

# Extremal Point Queries with Lines and Line Segments and Related Problems<sup>1</sup>

Ovidiu Daescu<sup>2</sup> Robert Serfling

*Department of Computer Science, University of Texas at Dallas,  
Richardson, TX 75080, USA*

*Department of Mathematical Sciences, University of Texas at Dallas,  
Richardson, TX 75080, USA*

---

## Abstract

We address a number of extremal point query problems when  $P$  is a set of  $n$  points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, including the computation of the farthest point from a query line and the computation of the farthest point from each of the lines spanned by the points in  $P$ . In  $\mathbb{R}^3$ , we give a data structure of size  $O(n^{1+\epsilon})$ , that can be constructed in  $O(n^{1+\epsilon})$  time and can report the farthest point of  $P$  from a query line segment in  $O(n^{2/3+\epsilon})$  time, where  $\epsilon > 0$  is an arbitrarily small constant. Applications of our results also include: (1) Sub-cubic time algorithms for fitting a polygonal chain through an indexed set of points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, and (2) A sub-quadratic time and space algorithm that, given  $P$  and an anchor point  $q$ , computes the minimum (maximum) area triangle defined by  $q$  with  $P \setminus \{q\}$ .

*Key words:* Algorithm, computational geometry, segment, query, farthest point.

---

## 1 Introduction

Approximating point sets by simple geometric objects is a major topic in geometric optimization. While such problems are well studied in the plane ( $\mathbb{R}^2$ ), there are few results in higher dimensions, most related to computing simple geometric objects (smallest ball, minimum radius cylinder, etc.) enclosing a set of points. Computing the smallest enclosing cylinder (ball, cylindrical shell, etc.) is a fundamental problem in data analysis, computational

---

<sup>1</sup> This work was supported in part by the NSF award CCF-0430366.

<sup>2</sup> Corresponding author. Phone: 972-883-4196. Fax: 972-883-2349. Email: daescu@utdallas.edu.

metrology and data compression; see [5, 9] and the references within. For example, in data analysis, for a given data set  $P$  of  $n$  points in  $\mathbb{R}^d$ , one may want to fit a line  $L$  through  $P$  that minimizes  $\max\{d(L, p) : p \in P\}$ , where  $d(L, p) = \min\{d(q, p) : q \in L\}$  and  $d(q, p)$  denotes the Euclidean distance between the points  $q$  and  $p$ . This is equivalent to finding the smallest radius cylinder enclosing  $P$ .

In some cases, a line  $L$  may be provided, corresponding to some hypothesis on the data set, and one would like to compute  $\max\{d(L, p) : p \in P\}$  or  $\min\{d(L, p) : p \in P\}$  (the farthest or closest point from the query line  $L$ ), to validate or invalidate the hypothesis. In other problems, it is useful to index the points in the input set  $P$ . In the indexed version, for example, the index of a point may give an order on the time at which the data was collected, thus “weakly” encoding a  $(d + 1)$ -dimension of the data set. Then, one could make queries *in the past* to extract some features of the data set, such as computing the extent of a subset of  $P$  collected in a given time frame. Some other applications of these scenarios include data compression, computational biology (in modeling neurons and molecules) and face recognition. For example, in [24], Li and Lu propose a novel classification method for face recognition that is based on computing the nearest feature line to a query point. A similar approach, when used to identify individuals in crowds, would result in computing closest points from query lines (a line would correspond to two distinct prototype feature points of a *query* individual).

In this paper we discuss a number of query problems on computing extremal points in a point set  $P \in \mathbb{R}^d$ ,  $d \geq 3$  a constant. In some of these problems the queries are known in advance (off-line queries) while in the others the queries are given on-line. All queries are related to measuring the extent of  $P$ . One of our main results is a data structure for answering farthest point queries for line segments in  $\mathbb{R}^3$ . The solution for answering farthest point from line segment queries relies on a solution for the following problem.

**Farthest Point From Line (FPFL).** Preprocess a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , where  $d \geq 3$  is a constant, such that given a query line  $L$  one can efficiently report the farthest point of  $P$  from  $L$ .

We also show how to use our solution to this problem to solve the following problem on indexed points.

**Farthest Point - Indexed Set (FPIS).** Given a set of indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ , where  $d \geq 3$  is a constant, for each line  $L(p_i, p_j)$  defined by two points  $p_i, p_j \in P$ ,  $1 \leq i < j \leq n$ , find the farthest point  $p_k \in P$  such that  $i < k < j$ .

Similarly, one may want to evaluate the extent of a data set by computing the minimum-width cylindrical shell with central axis  $L$  enclosing the data set  $P$ ,

where a cylindrical shell is the region enclosed between two co-axial cylinders, and the width of a cylindrical shell is defined as the absolute value of the difference of the radii of the two cylinders. If no axis is specified, then the minimum-width cylindrical shell for a set of  $n$  points in  $\mathbb{R}^3$  can be computed in  $O(n^5)$  time [2]. However, if more complex geometric objects are sought to capture the features of  $P$ , the problem may become significantly more difficult. For example, one may want to fit a polygonal chain through  $P$ , with the condition that the vertices of the polygonal chain are a subset of the points in  $P$  and such that it approximates the extent of  $P$  based on some given measure. In the indexed version, with the vertices of the polygonal chain an ordered subset of  $P$ , the problem is equivalent to the polygonal chain simplification problem [22,36], for which the known solutions in  $\mathbb{R}^d$ ,  $d \geq 3$ , with some exceptions [7], are brute force, cubic or super-cubic time algorithms. The results in [7] achieve sub-cubic time but only for one version of the problem, and do not generalize to other versions, including those in the same class (e.g., from min-# to min- $\epsilon$ , see Section 5).

As a first application of our results on extremal point queries, we prove that our solution to problem **FPIS** can be used to obtain algorithms, with similar time and space complexities as for **FPIS**, for fitting a polygonal chain through an indexed set of points. The resulting algorithms improve over previously known solutions in most cases and provide an unified approach. As a second application, we show how to use the solution for problem **FPFL** to obtain a sub-quadratic time and space algorithm for the following problem.

**Minimum (maximum) area anchored triangle (MAAT).** Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$  and an anchor point  $q$ , compute the minimum (maximum) area triangle defined by  $q$  and a pair of points in  $P$ .

The paper is organized as follows. In the remaining of this section we summarize our results and previous work. In Section 2 we briefly discuss the linearization technique, a key component in our solutions, and present our solution for the problem **FPFL**. In Section 3 we address farthest point from line segment queries in  $\mathbb{R}^3$ . The solution for problem **FPIS** is discussed in Section 4. Applications of our results for problems **FPIS** and **FPFL** to fit a polygonal chain through a set of indexed points and to find the minimum (maximum) area anchored triangle are presented in Section 5. We conclude the paper in Section 6.

## 1.1 Results

It is known that in  $\mathbb{R}^d$  the Euclidean distance between a point and a line, as a function of the line, admits a linearization into a space of dimension

$f(d) = O(d^2)$  [3] ( $f(d)$  is a constant that depends on  $d$ ; see the discussion in Section 2). Throughout the paper, the notation  $f(d)$  refers to the dimension of this linearization, where  $d$  is a constant. Our main results are as follows.

