Capturing the Shape of a Point Set with a Line Segment

Nathan van Beusekom
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Marc van Kreveld
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Max van Mulken
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Marcel Roeloffzen
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Bettina Speckmann
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Jules Wulms
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

Detecting location-correlated groups in point sets is an important task in a wide variety of applications areas. In addition to merely detecting such groups, the group's shape carries meaning as well. In this paper, we represent a group's shape using a simple geometric object, a line segment. Specifically, given a radius r, we say a line segment is *representative* of a point set P of n points if it is within distance r of each point $p \in P$. We aim to find the shortest such line segment. This problem is equivalent to stabbing a set of circles of radius r using the shortest line segment. We describe an algorithm to find the shortest representative segment in $O(n \log h + h \log^3 h)$ time, where h is the size of the convex hull of P. Additionally, we show how to maintain a stable approximation of the shortest representative segment when the points in P move.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Shape descriptor, Stabbing, Rotating calipers

Digital Object Identifier 10.4230/LIPIcs.MFCS.2024.26

Related Version *Full Version*: https://arxiv.org/abs/2402.12285

1 Introduction

Studying location-correlated groups or clusters in point sets is of interest in a wide range of research areas. There are many algorithms and approaches to find such groups; examples include the well-known *k-means clustering* [23] or *DBSCAN* [18]. In addition to the mere existence of such groups, the group's characteristics can carry important information as well. In wildlife ecology, for example, the perceived shape of herds of prey animals contains information about the behavioral state of animals within the herd [30]. Since shape is an abstract concept that can get arbitrarily complex, it is often useful to have a simplified representation of group shape that can efficiently be computed. The simplest shape (besides a point) that may represent a group is a line segment, suggesting that the group is stretched in a single direction.

When the points move in the plane, as is the case for animals, the representing line segment may change orientation and length. Also, it may disappear if the shape of the points is no longer captured well by a line segment. Conversely, it can also appear when the points form a segment-like shape again.

Let us concentrate on the static version of the problem first. There are a few simple ways to define a line segment for a set of points. We can use the width of the point set to define a narrowest strip, put tight semi-circular caps on it, and use the centers of these semi-circles as the endpoints of the line segment. We can also use the focal points of the smallest enclosing ellipse, and use them as the endpoints. We can also use a maximum allowed distance from the points to the line segment, and use the shortest line segment possible. The first and third option are based on a hippodrome shape (the Minkowski sum of a line segment and a disk). We note that the second and third option still need a threshold distance to rule out that points are arbitrarily far from the defining line segment. In particular, the case that a line segment is *not* a suitable representation should exist in the model, and also the case where the line segment can become a single point. There are multiple other options besides the three given, for example, by using the first eigenvector of the points, or the diameter.

In this paper we study the model given by the third option: given a set P of n points in general position, we want to find the shortest line segment q_1q_2 such that all points are within distance r. This model has several advantages: (i) It is a simple model. (ii) It naturally includes the case that no line segment represents the points, or a single point already represents the points. (iii) It guarantees that all points are close to the approximating line segment. (iv) It has desirable properties when the points move: in the first two options, there are cases where the points intuitively remain equally stretched in the direction of the line segment, but points moving orthogonally away from it yields a shorter(!) line segment. This issue does not occur in the chosen model. Moreover, it was studied before in computational geometry, and we can build on existing algorithmic methods and properties.

The first algorithm published that solves the optimization version of the static problem (in fact, the first option) uses $O(n^4 \log n)$ time, for a set of n points [24]. This was improved by Agarwal et al. [1] to $O(n^2\alpha(n)\log^3 n)$, where $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function. The first subquadratic bound was given by Efrat and Sharir [17], who presented an $O(n^{1+\varepsilon})$ time algorithm, for any constant $\varepsilon > 0$. They use the fixed-radius version as a subroutine and then apply parametric search. Their fixed-radius algorithm already has the bound of $O(n^{1+\varepsilon})$, as it uses vertical decompositions of a parameter space in combination with epsilon-nets. They remark that their methods can solve the shortest stabber problem for unit disks within the same time, which is our problem.

In this paper we present an improved static result and new kinetic results. We solve the static version in $O(n \log^3 n)$ time by exploiting the geometry of the situation better, which allows us to avoid the use of parametric search and epsilon-nets. Our new algorithm uses a rotating calipers approach where we predict and handle events using relatively simple data structures. We still use a key combinatorial result from [17] in our efficiency analysis. For the kinetic problem, we are interested in developing a strategy to maintain a "stable" line segment that does not frequently appear and disappear, and whose endpoints move with bounded speed. To accomplish this, we must relax (approximate) the radius around the line segment in which points can be. We show that with constant speeds and a constant factor approximation in radius, the endpoints of the line segment move at a speed bounded by a linear function in r, while also avoiding frequent (dis)appearances of the line segment. These results complement recent results on stability.

Related work. A number of shape descriptors have been proposed over the years. A few popular ones are the *alpha shape* of a point set [15] and the *characteristic shape* [12], both of which generate representative polygons. Another way to generate the shape of a point set is to fit a function to the point set [6, 22, 32]. Bounding boxes and strips are much closer related

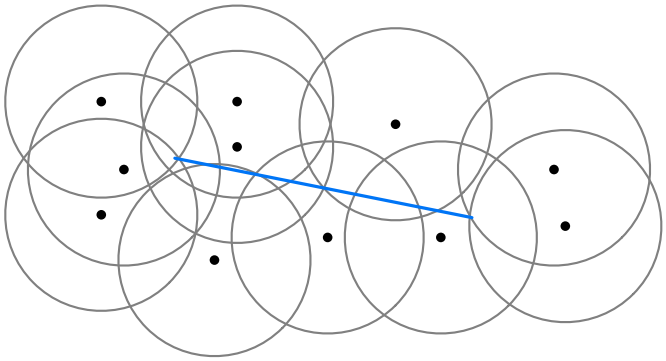

Figure 1 The line segment (blue) must hit every circle of radius r, centered at the points in P.

to the line segment we propose. In the orientation of the first eigenvector, bounding boxes (and strips) have been shown to not capture the dimensions of a point set well [11]. Optimal bounding boxes and strips, of minimum area and width, respectively, align with a convex hull edge and can be computed in linear time given the convex hull [19, 31]. Problems of finding one or more geometric objects that intersect a different set of geometric objects are known as *stabbing* problems [16], and several variants have been studied [5, 10, 29]. As mentioned, stabbing a set of unit circles with the shortest line segment was studied in [17]. The inverse variant, line segments stabbed by one or more circles, has also been studied [7, 25].

Recently, considerable attention has been given to stability of structures under the movement of a set of points or motion of other objects. Stability is a natural concern in, for example, (geo)visualization and automated cartography: In air traffic control planes may be visualized as labeled points on a map, and the labels are expected to smoothly follow the locations of the moving points [8]. Similarly, for interactive maps that allow, for example zooming and panning, labels should not flicker in and out of view [2, 20, 21, 28]. In computational geometry, only the stability of k-center problems was studied [3, 9, 13, 14], until Meulemans et al. introduced a framework for stability analysis [26]. Applying the framework to shape descriptors, they proved that an $O(1)$-approximation of an optimal oriented bounding box or strip moves only a constant-factor faster than the input points [27].

2 Computing the Shortest Representative Segment

Given a set P of n points and a distance bound r, we show how to construct the shortest segment q_1q_2 with maximum distance r to P. See Figure 1 for an example. Omitted proofs can be found in the full version.

Our algorithm uses the rotating calipers approach [31]. We start by finding the shortest representative segment for fixed orientation α, after which we rotate by π while maintaining the line segment, and return the shortest one we encounter. Note that, even though a representative segment does not exist for every orientation, we can easily find an initial orientation α for which it does exist using rotating calipers; these are the orientations at which the rotating calipers have width $\leq 2r$. Although our input point set P can be of any shape, the following lemma shows that it suffices to consider only its convex hull $\mathsf{CH}(P)$.

