An Efficient Algorithm for the Three-Dimensional Diameter Problem

Sergei N. Bespamyatnikh *†

Abstract

We give a deterministic algorithm for computing the diameter of an n-point set in three dimensions with $O(n \log^2 n)$ running time. This bound matches the best previous algorithm of Ramos [18]. However, our algorithm applies another technique and is simpler.

1 Introduction

We address the well-known diameter problem:

The diameter problem. Given a set S of n points in d-dimensional space. Compute the diameter of S, defined as the maximum distance between two points of S.

Let $diam(S)$ denote the diameter of set S. In this paper we consider three-dimensional diameter problem. This problem was solved by Clarkson and Shor [7] by a randomized algorithm with optimal expected running time $O(n \log n)$. The diameter can be computed in $O(n^2)$ time by brute force. Yao solved this problem in $O((n \log n)^{1.8})$ time [20] (in higher dimension $d \geq 4$ he obtained $O(n^{2-a(d)} \log^{1-a(d)} n)$ time, where $a(d) = 2^{-(d+1)}$). By computing a structure that allows point location in the (implicitly represented) furthest-point Voronoi diagram of S, it is possible to achieve running time $O(n^{4/3+\epsilon})$ [13].

The further solutions [6, 14, 3] are based on the same approach. Clarkson and Shor [7] first transform 3-dimensional diameter problem to the following problem.

The ball problem. Given n unit-balls and n points in 3-dimensional space, determine whether any point lies outside the common intersection of the balls.

The diameter of S is the smallest r such that the intersection of the balls of radius r centered at the points of S contains S. The algorithms of [6, 14, 3] use Megiddo's parametric search technique [15]. For a specific value r, an oracle O decides whether $r < diam(S)$, $r = diam(S)$ or $r > diam(S)$ using an algorithm for the ball problem.

Chazelle et al. [6] obtained a deterministic $O(n^{1+\epsilon})$ algorithm, where $\epsilon > 0$ is an arbitrary small constant. Matoušek and Schwarzkopf [14] improved this running time to $O(n \log^c n)$ where $c > 0$ is a constant. Amato et al. [2] and Brönninman et al. [3] gave $O(n \log^3 n)$ algorithms for computing diameter. Recently, Ramos [18] improved the running time by a $\log n$ factor, using 1-d lower envelope algorithm, and further insight into geometric optimization in an arrangement of surfaces.

In this paper we study another way to compute the diameter by constructing a restriction of the furthest-point Voronoi diagram to the convex hull (in a special case). We present an algorithm with running time $O(n \log^2 n)$. Our algorithm uses practical data structures such as the lists (including the double-connected-edge-lists), the balanced binary search tree, the hierarchy of Dobkin and Kirkpatrick [9]. I hope that it will be good in practice.

2 Geometric Preliminaries

We introduce some notation that will be used throughout this paper. For a finite set $A \subset R^3$,

*Department of Mathematics and Mechanics, Ural State University, 51 Lenin St., Ekaterinburg 620083, Russia.

†e-mail: Sergei.Bespamyatnikh@usu.ru

$CH(A)$ denotes the convex hull of A, $Vor(A)$ denotes the furthest-point Voronoi diagram of A. For a point $p \in A$, $Vor(p)$ denotes the Voronoi polytope of p in $Vor(A)$.

The following Lemmas are useful.

Lemma 1 *Let A be a finite set of points in R^d, p be a point of A and q be a point in the furthest-point Voronoi cell of p. Then $Vor(p)$ contains a ray starting at q in the direction from p to q.*

Lemma 2 *Let A be a finite set of points in R^3 and P be a convex polyhedron such that every point of A lies on or inside P. For a point $p \in A$, the intersection of the furthest-point Voronoi cell of p and the polyhedron P is connected set.*

Lemma 3 *Given a convex polytope with the set of vertices S in R^3. 1) For a point $p \in S$, the Voronoi cell of p is non-empty. 2) For a point $p \in S$ and a plane α, $O(1)$ operations is suffice to determine a side of α such that the corresponding halfspace and the Voronoi cell of p intersect.*

3 Reduction

At first we reduce the problem of finding the diameter of set to the problem of finding the furthest distance between points of separated sets.

The red-blue diameter problem. Given a red set R and a blue set B that are separated by a plane π. Find the red-blue diameter of R and B, defined as the maximum distance between two points such that one lies in R and the other lies in B.

Let $diam(R, B)$ denotes the red-blue diameter of sets R and B. The reduction of the diameter problem to the red-blue diameter problem can be done easy. Compute the minimum axes-parallel box that contains given set. Construct three planes perpendicular to coordinate axes such that each plane divide the covering box into two boxes of equal size. Consider one plane. It divides set S into two subsets. Solve red-blue diameter problem for these sets. The diameter of S is maximum of these diameters. This reduction gives three red-blue diameter problems.