- Let  $n \leq s \leq n^{f(d)/2}$  be a parameter that controls the trade-off between the query time and the space and preprocessing time. We prove that a set  $P$  of  $n$  points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, can be preprocessed with  $O(s \cdot \log^{O(1)} n)$  space and time such that given a query line  $L$  the farthest point of  $P$  from  $L$  (and then the minimum enclosing cylinder of  $P$  with central axis  $L$ ) can be found in  $O(n \log n / s^{1/\lfloor f(d)/2 \rfloor})$  time, thus solving problem **FPFL**.
- In  $\mathbb{R}^3$ , we use the result for problem **FPFL** and simplicial partitions to construct a data structure of size  $o(n^{4/3})$  that can report the farthest point from a query line segment in  $O(n^{2/3+\epsilon})$  time, where  $\epsilon > 0$  is an arbitrarily small constant. The data structure can be constructed in  $o(n^{4/3})$  time. This result naturally extends to higher dimensions. Alternatively, using an improved data structure for **FPFL** in  $\mathbb{R}^3$ , that requires  $O(n^{1+\epsilon})$  preprocessing time and space and answers farthest point from line queries in  $O(n^{1/2+\epsilon})$  [35], we obtain a data structure with  $O(n^{1+\epsilon})$  preprocessing time and space that answers queries in  $O(n^{2/3+\epsilon})$  time. Some trade-offs are also possible. For example, using half-space range searching based on cuttings [4, 27] one could obtain  $O(n^{1/2+\epsilon})$  query time by increasing the preprocessing time and space to  $O(n^{3/2} \log^{O(1)} n)$ . Our results are the first known for answering farthest point queries with line segments in  $\mathbb{R}^3$ .
- We give two sub-cubic time algorithms for solving problem **FPIS**. The first algorithm requires  $O(n^{3-2/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  space, and the second one requires  $O(n^{3-3/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  time and space.
- We show how to preprocess a set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, in  $O(n^{\lfloor f(d)/2 \rfloor} \log^{O(1)} n)$  time and space such that for a query line  $L$  the farthest point  $p_k$  to  $L$  that is in a specified range  $\{p_i, p_{i+1}, \dots, p_j\}$ ,  $1 \leq i < k < j \leq n$ , can be reported in  $O(\log^2 n)$  time.
- We apply the result for problem **FPIS** to obtain an algorithm with time and space  $O(n^{3-3/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  that, given a set  $P$  of  $n$  indexed points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, computes a “fitting” polygonal chain  $P'$  whose vertices are an ordered subsequence of the points in  $P$ , for two fitting measures and two optimization criteria.
- We solve the minimum (maximum) area anchored triangle in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, with time and space  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$ . In  $\mathbb{R}^3$  the time and space requirements are  $O(n^{1.6} \log^{O(1)} n)$ .

We note that in some of the problems we address the queries are known in advance and thus a batching approach may be possible. However, we do not know how to efficiently apply batching for these problems.

Throughout the paper, the notation  $O^*(\cdot)$  is used to hide an  $O(\log^{O(1)} n)$  factor in the standard  $O(\cdot)$  notation.

## 1.2 Previous Work

Recently, there have been a significant body of work related to computing or approximating the extent of point sets. In [33], Yap *et.al.* presented an  $O(n^4 \log^{O(1)} n)$  time solution for computing the smallest enclosing cylinder of  $n$  points in  $\mathbb{R}^3$ . A near cubic solution in  $\mathbb{R}^3$  was later presented by Agarwal, Aronov and Sharir [1]. A general technique for approximating various descriptors of the extent of points is presented in [3]. Among other applications, their technique results in approximation algorithms for computing the smallest enclosing ball, cylinder and cylindrical shell, and for fitting spheres and cylinders through point sets. A few other related results for fitting a shape or for approximating the extent of a point set  $P$  in constant or in high dimension have appeared in [9, 18, 19]. Some of these algorithms are based on computing a *core set* of  $P$  that approximates well the extent of  $P$ .

There is also related work on query problems but mainly for the planar case. The problem of computing the closest point to a query line or line segment in the plane has been studied [8, 12, 23, 30, 31]. In particular, for  $n$  points in the plane, the closest point to a query line can be found in  $O(\log n)$  time with  $O(n^2)$  preprocessing time and space [12, 23] or in  $O(n^{0.695})$  time with  $O(n \log n)$  time and  $O(n)$  space preprocessing [30]. Using simplicial partitions, a data structure of size  $O(n \log n)$ , that answers closest point from line queries in  $O(n^{1/2+\epsilon})$  time, can be constructed in  $O(n^{1+\epsilon})$  time, for arbitrary  $\epsilon > 0$  [31]. For line segment queries on a set  $P$  of  $n$  points in  $\mathbb{R}^2$ , Bespamyatnikh and Snoeyink [8] presented a data structure of size  $O(n)$ , that can be constructed in  $O(n \log n)$  time, and can report the closest point from a query line segment that is **outside** the convex hull of  $P$  in  $O(\log n)$  time.

Recently, a data structure to answer farthest point from line segment queries in  $\mathbb{R}^2$  has been given in [14]. Their data structure has size  $O(n \log n)$ , can be constructed in  $O(n \log n)$  time and can report the farthest point from a query line segment in  $O(\log^2 n)$  time. Very recently, using a variant of the shallow cuttings [26] for the semialgebraic ranges and an ingenious analysis of the decomposition of the common intersection of cylinders, Sharir and Shaul [35] have presented a data structure for **FPFL** in  $\mathbb{R}^3$  with  $O(n^{1+\epsilon})$  preprocessing time and space and  $O(n^{1/2+\epsilon})$  query time. So far, it is unclear whether their data structure could be modified to admit trade-offs.

We discuss previous work related to fitting a polygonal chain through  $P$  in Section 5.

## 2 Linearization and Farthest Point from Line

Let  $\mathcal{F} = \{f_1, f_2, \dots, f_n\}$  be a collection of  $n$  totally defined  $d$ -variate polynomials of constant maximum degree  $b$ .  $\mathcal{F}$  admits a *linearization* of dimension  $k$  if, for some integer  $b > 0$ , there exists a  $(d + b)$ -variate polynomial

$$g(x, a) = \psi_0(a) + \psi_1(a)\varphi_1(x) + \psi_2(a)\varphi_2(x) + \dots + \psi_k(a)\varphi_k(x) + \varphi_{k+1}(x)$$

for  $x \in \mathbb{R}^d$  and  $a \in \mathbb{R}^b$ , such that  $f_i(x) = g(x, a_i)$ , for  $i = 1, 2, \dots, n$  and some  $a_i \in \mathbb{R}^b$  [34]. An algorithm for computing a linearization of smallest dimension is given in [4].

A line  $L$  in  $\mathbb{R}^d$  is uniquely defined by  $2d - 2$  parameters. Let  $d(L, p)$  denote the Euclidean distance from a point  $p$  to a line  $L$ , as a function of  $L$ . The linearization of the line-point distance was first addressed in [4]. In [3], it has been shown that  $d(L, p)$  admits a linearization into a space of dimension  $f(d) = O(d^2)$ . For example, in  $\mathbb{R}^3$  the dimension of the space is  $f(3) = 9$  [2]. Then, given a set  $P$  of  $n$  points and a line  $\mathcal{L}$ , a point  $p_i \in P$  is the farthest point from  $\mathcal{L}$  if  $d(\mathcal{L}, p_i) = \max\{d(\mathcal{L}, p_k) : k = 1, 2, \dots, n\}$ , that is  $d(L, p_i)$  is on the upper envelope of the set of functions  $\{d(L, p_1), d(L, p_2), \dots, d(L, p_n)\}$  at  $\mathcal{L}$ , where the *upper (lower) envelope* is defined as the pointwise maximum (resp., minimum) of the functions  $d(L, p_i)$ ,  $i = 1, 2, \dots, n$ .

Using linearization, we reduce the problem of finding the farthest point  $p \in P$  from a query line  $\mathcal{L}$ , defined by the tuple  $\xi = (\xi_1, \xi_2, \dots, \xi_{2d-2})$ , to a ray shooting query in a set of  $n$  hyperplanes: shoot a vertical ray in the negative direction of the last coordinate from the  $f(d)$ -dimensional point  $(\varphi_1(\xi), \varphi_2(\xi), \dots, \varphi_{f(d)-1}(\xi), \infty)$ , where  $\varphi_i(\xi)$ ,  $i = 1, 2, \dots, f(d) - 1$ , are functions resulting from the linearization and can be evaluated in  $O(1)$  time each. The point of  $P$  associated with the first hyperplane hit by this ray is the farthest point of  $P$  from the line  $\mathcal{L}$ . The set of  $n$  hyperplanes can be produced in  $O(n)$  time for constant  $d$ . With  $s$  such that  $n \leq s \leq n^{f(d)/2}$ , we obtain the following result for **FPFL**.