▶ **Lemma 1.** *If a line segment q_1q_2 intersects all circles defined by the points in the convex hull $\mathsf{CH}(P)$, then q_1q_2 also intersects all circles defined by the points in P.*

We can compute $\mathsf{CH}(P)$ in $O(n \log h)$ time, where h is the size of the convex hull [4].

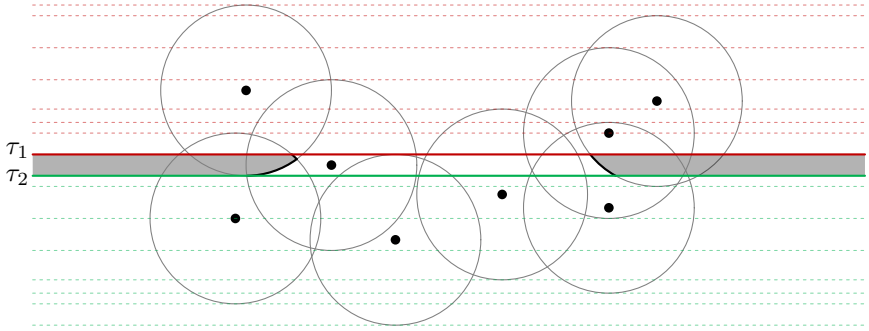

Figure 2 Two extremal tangents τ_1 and τ_2 for horizontal orientation α. The shortest line segment of orientation α that intersects all circles, ends at the boundary of the gray regions.

2.1 Fixed orientation

We describe how to find the shortest representative segment with fixed orientation α. Using rotating calipers [31], we can find all orientations in which a representative segment exists, and pick α such that a solution exists. For ease of exposition and without loss of generality, we assume α to be horizontal. Let the *left* and *right half-circle* of a circle C be the half-circle between $\pi/2$ and $3\pi/2$ and between $3\pi/2$ and $5\pi/2$, respectively. Lemma 1 permits us to consider only points of P on the convex hull, thus for the remainder of this paper we use $\mathcal{C}_P$ to indicate the set of circles of radius r centered at the points of P in $\mathsf{CH}(P)$.

Observe that every horizontal line that lies below the bottom-most top horizontal tangent τ_1 and above the top-most bottom horizontal tangent τ_2 of all circles crosses all circles (see Figure 2). If τ_1 lies below τ_2, then there exists no horizontal line that crosses all circles.

To place q_1q_2 in the strip between τ_1 and τ_2, we can define regions R_1, R_2 in which endpoints q_1 and q_2 must be placed such that q_1q_2 intersects all circles (see Figure 2).

The region R_1 is defined as the set of points below or on τ_1 and above or on τ_2 and right or on the right-most envelope of all left half-circles. The region R_2 is defined analogously using the left envelope of right half-circles. We use S_1 and S_2 to denote the envelope boundary of R_1 and R_2 respectively. Note that S_1 and S_2 are convex and consist of circular arcs from the left and right half-circles respectively. If R_1 and R_2 intersect, then we can place a single point in their intersection at distance at most r from all points in P. Otherwise, note that q_1 and q_2 must be on the convex sequences S_1 and S_2, respectively; otherwise, we can move the endpoint onto the convex sequence, shortening q_1q_2 and still intersecting all circles.

We will show that we can compute S_1 and S_2 in $O(h)$ time. First, we show that the half-circles on a convex sequence appear in order of the convex hull.

▶ **Lemma 2.** *The order of the circular arcs in S_1 or S_2 matches the order of their corresponding centers in $\mathsf{CH}(P)$.*

Now we can compute the convex sequences in linear time, given the tangents τ_1 and τ_2, which can easily be found in linear time.

▶ **Lemma 3.** *Given tangents τ_1 and τ_2, and $\mathsf{CH}(P)$, we can construct S_1 and S_2 in $O(h)$.*

Proof. We assume that a solution exists, which can easily be checked in $O(h)$ time. We describe only the construction of S_1, as S_2 can be constructed symmetrically. Without loss of generality, assume that τ_1 denotes the start of S_1 in clockwise order. We can find the first arc on S_1 by checking all intersections between τ_1 and the relevant half-circles, and

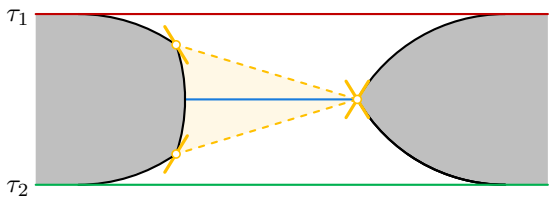

Figure 3 Two convex sequences between τ_1 and τ_2. There are multiple points on the left convex sequence that have the same tangent as the right yellow vertex. Still, there is only one line segment in horizontal orientation for which the tangents of its endpoints are equal (blue).

identifying the most extremal intersection in $O(h)$ time (see Figure 2) We add the part of the circle that lies between τ_1 and τ_2 to S_1. We then process each point p_i along the convex hull in clockwise order from the point defining our initial arc.

Next, let $\overset{\frown}{c}_1, \ldots, \overset{\frown}{c}_k$ denote the circular arc pieces for S_1 constructed so far. Let p_i be the next point on the convex hull that we process. Let c_i denote the left half-circle centered at p_i and let c_k be the support left half-circle of $\overset{\frown}{c}_k$. We find the intersection between c_i and c_k. If there is no intersection, then c_i must lie entirely to the left of c_k and it cannot contribute to S_1. If the intersection point is below τ_2 then between τ_1 and τ_2 we have that c_i lies left of c_k and it cannot contribute to S_1. If the intersection point lies within $\overset{\frown}{c}_k$ then we update S_1 to switch at the intersection point from c_k to c_i as then c_i must lie right of c_k below the intersection point. If the intersection point lies above $\overset{\frown}{c}_k$ then the entirety of $\overset{\frown}{c}_k$ lies to the left of c_i, therefore $\overset{\frown}{c}_k$ cannot contribute to S_1 and we can discard $\overset{\frown}{c}_k$. We then continue by comparing c_i to c_{k-1}.

Whenever a half-circle is possibly added it is compared to at most $O(|S_1|)$ arcs. However, when the half-circle is compared to i arcs, then $i-1$ arcs would be removed from S_1. Thus, by an amortization argument, this happens $O(h)$ times. ◀

Next, we must place q_1 and q_2 on S_1 and S_2, respectively, such that q_1q_2 is shortest. We show that q_1q_2 is the shortest line segment of orientation α when the tangents of S_1 at q_1 and S_2 at q_2 are equal. Vertices on S_1 and S_2 have a range of tangents (see Figure 3).

▶ **Lemma 4.** *Let S_1 and S_2 be two convex sequences of circular arcs, and let q_1 and q_2 be points on S_1 and S_2, respectively, such that line segment q_1q_2 has orientation α. If the tangent on S_1 at q_1 is equal to the tangent on S_2 at q_2, then q_1q_2 is minimal.*

Observe that the length of q_1q_2 is unimodal between τ_1 and τ_2. We can hence binary search in $O(\log h)$ time for the optimal placement of q_1 and q_2. By Lemmata 3 and 4 we can compute the shortest representative segment of fixed orientation α in $O(h)$ time.

2.2 Rotation

After finding the shortest line segment for a fixed orientation α, as described in the previous section, we sweep through all orientations α while maintaining τ_1, τ_2, S_1, S_2, and the shortest representative segment q_1q_2 of orientation α. We allow all of these maintained structures to change continuously as the orientation changes, and store the shortest representative segment found. Any time a discontinuous change would happen, we trigger an *event* to reflect these changes. We pre-compute and maintain a number of *certificates* in an event queue, which indicate at which orientation the next event occurs. This way we can perform the continuous motion until the first certificate is violated, recompute the maintained structures, repair the event queue, and continue rotation.

