

# Critical Placements of a Square or Circle amidst Trajectories for Junction Detection

Ingo van Duijn\*    Irina Kostitsyna†    Marc van Kreveld‡  
 Maarten Löffler§

## Abstract

Motivated by automated junction recognition in tracking data, we study a problem of placing a square or disc of fixed size in an arrangement of lines or line segments in the plane. We let distances among the intersection points of the lines and line segments with the square or circle define a clustering, and study the complexity of *critical* placements for this clustering. Here critical means that arbitrarily small movements of the placement change the clustering.

A parameter  $\varepsilon$  defines the granularity of the clustering. Without any assumptions on  $\varepsilon$ , the critical placements have a trivial  $\mathcal{O}(n^4)$  upper bound. When the square or circle has unit size and  $0 < \varepsilon < 1$  is given, we show a refined  $\mathcal{O}(n^2/\varepsilon^2)$  bound, which is tight in the worst case.

We use our combinatorial bounds to design efficient algorithms to compute junctions. As a proof of concept for our algorithms we have a prototype implementation that showcases their application in a basic visualization of a set of  $n$  trajectories and their  $k$  most important junctions.

## 1 Introduction

Many analysis problems in geography have an inherent scale component: the “granularity” or “coarseness” at which the data is studied. The most direct way to model spatial scale in geographic problems is by using a fixed-size neighborhood of locations, such as a fixed-size square or circle. For example, *population density* can be studied at the scale of a city or at the scale of a country, where one may consider the population in units with an area of  $10^4 m^2$  or  $25 km^2$ , respectively. There are many other cases where local geographic phenomena are studied at different spatial scales. The Geographical Analysis Machine is an example of a system that supports such analyses on point data sets [13].

---

\*MADALGO, Aarhus University, [ivd@cs.au.dk](mailto:ivd@cs.au.dk)

†Université libre de Bruxelles, [irina.kostitsyna@ulb.ac.be](mailto:irina.kostitsyna@ulb.ac.be)

‡Utrecht University [m.j.vankreveld@uu.nl](mailto:m.j.vankreveld@uu.nl)

§Utrecht University [m.loffler@uu.nl](mailto:m.loffler@uu.nl)

In computational geometry, the problem of computing the placement of a square or circle to optimize some measure has received considerable attention. For a set of  $n$  points in the plane, one can compute the (fixed-size, fixed-orientation) square that maximizes the number of points inside in  $\mathcal{O}(n \log n)$  time (expand every point to a square, and the problem becomes finding a point in the maximum number of squares which is solved by a sweep). Mount et al. [11] study the overlap function of two simple polygons under translation and show, among other things, that one can compute the placement of a square that maximizes the area of overlap with a simple polygon with  $n$  vertices in  $\mathcal{O}(n^2)$  time. For a weighted subdivision, one can compute a placement that maximizes the weighted area inside in  $\mathcal{O}(n^2)$  time as well [2]; this problem is motivated by clustering in aggregated data. In the context of diagram placement on maps, various other measures to minimize or maximize when placing squares were considered, like the total length of border overlap [18].

Our interest lies in a problem concerning trajectory data. A trajectory is represented by a sequence of points with associated time stamps, and models the movement of an entity through space; we will assume in this paper that the movement space is two-dimensional. The identification of “interesting regions” in the plane defined by a collection of trajectories has been studied in several papers recently. These regions can be characterized as meeting places [5], popular or interesting places [1, 6, 14, 15], and stop regions [10]. In several cases, interesting regions are also defined as squares of fixed size, placed suitably. The more algorithmic papers show such regions of interest can typically be computed in  $\mathcal{O}(n^2)$  or  $\mathcal{O}(n^3)$  time; what is possible of course depends on the precise definition of the problem. There are many other types of problems that can be formulated with trajectory data. For overviews, see [4, 9].

Besides exact algorithms for optimal square or circle placement, approximation algorithms have been developed for several problems on trajectory and other data, see e.g. [5, 8, 17].

**Motivation and problem description.** We consider a problem on trajectories related to common movements and changes of movement directions at certain places. Imagine a large open space like a town square, a large entrance hall, or a grass field. People tend to traverse such areas in ways that are not random, and the places where a decision is made and possibly a change of direction is initiated may be specific. Also for data like ant tracks, the identification of places where tracks go different ways is of interest. Without going into details, these observations motivate us to study placements of a square or circle of fixed size such that bundles of incoming and outgoing entities arise.

Consider the tracks in Figure 1. We observe that to define places where the tracks of entities cross and where decisions are made, we can use the placement of a square of a certain size and where the tracks enter and leave the square. We are interested in placements that give rise to bundles: large subsets of tracks that enter and leave the square at a similar place on its boundary, and with a gap to the next location along the boundary where this happens. The left square in the figure has five bundles where one bundle consists of only one trajectory,

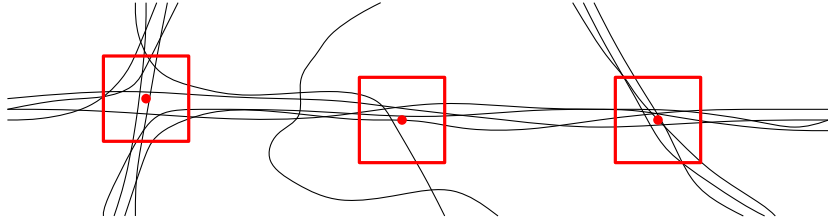


Figure 1: A set of trajectories and square placements at junctions of varying significance.

and the middle and right square have only two bundles (albeit with different “topology”). We see that the left and right squares indicate regions that should be considered more significant than the middle one for being a junction. The main difference between the left and right junctions is that on the left, decisions are made to go straight or change direction, whereas on the right, all moving entities went straight and no different decisions were taken.

We study an abstracted version of this placement problem and ignore many of the practical issues that our simplified definition of a junction would have. Later in the paper we briefly address such issues by giving a slightly more involved definition of a junction. The majority of the paper concentrates on an abstract combinatorial and computational problem that lies at the core of junction detection.

Let  $\mathcal{T}$  be a collection of trajectories, let  $\square_p$  be the boundary of a unit square  $S$  centered at  $p$  placed axis-aligned amidst the trajectories, and let  $\varepsilon$  be a positive real constant. Let  $Q = \mathcal{T} \cap \square_p$  be the set of all intersections of trajectories with the boundary of the square (here we can ignore the time component of trajectories; they are considered polygonal lines). Two points in  $Q$  are  $\varepsilon$ -close if their distance along  $\square_p$  is at most  $\varepsilon$ .  $\mathcal{T}$  and  $\square_p$  give rise to a *clustering* of  $Q$  on  $\square_p$  by the transitive closure relation of  $\varepsilon$ -closeness. Different clusters are separated by a distance larger than  $\varepsilon$  along  $\square_p$ . A single cluster of  $Q$  corresponds to parts of trajectories that enter or leave  $S$  in each other’s proximity (in Figure 1 from left to right the squares define four, three, and four clusters respectively).