**Theorem 1** *A set  $P$  of  $n$  points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, can be preprocessed with  $O(s \cdot \log^{O(1)} n)$  space and time such that for a query line  $L$  the farthest point of  $P$  from  $L$ , and thus the smallest enclosing cylinder of  $P$ , with central axis  $L$ , can be found in  $O(n \log n / s^{\lfloor f(d)/2 \rfloor})$  time.*

**Proof.** Let  $f(d) = O(d^2)$  be the dimension of the space obtained through a linearization of  $d(L, p)$  as above. For the point set  $P$ , this gives a collection of  $n$  hyperplanes in  $f(d)$ -dimension. Then, we build a static data struc-

ture for answering ray-shooting queries [15, 28], that can answer a query in  $O(n \log n / s^{1/\lfloor f(d)/2 \rfloor})$  time with  $O(s \cdot \log^{O(1)} n)$  space and preprocessing time, where  $n \leq s \leq n^{f(d)/2}$  is a parameter that controls the trade-off between the query time and the space and preprocessing time. For example, using this ray-shooting data structure, we can answer a query in  $O(\log n)$  time with  $O^*(n^{\lfloor f(d)/2 \rfloor})$  space and time preprocessing. When  $P \subset \mathbb{R}^3$ ,  $f(d) = 9$  and we have  $O^*(n^4)$  preprocessing time.  $\square$

A cylindrical shell for  $P$  is a region enclosed by two co-axial cylinders containing  $P$ . The width of a cylindrical shell is defined as the absolute value of the difference of the radii of the two cylinders defining the shell. By a similar argument as above we have:

**Corollary 1** *A set  $P$  of  $n$  points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, can be preprocessed with space and time  $O(s \cdot \log^{O(1)} n)$  such that for a query line  $L$  the minimum-width cylindrical shell enclosing  $P$ , with central axis  $L$ , can be found in  $O(n \log n / s^{1/\lfloor f(d)/2 \rfloor})$  time.*

### 3 Farthest Points from Line Segments in $\mathbb{R}^3$

The distance from a line segment  $s$  to a point  $p$  is defined as  $d(s, p) = \min\{d(q, p) : q \in s\}$ , where  $d(q, p)$  denotes the Euclidean distance between the points  $q$  and  $p$ . Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , and a line segment  $s$ , the *farthest point* of  $P$  from  $s$  is defined as  $\max\{d(s, p) : p \in P\}$ . That is, the farthest point of  $P$  from  $s$  is  $\max\{\min\{d(q, p) : q \in s\} : p \in P\}$ . Note that this is different from the *farthest distance* from  $s$  to  $P$ , which is defined as  $\max\{\max\{d(q, p) : q \in s\} : p \in P\}$  [35].

Our solution for answering farthest point from line segment queries is based on simplicial partitions. A *simplicial partition* for a set  $P$  of  $n$  points in  $\mathbb{R}^3$  is a collection  $\Psi(P) = \{(P_1, t_1), (P_2, t_2), \dots, (P_r, t_r)\}$  of pairs, where the  $P_i$ 's are disjoint subsets of  $P$  whose union is  $P$ , and  $t_i$  is a tetrahedron containing  $P_i$ . For a given simplicial partition  $\Psi(P)$ , the *crossing number* of a plane  $h$  is the number of tetrahedrons of  $\Psi(P)$  that  $h$  intersects. The crossing number of  $\Psi(P)$  is the maximum crossing number over all possible planes. A simplicial partition  $\Psi(P)$  is *fine* if  $|P_i| \leq 2n/r$ , for every  $1 \leq i \leq r$ . Matousek [25] proved the following result.

**Theorem 2** [25] *For any given set  $P$  of  $n$   $d$ -dimensional points,  $d \geq 2$ , and a parameter  $r$ ,  $1 \leq r \leq n/2$ , a fine simplicial partition of size  $r$  and crossing number  $O(r^{1-1/d})$  exists. If  $r$  is a constant such a simplicial partition can be constructed in  $O(n)$  time with  $O(n)$  space.*

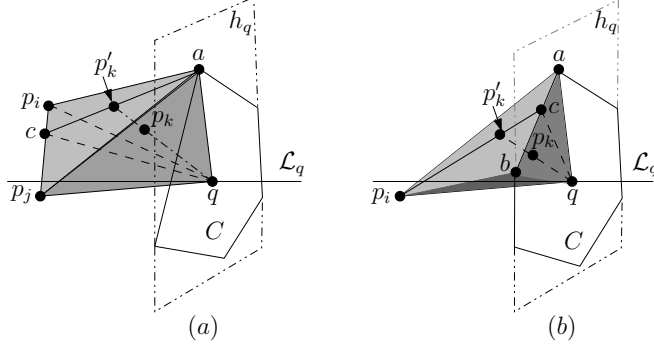


Fig. 1. The point  $p_k$  farthest from  $q$  is a vertex of  $CH(P, q)$  if  $|qp_k| > \Delta$ .

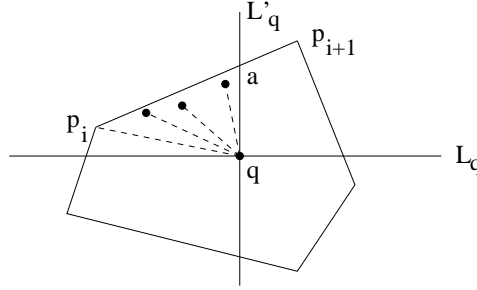


Fig. 2. In  $\mathbb{R}^2$ , when  $|qp_k| > \Delta$ , the farthest point  $p_k$  from  $q$  must be a vertex of the convex hull of  $P$ .

A simplicial partition can be used to construct an efficient data structure, called a *partition tree*: the root of the partition tree has  $r$  children, each associated with a simplex  $t_i$  in a simplicial partition  $\Psi(P)$ , as well as any secondary information related to the enclosed point set, and being the root of a recursively defined partition tree on the point set that belongs to this node.

**Lemma 1** *Let  $P$  be a set of  $n$  points in  $\mathbb{R}^3$  and let  $(q, \mathcal{L}_q, \Delta)$  be a triplet with  $q$  a point and  $\mathcal{L}_q$  a line through  $q$  such that all points in  $P$  are within distance  $\Delta$  from  $\mathcal{L}_q$ . Let  $h_q$  denote the plane orthogonal to  $\mathcal{L}_q$  at  $q$ . If there is a point  $p \in P$  such that  $|qp| > \Delta$ , then the point of  $P$  farthest from  $q$  in the halfspace defined by  $h_q$  and containing  $p$  is a vertex of the convex hull of  $P$ .*

**Proof.** Let  $P = \{p_1, p_2, \dots, p_n\}$ . Let  $CH(P)$  be the convex hull of  $P$ , let  $C = CH(P) \cap h_q$ , and consider the part  $CH(P, q)$  of  $CH(P)$  in one of the open halfspaces defined by  $h_q$ . For each tetrahedron  $qp_i p_k p_j$ , with  $p_i, p_k, p_j \in CH(P, q)$ , one of  $p_i, p_k$  or  $p_j$  is the farthest point to  $q$ . For each tetrahedron of the form  $qp_i p_j a$ , with  $p_i, p_j \in CH(P, q)$  and  $a$  being a vertex of the boundary  $\partial C$  of  $C$ , let  $p_k$  be the farthest point to  $q$  within that tetrahedron (see Fig. 1(a)). Let  $p'_k$  be the intersection of the line  $qp_k$  with the triangle  $p_i p_j a$ . Let  $c = ap'_k \cap p_i p_j$ . Observing that  $|qa| \leq \Delta$  and  $|\overline{qc}| \leq \max\{|\overline{qp_i}|, |\overline{qp_j}|\}$ , we have that the farthest point to  $q$  satisfying the condition in the lemma could only be one of  $p_i$  or  $p_j$ . For each tetrahedron of the form  $qp_i ab$ , with  $p_i \in CH(P, q)$  and  $a, b$  vertices of  $\partial C$ , let  $p_k$  be the farthest point to  $q$  within that tetrahedron

(see Fig. 1(b)). Let  $p'_k$  be the intersection of the line  $qp_k$  with the triangle  $p_iab$ . Let  $c = p_i p'_k \cap ab$ . Clearly,  $c$  is within distance  $\Delta$  from  $q$ . Thus, by a similar argument as above on the triangle  $qp_i c$ , the farthest point to  $q$  satisfying the condition in the lemma could only be  $p_i$ .