We distinguish five types of events:

1. q_1 or q_2 moves onto/off a vertex of S_1 or S_2;
2. τ_1 or τ_2 is a bi-tangent with the next circle on the convex hull;
3. τ_1 and τ_2 are the same line;
4. τ_1 or τ_2 is tangent to S_1 or S_2 and rotates over a (prospective) vertex of S_1 or S_2;
5. τ_1 or τ_2 is not tangent to S_1 or S_2 and rotates over a (prospective) vertex of S_1 or S_2.

Since the shortest line segment q_1q_2 in orientation α is completely determined by τ_1, τ_2, S_1, and S_2, the above list forms a complete description of all possible events. Thus, we maintain at most two certificates for events of type 1 (one for each convex sequence) and 2 (one for each tangent), and a single type-3 certificate. Additionally, there must be exactly one type-4 or type-5 certificate for each endpoint of S_1 and S_2, so four in total. These are stored in a constant-size event queue Q, ordered by appearance orientation. Insert, remove, and search operations on Q can hence be performed in $O(1)$ time.

We will describe below how all events over a full rotational sweep can be handled in $O(h \log^3 h)$ time in total. Combined with the computation of the convex hull of P this yields the following theorem. Note that in the worst case P is in convex position, and $n = h$.

▶ **Theorem 5.** *Given a point set P consisting of n points and a radius r, we can find the shortest representative segment in $O(n \log h + h \log^3 h)$ time, where $|\mathsf{CH}(P)| = h$.*

Event handling. In the following descriptions, we assume that an event happens at orientation α, and that ε is chosen such that no other events occur between $\alpha - \varepsilon$ and $\alpha + \varepsilon$. We also assume that no two events happen simultaneously, which is a general position assumption. We describe, for each event type, the time complexity of computing a new certificate of that type, the time complexity of resolving the event, and the number of occurrences.

(1) q_1/q_2 moves onto/off of a vertex of S_1/S_2. We describe, without loss of generality, how to handle the event involving q_1 and S_1; the case for q_2 and S_2 is analogous. See Figure 4 for an example of this event. First, observe that we can compute certificates of this type in $O(1)$ time, simply by walking over S_1 to find the next vertex/arc q_1 should move onto.

▶ **Observation 6.** *We can construct a new certificate of type 1 in $O(1)$ time.*

Observe that, since vertices of S_1 cover a range of tangents, there are intervals of orientations at which q_1 remains at a vertex of S_1. As such, we describe two different cases for this event: q_1 moves *onto* or *off* a vertex of S_1.

If q_1 was moving over an arc of S_1 at $\alpha - \varepsilon$ and encounters a vertex at α, then the movement path of q_1 is updated to remain on the encountered vertex. Additionally, we place a new type-1 certificate into the event queue that is violated when q_1 should move off the vertex, when the final orientation covered by the vertex is reached.

If q_1 is at a vertex at $\alpha - \varepsilon$ and orientation α is the final orientation covered by that vertex, then the movement path of q_1 must be updated to follow the next arc on S_1. Additionally, we place a new type-1 certificate into the event queue that is violated when q_1 encounters the next vertex, at the orientation at which this arc of S_1 ends.

▶ **Lemma 7.** *Throughout the full π rotation, type-1 events happen at most $O(h)$ times, and we can resolve each occurrence of such an event in $O(1)$ time.*

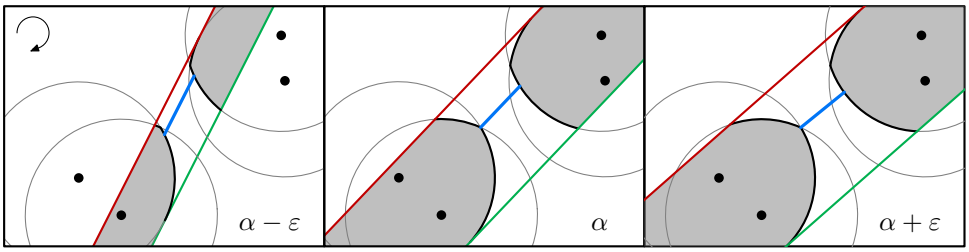

Figure 4 When q_1/q_2 is at a vertex of S_1/S_2, it stops moving.

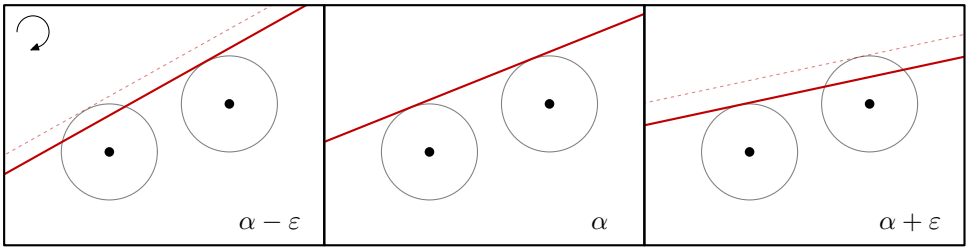

Figure 5 When the defining circle of τ_1/τ_2 changes, τ_1/τ_2 is parallel to a convex hull edge.

(2) τ_1 or τ_2 is bi-tangent with the next circle on the convex hull. We describe, without loss of generality, how to handle the event involving τ_1; handling τ_2 is analogous. See Figure 5 for an illustration. First, observe that we can compute certificates of this type in $O(1)$ time, since these certificates depend only on the orientation of the next convex hull edge.

▶ **Observation 8.** *We can construct a new certificate of type 2 in $O(1)$ time.*

When τ_1 is a bi-tangent of two circles defined by their centers $u, v \in P$ then, by definition of τ_1, u and v must both be the extremal points in the direction θ perpendicular to α. Therefore, (u, v) must be an edge on the convex hull. Suppose that, without loss of generality, u was the previous extremal vertex in direction $\theta - \varepsilon$, then v is extremal in direction $\theta + \varepsilon$. As such, τ_1 belongs to u at $\alpha - \varepsilon$, and to v at $\alpha + \varepsilon$. When this happens, we insert a new type-2 certificate into the event queue that is violated at the orientation of the next convex hull edge. Additionally, we must recompute the certificates of type 3, 4 and 5 that are currently in the event queue, since these are dependent on τ_1.

▶ **Lemma 9.** *Throughout the full π rotation, type-2 events happen at most $O(h)$ times, and we can resolve each occurrence of such an event in $O(\log^2 h)$ time.*

(3) τ_1 and τ_2 are the same line. When this event takes place, τ_1 and τ_2 are the inner bi-tangents of their two respective defining circles. See Figure 6 for an example. First, observe that we can compute certificates of this type in $O(1)$ time by simply finding the inner bi-tangent of the circles corresponding to τ_1 and τ_2.

▶ **Observation 10.** *We can construct a new certificate of type 3 in $O(1)$ time.*

We distinguish two different cases for this event: either there is a solution at $\alpha - \varepsilon$ and no solution at $\alpha + \varepsilon$, or vice versa.

If there was a solution at $\alpha - \varepsilon$ and there is none at $\alpha + \varepsilon$, we simply stop maintaining q_1q_2, S_1 and S_2 until there exists a solution again. As such, we remove all type-1, type-5 and type-4 certificates from the event queue and place a new type-3 certificate into the event queue that is violated at the next orientation where τ_1 and τ_2 are the same line.