Theorem 4 *The algorithm for reduction is correct.*

4 Red-Blue Diameter Algorithm

In this section we solve the red-blue diameter problem. Given a set R of n points and a set B of m points. The plane π separates these sets. Without loss of generality we can assume that $n \geq m$. For any points $p \in R$ and $q \in B$, if they achieve the red-blue diameter $dist(p, q) = diam(R, B)$ then they lie on the convex hull of the union of red and blue sets. First we construct the convex hull $H = CH(R \cup B)$ and remove points that lie inside it.

Let π_1 be a plane that

• is parallel to the plane π, and

• divides the set R into two subsets of (almost) equal size.

Let R_1 denotes the subset of R which lies in the same halfspace defined by π_1 as the set B. Let R_2 denotes another subset of R, i.e. $R_2 = R \setminus R_1$.

Construct the intersection of the furthest-point Voronoi diagram $Vor(B)$ and the plane π_1. The intersection has size $O(n)$. The algorithm for constructing $Vor(B) \cap \pi_1$ is described in Section 5. We also construct a polygon which is the intersection of the convex hull H and the plane π_1. Denote it by Q. It is clear that Q is a convex polygon.

For a point $p \in B$, there are five cases.

Case 1. The Voronoi cell of p does not intersect the plane π_1.

There are two subcases depending on which side of π_1 the Voronoi cell is on. By Lemma 3, the side can be determined in $O(1)$ time.

Case 1.1 The Voronoi cell of p is on the same side of π_1 as the points of B. The Voronoi cell of p does not contain any point of R_2.

Case 1.2 The plane π_1 separates the Voronoi cell of p and the set B. The Voronoi cell of p does not contain any point of R_1.

In the following cases the Voronoi cell of p intersects the plane π_1. Denote this polygon by $P = Vor(p) \cap \pi_1$.

Case 2. The polygon P lies inside the polygon Q. We show that the Voronoi cell of p does not contain any point of R_1. Suppose that, for a point $q \in R_1$, the Voronoi cell of p contains q. By Lemma 1, there exists a point $r \in H$ on the other side of π_1 (we can construct a ray starting at any

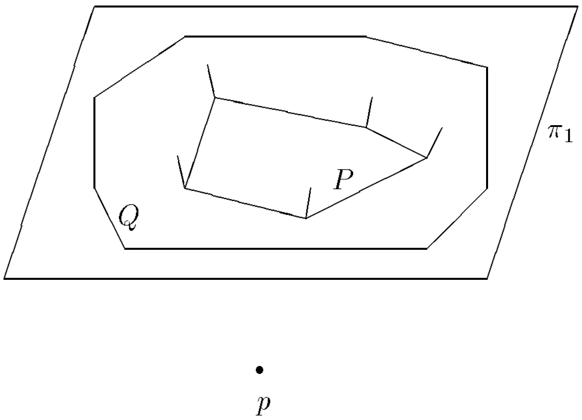

Figure 1: Case 2. $P \subset Q$.

point $r' \in P$ in the direction from p to r'). By Lemma 2, there is a path in $Vor(p) \cap H$ from q to r. This path crosses the boundary of polygon Q. Hence the Voronoi cell of p (and the polygon P) contains at least one point of the boundary of polygon Q. But $P \subset Q$. Contradiction.

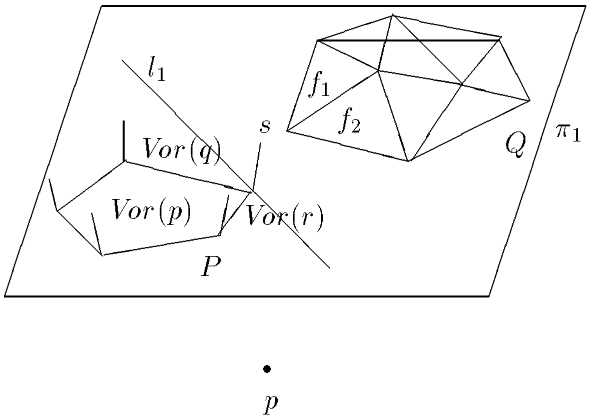

Figure 2: Case 3. $P \cap Q = \emptyset$.

Case 3. The polygon P lies outside the polygon Q. Construct a line l_1 (see fig. 2) such that

- l_1 separates the polygons P and Q, and
- l_1 is tangent to the polygon P.

To compute the line l_1 we can use an optimal $O(\log n)$ algorithm of Edelsbrunner [10] for finding the minimum distance between two convex polygons. Consider the tangent point of l_1 and edges of P incident to it. These edges share the Voronoi region of p and the Voronoi regions of two points, say q and r. It is clear that the Voronoi cell of p lies in the halfspace $\{t \mid dist(p,t) \leq dist(q,t)\}$ and in the halfspace $\{t \mid dist(p,t) \leq dist(r,t)\}$. The intersection of these halfspaces lies in the halfspace defined by a plane passing through the line l_1 and the line $\{t \mid dist(p,t) = dist(q,t) = dist(r,t)\}$. We denote this plane by π_2.