To help the intuition, Fig. 2 illustrates the main idea of the proof in  $\mathbb{R}^2$ , with  $h_q$  replaced by the line  $\mathcal{L}'_q$ , orthogonal to  $\mathcal{L}_q$  at  $q$ .  $\square$

**Lemma 2** *Given a set  $P$  of  $n$  points in  $\mathbb{R}^3$ , in  $o(n^{4/3})$  time one can construct a data structure of size  $o(n^{4/3})$  such that the farthest point of  $P$  from a query line segment  $s$  can be reported in  $O(n^{2/3+\epsilon})$  time, where  $\epsilon > 0$  is an arbitrarily small constant.*

**Proof.** Let  $L_s$  be the line supporting  $s$ . The farthest point from  $s$  is either (i) the farthest point from  $L_s$  or (ii) the farthest point  $p \in P$  from one of the endpoints of  $s$ , with the property that the plane orthogonal to  $s$  at that endpoint has  $s$  and  $p$  on different sides. The point for (i) can be obtained using **FPFL** (see Section 2). This gives us a value  $\Delta$  such that all points in  $P$  are within distance  $\Delta$  from  $L_s$ . The point for (ii) can then be found by performing two queries, each on a data structure associated with  $CH(P, q_i)$ , where  $q_i, i = 1, 2$ , is an endpoint of  $s$ , and  $CH(P, q_i)$  and  $q_{3-i}$  are on different sides of the defining plane for  $CH(P, q_i)$ . Consider the query for  $CH(P, q_1)$  (the query for  $CH(P, q_2)$  is similar). The problem is to report the farthest point  $p \in P$  to  $q_1$  such that  $p \in CH(P, q_1)$  and  $|pq_1| > \Delta$ , if such a point exists. From Lemma 1, it follows that one should only consider the vertices of  $CH(P, q_1)$ . Thus, we can find the desired point as follows. We first construct a partition tree based on a fine simplicial partition  $\Psi(P)$ . Let  $P_v$  be the subset of  $P$  stored at a node  $v$  of the tree, and let  $t_v$  be the tetrahedron of  $\Psi(P)$  that contains  $P_v$ . For each node  $v$ , we compute the convex hull  $CH_v$  of  $P_v$ . We also compute and store a data structure for the subset of  $P_v$  corresponding to the vertices of  $CH_v$ , that can answer farthest neighbor queries in  $\mathbb{R}^3$ . In  $\mathbb{R}^d$ ,  $d$  a constant, one can construct in  $O^*(m)$  time a data structure of size  $O(m)$  that can answer farthest neighbor queries in  $O(n/m^{1/\lceil d/2 \rceil})$ , for any  $m$  such that  $n \leq m \leq n^{\lceil d/2 \rceil}$  (see [17], page 594). Thus, in  $\mathbb{R}^3$ , the query time is  $O(n/m^{1/2})$ . From Theorem 2, it follows that there are  $r - O(r^{2/3})$  tetrahedrons that are not intersected by the plane  $h_{q_1}$ . For the point sets  $P_v$  that are in  $CH(P, q_1)$  and have  $t_v \cap h_{q_1} = \emptyset$ , we find the farthest point from  $q_1$  using the farthest point data structures associated with these point sets. For the remaining  $O(r^{2/3})$  point sets for tetrahedrons that are intersected by  $h_{q_1}$ , we recursively search the corresponding simplicial partitions.

We now analyze the preprocessing time and space and the query time for the data structure constructed. Assuming  $r$  is a constant, the construction of the partition tree takes  $O(n \log n)$  time and space, which follows from Theorem 2. After building the partition tree, we compute the convex hull of the points

associated with each node  $v$  in the tree in  $O(n \log n)$  time [32]. We then construct the data structure for answering farthest neighbor queries with  $O(n)$  space and  $O^*(n)$  preprocessing time. Then, if we let  $T(n)$  be the time required to compute this secondary information at every node of the partition tree, it satisfies the recurrence inequality  $T(n) \leq O^*(n) + \sum_1^r T(n_v)$ , where  $n_v$  is the number of points stored at node  $v$ . By choosing  $r > 2$  a constant, it follows that  $T(n) = O^*(n)$ . The space  $S(n)$  satisfies the recurrence inequality  $S(n) \leq O(n) + r + \sum_1^r S(n_v)$  and thus we obtain  $S(n) = O(n \log n)$ .

For the query phase, the time to answer a farthest neighbor query for  $q_1$  is  $O(n^{1/2})$ . We query with  $q_1$  in at most  $r - cr^{2/3}$  point sets, that have  $t_v \cap h_{q_1} = \emptyset$ , and recursively search on the remaining  $cr^{2/3}$  point sets with  $t_v \cap h_{q_1} \neq \emptyset$ . Therefore, we have  $Q(n) \leq r + \sum_{t_v \cap h_{q_1} = \emptyset} O(n_v^{1/2}) + \sum_{t_v \cap h_{q_1} \neq \emptyset} Q(n_v)$ . If we set  $r = c^{1/\epsilon} 2^{1+2/(3\epsilon)}$ , for any constant  $\epsilon > 0$ , then  $r > 2$ ,  $r$  is a constant, and the query time is  $O(n^{2/3+\epsilon})$ .

Balancing the space and preprocessing time for **FPFL** to obtain  $O(n^{2/3+\epsilon})$  query time, we get  $O^*(n^{4/3-4\epsilon})$  time and space. Then, adding up, we obtain the claimed bounds.  $\square$

We note the result in Lemma 2 could be extended to higher dimensions. Very recently, using a variant of the shallow cuttings [26] for the semialgebraic ranges and an ingenious analysis of the decomposition of the common intersection of cylinders, Sharir and Shaul [35] have presented a data structure for **FPFL** in  $\mathbb{R}^3$  with  $O(n^{1+\epsilon})$  preprocessing time and space and  $O(n^{1/2+\epsilon})$  query time. Applying the **FPFL** result in [35] leads to improved space and preprocessing time in  $\mathbb{R}^3$ .

**Theorem 3** *Given a set  $P$  of  $n$  points in  $\mathbb{R}^3$ , in  $O(n^{1+\epsilon})$  time one can construct a data structure of size  $O(n^{1+\epsilon})$  such that the farthest point of  $P$  from a query line segment  $s$  can be reported in  $O(n^{2/3+\epsilon})$  time, where  $\epsilon > 0$  is an arbitrarily small constant.*

Using the result in Theorem 3, some trade-offs are also possible. For example, using half-space range searching based on cuttings [4, 27] one could obtain  $O(n^{1/2+\epsilon})$  query time by increasing the preprocessing time and space to  $O(n^{3/2} \log^{O(1)} n)$ . It is unclear whether a trade-off is possible for the **FPFL** data structure in [35].