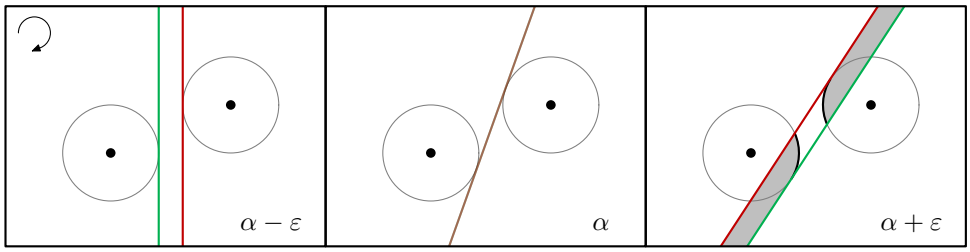

Figure 6 When τ_1 and τ_2 are the same line, they are an inner bi-tangent of their two defining circles.

If there was no solution at $\alpha - \varepsilon$ and there is a solution at $\alpha + \varepsilon$, we must recompute S_1, S_2, and q_1q_2 at orientation α. At orientation α, S_1 and S_2 are single vertices where τ_1 and τ_2 intersect the extremal half-circles of the arrangement. Then, q_1q_2 is the line segment between these single vertices of S_1 and S_2. We place new type-1, type-4 and type-5 certificates into the event queue reflecting the newly found S_1, S_2, q_1 and q_2. Additionally, we insert a new type-3 certificate that is violated at the next orientation where τ_1 and τ_2 are the same line.

▶ **Lemma 11.** *Throughout the full π rotation, type-3 events happen at most $O(h)$ times, and we can resolve each occurrence of such an event in $O(\log^2 h)$ time.*

(4) τ_1 or τ_2 is tangent to S_1 or S_2 and rotates over a (prospective) vertex of S_1 or S_2. We describe, without loss of generality, how to handle the event involving τ_1 and S_1; the case for τ_2 and S_2 is analogous. See Figure 7 for an example of this event. First, observe that we can compute certificates of this type in $O(1)$ time: Let C_i be the circle to which τ_1 is tangent. Then, to construct a certificate, we find the orientation at which τ_1 hits the intersection point of C_i and S_1. Note that this intersection is part of S_1 or will appear at τ_1.

▶ **Observation 12.** *We can construct a new certificate of type 4 in $O(1)$ time.*

Let vertex v be the vertex of the convex chain S_1 that is intersected by τ_1 at orientation α. Then either vertex v is a vertex of S_1 at orientation $\alpha - \varepsilon$ but not at $\alpha + \varepsilon$, or vice versa.

In the prior case, at orientation α the arc to which τ_1 is a tangent is completely removed from S_1. Vertex v becomes the endpoint of S_1 and starts moving along the next arc of S_1. If the affected arc or vertex appeared in a type-1 certificate in the event queue, it is updated to reflect the removal of the arc and the new movement of the vertex. Additionally, we place a new type-5 certificate into the event queue.

In the latter case, at orientation α an arc of the incident circle to τ_1 needs to be added to S_1. If the arc that was previously the outer arc of S_1 appeared in a type-1 certificate in the event queue, it may need to be updated to reflect the addition of the new arc. Additionally, we place a new type-4 certificate into the event queue.

▶ **Lemma 13.** *Throughout the full π rotation, type-4 events happen at most $O(h)$ times, and we can resolve each occurrence of such an event in $O(\log^2 h)$ time.*

(5) τ_1 or τ_2 is not tangent to S_1 or S_2 and rotates over a (prospective) vertex of S_1 or S_2. We describe, without loss of generality, how to handle the event involving τ_1 and S_1; the case for τ_2 and S_2 is analogous. See Figure 8 for an example of this event. The time complexity of constructing a certificate of this type is stated in the following lemma.

▶ **Lemma 14.** *We can construct a new certificate of type 5 in $O(\log^2 h)$ time.*

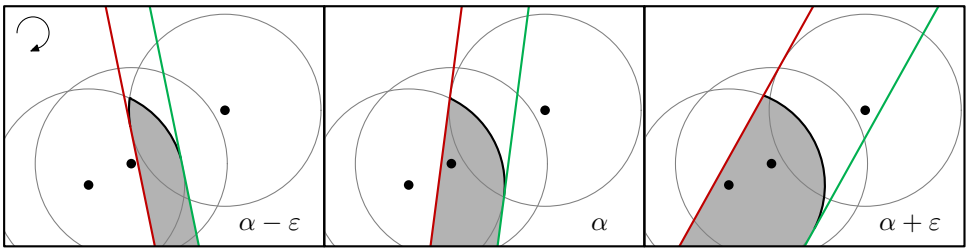

Figure 7 When τ_1/τ_2 hits an intersection of its defining circle that is also on S_1/S_2, an arc is removed from S_1/S_2.

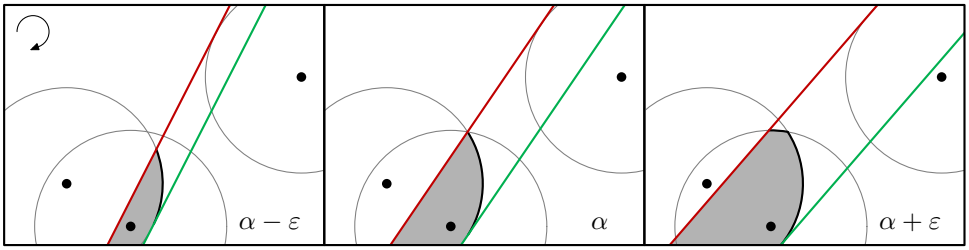

Figure 8 When τ_1/τ_2 hits an intersection of two circles, an arc needs to be added to S_1/S_2.

Additionally, we get the following bounds on handling type-5 events.

▶ **Lemma 15.** *Throughout the full π rotation, τ_1 or τ_2 hits a vertex of S_1 or S_2 at most $O(h \log h)$ times, and we handle each occurrence of this event in $O(\log^2 h)$ time.*

We prove Lemmata 14 and 15 using an additional data structure in the following section.

2.3 Finding and maintaining the convex sequence

In this section, we describe an additional data structure necessary to maintain the convex sequences S_1 and S_2 efficiently. We can use this data structure to construct and handle violations of type-5 certificates efficiently, as well as to find new starting positions of S_1 and S_2 after a type-3 event.

Let $p_1, \ldots, p_h$ be the vertices of the convex hull in clockwise order. At a given orientation α, let p_i be the point corresponding to the circle C_i to which τ_1 is tangent. We use v_τ to denote the intersection point between τ_1 and S_1, if it exists, which is simultaneously an endpoint of S_1. Let p_j be the point corresponding to the circle C_j on which v_τ is located. This implies that the arc on S_1 intersected by τ_1 belongs to circle C_j. Then, during our rotational sweep, v_τ is moving over C_j. A type-5 event takes place when v_τ hits the intersection of C_j with another circle C_k corresponding to point p_k.

If, before a type-5 event, the arc of C_j on S_1 was shrinking due to the movement of v_τ, then C_j is fully removed from S_1 at the event, and v_τ continues moving over S_1. Constructing the certificate in this case is very easy, since all we need to do is walk over S_1 from v_τ to find the next vertex. As such, for the remainder of this section, we consider only the more complicated type-5 event, where the arc of C_j on S_1 is growing due to the movement of v_τ.

In that case, when the type-5 event happens, an arc of C_k is added to S_1, and v_τ starts moving over C_k instead of C_j. As such, to construct a type-5 certificate, we must find the intersection between C_j and another circle C_k belonging to a point $p_k \in P$, such that the intersection between C_j and C_k is the first intersection hit by v_τ. To do this, we will first state some characteristics of C_k and p_k.