Now consider the two-dimensional space of all placements of a square of fixed size by choosing its center as its reference point. We say that a placement  $q$  is *critical*, if any arbitrarily small neighborhood of  $q$  contains points inducing different clusters. This can be due to a change in size of a cluster on  $\square_q$  or to a change in the clustering. The latter corresponds to placements where the distance between two points of  $\mathcal{T} \cap \square_p$  on  $\square_p$  is exactly  $\varepsilon$ , with no other points of  $\mathcal{T} \cap \square_p$  in between. Since noncritical points are part of a region that defines the same clustering, one can think of the set of critical points to be the boundary between these regions.

**Results and organization.** In Section 2 we analyze the complexity of the space of critical placements for fixed-size squares in an arrangement of  $n$  lines. We

show that the placement space has  $\mathcal{O}(n^2/\varepsilon^2)$  total complexity in the worst case, and can be constructed in  $\mathcal{O}(n^2 \log n + k)$  time where  $k$  is the true complexity (the latter appearing in the appendix due to space constraints). In Section 4, we show that these results are tight by presenting an explicit construction that exhibits the worst-case behavior. In Section 6 we discuss our application to junction detection further and show output from a prototype implementation. We conclude in Section 7. In the appendix we further show how to extend our approach to more realistic settings, such as placing a square on an arrangement of line segments or placing a circle rather than a square.

## 2 Square on lines

We begin by studying the simplest version of the problem: placing a unit square over an arrangement of lines. The lines “cut” the boundary of the square into several pieces. We are interested in all placements of the square such that one of these pieces has length exactly  $\varepsilon$ , in which case we call the piece an  $\varepsilon$ -segment and the placement *critical*. When a piece contains one of the corners of the square, its length is the sum of its two incident line segments. Note that this definition of a critical placement is a simplification of the one defined earlier in terms of clusters; it only considers merging and splitting clusters. In the definition from the introduction, a placement is also critical if a corner of the square coincides with a line. However, these placements are simply four translates of the input lines, so the rest of this section focuses on the harder critical placements as defined here.

### 2.1 Placement space

Let  $\mathcal{L}$  be a set of lines and denote by  $\mathcal{A}$  the arrangement of  $\mathcal{L}$ . For a placement of the square on  $\mathcal{A}$ , consider all cells that contain part of its boundary. To characterize all critical placements, we look at how the square can be “moved around” such that a given cell of  $\mathcal{A}$  contains an  $\varepsilon$ -segment throughout the motion. For instance, in Figure 2 on the left, we can move the square slightly left and right such that there is always an  $\varepsilon$ -segment in the same cell. We use the intuition of moving the square to argue that the critical placements can be characterized as a set of line segments, and then prove an upper bound on how many times these line segments can intersect.

**Definition 1.** *Let  $\mathcal{L}$  be a set of lines and  $\square_p$  be the boundary of an axis-aligned unit square whose center is denoted by  $p$ , and let  $\varepsilon > 0$  be a constant. A placement of  $\square_p$  (or  $p$ ) is an  $\varepsilon$ -placement if at least one connected component in  $\square_p \setminus \mathcal{L}$  has length exactly  $\varepsilon$ . An  $\varepsilon$ -segment is a connected component of  $\square_p \setminus \mathcal{L}$  with length exactly  $\varepsilon$ .*

Consider the example from the figure again. When moving  $p$  to the right we maintain the indicated  $\varepsilon$ -segment; this reduces  $p$ ’s movement to one degree of freedom. It can happen that when moving  $p$ , another  $\varepsilon$ -segment is created

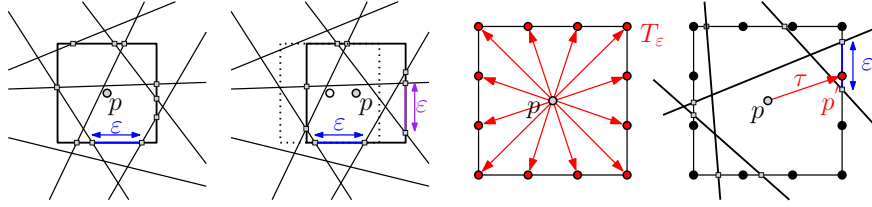


Figure 2: Left two: placement of square with one  $\varepsilon$ -segment and two  $\varepsilon$ -segments. Right two: the set of vectors  $T_\varepsilon$  and an  $\varepsilon$ -placement  $p$  with the vector in  $T_\varepsilon$  indicated.

in another cell. This means that this  $\varepsilon$ -placement gives rise to two  $\varepsilon$ -segments. We assume  $\mathcal{L}$  to be in general position such that no two  $\varepsilon$ -segments give  $p$  the same allowed movement (a condition that is met after perturbing the input). Therefore no more than two  $\varepsilon$ -segments will ever occur simultaneously.

Since  $\varepsilon$ -segments are part of  $\square_p$ , it is convenient to fix a point  $p'$  on  $\square_p$  and look at the movement of  $p'$  instead of  $p$ . Thus, by moving the square such that an  $\varepsilon$ -segment is maintained inside a cell  $c$  in  $\mathcal{A}$ , the point  $p'$  traces a curve. If we consider only those parts of this curve corresponding to placements where  $p'$  lies on the  $\varepsilon$ -segment, we observe that this subcurve is contained in  $c$ . For instance, in Figure 2 on the bottom right, the fixed point  $p'$  can be moved vertically ( $p$  and the square move accordingly) exactly between the intersection points with  $\mathcal{L}$ . To facilitate our analysis, we will choose a set of fixed points on  $\square_p$  such that any  $\varepsilon$ -segment will contain exactly one of these points. For ease of presentation we assume that  $\frac{1}{\varepsilon}$  is integer, so that we can place exactly  $\frac{1}{\varepsilon}$  fixed points with distance  $\varepsilon$  apart along  $\square_p$ . If it is not integer, we can pick fixed points such that an  $\varepsilon$ -segment contains one or two such points, in which case our analysis overcounts by a constant factor.

Since these points are fixed with respect to  $p$ , we define them in terms of *translation vectors*. We write  $\tau(X)$  for the translation by  $\tau$  of any object  $X$ . The inverse of  $\tau$  is denoted  $\tau^{-1}$ . Thus, in Figure 2,  $p' = \tau(p)$ .