## 4 Farthest Point from Line: Indexed Version

In this section we discuss problem **FPIS**: Given a set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, for each line  $L(p_i, p_j)$ ,  $1 \leq i < j \leq n$ ,

find the farthest point  $p_k$  such that  $i < k < j$ . Clearly, this problem admits a simple  $O(n^3)$  time,  $O(n)$  space solution. Below, we describe two sub-cubic time solutions based on our solution for problem **FPFL**.

**Lemma 3** *Given a set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, with  $O(n^{3-2/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  space one can find for each line  $L(p_i, p_j)$ ,  $1 \leq i < j \leq n$ , the farthest point  $p_k$  such that  $i < k < j$ .*

**Proof.** For each  $p_i \in P$ ,  $i = 1, 2, \dots, n-2$ , we construct a data structure to report the farthest point  $p_k$  to  $L(p_i, p_j)$  for all  $j$  such that  $i < j \leq n$ , with  $i < k < j$ . The data structure for  $p_i$  is a balanced binary tree  $T_i$ , that is defined on the subset of points  $P_i = \{p_i, p_{i+1}, \dots, p_n\}$  of  $P$ . The  $k$ -th leaf of  $T_i$ , in order, stores the vertex  $p_{i+k-1}$  of  $P$ , the line  $L(p_i, p_{i+k-1})$ , and the “current” farthest point  $p_{i+k-1}^*$  for  $L(p_i, p_{i+k-1})$  ( $p_{i+k-1}^*$  is initially empty and  $L(p_i, p_i)$  can be any line containing  $p_i$ ). Each internal node  $v$  of  $T_i$  is associated with a ray shooting data structure  $R_i(v)$  as above, for the points stored at the leaves of the subtree  $T_i(v)$  of  $T_i$  rooted at  $v$ , and it is also associated with the set  $L_i(v)$  of all the lines  $L(p_i, p_j)$  stored at the leaves of  $T_i(v)$ .

We start from the leaves and go up  $T_i$ , level by level, until the root is reached. Suppose that the computation at the level of the children of a node  $v$  has just been completed. Then,  $L_i(v_l)$  and  $L_i(v_r)$  are available from  $v$ 's left and right children,  $v_l$  and  $v_r$ , respectively. At the node  $v$ , we do the following: (1) Compute  $R_i(v)$ ; (2) for each line  $L(p_i, p_j)$  in  $L_i(v_r)$ , find the farthest point from  $L(p_i, p_j)$  among the points that are leaf descendants in  $T_i(v_l)$  (and update the farthest point if needed), and (3) let  $L_i(v) = L_i(v_l) \cup L_i(v_r)$ . Then, at the root of  $T_i$ , we have the farthest point for each line  $L(p_i, p_j)$ , with  $i < j \leq n$ .

We now analyze the complexity bounds of the algorithm. At level  $k$  of  $T_i$  we have  $2^k$  ray shooting data structures, each of which answers  $O(n/2^k)$  queries. For an internal node at level  $k$ , the trade-off between the preprocessing and query time gives  $s = O((n/2^k)^{2\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)})$ , which results in  $O^*(2^k (n/2^k)^{2\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)})$  time at level  $k$  and  $O^*(n^{2\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)}) = O^*(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)})$  time over all the nodes of  $T_i$ . Considering the  $O(n)$  iterations, the time to answer all queries for problem **FPIS** is  $O^*(n^{3-2/(\lfloor f(d)/2 \rfloor + 1)})$ , which is sub-cubic for any constant dimension  $d$ . The space bound of the algorithm is  $O^*(n^{2\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)}) = O^*(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)})$ . (Note that we only need to store the information on at most two consecutive levels of  $T_i$ , and after the queries for  $T_i$  are answered we can discard it). For example, when  $d = 3$ ,  $f(d) = 9$  and we can answer all queries in time  $O^*(n^{13/5}) = O^*(n^{2.6})$ , with  $O^*(n^{8/5}) = O(n^{1.6})$  space.  $\square$

**Theorem 4** *Given a set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, with  $O(n^{3-3/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and space one can report*

for each line  $L(p_i, p_j)$ ,  $1 \leq i < j \leq n$ , the farthest point  $p_k$  such that  $i < k < j$ .

**Proof.** We construct a data structure on the set of points  $P$ , to report the farthest point  $p_k$  to  $L(p_i, p_j)$  for all pairs  $(i, j)$  such that  $1 \leq i < j \leq n$ , with  $i < k < j$ . It can also be used to report the farthest point  $p_k$  in a specified range  $\{p_i, p_{i+1}, \dots, p_j\}$  to an arbitrary query line  $L$ , where  $1 \leq i < k < j \leq n$ .

The data structure is a balanced binary tree  $T$  on the set of points  $P$ . The points in  $P$ , in order, are associated with the leaves of  $T$ . Thus, the first leaf corresponds to the point  $p_1$ , the second leaf corresponds to point  $p_2$  and so on. At each internal node  $v$ , we compute and store a ray shooting data structure for the set of points that are the leaf descendants of  $v$ .

At level  $k$  of  $T$  we have  $2^k$  ray shooting data structures, each of which is constructed on  $n/2^k$  points. As we will see next (Lemma 4), the ray shooting data structure for a node at level  $k$  will be used to answer  $O(n^2/2^k)$  queries. For a node at level  $k$ , the trade-off between the preprocessing and query time gives

$$\begin{aligned} s &= O((n^3/2^{2k})^{\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)}) \\ &= O(n^{3\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)} (2^k)^{-2\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)}). \end{aligned}$$

The time and space at level  $k$  is

$$O^*(2^k s) = O^*(n^{3\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)} (2^{(1-\lfloor f(d)/2 \rfloor) / (\lfloor f(d)/2 \rfloor + 1)})^k).$$

Then, since  $T$  has  $O(\log n)$  levels, the total time and space to construct  $T$  is  $O^*(n^{3\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)} \sum_{k=1}^{\log n} (2^{(1-\lfloor f(d)/2 \rfloor) / (\lfloor f(d)/2 \rfloor + 1)})^k) = O^*(n^{3-3/(\lfloor f(d)/2 \rfloor + 1)})$ , which is sub-cubic and improves over the preprocessing time of our first algorithm (Lemma 3) by an  $O(n^{1/(\lfloor f(d)/2 \rfloor + 1)})$  factor.

To answer a query with a line  $L(p_i, p_j)$  we select the two leaves of  $T$  that store the points  $p_i$  and  $p_j$  and form a search path from  $p_i$  to  $p_j$  in  $T$ . We then obtain a set  $\mathcal{S}$  of  $O(\log n)$  ray shooting structures as follows. For each internal node  $v$  of  $T$ , we add the ray shooting data structure stored at the right child  $v_r$  of  $v$  to the set  $\mathcal{S}$  if the search path comes from the left child  $v_l$  of  $v$ . Similarly, we add the ray shooting data structure stored at the left child  $v_l$  of  $v$  to the set  $\mathcal{S}$  if the search path goes from  $v$  to the right child  $v_r$  of  $v$ . Since the search path is of  $O(\log n)$  length, the size of  $\mathcal{S}$  is  $O(\log n)$ . Clearly, all points  $p_k$  associated with these data structures are such that  $i < k < j$ .