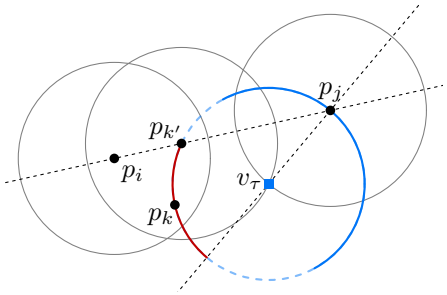

Figure 9 Point p_k is placed on the red arc, which is a subset of the (blue dashed) convex semi-circle centered at v_τ, and disjoint from the (blue solid) concave semi-circle centered at v_τ.

First, observe that finding the first intersecting circle C_k of C_j is not necessarily enough. We are only interested in the semi-circles of all circles in $\mathcal{C}_P$ that have the same "opening direction" as the convex chain S_1 for a given orientation α. As such, let the *convex semi-circle* of a given circle C be the semi-circle of C that is convex with respect to S_1. Conversely, let the *concave semi-circle* of C be the opposite semi-circle of C. Then, circle C_k appears on S_1 after a hit by v_τ at orientation α if v_τ is on the convex semi-circle of C_k at orientation α. If this is not the case, C_k should be skipped. We show that, for this reason, we never have to consider points p_l such that $l < i$ or $j < l$ when constructing a type-5 certificate.

▶ **Lemma 16.** *Let C_i be the circle defining τ_1, and let C_j be the circle on which v_τ is located, for $i \neq j$. Let C_k be the first circle hit by v_τ during rotation at orientation α. If $k < i$ or $j < k$, then at orientation α, v_τ lies on the concave semi-circle of C_k with orientation α.*

Proof. Without loss of generality assume p_i is positioned left of p_j, then assuming that $k < i$ or $j < k$, p_k must lie below the line through p_i and p_j (since $i < j$ and the points are ordered in clockwise order). See Figure 9 for the following construction.

Consider point $p_{k'}$ placed on the line through p_i and p_j, such that v_τ could be the bottom intersection of C_j and a radius-r circle centered at $p_{k'}$. Let p_k be a point placed on the circle of radius r centered at v_τ. If p_k is placed on the other side of the line through v_τ and p_j, compared to $p_{k'}$, then v_τ would enter C_k at orientation α, which does not induce an event. As such, p_k must be on the same side of the line through v_τ and p_j as $p_{k'}$. Additionally, since $k < i$ or $j < k$, p_k must be below the line through p_i and p_j.

It is easy to see that all points on the convex semi-circle of v_τ will have v_τ on their concave semi-circle. Since the arc on which p_k is placed is a strict subset of the concave semi-circle of v_τ, any placement of p_k must have v_τ on its concave semi-circle. ◀

Lemma 16 implies that, while constructing a type-5 certificate, we need to consider only candidate points p_k such that $i < k < j$. Note that, if $i = j$, we get a type-4 event. Then, all that is left is to find the first circle C_k with $i < k < j$ that is intersected by v_τ as it moves over C_j. To this end, we describe a data structure that allows us to perform a circular ray shooting query along C_j from the orientation at which the certificate must be constructed.

Data structure. Our data structure is essentially a balanced binary tree $\mathcal{T}$ on the vertices of the convex hull in clockwise order, where each node stores an associated structure (see Figure 10). For any i, let D_i be the disk bounded by circle C_i. Suppose a node in $\mathcal{T}$ is the root of a subtree with $p_i, \ldots, p_j$ in the leaves. Then its associated structure stores the boundary of $\bigcap_{i \leq l \leq j} D_l$ as a sorted sequence of circular arcs. Given a range (i, j) we can

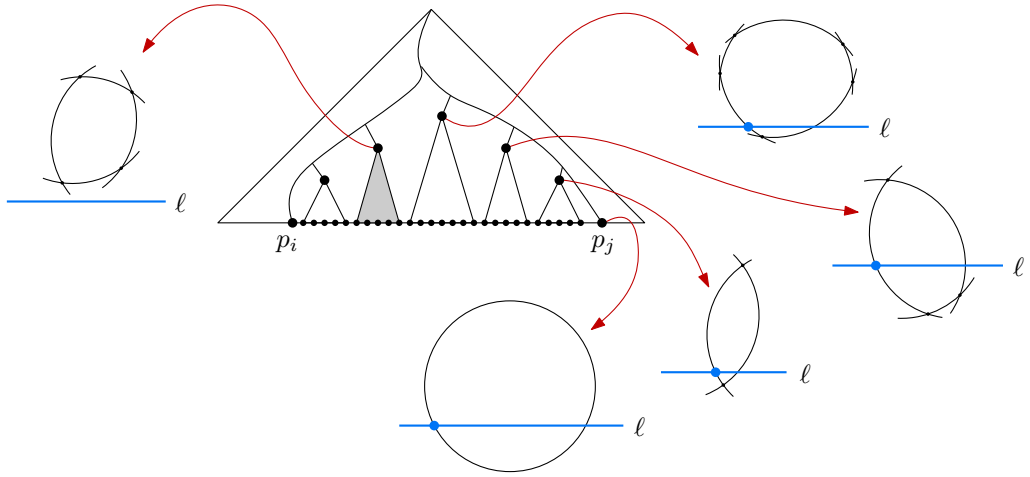

Figure 10 A schematic representation of the data structure $\mathcal{T}$ and a query with ray ℓ.

query this data structure with a (circular) ray in $O(\log^2 h)$ time to find the first intersection of the ray with the boundary of $\bigcap_{i \leq l \leq j} D_l$. A more detailed description of the data structure can be found in the full version of this paper, along with the query algorithms needed to handle certain events and find new certificates.

Event handling. Whenever a certificate of type 5 is violated at orientation α, it means some circle C_k is hit by v_τ at orientation α. It is still possible, however, that this hit happens on the concave semi-circle of C_k. In that case, C_k should not be added to S_1, and v_τ should simply continue moving over its original trajectory. We call these events, where a type-5 certificate is violated but S_1 is not updated, *internal events*. To handle an internal event, we merely need to construct another new type-5 certificate and continue our rotational sweep. In this case, however, we do not have to search the entire range $p_i, \ldots, p_j$ when constructing a new certificate, as shown in the following lemma.

▶ **Lemma 17.** *Let p_i be the point corresponding to τ_1, and let v_τ be at the intersection of circles C_j and C_k, where C_j is the circle that defines the current trajectory of v_τ and where v_τ is on the concave semi-circle of C_k. Then if the next circle hit by v_τ is C_l for $i < l < k$, this circle is hit on its concave semi-circle.*

Proof. Let C_l be the next circle hit by v_τ for $i < l < k$, and see Figure 11 for the following construction. Since v_τ is on the concave semi-circle of p_k, p_k must be on the convex semi-circle of v_τ. Furthermore, as C_k was hit by v_τ, p_k must lie on the same side of the line through p_j and v_τ as p_i. Let the endpoint of the concave semi-circle of v_τ that lies clockwise from p_k be denoted v_c, and observe that $d(p_k, p_j) > d(v_c, p_j)$, where $d(a, b)$ denotes the Euclidean distance between a and b. Additionally, since we consider only points on the convex hull, point p_l must lie above line $p_i p_k$ but below line $p_k p_j$, resulting in a cone with its apex at p_k.

Every point in this cone is further away from p_j than p_k: Since v_τ is part of S_1, placing q_1 at v_τ must yield a valid solution for $q_1 q_2$. This means that all points must be in the Minkowski sum of a radius r disc and the ray with orientation α originating from v_τ, and hence p_k lies below the line $v_c p_i$ (see Figure 12). Finally, p_j lies on the concave semi-circle around v_τ, and p_k lies on the convex semi-circle around v_τ, which shows that the cone lies on the far side of p_k with respect to p_j. This implies that $d(p_l, p_j) > d(p_k, p_j)$.