The plane π_2 (and the Voronoi cell of p) cannot intersect both parts of the convex hull H into which it is divided by the plane π_1. We want to determine a part that does not intersect the plane π_2. Find a vertex s of the polygon Q that is closest to the line l_1. Let f_1 and f_2 be the facets of the convex hull H that

- are incident to the point s, and
- intersect the plane π_1 (see fig. 2).

Compute a line l_2 that is an intersection of planes passing through the facets f_1 and f_2. Consider the point of intersection of the line l_2 and the plane π_2. If the plane π_1 separates it and the point p, then the Voronoi cell of p does not contain any point of R_1. If these points lie at the same halfspace defined by π_1, then the Voronoi cell of p does not contain any point of R_2. If the line l_2 is parallel to the plane π_2, then the Voronoi cell of p does not contain any point of R.

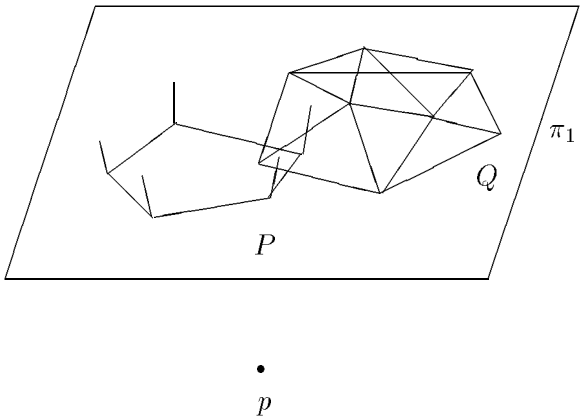

Figure 3: Case 4.$P \cap Q \neq \emptyset$ and $P \setminus Q \neq \emptyset$.

Case 4. The polygon P intersects the boundary of the polygon Q (the vertices and the segments). Unfortunately, the point p can be the furthest neighbor of points in both R_1 and R_2.

Thus, we partition the set B into three subsets $B = B_1 \cup B_2 \cup B_3$, where B_1 (B_2) is the set of points $p \in B$ such that the Voronoi cell of p does not contain any point of R_1 (R_2). The set $B_3 = B \setminus B_1 \cup B_2$ (the case 4). An algorithm to determine the sets B_1, B_2 and B_3 is given in Section 6.

The red-blue diameter $diam(R, B)$ is maximum of the red-blue diameters $diam(R_1, B_2)$, $diam(R_2, B_1)$ and $diam(R, B_3)$. We recursively call the red-blue diameter algorithm of this Section to compute the first two diameters. We show how to compute the red-blue diameter of R and B_3 in Section 7.

5 Voronoi Diagram in Plane

In this section we consider the problem of constructing the furthest-point Voronoi diagram in the plane. This problem differs from classical one, because the data points are given in 3 dimensions. In fact the data points can be given in higher dimensions.

Problem. Given n points in d-dimensional space and the plane π. Construct the intersection of the furthest-point Voronoi diagram and the plane π.

We apply well-known divide-and-conquer technique [17] to construct the Voronoi diagram in the plane. We need an auxiliary result.

Definition 5 *Given a finite set S in R^d, a subdivision $\{S_1, S_2\}$ of S and a plane π. An edge of the furthest-point Voronoi diagram $Vor(S) \cap \pi$ is* shared *if it is common for pair of the Voronoi polygons $Vor(p) \cap \pi$ and $Vor(q) \cap \pi$ where $p \in S_1$ and $q \in S_2$. $\sigma(S_1, S_2, \pi)$ denotes the set of shared edges.*

Definition 6 *Given a finite set S in the plane and a subdivision $\{S_1, S_2\}$ of S. The sets S_1 and S_2 are* linearly separated *if there is a line l such that*

- *the points of S_1 (and S_2 respectively) lies in the same halfplane defined by l, and*
- *if the line l contains points of S then these points lies in the same subset of subdivision (S_1 or S_2).*

Now we can formulate the key lemma.

Lemma 7 *Given a finite set S in R^d, a subdivision $\{S_1, S_2\}$ of S and a plane π. The set of shared edges has the following properties:*

1) $\sigma(S_1, S_2, \pi)$ is the of cycles and chains without common edges. If a chain contains only one edge then it is a line. Otherwise the first and last edges are rays.

2) If the projections of sets S_1 and S_2 into π are linearly separable then $\sigma(S_1, S_2, \pi)$ is a monotone chain.

5.1 Constructing Voronoi diagram.

Now we show how to construct the furthest-point Voronoi diagram in the plane π.

Compute the median m_0 of the x-coordinates of the points of S. Partition S into two subsets S_1 and S_2 of (almost) equal size such that all points of S_1 (resp. S_2) are on or to the left (resp. on or to the right) of the plane $x = m_0$. Using the same algorithm recursively, construct the furthest-point Voronoi diagrams in the plane π for the sets S_1 and S_2 separately. Then construct the chain $\sigma(S_1, S_2, \pi)$ that will be called the *merge chain*. Remove the edges of the Voronoi diagram $Vor(S_2) \cap \pi$ on the left of σ and the edges of $Vor(S_1) \cap \pi$ on the right of σ. As a result, we have the furthest-point Voronoi diagram $Vor(S) \cap \pi$.