**Definition 2.** Let  $\frac{1}{\varepsilon}$  be integer.  $T_\varepsilon$  is the set of  $\frac{1}{\varepsilon}$  vectors such that  $\square_p \setminus \{\tau(p) \mid \tau \in T_\varepsilon\}$  is a set of open line segments each of length  $\varepsilon$ , and  $T_\varepsilon$  includes all vectors  $\sigma$  such that  $\sigma(p)$  is a corner of  $\square_p$ .

Let  $c$  be a cell of  $\mathcal{A}$  and  $\tau \in T_\varepsilon$  a vector. We denote by  $S(c, \tau)$  the set of all critical placements  $p$  such that  $\tau(p)$  lies on an  $\varepsilon$ -segment in  $c$ . That is,  $\tau(S(c, \tau))$  is the set of curves traced out by  $\tau(p)$  under our previous interpretation of moving  $p'$ . We note that some cells are “too small” to contain an  $\varepsilon$ -segment, in which case  $S(c, \tau)$  is empty; other cells may contain up to two disconnected curves (see Figure 3). As shorthand notation, let  $S(\tau)$  denote the union of all  $S(c, \tau)$  over all cells in  $\mathcal{A}$ .

If  $\tau(p)$  is not a corner of  $\square_p$ , then  $\tau(S(c, \tau))$  coincides with (one<sup>1</sup> or) two

<sup>1</sup>Degenerate case where the width or height of  $c$  is exactly  $\varepsilon$ .

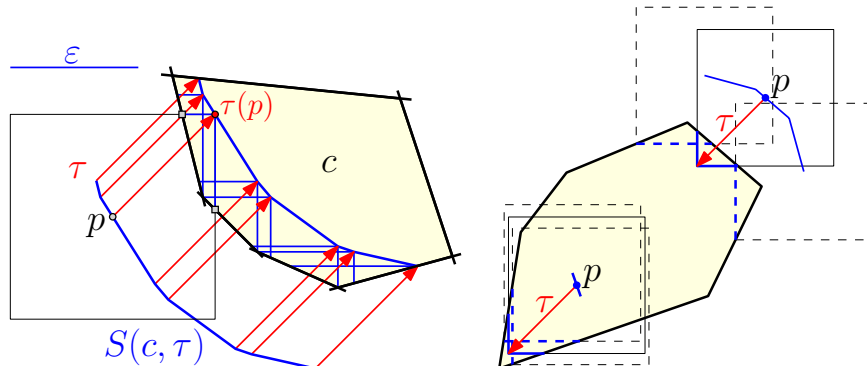


Figure 3: Left: construction of the curve  $S(c, \tau)$ . Right:  $S(c, \tau)$  can consist of multiple pieces.

parallel axis-aligned  $\varepsilon$ -segments contained in  $c$ . Therefore, the number of such segments is bounded by twice the number of cells in the arrangement of lines. If  $\tau(p)$  is a corner of  $\square_p$ , the shape of  $S(c, \tau)$  is more complex. This means that a similar bound on  $S(\tau)$  as before is not as easy to achieve. Figure 3 shows the construction of  $S(c, \tau)$  when  $\tau(p)$  is the upper right corner of  $\square_p$ . The curve inside  $c$  shows exactly how  $\tau(p)$  can be moved such that  $c$  contains an  $\varepsilon$ -segment containing  $\tau(p)$ . Translated by the inverse of  $\tau$ , we get the actual  $\varepsilon$ -placements of  $p$  corresponding to the particular cell  $c$  and vector  $\tau$ . Note that the figure also shows the only case when  $S(c, \tau)$  can be disconnected, i.e. when  $c$  contains an acute angle in the same “direction” as  $\tau$ . We will show that for the four corner vectors  $\tau$ , the curves in  $S(\tau)$  have properties that allow us to bound the complexity of the space of all  $\varepsilon$ -placements.

**Lemma 1.** *For  $\tau \in T_\varepsilon$  and a cell  $c$  in  $\mathcal{A}$ ,  $S(c, \tau)$  consists of at most two connected subsets of a piecewise linear and convex curve.*

*Proof.* If  $\tau(p)$  is not a corner point of  $\square_p$  then  $S(c, \tau)$  is either empty or consists of two single line segments.

Without loss of generality let  $\tau$  translate  $p$  to the upper right corner of  $\square_p$ . Because the result must hold for arbitrary  $\varepsilon$ , we define the following function  $f$  related to the notion of an  $\varepsilon$ -segment, but without being dependent on  $\varepsilon$ .

For a point  $a$ , let  $x_a$  be the distance to the closest line in  $\mathcal{L}$  left of  $a$ , and let  $y_a$  be the distance to the closest line below  $a$ ; define  $f(a) = x_a + y_a$ . Let  $f_c$  be  $f$  restricted to the points in a cell  $c$  of  $\mathcal{A}$ , and consider two points  $a$  and  $b$  inside  $c$ . For a point  $v$  halfway between  $a$  and  $b$ , we have  $x_v \geq (x_a + x_b)/2$  and  $y_v \geq (y_a + y_b)/2$  by the convexity of  $c$ . It follows that  $f_c(v) \geq (f_c(a) + f_c(b))/2$ . Thus,  $f_c$  is a concave function inside  $c$ , and taking the level set of a concave function yields a convex curve (possibly extending outside  $c$ ). Observe that the set of all points  $a \in c$  such that  $f(a) = \varepsilon$  is the same as  $S(c, \tau)$  and is a level set of  $f_c$ , so it is convex. Similarly,  $f_c$  is piecewise linear and hence, so is  $S(c, \tau)$ .

Since there are at most 4 points on the boundary of  $c$  where  $f_c$  has value  $\varepsilon$ ,  $S(c, \tau)$  consists of at most two connected components.  $\square$

## 2.2 Complexity

**Lemma 2.** *For all  $\tau \in T_\varepsilon$ ,  $S(\tau)$  consists of  $\mathcal{O}(n^2)$  line segments.*

*Proof.* For any  $\tau$  not corresponding to a corner,  $S(\tau)$  contains at most two horizontal or two vertical line segments for each cell of  $\mathcal{A}$ . Next, assume that  $\tau$  corresponds to a corner of  $\square_p$ , say, the upper right corner. Let  $c$  be a cell with  $k$  edges in  $\mathcal{A}$ , and let  $p$  be an  $\varepsilon$ -placement such that  $\tau(p) \in c$ . For a fixed  $y$ -coordinate of  $p$ , there are at most two  $\varepsilon$ -placements with  $\tau(p) \in c$  by convexity of the cell  $c$ . Furthermore, for each vertex of  $S(c, \tau)$ , the point  $\tau(p)$  aligns horizontally or vertically with a vertex of  $c$ . It follows that the complexity of  $S(c, \tau)$  is  $\mathcal{O}(k)$ . Summing over the complexities of all cells in the arrangement gives the bound.  $\square$

We are interested in the complexity of the placement space of  $\square_p$ , and it is therefore important to know how many times two sets of curves  $S(\tau)$  and  $S(\sigma)$  intersect, for  $\tau, \sigma \in T_\varepsilon$ . An intersection of these two sets corresponds to a placement of  $\square_p$  such that  $\tau(p)$  and  $\sigma(p)$  both lie on an  $\varepsilon$ -segment of  $\square_p$ . The following lemma shows that the number of intersections is bounded by the complexity of  $\mathcal{A}$ .