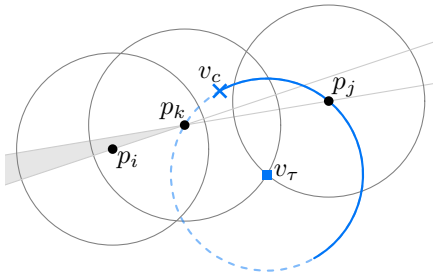

Figure 11 p_l must be located in the gray cone.

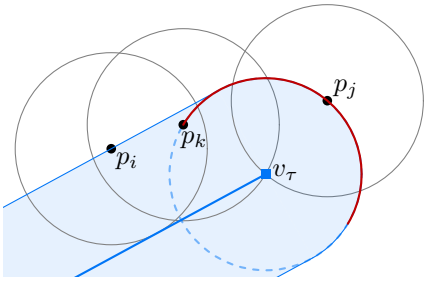

Figure 12 All points in $\mathsf{CH}(P)$ must be in the blue shaded area. When an internal event happens, the red semi-circle is a witness that (p_k, p_j) is conjugate.

During our monotone rotational sweep, v_c must continuously move closer to p_j as we continue our sweep. Therefore, when v_τ intersects C_l, we must have $d(p_l, p_j) > d(v_c, p_j)$: This directly implies that v_c must lie clockwise from p_l on the radius r circle centered at v_τ. Thus, v_τ lies on the concave semi-circle of C_l. ◀

Lemma 17 implies that, after an internal event with circle C_l, it is sufficient to consider only points p_k with $l < k < j$ when constructing a new type-5 certificate.

When a type-5 certificate is violated and v_τ is on the convex semi-circle of C_k, however, we do need to update S_1 to reflect v_τ moving over the intersection between C_k and C_j. Additionally, we must construct new certificates: We possibly need to compute a new type-1 certificate, as well as either a type-4 certificate or a new type-5 certificate using C_k as the new trajectory of v_τ and searching for the next hit with C_l for $i < l < k$. We are now ready to prove Lemmata 14 and 15.

▶ **Lemma 14.** *We can construct a new certificate of type 5 in $O(\log^2 h)$ time.*

Proof. Let p_i be the point corresponding to τ_1, C_j be the circle over which v_τ is currently moving, and α be the current orientation. To construct a type-5 certificate, we find the first circle C_k with $i \leq l < k < j$ that is intersected by v_τ. Here, if this certificate is constructed during an internal event, l is the index of the circle C_l intersected by v_τ during that event. Otherwise, $l = i$. Since v_τ moves over C_j, we can use the data structure described in the full version of this paper to perform a circular ray shooting query on the sequence $C_l, \ldots, C_{j-1}$ with starting point v_τ to find C_k in $O(\log^2 h)$. This gives us the point p_k to include in the certificate, and we can find the orientation at which p_k is hit by drawing the tangent of C_i through the intersection point between C_j and C_k. ◀

▶ **Lemma 15.** *Throughout the full π rotation, τ_1 or τ_2 hits a vertex of S_1 or S_2 at most $O(h \log h)$ times, and we handle each occurrence of this event in $O(\log^2 h)$ time.*

Proof. To show that this event happens $O(h \log h)$ times during a π rotation, we need the following definition. Let an ordered pair (p_g, p_h) of vertices of $\mathsf{CH}(P)$ be *conjugate* if we can place a semi-circle of radius r so that it hits p_g and p_h, p_h lies clockwise from p_g on this semi-circle, and the semi-circle does not intersect the interior of $\mathsf{CH}(P)$. Efrat and Sharir prove that any convex polygon with n vertices has at most $O(n \log n)$ conjugate pairs if the vertices are placed in general position [17]. We charge each occurrence of a type-5 event to a conjugate pair, and prove that each pair is charged only a constant number of times. This immediately yields the bound of $O(h \log h)$ on the number of type-5 events.

Consider an internal type-5 event. We charge these events to the pair (p_k, p_j). To this end, we show that this pair of points must be conjugate. Consider orientation α at which this event takes place. At that point, we can draw a circle of radius r, centered at v_τ, through p_k and p_j. Since, by definition of an internal event, v_τ is on the concave semi-circle of p_k, p_k must be on the convex semi-circle centered around v_τ.

Now again observe that, since v_τ is part of S_1, placing q_1 at v_τ must yield a valid solution for q_1q_2. This means that all points must be in the Minkowski sum of a radius r disc and the ray with orientation α originating from v_τ. See Figure 12. Therefore, the concave semi-circle of radius r centered at v_τ does not intersect $\mathsf{CH}(P)$. If we rotate this semi-circle counter-clockwise until one of its endpoints coincides with p_k, we obtain a semi-circle through p_k and p_j that does not intersect $\mathsf{CH}(P)$. This means it is a witness that (p_k, p_j) is conjugate.

Next, consider a type-5 event that is not internal. The same argument as above holds, except p_k is already on the concave semi-circle of radius r centered at v_τ. Since this semi-circle does not intersect $\mathsf{CH}(P)$, that semi-circle is a direct witness that (p_k, p_j) is conjugate.

Every conjugate pair only induces at most one type-5 event. In order for conjugate pair (p_k, p_j) to induce a second type-5 event, v_τ must again hit the intersection between p_k and p_j while it is moving in the same angular movement direction. This can only happen if we perform a full 2π rotational sweep, or if v_τ first moves over this intersection in the opposite direction. The prior is not possible in a π rotational sweep. The latter is only possible if p_k is first involved in a type-4 event. But then, $k = i$, and by construction p_k can never be involved in another type-5 certificate.

Handling a type-5 event consists of updating S_1 and the movement trajectory of v_τ, which can be done in $O(1)$ time. Additionally, we construct new type-1 certificate and either a type-4 or type-5 certificate, which can be done in $O(1)$ and $O(\log^2 h)$ time by Observations 6 and 12 and Lemma 14. ◀

After analyzing all events that occur during the rotational sweep, we can prove Theorem 5.

▶ **Theorem 5.** *Given a point set P consisting of n points and a radius r, we can find the shortest representative segment in $O(n \log h + h \log^3 h)$ time, where $|\mathsf{CH}(P)| = h$.*

Proof. We initialize the algorithm by computing the convex hull $\mathsf{CH}(P)$. By Lemma 1, it is sufficient to consider only points in $\mathsf{CH}(P)$ to find a representative segment of P. We use rotating calipers to check that the shortest representative segment is not a point, and find an orientation α in which a solution exists. For this fixed α we find τ_1 and τ_2, compute S_1 and S_2, as well as the shortest line segment q_1q_2 in orientation α. Computing the convex hull can be done in $O(n \log h)$ [4]. By Lemma 3, S_1 and S_2 can be initialized in $O(h)$, and we can initialize q_1q_2 in $O(\log h)$ time. As such, initialization of the algorithm can be done in $O(n \log h)$ time in total.

Next, we rotate orientation α over π in total, maintaining τ_1, τ_2, S_1, and S_2, as well as q_1q_2. Note that a rotation of π is sufficient, since we consider orientations, which identify opposite directions of a 2π rotation. Throughout the rotation we maintain the shortest

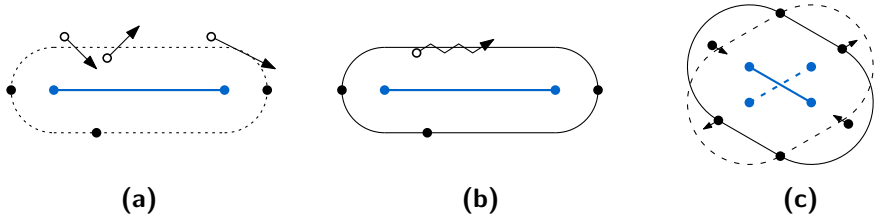

Figure 13 Examples of representative segments (blue) for moving points. These show that a representative segment may appear and disappear frequently **(a)** due to the motion of two points (top left) or one point (top right) or **(b)** due to minor oscillations in a movement path. **(c)** Small movements can trigger a discrete change of the representative segment.

representative segment, and return the shortest such line segment found over all orientations. Over the full rotation, we encounter five different types of events. By Lemmata 7–15, these events can be handled in $O(h \log^3 h)$ time in total.