We perform a ray shooting query in each of the  $O(\log n)$  ray shooting data structures and keep track of the farthest point  $p_k$  to  $L(p_i, p_j)$ . A ray shooting

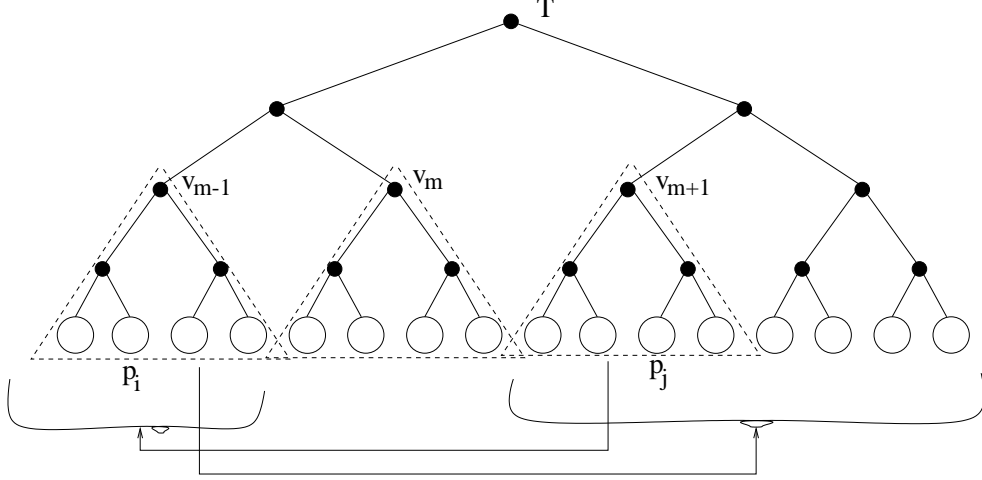


Fig. 3. Illustrating that the number of queries at a node  $v_m$  is  $O(\frac{n^2}{2^k})$ .

query at a level  $k$  takes

$$\begin{aligned}
O^*(s/(n^2/2^k)) &= O^*(2^k s/n^2) \\
&= O^*(n^{3\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1) - 2} (2^k)^{1 - 2\lfloor f(d)/2 \rfloor / (\lfloor f(d)/2 \rfloor + 1)}) \\
&= O^*(n^{1 - 3/(\lfloor f(d)/2 \rfloor + 1)} (2^k)^{(1 - \lfloor f(d)/2 \rfloor) / (\lfloor f(d)/2 \rfloor + 1)}).
\end{aligned}$$

Thus, we can answer a query with line  $L(p_i, p_j)$  in time

$$O^*(n^{1 - 3/(\lfloor f(d)/2 \rfloor + 1)}) \sum_{k=1}^{\log n} (2^k)^{(1 - \lfloor f(d)/2 \rfloor) / (\lfloor f(d)/2 \rfloor + 1)} O^*(n^{1 - 3/(\lfloor f(d)/2 \rfloor + 1)}).$$

The  $O(n^2)$  queries with lines  $L(p_i, p_j)$ ,  $1 \leq i < j \leq n$ , can then be answered in a total of  $O^*(n^{3 - 3/(\lfloor f(d)/2 \rfloor + 1)})$  time, which matches the preprocessing time and space for constructing  $T$ . For example, when  $d = 3$ , we have  $f(d) = 9$  and we can answer all queries in time  $O^*(n^{2.4})$ , with  $O^*(n^{2.4})$  space.  $\square$

**Lemma 4** *There are a total of  $O(n^2)$  queries for the data structures associated with a level  $k$  of  $T$ , with  $O(n^2/2^k)$  queries at each node on level  $k$ .*

**Proof.** Let  $T(v)$  denote the subtree of  $T$  rooted at node  $v$ . Let  $v_{m-1}$ ,  $v_m$  and  $v_{m+1}$  be three consecutive nodes at level  $k$ . From the query answering procedure we have that the ray shooting data structure at  $v_m$  answers queries only for lines  $L(p_i, p_j)$  with  $p_i \in T(v_{m-1})$  or  $p_j \in T(v_{m+1})$  (see Fig. 3). Thus, the number of queries on the ray shooting data structure at  $v_m$  is  $O(n(n/2^k)) = O(n^2/2^k)$ .  $\square$

**Corollary 2** *Given a set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ , where*

$d \geq 3$  is a constant, with  $O(n^{3-3/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and space one can report for each line  $L(p_i, p_j)$ ,  $1 \leq i < j \leq n$ , the minimum-width cylindrical shell with central axis  $L(p_i, p_j)$  and enclosing the points  $\{p_i, p_{i+1}, \dots, p_j\}$ .

The data structure constructed in the proof of Theorem 4 can also be used to answer a query with an arbitrary line  $L$ , that asks to report the farthest point  $p_k$  to  $L$  that is in a specified range  $\{p_i, p_{i+1}, \dots, p_j\}$ , where  $1 \leq i < k < j \leq n$ .

**Theorem 5** *A set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, can be preprocessed in  $O(n^{\lfloor f(d)/2 \rfloor} \cdot \log^{O(1)} n)$  time and space such that for a query line  $L$  one can report in  $O(\log^2 n)$  time the farthest point  $p_k$  to  $L$  that is in a specified range  $\{p_i, p_{i+1}, \dots, p_j\}$ , where  $1 \leq i < k < j \leq n$ .*

**Proof.** To obtain  $O(\log n)$  query time at an internal node of  $T$ , we need to balance for  $(n/2^k) \log n / s^{1/\lfloor f(d)/2 \rfloor} = O(\log n)$ , which gives  $s = O((n/2^k)^{\lfloor f(d)/2 \rfloor})$ . At level  $k$  we obtain  $O^*(2^k s) = O^*(n^{\lfloor f(d)/2 \rfloor} (2^k)^{1-\lfloor f(d)/2 \rfloor})$  preprocessing time and space, resulting in a total of  $O^*(n^{\lfloor f(d)/2 \rfloor})$  preprocessing time and space over  $T$ . For example, when  $d = 3$ , we have  $f(d) = 9$  and the data structure requires  $O^*(n^4)$  space. Using this data structure and the query procedure on  $T$  described earlier, to answer a query with  $L$  we need to query  $O(\log n)$  nodes of  $T$  at  $O(\log n)$  time per node, for a total of  $O(\log^2 n)$  query time.  $\square$

**Corollary 3** *A set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$  in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, can be preprocessed in  $O(n^{\lfloor f(d)/2 \rfloor} \cdot \log^{O(1)} n)$  time and space such that for a query line  $L$  and a pair  $(i, j)$ , with  $1 \leq i < j \leq n$ , the minimum-width cylindrical shell of central axis  $L$  and enclosing  $\{p_i, p_{i+1}, \dots, p_j\}$  can be found in  $O(\log^2 n)$  time.*

## 5 Applications

In this section we show how to use our results for problems **FPIS** and **FPFL** to fit a polygonal chain through a set of indexed points and to solve the minimum (maximum) area anchored triangle problem.

### 5.1 Fitting Indexed Points with Polygonal Chains

Consider a set of  $n$  indexed points  $P = \{p_1, p_2, \dots, p_n\}$ . The problem is to compute a fitting polygonal chain  $P' = (p_{i_1} = p_1, p_{i_2}, \dots, p_{i_m} = p_n)$  with fewer than  $n$  vertices. Specifically, we ask that the vertices of  $P'$  are an ordered subsequence of the vertices of  $P$  (i.e.,  $i_j < i_{j+1}$  and  $i_j, i_{j+1} \in \{1, 2, \dots, n\}$ , for  $j \in \{1, 2, \dots, m-1\}$ ) and that the subset of points  $\{p_{i_j}, p_{i_{j+1}}, \dots, p_{i_{j+1}}\}$  of  $P$

is contained in some *fitting* region of the line segment  $\overline{p_i p_{i+1}}$ .

The problem is equivalent to the polygonal chain simplification problem, in which  $P$  is treated as the set of vertices of a polygonal chain, and we discuss the later. There are two optimization versions of the polygonal chain simplification problem. In the first one, called **min-#**, the input consists of a polygonal chain  $P$  and an error tolerance  $\epsilon$  (defining the “fitting” region), and the goal is to find an  $\epsilon$ -approximating chain  $P'$  with the smallest number of vertices. In the second one, called **min- $\epsilon$** , the input is a polygonal chain  $P$  and an integer  $m$ , where  $m < n$  and  $n$  is the number of vertices of  $P$ , and the goal is to compute a chain  $P'$  with at most  $m$  vertices that minimizes the error  $\epsilon$ .