**Lemma 3.** *For distinct  $\tau, \sigma \in T_\varepsilon$ , the intersection of  $S(\tau)$  and  $S(\sigma)$  contains  $\mathcal{O}(n^2)$  points.*

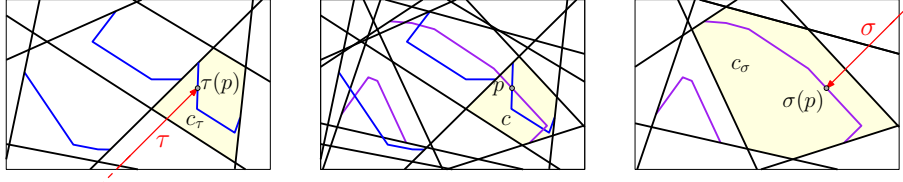


Figure 4: Example arrangement with  $\varepsilon$ -placements for two vectors  $\tau$  and  $\sigma$ . Left:  $\mathcal{L}$  and  $\tau(S(\tau))$ . Right:  $\mathcal{L}$  and  $\sigma(S(\sigma))$ . Center: Both combined under appropriate translations.

*Proof.* Let  $p$  be an  $\varepsilon$ -placement such that  $p \in S(\tau) \cap S(\sigma)$ , for distinct  $\tau, \sigma \in T_\varepsilon$ . Let  $\mathcal{A}' = \mathcal{A}(\tau^{-1}(\mathcal{L}) \cup \sigma^{-1}(\mathcal{L}))$  be the arrangement of two copies of  $\mathcal{L}$  translated by the inverses of  $\tau$  and  $\sigma$ . Let  $c$  denote the cell in  $\mathcal{A}'$  that contains  $p$ , and let  $c_\tau$  and  $c_\sigma$  denote the cells in  $\mathcal{A}$  that respectively contain  $\tau(p)$  and  $\sigma(p)$  (see Figure 4). Observe that  $c = \tau^{-1}(c_\tau) \cap \sigma^{-1}(c_\sigma)$ .

Let  $s(\tau)$  denote the line segment in  $S(c_\tau, \tau)$  on which  $p$  lies. Distinguish the following two cases for  $p$ : either at least one end point of  $s(\tau)$  or  $s(\sigma)$  lies in  $c$ ,

or none. In the first case, without loss of generality assume that one end point of  $s(\tau)$  lies in  $c$ . By Lemma 1,  $S(c_\sigma, \sigma)$  consists of at most two convex curves, hence  $s(\tau)$  intersects  $S(c_\sigma, \sigma)$  at most four times. Because  $S(c_\sigma, \sigma)$  is the only part of  $S(\sigma)$  intersecting  $c$ , it also holds that the part of  $s(\tau)$  inside  $c$  intersects  $S(\sigma)$  only four times. Thus, all intersections of this kind are bounded by the number of vertices in  $S(\tau)$ , which by Lemma 2 is  $\mathcal{O}(n^2)$ .

In the other case, no end point of  $s(\tau)$  or  $s(\sigma)$  lies in  $c$ . Consider an edge  $e$  of  $c$  in  $\mathcal{A}'$  that intersects  $s(\tau)$ . Since  $e$  intersects  $s(\tau)$  it must be an edge of  $c_\sigma$ . Therefore, it intersects  $S(c_\tau, \tau)$  and thus  $S(\tau)$  at most twice.

Therefore, the number of intersections of this kind is bounded by the number of edges in  $\mathcal{A}'$ , which is  $\mathcal{O}(n^2)$ .  $\square$

Since there are  $\mathcal{O}(\frac{1}{\varepsilon^2})$  pairs of vectors in  $T_\varepsilon$ , we immediately obtain:

**Theorem 4.** *Given a set of  $n$  lines  $\mathcal{L}$ , the arrangement of  $\varepsilon$ -placements of a unit square has complexity  $\mathcal{O}(\frac{n^2}{\varepsilon^2})$ .*

### 3 Extensions

Here we describe how to extend the general approach when placing a square on lines to different, more realistic settings. We describe what happens when we replace the line arrangement with an arrangement of line *segments*, denoted  $\mathcal{A}(\mathcal{T})$ . Additionally, we describe how to adapt the approach when placing a circle rather than a square.

#### 3.1 Square on line segments

The definition of an  $\varepsilon$ -placement remains the same when we place a square amidst line segments, and again we study the complexity of the  $\varepsilon$ -placement space. In a nonconvex cell  $c$  of the arrangement, the curve  $S(c, \tau)$  is no longer a piecewise linear convex curve, but it can have several disconnected pieces; see Figure 5.

**Theorem 5.** *Given a set of line segments  $\mathcal{T}$ , the arrangement of  $\varepsilon$ -placements of a unit square has complexity  $\mathcal{O}(\frac{n^2}{\varepsilon^2})$ .*

*Proof.* For analysis purposes, we imagine that any nonconvex cell  $c$  of  $\mathcal{A}(\mathcal{T})$  is partitioned into convex subcells as follows. For each endpoint of a line segment, shoot a ray in each orthogonal direction inside the cell until it hits another line segment. Use these rays to subdivide the cell into convex subcells, see Figure 5. If a cell  $c$  has  $k$  endpoints in its boundary, it is partitioned into  $\mathcal{O}(k^2)$  subcells. Within each subcell of  $c$ , the curve  $S(c, \tau)$  has the same properties as in a convex cell, and hence we can apply the same arguments as before. The total number

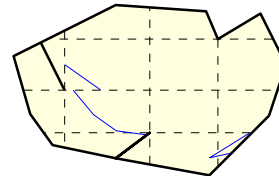


Figure 5: A nonconvex cell  $c$ , its subdivision, and the (blue) curves  $S(c, \tau)$ .