It remains to describe the finding the merge chain σ. It is similar to the finding the merge chain σ for the planar case $d = 2$ [19, 11, 12]. The merge chain can be constructed edge by edge, starting with unbounded one. For a set $A \subset R^d$, let us denote by A' the projection of a set A into the plane π. Let us look at the unbounded edges of the furthest-point Voronoi diagram $Vor(S_i) \cap \pi$. In fact they correspond to the edges of convex hull of S_i'. There are exactly two edges of the convex hull of S' that are not included in the convex hulls of S_1' and S_1'. It is suffices to determine one of these edges, for instance, the upper edge. This can be accomplished in linear time by finding a line tangent to the convex hulls of respective S_1' and S_2'. Each further edge of the merge chain is found by tracing the boundaries of the current regions of $Vor(S_1) \cap \pi$ and $Vor(S_2) \cap \pi$ in an appropriate direction until they intersect the actual separator. Easy counting arguments show that only $O(|S|)$ edges are processed in total. This implies an overall running time of $O(|S|)$ for the merge process. The time for computing the furthest-point Voronoi diagram in the plane π is $O(|S| \log |S|)$ by the recurrence relation.

6 Partitioning of Set B

We have the furthest-point Voronoi diagram $Vor(B)$ in the plane π and the polygon Q which is the intersection of the plane π and the convex hull of the red and blue sets. The polygon Q is convex. We will solve the following problem.

Problem. Given a convex polygon Q and a set $\mathcal{P}$ of k convex polygons with disjoint interiors covering the plane.

Determine the polygons of $\mathcal{P}$ that

- lie inside Q,
- lie outside Q,
- intersect the boundary of Q

We restrict the problem to finding the last set $\mathcal{P}'$ of polygons. We can find the intersection of edges of Q and the edges of polygons of $\mathcal{P}$. The number of intersections is bounded by the amount of edges (in fact it is $O(k)$). We can apply the algorithm of Bentley and Ottmann [4] and compute the polygons of $\mathcal{P}'$ in $O(k \log k)$ running time.

We present a simple way of finding $\mathcal{P}'$ in linear time. Denote the vertices of Q by $q_1, \ldots, q_m$ in clockwise order. We will move the point p on the boundary Q and find the polygons of $\mathcal{P}$ that contains p. The moving starts at point $p = q_1$. Determine the polygon of $\mathcal{P}$ containing p (simple scan of edges of polygons of $\mathcal{P}$). Denote it by P. It is clear that $P \in \mathcal{P}'$.

The point p moves in direction from q_1 to q_2. Determine the side e of P that intersects the line through p and q. We distinguish 3 cases to process the point p.

Case 1. The edge e intersects the line segment $[p, q]$. The edge e is common to two polygons of $\mathcal{P}$. One of them is P. It is clear that the second polygon belongs to $\mathcal{P}'$. Add it into $\mathcal{P}'$ and assign it as P.

Case 2. The edge e does not intersect the line through p and q. (This case can occur after the Case 1.) Take the next edge of P after e in clockwise order as e.

Case 3. The edge e intersects the line through p and q but does not intersect the line segment $[p, q]$. Move the point p into q and take the next point after p as q.

The proof of correctness of the algorithm is straightforward.

7 Processing the Set B_3

In this section we show how to find the diameter of the sets B_3 and R. Recall that B_3 is the set of points $p \in B$ such that

$$Vor(p) \cap H \cap \pi_1 \neq \emptyset.$$

The polygon $Q = H \cap \pi_1$ is partitioned into chains by the Voronoi cells $Vor(p)$, $p \in B_3$. Consider a point p in B_3. Unfortunately the Voronoi cell $Vor(p)$ gives $|Q|$ chains in the polygon Q in worst case ($|Q|$ means a number of vertices of Q). It follows from the fact that, for an edge e of the polygon Q, the intersection of the edge e and the Voronoi cell $Vor(p)$ is a segment (if any)). The Figure 4 shows when this case can occur. Fortunately the total number of chains is $O(|Q|)$ (by Lemma 8).

