshop on Computational Geometry,* 277–280, 2009.

[9] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM J. Comput.*, 12(4):759-776, 1983.

[10] J. S. B. Mitchell. *Approximation Algorithms for Geometric Separation Problems.* Technical Report, State University of New York at Stony Brook, 1993.

[11] J. O'Rourke, S. R. Kosaraju, and N. Megiddo. *Computing circular separability.* Discr. Comput. Geom., 1(1):105-113, 1986.

[12] C. Seara. *On Geometric Separability.* Ph.D. Thesis, Univ. Politècnica de Catalunya, June 2002.

[13] M. Sharir and P.K. Agarwal. Davenport-Schinzel Sequences and their Geometric Applications. *Cambridge University Press,* 1995.

[14] G. Toussaint. Solving geometric problems with the rotating calipers. *In Proc. of the IEEE MELECON,* A10.02/1–4, 1983.