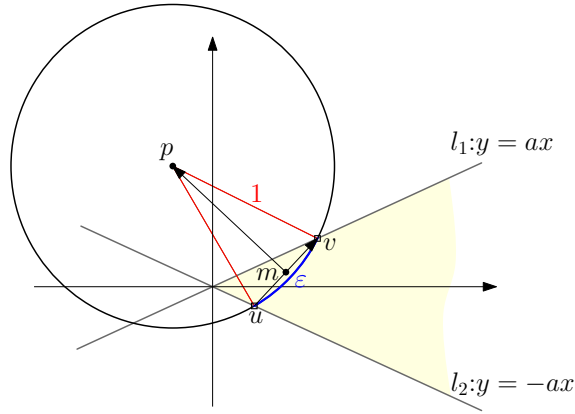


Figure 6: Two lines  $\ell_1, \ell_2 \in \mathcal{L}$  and an  $\varepsilon$ -placement of  $p$  such that the corresponding  $\varepsilon$ -segment is in the highlighted region.

of endpoints is  $\mathcal{O}(n)$ , and hence the total number of subcells analyzed in the arrangement  $\mathcal{A}(\mathcal{T})$  is  $\mathcal{O}(n^2)$ . The bound follows.  $\square$

### 3.2 Circle on lines

We now place a unit circle,  $\mathcal{O}_p$ , on  $\mathcal{L}$ , rather than a square. We again define a set of translation vectors  $T_\varepsilon$  to points  $\varepsilon$ -spaced on the boundary of  $\mathcal{O}_p$ , this time assuming  $2\pi/\varepsilon$  is an integer. Consider cell  $c$  in the arrangement  $\mathcal{A}$ . The segments in  $S(c, \tau)$  are no longer straight, but are generally elliptic.

**Lemma 6.** *The  $\varepsilon$ -placements of a circle such that the corresponding  $\varepsilon$ -segments lie in  $c$  form a collection of segments and elliptic arcs.*

*Proof.* Consider two lines  $\ell_1$  and  $\ell_2 \in \mathcal{L}$ . Consider a coordinate system with the origin in the intersection point of  $\ell_1$  and  $\ell_2$  and oriented such that  $\ell_1$  has equation  $y = ax$ , and  $\ell_2$  has equation  $y = -ax$ . Consider an  $\varepsilon$ -placement of a unit circle such that the  $\varepsilon$ -segment is in the I- and IV-quadrants below line  $\ell_1$  and above  $\ell_2$  as in Figure 6.

Let  $u$  be the intersection point of the circle and line  $\ell_2$ ,  $v$  be the intersection of the circle and  $\ell_1$ , and  $m$  be a middle point between  $u$  and  $v$ . From the following equations:

$$\begin{aligned} |uv|^2 &= 2 - 2 \cos \varepsilon, \\ |up| &= 1, \\ |vp| &= 1, \\ \vec{up} \times \vec{mp} &= |up| \cdot |mp|, \end{aligned}$$

we can derive that the segment of the  $\varepsilon$ -placement curve that corresponds to all such  $\varepsilon$ -segments that have one end point on line  $\ell_1$  and another end on  $\ell_2$  is an

arc of an ellipse given by the following equation (refer to Figure 7):

$$\frac{a^2 x^2}{\left(\sin\left(\frac{\varepsilon}{2}\right) - a \cos\left(\frac{\varepsilon}{2}\right)\right)^2} + \frac{y^2}{\left(a \sin\left(\frac{\varepsilon}{2}\right) + \cos\left(\frac{\varepsilon}{2}\right)\right)^2} = 1.$$

□

From this analysis we see that a sequence of adjacent arcs is not necessarily convex: when  $a > \tan \frac{\varepsilon}{2}$  the point  $p$  is tracing the left arc of an ellipse (the convex part), when  $a < \tan \frac{\varepsilon}{2}$  point  $p$  is tracing the right arc of an ellipse (the concave part), and when  $a = \tan \frac{\varepsilon}{2}$  the ellipse degenerates to a straight segment. Nonetheless, we can show that this cannot happen arbitrarily often.

**Lemma 7.** *For  $\varepsilon < 1$ ,  $S(c, \tau)$  consists of a constant number of piece-wise elliptic convex curves.*

*Proof.* Consider the sequence of angles  $a_1, a_2, \dots, a_k$  between consecutive edges of  $c$ . Because  $c$  is convex, we know that  $\sum_{i=1}^k (\pi - a_i) = 2\pi$ . For sufficiently small  $\varepsilon$  ( $\varepsilon < 1$  suffices), this implies that no more than two angles  $a_i$  can be smaller than  $\pi - \tan \frac{\varepsilon}{2}$ .

By the previous lemma, these at most two angles correspond to concave elliptic arcs, which we report as separate curves. The remainder of the arcs and segments formed by boundary of  $c$  is subdivided by these gaps into at most two piece-wise elliptic convex curves.

As in the case for squares (refer to Lemma 1), at most two disconnected curves, each of which can be decomposed into at most four convex subcurves, may exist for any given  $\tau$ . To see this, consider any line with direction  $\tau$  that intersects  $c$  in a segment  $s$ . As we slide  $\tau(p)$  along  $s$ , the length of the boundary piece of  $\bigcirc_p$  containing the point  $\tau(p)$  in  $c$  changes as a concave function. Hence, the piece is an  $\varepsilon$ -segment at most twice. (Note that, in contrast to the square case, it is essential that  $s$  has orientation  $\tau$ .) □

To bound the complexity of the placement space, it is sufficient to bound the number of intersection points between  $S(\tau)$  and  $S(\sigma)$ .

**Lemma 8.** *For distinct  $\tau, \sigma \in T_\varepsilon$ , the intersection of  $S(\tau)$  and  $S(\sigma)$  contains  $\mathcal{O}(n^2)$  points.*

*Proof.* Let  $\mathcal{A}'$  be the overlay of  $\tau^{-1}(\mathcal{A})$  and  $\sigma^{-1}(\mathcal{A})$ : the arrangement of two copies of  $\mathcal{L}$  translated by the inverses of  $\tau$  and  $\sigma$ . Let  $c$  be a cell in  $\mathcal{A}'$ , let  $c_\tau$  and  $c_\sigma$  denote the cells in  $\mathcal{A}$  which respectively contain  $\tau(p)$  and  $\sigma(p)$ . Observe that  $c = \tau^{-1}(c_\tau) \cap \sigma^{-1}(c_\sigma)$ .