As we mentioned in Section 2.1, the shortest representative segment is defined by only τ_1, τ_2, S_1, and S_2. Each tangent is defined by a circle. Hence, τ_1 and τ_2 can only change when they are defined by a new circle. Thus, by Lemma 9, we correctly maintain τ_1 and τ_2 throughout the full π rotation. Furthermore, S_1 and S_2 can only exist when τ_1 and τ_2 appear in the correct order. By Lemma 11 we correctly maintain when S_1 and S_2 exist. Finally, S_1 and S_2 can only make a discrete change as the tangents τ_1 and τ_2 hit intersections between circles in $\mathcal{C}_P$. If the tangents do not touch a vertex, then S_1 and S_2 must change continuously along the arcs that τ_1 and τ_2 cross. By Lemmata 13 and 15, we correctly maintain S_1 and S_2 when they exist. In conclusion, we correctly maintain τ_1, τ_2, S_1, and S_2 throughout the full π rotation. Since we maintain the shortest line segment between S_1, and S_2 in any orientation using Lemma 7, we also maintain the shortest representative segment for any orientation. ◀

3 Stable Representative Segments for Moving Points

In this section, we consider maintaining representative segment q_1q_2 while the points in P move. We first show what can happen if we would maintain the optimal solution explicitly under continuous motion of the points.

There are examples where a representative segment exists for a value of r for an arbitrarily short duration. This can happen because one point moves towards a hippodrome shape and another point moves out of it, giving a brief moment with a valid hippodrome. The same effect can be caused by a single linearly moving point that grazes the hippodrome at a join point. These examples are illustrated in Figure 13(a). It can also happen that a minor oscillating movement of a point causes a quick sequence of changes between a valid and no valid segment, see Figure 13(b).

Now, consider the point set P in which the points form a regular k-gon (see Figure 13(c)). When r is equal to half the width of the k-gon, then we can force a discrete change in the placement of q_1 and q_2, with very slow movement of points in P. In this case there is always a valid representative segment, but its endpoints make a jump.

We see that continuously maintaining the optimal solution has two artifacts that are undesirable to a human observer: (1) the segment can appear and disappear frequently within an arbitrarily short time frame, and (2) a segment may jump to a new location, leading to infinitely high speeds of the endpoints even if the points themselves move slowly.

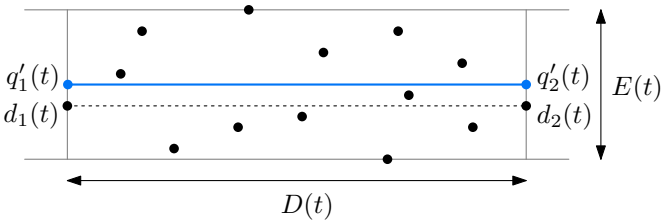

Figure 14 The prospective segment $q_1'q_2'$ in relation to D, E, and diametrical line d_1d_2 at time t.

Our goal is therefore to ensure that the segment movement is *stable* over time: We want q_1 and q_2 to move continuously and with bounded speed, preventing discrete or (near) instantaneous changes. We accomplish this as follows. First, we will sample the positions of the moving points only at integer moments, and then decide immediately to show or not show a solution in the next time unit. This implies that existence and non-existence of a solution lasts at least one time unit, and we avoid the solution (dis)appearing frequently. Second, we make the assumption that the maximum speed of the points is unit. We have to make some bounded speed assumption otherwise we cannot hope to get a bounded speed of the endpoints either. Third, we allow more flexibility in when we have a solution. We do this using a less strict regime on r, and by assuming that $r \geq 1$. Whenever the real solution exists (with the actual r), then we guarantee that we also have a solution. Whenever there is no solution even for a radius of $2\sqrt{2} \cdot r + 4$, then we never give a solution. When the radius is in between these bounds, we may have a solution or not. Our algorithm can then ensure that the speeds of the endpoints are bounded, and the length of our chosen segment always approximates the true optimum (when it exists), at any moment in time, also between the integer sampling moments.

So we assume that a point $p \in P$ is described by a trajectory that is sampled at integer timestamps. That is, each point in P is described by $p(i) \to \mathbb{R}^2$ where $i \in \mathbb{Z}$ is the timestamp. We also assign each endpoint of the representative segment a position as a function of $t \in \mathbb{R}$, which means that the representative segment at time t is now defined by $q_1(t)q_2(t)$. Furthermore, we use $D(t)$ and $W(t)$ as the *diameter* and *width* of the point set, which are respectively the maximum pairwise distance of points in P and the width of the thinnest strip containing P. Let $d_1(t), d_2(t) \in P$ be a pair of points defining the diameter. Lastly, we also use the *extent* $E(t)$ of P in the direction orthogonal to the line segment $d_1(t)d_2(t)$. For all of the above definitions, we omit the dependence on t when it is clear from the context.

In the following, we describe how to specify $q_1(t)$ and $q_2(t)$ such that they move with bounded speed, and such that the length of the segment $q_1(t)q_2(t)$ as well as the proximity of the segment to P can be bounded at any time t. In particular, we define such a segment $q_1(t)q_2(t)$ as an *approximating* segment, and prove that the length of an optimal representative segment is approximated by an additive term l, and at the same time the maximum distance from any point in P to the segment $q_1(t)q_2(t)$ is at most $h \cdot r$, for some constant h.

Algorithm. Our algorithm $A(t)$ is *state-aware*. This means that the output of $A(t)$ is dependent only on the input at or before time t, but it has no knowledge of the input after time t. At every integer timestamp $i \in \mathbb{Z}$, we compute a *canonical solution* $q_1'(i)q_2'(i)$. The endpoints $q_1'(i)$ and $q_2'(i)$ of this canonical solution are placed on the lines orthogonal to the diametric line through $d_1(i)$ and $d_2(i)$, respectively, such that $q_1'(i)q_2'(i)$ lies in the middle of the narrowest strip containing P in the diametric orientation, see Figure 14.

Our algorithm is now a special kind of state-aware algorithm called a *chasing algorithm* [27]: At at every integer time step $i \in \mathbb{Z}$, the algorithm computes the canonical solution $q'_1(i)q'_2(i)$ and then linearly interpolates from $q'_1(i-1)q'_2(i-1)$ to $q'_1(i)q'_2(i)$, arriving there at time $i+1$. At that point, $q'_1(i+1)q'_2(i+1)$ can be computed, and we continue in a similar manner.

However, if the maximum distance from P to the canonical solution becomes too large, we no longer want the algorithm to output any solution. In this case, no (optimal) representative solution exists, and we do not produce an approximating segment either.

Formally, for any timestamp $t \in (i, i+1)$, we linearly interpolate q_1 and q_2 between their previous canonical placements as follows. We define $\alpha = (t-i)$ and set $q_j(t) = \alpha \cdot q'_j(i-1) + (1-\alpha) \cdot q'_j(i)$, for $j \in \{1, 2\}$. Then, the output of our algorithm is $A(t) = q_1(t)q_2(t)$ if $E(\lfloor t \rfloor) \leq 2r\sqrt{2} + 2$ and $\varnothing$ otherwise.

The above algorithm yields the bounds stated in the following theorem. Detailed proofs for these bounds can be found in the full version of this paper.