The error criterion that defines the error tolerance region of a line segment  $\overline{p_i p_{i+1}}$  determines the quality of the approximation in terms of the deviation between the approximated and approximating paths. A few error criteria have been used in solving the polygonal chain approximation problem [6, 7, 10, 11, 13, 16, 20–22, 29, 36]. Two of the most popular ones are the *tolerance zone* [7, 20, 22, 29] and the *infinite beam* [11, 16, 22, 36] criteria. With the *infinite beam* criterion the  $\epsilon$ -tolerance region of a line segment  $\overline{p_i p_j}$  is the set of points within distance  $\epsilon$  from the line  $L(p_i, p_j)$  supporting the line segment  $\overline{p_i p_j}$ . With the *tolerance zone* criterion the  $\epsilon$ -tolerance region of a line segment  $\overline{p_i p_j}$  is the set of points within distance  $\epsilon$  from the line segment.

The problem appears in computer graphics, computational biology (for modeling neurons and molecules) and data compression. The theoretical framework for solving the problem has been developed in [20, 22]. However, it could be successfully applied only for the planar version of the problem [10, 11]. In  $\mathbb{R}^d$ ,  $d \geq 3$ , in most cases, the only known solutions are brute force algorithms that take  $O(n^3)$  and  $O(n^3 \log n)$  time to solve the min-# and min- $\epsilon$  problems, respectively. Recently, a sub-cubic,  $O^*(n^{3-2/(\lfloor d/2 \rfloor + 1)})$  time algorithm for solving the min-# problem in  $\mathbb{R}^d$ ,  $d \geq 3$ , with the tolerance zone criterion, was presented in [7] but their results cannot be extended to the infinite beam criterion. They also give  $O(n^2 \log^3 n)$  and  $O^*(n^{7/3})$  time algorithms for the min- $\epsilon$  problem in  $\mathbb{R}^3$  and  $\mathbb{R}^4$ , respectively, based on parametric search. Their results for the min- $\epsilon$  problem cannot be extended to  $d > 4$  since the underlying techniques do not extend for  $d > 4$ . We show below how to solve this problem in sub-cubic time, using a unified approach. We are able to address the min- $\epsilon$  problem for any constant  $d \geq 3$  by avoiding parametric search. Unlike in [7], we use the **FPIS** data structure to compute the set  $\mathcal{E}$  of  $O(n^2)$  possible approximation errors, sort it and then perform a binary search on the sorted set, at each step running the min-# algorithm.

No sub-cubic time solutions for the min-# and min- $\epsilon$  problems in  $\mathbb{R}^d$ ,  $d \geq 3$ , with the infinite beam criterion, were known until now.

**Theorem 6** *The min- $\epsilon$  version of the polygonal chain simplification problem in  $\mathbb{R}^d$ , with  $d \geq 3$  a constant, using the tolerance zone measure of error, can be solved in time and space  $O(n^{3-3/(\lfloor d/2 \rfloor + 1)} \log^{O(1)} n)$ .*

**Proof.** We use the general approach to solve the min- $\epsilon$  problem: compute the set  $\mathcal{E}$  of  $O(n^2)$  possible approximation errors, sort them and do binary search on the sorted set, at each step running the min-# algorithm. Since the min-# algorithm takes  $O(n^{3-2/(\lfloor d/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time [7], to obtain the claimed bounds we need to show how to compute  $\mathcal{E}$  within those time and space bounds. We compute a superset of errors that includes  $\mathcal{E}$ , it is of size  $O(n^2)$  and it is formed by (1) the error set  $\mathcal{E}_1$  containing  $d(p_i, p_j)$ , for all  $1 \leq i < j \leq n$ , where  $d(p_i, p_j)$  is the Euclidean distance from  $p_i$  to  $p_j$ , and (2) the error set  $\mathcal{E}_2$  containing  $d(p_k, L(p_i, p_j))$ , where  $p_k$  is the farthest point from the line  $L(p_i, p_j)$  such that  $i < k < j$  and  $d(p_k, L(p_i, p_j))$  denotes the shortest Euclidean distance from  $p_k$  to  $L(p_i, p_j)$ . Clearly,  $\mathcal{E}_1 \cup \mathcal{E}_2$  is a superset of  $\mathcal{E}$ , the sought error  $\epsilon \in \mathcal{E}$  that is the solution for the min- $\epsilon$  problem is contained in  $\mathcal{E}_1 \cup \mathcal{E}_2$ , and the size of  $\mathcal{E}_1 \cup \mathcal{E}_2$  is  $O(n^2)$ . Since  $\mathcal{E}_1$  can be easily computed in  $O(n^2)$  time and space and from Theorem 4 it follows that  $\mathcal{E}_2$  can be computed with  $O(n^{3-3/(\lfloor d/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and space, we obtain the claimed bounds.  $\square$

**Theorem 7** *The min-# version of the polygonal chain simplification problem in  $\mathbb{R}^d$ , with  $d \geq 3$  a constant, using the infinite beam measure of error, can be solved in time and space  $O(n^{3-3/(\lfloor d/2 \rfloor + 1)} \log^{O(1)} n)$ .*

**Proof.** We construct the data structure in Theorem 4 and use it in a query based, breadth first traversal approach to solve the min-# problem. Define the value of a shortest path  $s_j$  at  $p_j$ ,  $1 \leq j \leq n$ , as the number of vertices in an optimal solution to the min-# problem on the subpath from  $p_1$  to  $p_j$ . Set  $s_1 = 1$  and  $s_j = \infty$ , for  $j = 2, 3, \dots, n$ , and place the index 1 into an initially empty queue  $\mathcal{Q}$ . Then, we repeat the following operations until  $s_n$  is updated for the first time. Let  $i$  be the value of the index at the head of  $\mathcal{Q}$ . Remove  $i$  from  $\mathcal{Q}$  and, for each  $p_j$  such that  $i < j$  and  $s_j = \infty$ , query the data structure for the farthest point  $p_k$  to the line  $L(p_i, p_j)$ , with  $i < k < j$ . If  $d(p_k, L(p_i, p_j)) \leq \epsilon$  then we set  $s_j = s_i + 1$  and place  $j$  at the tail of  $\mathcal{Q}$ . Then, the complexity of the algorithm is dominated by the time to answer the queries and Theorem 4 gives  $O(n^{3-3/(\lfloor d/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and space.  $\square$

**Theorem 8** *The min- $\epsilon$  version of the polygonal chain simplification problem in  $\mathbb{R}^d$ , with  $d \geq 3$  a constant, using the infinite beam measure of error, can be solved in time and space  $O(n^{3-3/(\lfloor d/2 \rfloor + 1)} \log^{O(1)} n)$ .*

**Proof.** The set of possible approximation errors corresponds to the distances  $d(p_k, L(p_i, p_j))$ , where  $p_k$  is the farthest point from the line  $L(p_i, p_j)$  and  $1 \leq i < k < j \leq n$ . Then, we only need to compute the set of errors  $\mathcal{E}_2$  as defined in

the proof of Theorem 6. From Theorem 4 it follows that  $\mathcal{E}_2$  can be computed with  $O(n^{3-3/(\lfloor f(d)/2 \rfloor + 1)} \cdot \log^{O(1)} n)$  time and space. Since the corresponding min-# algorithm has similar bounds and it is called  $O(\log n)$  times in the binary search process, we obtain the claimed bounds.  $\square$

## 5.2 Maximum (Minimum) Area Anchored Triangle

In this section we show how to use our result for problem **FPFL** to find the maximum area anchored triangle defined by an anchor point  $q$  with the points in  $P$ , where  $P$  is a set of  $n$  points and  $q, P \subset \mathbb{R}^d$ ,  $d$  a constant.