Now, by Lemma 7,  $c_\tau$  and  $c_\sigma$  both contain a constant number of (pieces of) convex curves. Each curve piece itself may consist of many elliptic arcs. We observe that a convex curve, consisting of  $k$  elliptic arcs, and a second convex curve, consisting of  $h$  elliptic arcs, may cause at most  $\mathcal{O}(k + h)$  intersections. We thus charge the intersections to the pieces of  $S(\tau)$  or  $S(\sigma)$ , and note that each piece is charged at most constantly often. □

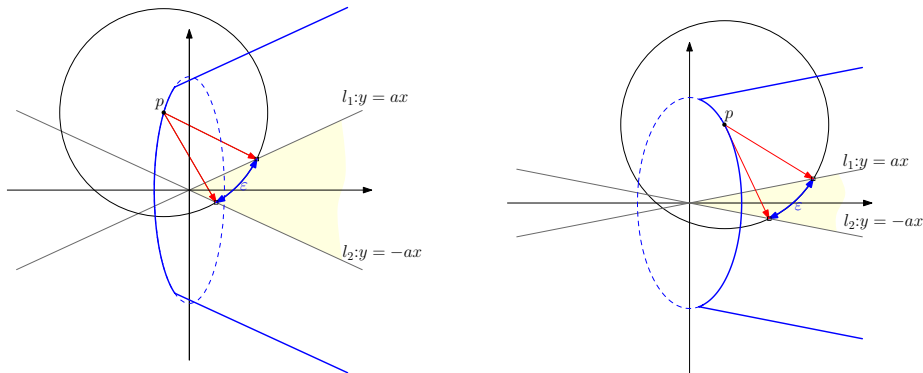


Figure 7: Two lines  $\ell_1, \ell_2 \in \mathcal{L}$  and the curve (blue) that  $p$  traces over all  $\varepsilon$ -placements such that the corresponding  $\varepsilon$ -segments are within the highlighted region.

As in the case of squares, we have  $\mathcal{O}(1/\varepsilon^2)$  distinct translation vectors, so we can bound the total complexity by  $\mathcal{O}(n^2/\varepsilon^2)$ .

**Theorem 9.** *Given a set of  $n$  lines  $\mathcal{L}$ , the arrangement of  $\varepsilon$ -placements of a unit circle has complexity  $\mathcal{O}(\frac{n^2}{\varepsilon^2})$ .*

## 4 Lower bounds

By a worst case construction we show a tight bound on the complexity of the placement space of a square (the construction for placing a circle is analogous). Consider a (slightly tilted) square cell of size  $\mathcal{O}(\varepsilon)$ . The curve traced by  $p$  of all  $\varepsilon$ -placements of a unit side square such that an  $\varepsilon$ -segment is inside the cell is shown in Figure 8. It forms an “offset” curve around the cell of width  $\approx 2$ . Place  $\frac{n}{2}$  almost horizontal lines and  $\frac{n}{2}$  almost vertical lines to form a grid with cells of size  $\mathcal{O}(\varepsilon)$  (consider all lines slightly tilted). Figure 8 shows this construction.

A trivial upper bound<sup>2</sup> on the complexity of the placement space is  $\mathcal{O}(n^4)$ , which by our construction is worst case tight if  $\varepsilon = \mathcal{O}(\frac{1}{n})$ . If  $\varepsilon$  is bounded from below, the parametrized complexity of the arrangement is also worst-case tight.

**Corollary 10.** *Given a set of  $n$  lines  $\mathcal{L}$  and a parameter  $\varepsilon = \Omega(\frac{1}{n})$ , the arrangement of  $\varepsilon$ -placements has a worst case complexity  $\Omega(\frac{n^2}{\varepsilon^2})$ .*

<sup>2</sup>If  $\varepsilon$  is arbitrarily small, the arrangement of the placement space is a collection of  $n^2$  unit squares which can intersect pairwise.

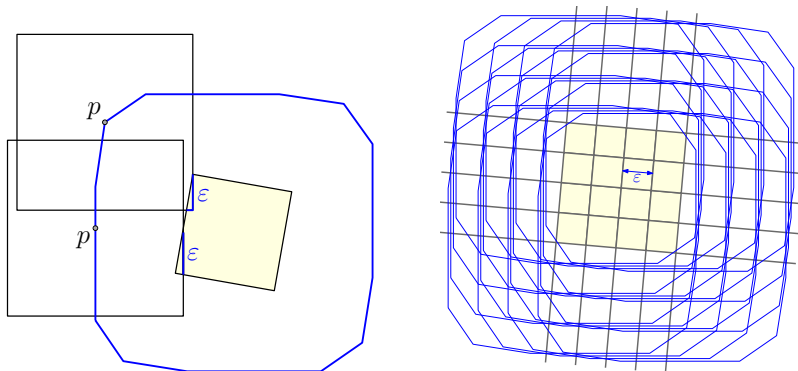


Figure 8: Lower bound construction for squares. (left) Point  $p$  traces an “offset” around square cell. (right) A  $\frac{n}{2} \times \frac{n}{2}$  grid formed by lines from  $\mathcal{L}$ . Each cell has size  $\mathcal{O}(\epsilon)$ .

## 5 Computation

To compute the critical placement of a square on  $n$  lines  $\mathcal{L}$ , we first compute  $\mathcal{A}(\mathcal{L})$  in  $\mathcal{O}(n^2)$  time. Next, we traverse all cells  $c$  in  $\mathcal{A}$  and calculate  $\cup_{\tau \in T_\epsilon} S(c, \tau)$ . We do this by scanning the boundary of  $c$ , taking linear time in the complexity of  $c$ ; therefore, in total we spend  $\mathcal{O}(n^2)$  time finding all  $\mathcal{O}(n^2)$  critical segments. Given this set of line segments we can compute their arrangement with standard techniques [3, 7, 12], giving an  $\mathcal{O}(n^2 \log n + k)$  running time for output size  $k$ .

**Theorem 11.** *An arrangement of critical placements of a unit axis-aligned square among a set of  $n$  lines  $\mathcal{L}$  can be computed in  $\mathcal{O}(n^2 \log n + k)$  time, where  $k$  is the output size.*

**Extensions.** When placing a square over line segments  $\mathcal{T}$ , we first compute  $\mathcal{A}(\mathcal{T})$ . With this arrangement we can find the subdivisions as in Figure 5, and compute  $\cup_{\tau \in T_\epsilon} S(c, \tau)$  for subdivisions  $c$  by scanning their convex boundary.