Consider the furthest-point Voronoi diagram of B_3 in R^3. By Lemma 2, the convex hull H is partitioned into $|B_3|$ connected domains, and we call them *surface Voronoi cells.* For a point $p \in B_3$, let $V(p) = Vor(p) \cap H$ denote the surface Voronoi cell corresponding to p. The surface Voronoi cells are determined by facets of polytopes $Vor(B_3)$ that intersect the convex hull H. We know some planes of such faces, namely the faces which intersect the polygon Q. The plane of a face is the bisector plane of points $p, q \in B_3$ (the set of points equidistant from two points

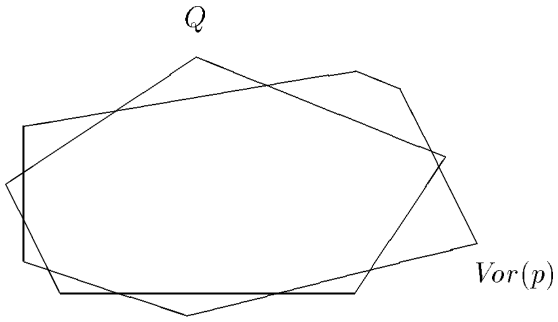

Figure 4: 5 polygonal chains of $Vor(p)$.

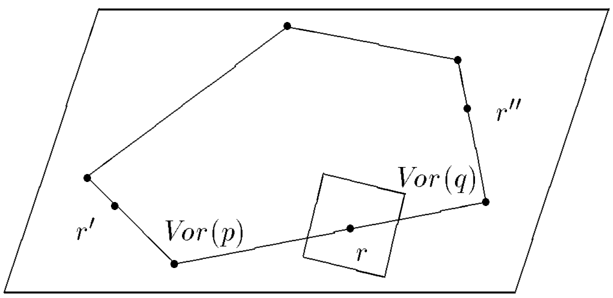

Figure 5: $[r', r]$ (resp. $[r, r'']$) is the polygonal chain of p (resp. q). The point r lies in bisector plane of p and q.

is called the *bisector plane*) such that the corresponding polygonal chains have a common point $r \in Q$ (see Fig. 5).

It should be noted that the description of $V(p)$ does not include all points of intersection the boundary of Voronoi cell $Vor(p)$ with the convex hull H, namely with the edges of the convex hull. It contains only

- the vertices of $V(p)$ that correspond the intersections of segments of $Vor(B_3)$ and the convex hull H. In other words, the point $q \in H$ is the vertex of $V(p)$ if and only if q is equidistant from at least three points in B_3 (p is one of them).
- the "edges" of $V(p)$. The edge of $V(p)$ connects two vertices of $V(p)$ and separates two surface Voronoi cells, one of them is $V(p)$. In fact any edge of $V(p)$ is the polygonal chain in the convex hull. In spite of this we store only endpoints of the edge and the pair of data points whose surface Voronoi cells are shared by this edge.

At first we solve the following problem.

Problem P1. Given a convex hull CH with n vertices, a plane π which intersects CH. Given a set S of m points in the convex hull which lie in the same halfspace bounded by π. For any point $p \in S$, its Voronoi cell in $Vor(S)$ intersects the polygon $\pi \cap CH$. Compute the surface Voronoi cells $V(p)$, $p \in S$.

We will solve this problem in $O(m \lg m \lg n)$ time. For convenience, we assume that the plane π is horizontal ($z = const$) and the points of S lie below π. We split the problem into two subproblems. In the first problem we construct the surface Voronoi cells in $Vor(S) \cap CH \cap \pi_+$ where π_+ is the upper halfspace bounded by π. In the second one we consider the lower halfspace π_- bounded by π. In order to obtain the general solution we glue the edges of surface Voronoi cells which intersect the plane π. (In fact we do not need to union the solutions, see Section 7.3) We show how to solve these problems in the following subsections.

7.1 Upper Halfspace.

We know that, for every point $p \in S$, its surface Voronoi cell intersects the polygon $\pi \cap CH$. The polygon $\pi \cap CH$ is partitioned into the polygonal chains $c_1, \ldots, c_k$ by the surface Voronoi cells. The polygonal chains are defined by their endpoints. Denote the endpoints of polygonal chain c_i by e_i and e_{i+1}. To find endpoints we can use the algorithms of Sections 5 and 6. (Initially, $S = B = B_3$ and we already know endpoints.) Note that k is at least m.

Let $p_i, i = 1, \ldots, k$ denote the point of S whose surface Voronoi cell contains the polygonal chain c_i. To characterize polygonal chains we use Davenport-Schinzel sequences [8, 1]. An (n, s) Davenport-Schinzel sequence, for a positive integers n and s, is a sequence composed of

n symbols with the properties that no two adjacent elements are equal, and that it does not contain, as a (possibly non-coniguous) subsequence, any alteration $a \ldots b \ldots a \ldots b \ldots$ of length $s+2$ between two distinct symbols a and b.

