Discrete Comput Geom 26:233–244 (2001)
DOI: 10.1007/s00454-001-0029-8

Discrete & Computational
Geometry
© 2001 Springer-Verlag New York Inc.

An Optimal Deterministic Algorithm for Computing the Diameter of a Three-Dimensional Point Set

E. A. Ramos

Max-Planck-Institut für Informatik,
Stuhlsatzenhausweg 85,
66123 Saarbrücken, Germany
ramos@mpi-sb.mpg.de

Abstract. We describe a deterministic algorithm for computing the diameter of a finite set of points in $\mathbb{R}^3$, that is, the maximum distance between any pair of points in the set. The algorithm runs in optimal time $O(n \log n)$ for a set of n points. The first optimal, but randomized, algorithm for this problem was proposed more than 10 years ago by Clarkson and Shor [11] in their ground-breaking paper on geometric applications of random sampling. Our algorithm is relatively simple except for a procedure by Matoušek [25] for the efficient deterministic construction of epsilon-nets. This work improves previous deterministic algorithms by Ramos [31] and Bespamyatnikh [7], both with running time $O(n \log^2 n)$. The diameter algorithm appears to be the last one in Clarkson and Shor's paper that up to now had no deterministic counterpart with a matching running time.

1. Introduction

We consider the problem of computing the *diameter* of a given set P of n points in $\mathbb{R}^3$, that is, the maximum distance between any pair of points in P.

Previous Work. Yao [33] first showed that the problem can be solved in subquadratic time; he gave a deterministic algorithm with running time $O((n \log n)^{1.8})$. Reporting on the status of the problem in 1985, Preparata and Shamos [29] wrote: "In spite of its apparent simplicity, the computation of the diameter of a three-dimensional set has been a source of frustration to a great many workers." Then, Clarkson and Shor [11], in their ground-breaking paper on applications of random sampling in computational geometry, gave a simple randomized algorithm that runs in expected time $O(n \log n)$, which is optimal [29]. They solve the diameter problem through a reduction to the problem of computing the intersection of *congruent* (equal radius) balls. Since then, it has been a

challenge to match that time complexity with a deterministic algorithm: First, Agarwal et al. [1] gave an algorithm with running time $O(n^{4/3+\varepsilon})$ and then Chazelle et al. [8] improved the time to $O(n^{1+\varepsilon})$, where $\varepsilon > 0$ is arbitrary and the constant in the O notation depends on ε; next, Matoušek and Schwarzkopf [27] further improved it to $O(n \log^c n)$ where c is a constant for which they do not give an explicit bound and is possibly very large; then Amato et al. [3] improved the time to $O(n \log^3 n)$; finally, Ramos [31] improved it to $O(n \log^2 n)$. All these algorithms use the parametric search technique and deterministic (derandomized) sampling. On a different track, using only elementary tools (in particular, without using derandomization techniques), Ramos gave an algorithm with running time $O(n \log^5 n)$ [30] (that still uses parametric search), and later Bespamyatnikh [7] gave an algorithm also with running time $O(n \log^2 n)$, thus providing an alternative approach simpler than that of [31].

Our Result. In this paper we present a new optimal randomized algorithm that can be easily derandomized using current standard techniques, while preserving the optimal running time. Specifically, the derandomization only makes use of the efficient construction of *epsilon-nets* by Matoušek [25]. The algorithm is closely related to that in [8]. A new essential tool, combined with the use of random sampling, is the clustering of subproblems via planar-graph separators. In this, we follow its use by Dehne et al. [12] (and also Kühn [22] and Deng and Zhu [13]) in the context of algorithms for Voronoi diagrams in coarse-grained multicomputers. This allows for a decomposition into subproblems for which the total size of the subproblems increases only by a small factor, and so that it is always within a constant factor of the initial size. Furthermore, we observe that parametric search is not necessary. The result is a relatively simple divide-and-conquer algorithm, modulo the epsilon-net computation.

This paper consists of two main sections. In the first we review some preliminaries: intersection of congruent balls, decomposition into spherical-simplices and epsilon-nets, conflict list computation via a Dobkin–Kirkpatrick hierarchy, and clustering via graph separators; in the second we present and analyze the algorithm. Finally, we make some concluding remarks.

2. Preliminaries

2.1. *Intersection of Congruent Balls*

Let P be a set of n points in $\mathbb{R}^3$. For $p \in P$ and $r \in \mathbb{R}^+$, let $b(p, r)$ be the closed ball of radius r centered at p and let $s(p, r)$ be the bounding sphere of $b(p, r)$ (p and r may be omitted when they are understood or not relevant). Let $B(P, r) = \{b(p, r)\colon\ p \in P\}$ and $S(P, r) = \{s(p, r)\colon\ p \in P\}$. For a set B of congruent balls, let $\mathcal{B} = \mathcal{B}(B) = \bigcap_{b \in B} b$. $\mathcal{B}$ is a convex body which we call a *spherical-polytope*. The boundary $\mathrm{bd}(\mathcal{B})$ of $\mathcal{B}$ consists of facets, edges, and vertices corresponding to the intersection of one, two, and three bounding spheres, respectively (we assume that the point set P is in general position so that more than three bounding spheres have empty intersection; this can be achieved using symbolic perturbation [16], [35], [17]). A facet of $\mathcal{B}$ is the intersection of *spherical-caps* on the sphere that supports the facet (each cap is the intersection of another ball