**Theorem 12.** *An arrangement of critical placements of a unit axis-aligned square among a set of  $n$  line segments  $\mathcal{T}$  can be computed in  $\mathcal{O}(n^2 \log n + k)$  time, where  $k$  is the output size.*

For placing a circle on lines, we compute  $\cup_{\tau \in T_\epsilon} S(c, \tau)$  per cell  $c$  of  $\mathcal{A}(\mathcal{L})$ , spending  $\mathcal{O}(n^2)$  time to find all arcs. Given a set of  $n$  Jordan arcs, the complexity of building an arrangement is  $\mathcal{O}((n + k) \log n)$  [7]. Therefore, in our case we get:

**Theorem 13.** *An arrangement of critical placements of a unit circle among a set of  $n$  lines  $\mathcal{L}$  can be computed in  $\mathcal{O}((n^2 + k) \log n)$  time, where  $k$  is the output size.*

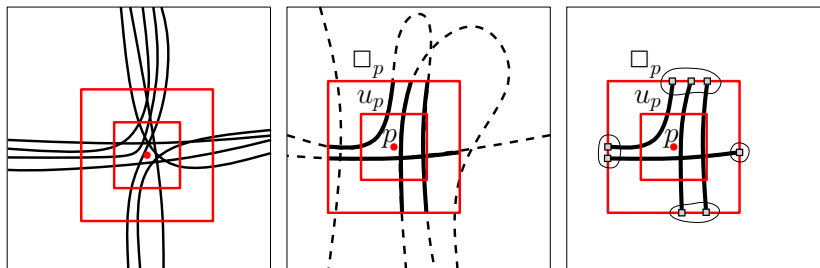


Figure 9: Left: Junction-like point with its two concentric squares. Middle: Solid subtrajectories are salient for the junction. Right:  $\varepsilon$ -clustering of the salient subtrajectory endpoints, revealing four arms.

## 6 Applications to junction detection

While junctions are normally features of (road) networks, our objective is to extract junctions purely based on trajectory data. This allows us to identify regions that serve as junctions in open spaces like city squares, or to identify places where animals pass by and choose one or the other direction.

We describe basic properties of a junction, which motivate corresponding definitions. Our approach is to treat any point in the plane as a potential junction; we define when a point is *junction-like* depending on the trajectories in its neighborhood. We show that the arrangement from Section 2 can be used to partition the plane into regions of points that are similarly junction-like, that is to say they have the same basic properties. In the appendix we present—as a proof of concept—the output of a prototype implementation run on idealized data.

**Properties of a junction.** A junction is a region where trajectories enter and leave in a limited number of directions or routes, called *arms* of the junction. While the moving entities initiate a direction to leave the junction in the core area, the arms are only “visible” somewhat further away from the core; see Figure 9. This motivates the use of two concentric squares to decide whether a point is junction-like. The way in which the trajectories enter and leave the two squares determines whether the point is junction-like and how significant that junction is. Since we can scale the plane with all trajectories, we can assume that the larger square is unit-size. We let the smaller square have side length  $\frac{1}{2}$ .

For a point  $p$  in the plane, let  $u_p$  and  $\square_p$  denote the boundaries of squares with side length  $\frac{1}{2}$  and 1, respectively, both centered at  $p$ .

**Definition 3.** A (sub)trajectory  $T$  is salient for a point  $p$  if it lies completely inside  $\square_p$ , it intersects  $u_p$ , and its endpoints lie on  $\square_p$ .

From a set  $\mathcal{T}$  of trajectories we consider all subtrajectories that are salient; see Figure 9. Such subtrajectories should be maximal: their endpoints must

be actual crossings with the boundary of  $\square_p$  and not just contacts. A single trajectory in  $\mathcal{T}$  can have multiple salient subtrajectories.

**Definition 4.** *Given a set of points  $Q$  on the boundary of a square  $\square_p$  and a constant  $\varepsilon > 0$ , an  $\varepsilon$ -clustering is a partitioning of  $Q$  such that for any two points  $q, q' \in Q$  belonging to the same cluster, there exists a sequence of points  $q = q_1, \dots, q_j = q'$ , all in  $Q$ , such that  $q_i$  and  $q_{i+1}$  for  $1 \leq i < j$  have distance at most  $\varepsilon$  along  $\square_p$ .*

**Definition 5.** *Let  $p$  be a point in the plane, let  $\mathcal{T}$  be a set of trajectories, let  $\varepsilon > 0$ , and let  $Q_p(\mathcal{T})$  be the set of endpoints of all salient subtrajectories from  $\mathcal{T}$ . A point  $p$  is junction-like if an  $\varepsilon$ -clustering of  $Q_p(\mathcal{T})$  has at least three clusters.*

When we move the point  $p$  over the plane, these clusters can grow, shrink, merge and split. A cluster may for instance split when two consecutive intersection points from  $Q_p(\mathcal{T})$  become more separated than  $\varepsilon$ . A cluster may shrink because a subtrajectory is no longer salient, which then may also cause a split of a cluster.

It should be clear that we can compute a subdivision of the plane into maximal cells where the  $\varepsilon$ -clustering is the same. It should also be clear that the theory of Section 2 discusses a simplified version of this problem where salience is not considered.

**Implementation and results.** There are many ways to obtain junctions from the junction-like points and their  $\varepsilon$ -clusterings. There are also many ways to attach a significance to a junction, based on the number of clusters and their sizes. Moreover, we may want to distinguish between *real junctions* and *crossings*, where a crossing is a junction-like region with four arms where all trajectories go straight, and a real junction has several different splits and merges of trajectories over the clusters. For both types we can define their significance in various ways; see [16] for further discussions.

Figure 10 shows some results of an implementation that samples points from a regular square grid and evaluates the significance of a junction according to such a measure. Junction-like points are indicated by a colored grid cell, which contains a measured value. What can be observed is that an area around each junction contains several junction-like points with the same measure, and that the measure drops quickly towards the edge of this region. It is easy to convert the output so that for each group of junction-like cells, only one junction is reported. Figure 11 shows that this type of definition can indeed identify junctions and assign a significance in a reasonable manner.

## 7 Conclusions and further research

We analyzed the complexity of the placement space of a unit square or circle in an arrangement of lines or line segments, a problem that lies at the core of junction detection in trajectory analysis. Our results include upper bounds that improve on the naive  $O(n^4)$  bound by showing that two of the linear factors in

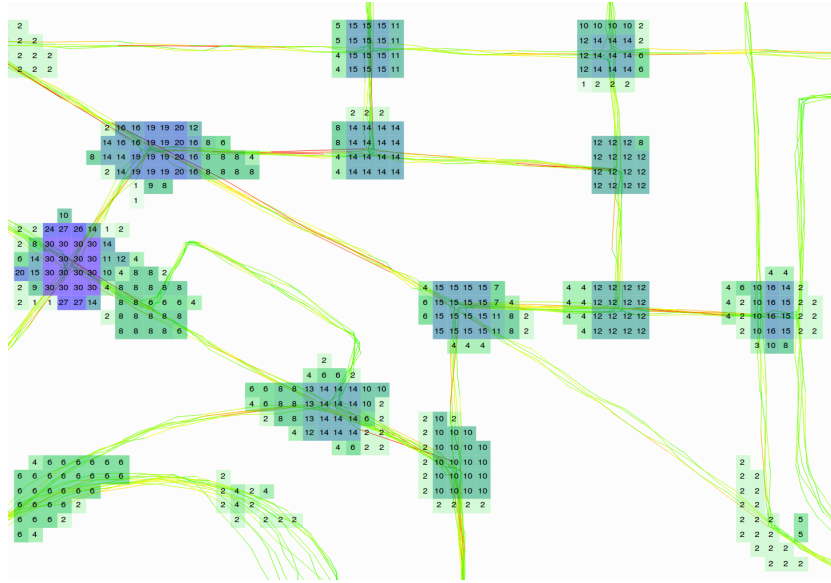


Figure 10: Junction-like points indicated by colored grid cells, darker cells contain a higher measure.