**Theorem 9** *Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$  and an anchor point  $q$  (possibly in  $P$ ), the maximum area triangle defined by  $q$  with  $P \setminus \{q\}$  can be computed in  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  time and space.*

**Proof.** For a point  $p_i \in P$ ,  $p_i \neq q$ , the maximum area triangle defined by the line segment  $\overline{qp_i}$  with the points in  $P \setminus \{q, p_i\}$  can be found by computing the farthest point from the line supporting  $\overline{qp_i}$ . Thus, over all  $p_i \in P$  we have  $O(n)$  **FPFL** queries. Balancing the preprocessing time and space for constructing the **FPFL** data structure with the query time results in  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  time and space for constructing the data structure. Each query on this data structure takes  $O(n^{1-2/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  time, for a total of  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  over the  $O(n)$  queries. Thus, the maximum area anchored triangle can be found within the claimed bounds. In  $\mathbb{R}^3$  the time and space requirements are  $O(n^{1.6} \log^{O(1)} n)$ .  $\square$

We note that the computation of the closest line from a query point is similar to the computation of the farthest line (consider ray shooting queries in the lower envelope of the distance functions, instead of the upper envelope), and thus the algorithm for finding the minimum area anchored triangle is similar to that for the maximum area anchored triangle (farthest point from line queries are replaced by closest point from line queries).

**Theorem 10** *Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$  and an anchor point  $q$  (possibly in  $P$ ), the minimum area triangle defined by  $q$  with  $P \setminus \{q\}$  can be found in  $O(n^{2-2/(\lfloor f(d)/2 \rfloor + 1)} \log^{O(1)} n)$  time and space.*

## 6 Conclusions

In this paper we presented a data structure for answering farthest point queries with line segments. For a set  $P$  of  $n$  points in  $\mathbb{R}^3$  the data structure has size

$O(n^{1+\epsilon})$ , can be constructed in  $O(n^{1+\epsilon})$  time and can report the farthest point of  $P$  from a query line segment in  $O(n^{2/3+\epsilon})$  time, for arbitrary  $\epsilon > 0$ .

We also addressed a number of extremal point query problems when  $P$  is a set of  $n$  points in  $\mathbb{R}^d$ ,  $d \geq 3$ , including the computation of the farthest point from a query line and the computation of the farthest point from each of the lines spanned by the points in  $P$ . We showed how to use these results to obtain: (1) Sub-cubic time algorithms for fitting a polygonal chain through an indexed set of points in  $\mathbb{R}^d$ ,  $d \geq 3$  a constant, and (2) A sub-quadratic time and space algorithm that, given  $P$  and an anchor point  $q$  (possibly in  $P$ ), computes the minimum (maximum) area triangle defined by  $q$  with  $P \setminus \{q\}$ .

## Acknowledgements

The authors are very thankful to Pankaj Agarwal and Boris Aronov for their suggestions on the problems addressed in this paper.

## References

- [1] P.K. Agarwal, B. Aronov, and M. Sharir. Line transversals of balls and smallest enclosing cylinders in three dimensions. *Disc. Comput. Geom.*, 21:373–388, 1999.
- [2] P.K. Agarwal, B. Aronov, and M. Sharir. Exact and approximation algorithms for minimum-width cylindrical shells. *Discrete Comput. Geom.*, 26:307–320, 2001.
- [3] P.K. Agarwal, S. Har-Peled, and K.R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.
- [4] P.K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11:393–418, 1994.
- [5] P.K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30:412–458, 1998.
- [6] P.K. Agarwal and K.R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete Computational Geometry*, 23:273–291, 2000.
- [7] G. Barequet, D.Z. Chen, O. Daescu, M.T. Goodrich, and J. Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002.
- [8] S. Bespamyatnikh and J. Snoeyink. Queries with segments in Voronoi diagrams. *Computational Geometry: Theory and Applications*, 16(1):23–33, 2000.

- [9] T.M. Chan. Approximating the diameter, width, smallest enclosing cylinder and minimum-width annulus. *Internat. J. Comput. Geom. Appl.*, 12(2):67–85, 2002.
- [10] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *Intl. Journal of Computational Geometry and Applications*, 6(1):59–77, 1996.
- [11] D.Z. Chen and O. Daescu. Space-efficient algorithms for approximating polygonal curves in two dimensional space. *Intl. J. Comput. Geom. Appl.*, 13(2):95–112, 2003.
- [12] Richard Cole and Chee K. Yap. Geometric retrieval problems. *Proc. IEEE 24th Symposium on Foundations of Computer Science*, pages 112–121, 1983.
- [13] L.P. Cordella and G. Dettoni. An  $O(n)$  algorithm for polygonal approximation. *Pattern Recognition Letters*, 3:93–97, 1985.
- [14] O. Daescu, N. Mi, C.-S. Shin, and A. Wolff. Farthest point queries with geometric and combinatorial constraints. *Proc. 20th European Workshop on Computat. Geom.* pg. 45-48, 2004.
- [15] J. Erickson. Space-time tradeoffs for emptiness queries. *SIAM J. on Comput.*, 29(6):1968–1996, 2000.
- [16] D. Eu and G.T. Toussaint. On approximation polygonal curves in two and three dimensions. *CVGIP: Graphical Models and Image Processing*, 56(3):231–246, 1994.
- [17] J.E. Goodman and J. O’Rourke. *Handbook of Disc. and Computat. Geom.*, CRC Press, 1997.
- [18] S. Har-Peled and K. Varadarajan. High dimensional shape fitting in linear time. *Discrete Computational Geometry*, 32(2):269–288, 2004.
- [19] S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM Journal on Computing*, 33(2):269–285, 2004.
- [20] H. Imai and M. Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics and Image Processing*, 36:31–41, 1986.
- [21] H. Imai and M. Iri. An optimal algorithm for approximating a piecewise linear function. *Journal of Information Processing*, 9(3):159–162, 1986.
- [22] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. *Computat. Morphology*, pages 71–86, 1988.
- [23] D.T. Lee and Y.T. Ching. The power of geometric duality revisited. *IPL*, 21:117–122, 1985.
- [24] S.Z. Li and J. Lu. Face recognition using the nearest feature line method. *IEEE Transactions on Neural Networks*, 10(2):439–443, 1999.

- [25] J. Matoušek. Efficient partition trees. *Discrete Computational Geometry*, 8:315–334, 1992.
- [26] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.
- [27] J. Matousek. Range searching with efficient hierarchical cuttings. *Discrete Computational Geometry*, 10:183–196, 1993.
- [28] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Computational Geometry*, 10:215–232, 1993.
- [29] A. Melkman and J. O’Rourke. On polygonal chain approximation. *Computational Morphology*, pages 87–95, 1988.
- [30] P. Mitra and B.B. Chaudhuri. Efficiently computing the closest point to a query line. *Pattern Recognition Letters*, 19(11):1027–1035, 1998.
- [31] Asish Mukhopadhyay. Using simplicial partitions to determine a closest point to a query line. *Pattern Recognition Letters*, 24(12):1915–1920, 2003.
- [32] F. P. Preparata and M. I. Shamos. Computational geometry: An introduction. *Springer-Verlag*, 2nd edition, 1990.
- [33] E. Schomer, J. Sellen, M. Teichmann, and C. Yap. Efficient algorithms for smallest enclosing cylinder problem. *Proc. Canad. Conf. Comp. Geom.*, pages 264–269, 1996.
- [34] M. Sharir and P.K. Agarwal. Davenport-schinzal sequences and their geometric applications. Cambridge University Press, 1995.
- [35] M. Sharir and H. Shaul. Ray shooting amid balls, farthest point from line, and range emptiness searching. *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, pages 525–534, 2005.
- [36] G.T. Toussaint. On the complexity of approximating polygonal curves in the plane. *Proc. IASTED International Symp. on Robotics and Automation*, 1985.