Lemma 8 *The sequense $(p_1, p_2, \ldots, p_k)$ is an $(m, 2)$ Davenport-Schinzel sequence. Hence $k \leq 2m-1$.*

Proof: The first condition of Davenport-Schinzel sequence is trivial. Suppose that the second one does not hold. There exist indexes $i_1 < i_2 < i_3 < i_4$ such that $p_1 = p_3$ and $p_2 = p_4$. The Voronoi cell $Vor(p_1)$ contains the quadrangle with vertices $e_{i_1}, e_{i_1+1}, e_{i_3}, e_{i_3+1}$. The interior of segment $[e_{i_1}, e_{i_3}]$ lies in the interior of Voronoi cell $Vor(p_1)$. The interior of segment $[e_{i_2}, e_{i_4}]$ lies in the interior of Voronoi cell $Vor(p_2)$. The point of intersection these segments has to interiors of Voronoi cells $Vor(p_1)$ and $Vor(p_2)$. Contradiction. $\Box$

Our first goal is to reduce the problem to the problems in which every surface Voronoi cell gives only one polygonal chain in intersection with the polygon $CH \cap \pi$. Consider the case $k > m$. Then there exists a point $p \in S$ whose surface Voronoi cell contains more than one polygonal chain. Let these chains have indexes $i_1, i_2, \ldots i_l$ in increasing order.

Using l polygonal chains of p we can reduce our problem to l subproblems. Then the sets of the subproblems are

$$\{p_{i_1}, \ldots, p_{i_2-1}\},$$

$$\{p_{i_2}, \ldots, p_{i_3-1}\},$$

$$\ldots$$

$$\{p_{i_l}, p_{i_l+1}, \ldots, p_m, p_1, \ldots, p_{i_1-1}\}.$$

The polygonal chains of the subproblems have the following endpoints

$$\{e_{i_1+1}, \ldots, e_{i_2}\},$$

$$\{e_{i_2+1}, \ldots, e_{i_3}\},$$

$$\ldots$$

$$\{e_{i_l+1}, \ldots, e_m, e_1, \ldots, e_{i_1}\}.$$

Using this technique we can reduce our problem to the subproblems in which every data point has only one polygonal chain. This can be done in $O(m)$ time such that the total number of data points of subproblems will be $O(m)$. It should be noted that the solution of our problem is the union of the solution of the subproblems.

We now sketch the algorithm for reduction. We go through the polygonal chains $c_1, \ldots, c_k$. For current polygonal chain c_j, if the point p_j has already occurred for the previous polygonal chain i then extract the polygonal chains with indexes from i to $j-1$ into the subproblem. Note that the point p_i is included into both extracted and remained subproblems. Note that the number of extracted chains can be less than $j-i$ because of some of the chains can be extracted before this moment. At the end, we form the last subproblem with the remaining chains. All of these subproblems satisfy the condition that the number of the polygonal chains is equal to the number of data points.

Now we consider the problem for the case $k = m$. We apply the divide-and-conquer approach. Divide the set S into two subsets

$$S_1 = \{p_1, \ldots, p_{\lfloor \frac{k}{2} \rfloor}\} \text{ and } S_2 = S \setminus S_1.$$

Using previous algorithm solve the problem for these sets. The surface Voronoi cells for S can be obtained from the surface Voronoi cells for S_1 and S_2 in usual manner.

Lemma 9 *Let A be a finite set of points in R^d, p be a point of A and q be a point in the furthest-point Voronoi cell of p. Then $Vor(p)$ contains a ray starting at q in the direction from p to q.*

It is not difficult to see that there is a path C of edges of the surface Voronoi cells for S such that

• every edge of C shares the surface Voronoi cells for the points from different subsets S_1 and S_2

• the path C starts from e_1 and ends in $e_{\lfloor k/2 \rfloor+1}$

• the path C partition the surface $CH \cap \pi_+$ into two parts corresponding to S_1 and S_2.

We explain how to find the endpoints of edges of the merge chain C. Note that endpoints lie on the edges of surface Voronoi cells for S_1 and S_2. Algorithm starts in the point e_1 which is endpoint of an edge that separates the surface Voronoi cells $V(p_1)$ and $V(p_k)$. Consider the moment when we found an endpoint e on the edge that separates the surface Voronoi cells $V(p_i)$ and $V(p_j)$. Without loss of generality we can assume that $p_i, p_j \in S_1$ and $i < j$. We are entering to the surface Voronoi cells $V(p_j)$. e is the second endpoint of an edge that separates $V(p_i)$ and the surface Voronoi cell of some point of S_2, say p_l. e is the first endpoint of an edge that separates the surface Voronoi cells $V(p_j)$ and $V(p_l)$.

We show how want to find an edge of $V(p_j)$ that occurs if we continue the motion of point e. The motion lies in the convex polygon which is the intersection of the convex hull CH and the bisector plane of points p_j and p_l. This can be done by tracing the boundary of $V(p_j)$ in direction opposite to the direction of polygonal chains $p_1, \ldots, p_k$ (in the projection on the plane π). Let current edge E of the boundary of $V(p_j)$ separates $V(p_j)$ and $V(p_s)$. To detect whether this edge occurs in the motion of e, we compute the intersection of the convex hull CH and the line L that is the set of points equidistant from the points p_j, p_l and p_s.

To do this we use the algorithm of Dobkin and Kirkpatrick [9]. Their algorithm takes

- $O(n \log n)$ time to preprocess the convex hull CH, and
- $O(\log n)$ time to find the intersection of the line L and the convex hull R.