fact only need to depend on  $1/\varepsilon$ . Increasing the number of trajectories is not a reason to increase the granularity of the clustering, so in practice we expect  $1/\varepsilon$  to be much smaller than  $n$ , if not constant. The resulting  $\Theta(n^2/\varepsilon^2)$  bound is tight in the worst case. The combinatorial problem is interesting from the theoretical perspective because it combines arrangements with distances in the arrangement.

A prototype implementation applies the result to junction detection. While the implementation demonstrates the value of the approach and indicates that the time complexity is not prohibitive, several simplifications of the problem have been made and no extensive experiments have been performed. In a follow-up study, it would be interesting to augment the implementation with circles or different shapes, and to run the algorithm on real-world trajectory data from various sources.

From a theoretical perspective, it would be interesting to explore further extensions of the approach (i.e. curved trajectories, trajectories embedded in higher-dimensional spaces), or apply it to other distance based arrangements.

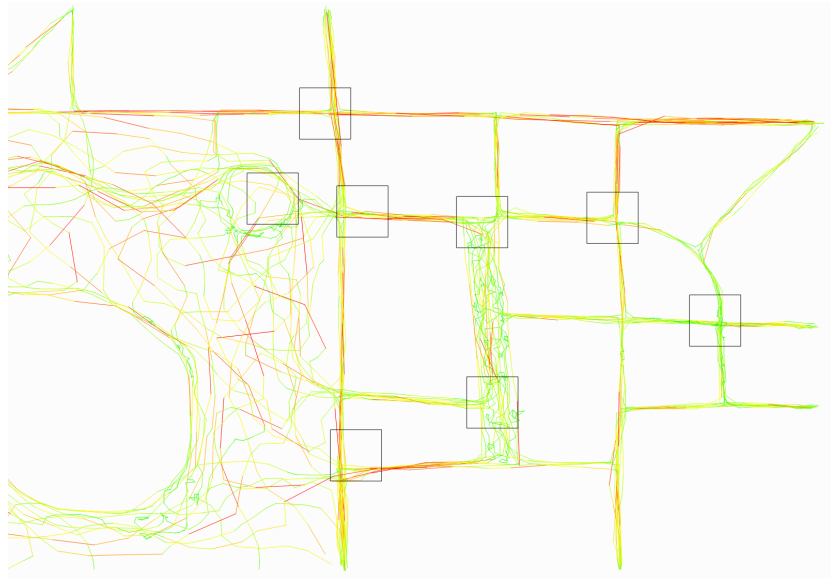


Figure 11: Locations of  $u_p$  indicated for the eight most significant junction-like points.

## Acknowledgments

M.L. is supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123. MADALGO Center for Massive Data Algorithmics is supported in part by the Danish National Research Foundation grant DNRF 84.



## References

- [1] M. Benkert, B. Djordjevic, J. Gudmundsson, and T. Wolle. Finding popular places. *Int. J. Comput. Geometry Appl.*, 20(1):19–42, 2010.
- [2] K. Buchin, M. Buchin, M. van Kreveld, M. Löffler, J. Luo, and R. I. Silveira. Processing aggregated data: the location of clusters in health data. *GeoInformatica*, 16(3):497–521, 2012.
- [3] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM (JACM)*, 39(1):1–54, 1992.
- [4] J. Gudmundsson, P. Laube, and T. Wolle. Movement patterns in spatio-temporal data. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*, pages 726–732. Springer, 2008.
- [5] J. Gudmundsson and M. van Kreveld. Computing longest duration flocks in trajectory data. In *14th ACM International Symposium on Geographic Information Systems*, pages 35–42, 2006.
- [6] J. Gudmundsson, M. van Kreveld, and F. Staals. Algorithms for hotspot computation on trajectory data. In *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 134–143, 2013.
- [7] D. Halperin. Arrangements. In J. E. Goodman and J. O’Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 529–562. CRC Press LLC, Boca Raton, FL, 2004.
- [8] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing circle. *Algorithmica*, 41(3):147–157, 2005.
- [9] H. Jeung, M. Yiu, and C. Jensen. Trajectory pattern mining. In Y. Zheng and X. Zhou, editors, *Computing with Spatial Trajectories*, pages 143–177. Springer, 2011.
- [10] B. Moreno, V.C. Times, C. Renso, and V. Bogorny. Looking inside the stops of trajectories of moving objects. In *Proc. XI Brazilian Symposium on Geoinformatics*, pages 9–20. MCT/INPE, 2010.
- [11] D. M. Mount, R. Silverman, and A. Y. Wu. On the area of overlap of translated polygons. *Computer Vision and Image Understanding*, 64(1):53–61, 1996.
- [12] K. Mulmuley. *Computational geometry - an introduction through randomized algorithms*. Prentice Hall, 1994.
- [13] S. Openshaw, M. Charlton, C. Wymer, and A. Craft. A mark 1 geographical analysis machine for the automated analysis of point data sets. *International Journal of Geographical Information System*, 1(4):335–358, 1987.
- [14] A.T. Palma, V. Bogorny, B. Kuijpers, and L. Otávio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proc. 2008 ACM Symposium on Applied Computing*, pages 863–868, 2008.
- [15] S. Tiwari and S. Kaushik. Mining popular places in a geo-spatial region based on GPS data using semantic information. In *Proc. 8th Workshop on Databases in Networked Information Systems*, volume 7813 of *LNCIS*, pages 262–276, 2013.
- [16] I. van Duijn. Pattern extraction in trajectories and its use in enriching visualisations. Master’s thesis, Department of Information and Computing Sciences, Utrecht University, 2014. <http://dspace.library.uu.nl/handle/1874/294075>.

- [17] S. van Hagen and M. van Kreveld. Placing text boxes on graphs. In *Graph Drawing, 16th International Symposium*, volume 5417 of *Lecture Notes in Computer Science*, pages 284–295. Springer, 2008.
- [18] M. van Kreveld, É. Schramm, and A. Wolff. Algorithms for the placement of diagrams on maps. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 222–231. ACM, 2004.