▶ **Theorem 18.** *Given a set P of points moving with at most unit speed, algorithm A yields a stable approximating segment with $l = 2r + 4$ and $h = 2\sqrt{2} + 4$, for which speed of the endpoints is bounded by $(2r+1)\sqrt{2} + 2$.*

4 Conclusion

In this paper, we presented an $O(n \log h + h \log^3 h)$ time algorithm to find the shortest representative segment of a point set, improving the previous $O(n^{1+\varepsilon})$ time solution. Additionally, we showed how to maintain an approximation of the shortest representative segment in a stable manner, such that its endpoints move with a speed bounded by a linear function in r.

There may be possibilities for improving the running time of our static solution to $O(n \log h + h \log^2 h)$, or even $O(n \log h)$. The $O(h \log^3 h)$ term comes from having to handle $O(h \log h)$ type-5 events in $O(\log^2 h)$ time each. However, it may be possible to show that there are at most $O(h)$ type-5 events, since the conjugate pairs used to bound the number of internal events each have a unique starting point. Additionally, it may be possible to improve the query time of the data structure described in the full version of this paper to $O(\log h)$ time using ideas like fractional cascading, but there is no straightforward way to make this work for the circular query.

References

1. Pankaj K. Agarwal, Alon Efrat, Micha Sharir, and Sivan Toledo. Computing a segment center for a planar point set. *Journal of Algorithms*, 15(2):314–323, 1993. doi:10.1006/jagm.1993.1043.
2. Ken Been, Martin Nöllenburg, Sheung-Hung Poon, and Alexander Wolff. Optimizing active ranges for consistent dynamic map labeling. *Computational Geometry: Theory and Applications*, 43(3):312–328, 2010. doi:10.1016/j.comgeo.2009.03.006.
3. Sergei Bespamyatnikh, Binay Bhattacharya, David Kirkpatrick, and Michael Segal. Mobile facility location. In *Proc. 4th Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, pages 46–53, 2000. doi:10.1145/345848.345858.
4. Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996. doi:10.1007/BF02712873.
5. Timothy M. Chan, Thomas C. van Dijk, Krzysztof Fleszar, Joachim Spoerhase, and Alexander Wolff. Stabbing rectangles by line segments–how decomposition reduces the shallow-cell complexity. In *Proc. 29th International Symposium on Algorithms and Computation (ISAAC)*, pages 61:1–61:13, 2018. doi:10.4230/LIPICS.ISAAC.2018.61.

6 Danny Z. Chen and Haitao Wang. Approximating points by a piecewise linear function. *Algorithmica*, 66(3):682–713, 2013. doi:10.1007/s00453-012-9658-y.

7 Merce Claverol, Elena Khramtcova, Evanthia Papadopoulou, Maria Saumell, and Carlos Seara. Stabbing circles for sets of segments in the plane. *Algorithmica*, 80:849–884, 2018. doi:10.1007/s00453-017-0299-z.

8 Mark de Berg and Dirk H. P. Gerrits. Labeling moving points with a trade-off between label speed and label overlap. In *Proc. 21st European Symposium on Algorithms (ESA)*, pages 373–384, 2013. doi:10.1007/978-3-642-40450-4_32.

9 Mark de Berg, Marcel Roeloffzen, and Bettina Speckmann. Kinetic 2-centers in the black-box model. In *Proc. 29th Symposium on Computational Geometry (SoCG)*, pages 145–154, 2013. doi:10.1145/2462356.2462393.

10 José Miguel Díaz-Báñez, Matias Korman, Pablo Pérez-Lantero, Alexander Pilz, Carlos Seara, and Rodrigo I. Silveira. New results on stabbing segments with a polygon. *Computational Geometry*, 48(1):14–29, 2015. doi:10.1016/j.comgeo.2014.06.002.

11 Darko Dimitrov, Christian Knauer, Klaus Kriegel, and Günter Rote. Bounds on the quality of the PCA bounding boxes. *Computational Geometry*, 42(8):772–789, 2009. doi:10.1016/j.comgeo.2008.02.007.

12 Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern recognition*, 41(10):3224–3236, 2008. doi:10.1016/j.patcog.2008.03.023.

13 Stephane Durocher and David Kirkpatrick. The steiner centre of a set of points: Stability, eccentricity, and applications to mobile facility location. *International Journal of Computational Geometry & Applications*, 16(04):345–371, 2006. doi:10.1142/S0218195906002075.

14 Stephane Durocher and David Kirkpatrick. Bounded-velocity approximation of mobile Euclidean 2-centres. *International Journal of Computational Geometry & Applications*, 18(03):161–183, 2008. doi:10.1142/S021819590800257X.

15 Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559, 1983. doi:10.1109/TIT.1983.1056714.

16 Herbert Edelsbrunner, Hermann A. Maurer, Franco P. Preparata, Arnold L. Rosenberg, Emo Welzl, and Derick Wood. Stabbing line segments. *BIT Numerical Mathematics*, 22:274–281, 1982. doi:10.1007/BF01934440.

17 Alon Efrat and Micha Sharir. A near-linear algorithm for the planar segment-center problem. *Discrete & Computational Geometry*, 16(3):239–257, 1996. doi:10.1007/BF02711511.

18 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996. doi:10.5555/3001460.3001507.

19 Herbert Freeman and Ruth Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, 18(7):409–413, 1975. doi:10.1145/360881.360919.

20 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Sliding labels for dynamic point labeling. In *Proc. 23rd Canadian Conference on Computational Geometry (CCCG)*, pages 205–210, 2011.

21 Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. Consistent labeling of rotating maps. *Journal of Computational Geometry*, 7(1):308–331, 2016. doi:10.20382/jocg.v7i1a15.

22 S. Louis Hakimi and Edward F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing*, 53(2):132–136, 1991. doi:10.1016/1049-9652(91)90056-P.

23 John A. Hartigan and Manchek A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society.*, 28(1):100–108, 1979. doi:10.2307/2346830.

24 Hiroshi Imai, DT Lee, and Chung-Do Yang. 1-segment center problems. *ORSA Journal on Computing*, 4(4):426–434, 1992. doi:10.1287/ijoc.4.4.426.

25 Konstantin Kobylkin. Stabbing line segments with disks: complexity and approximation algorithms. In *Proc. 6th International Conference on Analysis of Images, Social Networks and Texts (AIST)*, pages 356–367, 2017. doi:10.1007/978-3-319-73013-4_33.

26 Wouter Meulemans, Bettina Speckmann, Kevin Verbeek, and Jules Wulms. A framework for algorithm stability and its application to kinetic euclidean MSTs. In *Proc. 13th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 805–819, 2018. doi:10.1007/978-3-319-77404-6_58.

27 Wouter Meulemans, Kevin Verbeek, and Jules Wulms. Stability analysis of kinetic orientation-based shape descriptors. *arXiv preprint*, 2019. arXiv:1903.11445.

28 Martin Nöllenburg, Valentin Polishchuk, and Mikko Sysikaski. Dynamic one-sided boundary labeling. In *Proc. 18th International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 310–319, 2010. doi:10.1145/1869790.1869834.

29 Lena Marie Schlipf. *Stabbing and Covering Geometric Objects in the Plane*. PhD thesis, FU Berlin, 2014.

30 Ariana Strandburg-Peshkin, Damien R. Farine, Margaret C. Crofoot, and Iain D. Couzin. Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement. *eLife*, 6:e19505, 2016. doi:10.7554/eLife.19505.

31 Godfried T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. 2nd Mediterranean Electrotechnical Conference (MELECON)*, volume 1, page A10, 1983.

32 Der Perng Wang. A new algorithm for fitting a rectilinear x-monotone curve to a set of points in the plane. *Pattern Recognition Letters*, 23(1-3):329–334, 2002. doi:10.1016/S0167-8655(01)00130-1.