If line L does not intersect the convex hull we pass the edge E and take the next edge on the boundary $V(p_j)$.

Let us the line L intersects the convex hull in two points. Choose the point e' of them that is the first in the motion of e. (The direction of the motion in the polygon $CH \cap \{t \mid dist(t, p_j) = dist(t, p_l)\}$ can be defined using the halfplane $\{t \mid dist(t, p_j) = dist(t, p_l) \text{ and } dist(t, p_j) \leq dist(t, p_i)\}$ or $\pi_+ \cap \{t \mid dist(t, p_1) = dist(t, p_k)\}$ for $e = e_1$.) This point belongs to the bisector plane of p_j and p_s. We can test whether it belongs to the edge E in $O(1)$ time. Consider the point $\frac{p_j + p_s}{2}$ which is the median of the segment $[p_j, p_s]$. The edge E does not contain the point e' if and only if the line through endpoints of E separates the points e' and $\frac{p_j + p_s}{2}$ in the bisector plane of p_j and p_s. If the edge E does not contain the point e', we pass the edge and take the next edge on the boundary $V(p_j)$.

In a similar way we find the point on an edge of $V(p_l)$ that occurs in the motion of point e. Determine the point that occurs first in the motion of point e. This point is the next point and the next edge of merge chain is found. The tracing of the boundary of the surface Voronoi cell that are not changed can be continued. This is important for the running time because we try to detect the intersection with an edge at most two times.

To prove correctness of the algorithm we need the following lemmas.

Lemma 10 *For any $j \leq \lfloor k/2 \rfloor$ and $l > \lfloor k/2 \rfloor$, the intersection of the surface Voronoi cell $V(p_j)$ and the bisector plane of p_j and p_l is a polygonal chain.*

Let $\pi(i, j)$ denote the bisector plane of points p_i and p_j.

Lemma 11 *For any i, j, $i < j \leq \lfloor k/2 \rfloor$ and $l > \lfloor k/2 \rfloor$, the intersection of polygonal chains $V(p_l) \cap \pi(i, l)$ and $V(p_l) \cap \pi(j, l)$ contains at most one point. In other words, the surface Voronoi cell of p_l cannot contain two distinct points equidistant from p_i, p_j, and p_l.*

Theorem 12 *The algorithm for solving the problem P1 is correct.*

Proof: Lemma 10 provides an existence of endpoints of e in both $V(p_j)$ and $V(p_l)$. By Lemma 11, the endpoints which are computed by the algorithm in $V(p_l)$ form a monotone sequence. Hence the tracing of edges is correct. □

7.2 Lower Halfspace.

The algorithm for constructing the surface Voronoi cells in lower halfspace bounded by π is similar to one for the upper halfspace. We explain only differences. We do not need to reduce the problem to the subproblems with one polygonal chain per data point. We can think that the polygonal chains corresponding to the "distinct" data points because of the different polygonal chains of the same data point are not connected in the lower halfspace.

7.3 Furthest Neighbors.

Now we have solved the problem P1. This means that we know the intersection of the convex hull H with the Voronoi diagram of B_3. It remains to show how, for any point of R, find its furthest neighbor among the points of B_3.

We also divide the problem into two subproblems, for R_1 and R_2. Let us consider the subproblem for R_2. First we construct the Delaunay triangulation using duality. For a polygonal chain in $H \cap \pi$, take any point from it. These points are the vertices of the plane graph. Connect two vertices by edge if the corresponding surface Voronoi cells have a common edge. To obtain triangulation we add any edges that connect the vertices corresponding to the same data point. Any edge of triangulation partition the vertices of graph (except endpoints of the edge) into two parts. By Lemma 13, there exists an edge of the triangulation such that the number of vertices on any side is at least $\lfloor k/3 \rfloor - 1$ where k is the number of vertices of the graph.

Lemma 13 *Given a triangulation of a convex polygon with k vertices. There exists an edge of the triangulation such that the number of vertices on any side of the edge is at least* $\lfloor k/3 \rfloor - 1$.

Proof: We can assume that the convex polygon is regular. Consider a triangle of the triangulation that contains center of the polygon. Consider the longest side of the triangle. It is easy to see that this edge suits to the lemma. □

The edge which give fair split of the polygon can be found in $O(k)$ time. Denote it by $e = (u, v)$. Let p and q be the data points corresponding to the vertices u and v.

Case 1. $p \neq q$.

There is an edge of the surface Voronoi diagram that separates the surface Voronoi cells $V(p)$ and $V(q)$. Choose any point r from it (for example, one of the endpoints). Choose any points p' and q' from the corresponding polygonal chains. The upper half of the convex hull H can be partitioned into two parts as follows. The intersection of the convex hull and the plane passing through three points p, p' and r is the convex polygon. Take the part of it in the upper halfspace between the points p' and r. The polygonal path from q' and r can be constructed in similar way. Thus we have a path from p' and q' that separates $H \cap \pi_+$ into two parts. It is easy to partition the points of R_2 into two subsets corresponding these parts.

Consider any point t of R_2. Compute the halfspace bounded by the bisector plane of p and q that contains the point t. If t lies in the same halfspace as p' then compute the halfspace bounded by the plane passing through p, p' and r that contains T. Otherwise we test the side of the plane passing through q, q' and r. Now we divide the problem into two subproblems which can be solved recursively. Note that the vertices u and v have to be included in both tasks.

Case 2. $p = q$

The partition algorithm is very simple. The points of R_2 are divided by the plane passing through p, p' and q' (the points p' and q' are chosen as in previous case).

At the end of the recursion (if the number of data points is $O(1)$ or the number of vertices of the graph is $O(1)$) the computing of the furthest neighbors is straightforward.

The points of R_1 can be handled in similar way.

8 Analysis

We now analyze the running time of the algorithm. It is clear that the overall time is a constant times the running time of computing the red-blue diameter. Let $T(n, m)$ denotes the time for computing the red-blue diameter of n red points R and m blue points B.

Lemma 14 $T(n, m) \leq c(n + m)\log^2(n + m)$.

Thus we proved the following theorem

Theorem 15 *The diameter of a set of n points in three-dimensional space can be computed in* $O(n \log^2 n)$.

Acknowledgment. The author thanks Raimund Zeidel, Michiel Smid, Guenter Rote and Frank Dehne for useful comments and for pointing out errors in previous version of this paper.

References

[1] P. K. Agarwal and M. Sharir, *Davenport-Schinzel sequences and their geometric applications*, Cambridge University Press, Cambridge, UK, 1995.

[2] N. M. Amato, M. T. Goodrich, and E. A. Ramos, *Parallel algorithms for higher-dimensional convex hulls*, Proceedings 35th Annual IEEE Symposium on Foundations of Computer Science, 1994, pp. 340–347.

[3] H. Brönninman, B. Chazelle, and J. Matoušek, *Product range spaces, sensitive sampling, and derandomization*, Proceedings 34th Annual IEEE Symposium on Foundations of Computer Science, 1993, pp. 400–409.

[4] J. L. Bentley and T. A. Ottmann, *Algorithm for reporting and counting geometric intersections*, IEEE Transactions on Computers, 28, 1979, pp. 643–647.

[5] B. Bollobás, *Graph theory, An introductory course*, Springer-Verlag, New York, 1979.

[6] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, *Diameter, width, closest line pair, and parametric search*, Discrete & Computational Geometry, 10 (1993), pp. 145–158.

[7] K. L. Clarkson and P. W. Shor, *Applications of random sampling in computational geometry*, II. Discrete & Computational Geometry, 4 (1989), pp. 387–421.

[8] H. Davenport and A. Schinzel, *A combinatorial problem connected with differential equations*, Amer. J. Math., 87 (1965), pp. 684–689.

[9] D. P. Dobkin and D. G. Kirkpatrick, *Fast detection of polyhedral intersection*, Theor. Comput. Sci., 27 (1983), pp. 241–253.

[10] H. Edelsbrunner, *Computing the extreme distances between two convex polygons*, J. Algorithms, 6 (1985), pp. 213–224.

[11] D. T. Lee, *Proximity and reachability in the plane*, Tech. Rep. No.R-831, Coordinate Sci. Lab., Univ. of Illinois at Urbana, IL, 1978.

[12] D. T. Lee , *Two-dimensional Voronoi diagrams in the L_p-metric*, J. ACM, 27 (1980), pp. 604–618.

[13] J. Matoušek and O. Schwarzkopf, *On Ray Shooting in Convex Polytopes*, Discrete & Computational Geometry, 10 (1993), pp. 215–232.

[14] J. Matoušek and O. Schwarzkopf , *A deterministic algorithm for the three-dimensional diameter problem*, Comput. Geom. Theory Appl., 6 (1996), pp. 253–262.

[15] N. Megiddo, *Applying parallel computation algorithms in the design of serial algorithms*, J. ACM, 30 (1983), pp. 852–865.

[16] F. P. Preparata and S. J. Hong, *Convex hulls of finite sets of points in two and three dimensions*, Commun. ACM, 20, 2 (1977), pp. 87–93.

[17] F. P. Preparata and M. I. Shamos, *Computational geometry, an introduction*, Springer-Verlag, New York, 1985.

[18] E. A. Ramos, *Construction of 1-d lower envelopes and applications*, Proceedings 13th Annual ACM Symposium on Computational Geometry, 1997, pp. 57–66.

[19] M. I. Shamos and D. Hoey, *Closest-point problems*, Proceedings 16th Annual IEEE Symposium on Foundations of Computer Science, 1975, pp. 151–162.

[20] A. C. Yao, *On constructing minimum spanning trees in k-dimensional spaces and related problems*, SIAM Journal on Computing, 11 (1982), pp. 721–736.