with this supporting sphere), which we call a *spherical-polygon*. We point out that a spherical-polygon may be bounded by only two edges (or even one if B consists of just two balls), and that a circle which is the intersection of two bounding spheres can contribute more than one edge to $\mathrm{bd}(\mathcal{B})$. The *size* of $\mathcal{B}$, denoted $|\mathcal{B}|$, is the total number of facets, edges, and vertices. In general, if the radii of the balls are not equal, $|\mathcal{B}|$ can be $\Omega(|B|^2)$. However, for congruent balls, $|\mathcal{B}|$ is $O(|B|)$ as a result of a "convexity" property of the faces [21],[1] which implies that for the bounding sphere s of each ball in B, $\mathrm{bd}(\mathcal{B}) \cap s$ has at most one connected component.

The structure of $\mathcal{B}$ is similar to that of a Euclidean Voronoi diagram in the plane. In fact, Clarkson and Shor [11] used their random incremental approach to construct $\mathcal{B}$ in optimal expected time $O(n \log n)$ much in the way the Euclidean Voronoi diagram is constructed. An optimal deterministic algorithm was given by Amato et al. [3], by a derandomization of a divide-and-conquer approach. The previous best algorithm for the diameter problem in [31] was based on such an optimal algorithm, in combination with parametric search. In contrast, the new algorithm requires the computation of an intersection of congruent balls only for small collections, and so a simpler nonoptimal algorithm suffices.

2.2. *Spherical-Simplices and Epsilon-Nets*

Spherical-Simplices. A spherical-polytope $\mathcal{B}$ is decomposed into *spherical-simplices* as follows. First, each facet is decomposed into *spherical-triangles* by taking an arbitrary vertex of the facet and joining it to each of the other nonadjacent vertices of the facet with a geodesic segment (note that such geodesic segments do not intersect inside the facet). Finally, an arbitrary vertex of $\mathcal{B}$ is selected and joined to each of the spherical-triangles. This results in a decomposition of $\mathcal{B}$ into a collection of $O(|\mathcal{B}|)$ *spherical-simplices*.[2] For $\mathcal{B} = \mathcal{B}(B)$, we denote this decomposition by $\mathcal{T}(B)$. Its size is $O(|B|)$. For a set of spheres S and a spherical-simplex Δ, the *conflict list* $S_{|\Delta}$ is the set of those $s \in S$ such that the *outside* of s (the complement of the ball bounded by s) has a nonempty intersection with Δ (equivalently, $s \notin S_{|\Delta}$ iff the ball bounded by s contains Δ). Correspondingly, for a set of points P, $P_{|\Delta} \subseteq P$ denotes the points contained in Δ.

Range Spaces. A *range space* is a pair $(X, \mathcal{R})$ where X is a ground set and $\mathcal{R} \subseteq 2^X$ is a collection of *ranges*. $R \in \mathcal{R}$ is called a *range*. Spherical-simplices define a range space $(P, \mathcal{S}_P)$ on a set P of points as follows: $R \subseteq P$ is in $\mathcal{S}_P$ if for some radius $r > 0$ and some spherical-simplex Δ determined by spheres of radius r, $R = \mathrm{Centers}(S_{|\Delta})$, where $S = S(P, r)$ and Centers() denotes the corresponding set of center points. The

[1] For completeness: Let p, q be two points on $\mathrm{bd}(\mathcal{B})$ both on the same bounding sphere s (assuming $|B| \geq 2$, they cannot be antipodal). Then the *geodesic* segment $\widehat{pq}$ connecting p and q on s (a portion of a circle with radius equal to the radius of s) is also on $\mathrm{bd}(\mathcal{B})$, for otherwise another ball b' would contain p, q but not some other point in $\widehat{pq}$, and so b' would have a radius greater than that of b.

[2] We point out that we have no canonical way to choose the vertex used in the triangulation of each facet polygon, so that the resulting decomposition into spherical-simplices satisfies the properties of *configuration spaces* [11], [26], [28] needed for stronger sampling bounds to hold (choosing the bottom vertex does not work). Fortunately, we do not need such stronger bounds; it suffices to have a triangulation.

range space $(P, \mathcal{S}_P)$ has a constant *VC-dimension*[3] [20], [32]. This is most easily verified through *linearization* [2], [27], [34]. For this, we only need to verify that for a spherical-simplex Δ and a sphere s, whether the intersection between Δ and the outside of s is nonempty can be described by a first-order predicate of size $O(1)$ in the theory of real closed fields (one formed from polynomials inequalities using Boolean connectives and quantifiers). See [27] for a justification of this.

Epsilon-Nets. An ε-net N for a range space $(X, \mathcal{R})$ is a subset of X such that for each $R \in \mathcal{R}$, if $|R| > \varepsilon|X|$, then $N \cap R \neq \emptyset$ [20]. In particular, using the contrapositive (if $N \cap R = \emptyset$, then $|R| \leq \varepsilon|X|$) leads to an effective way to decompose a problem into subproblems with smaller size. Matoušek [25] has shown that, for a *linearizable* range space $(X, \mathcal{R})$ of constant VC-dimension d, there is a positive fraction $\delta = \delta(d)$ such that for $r \leq |X|^{\delta}$, a $(1/r)$-net of size $O(r \log r)$ can be computed in time $O(|X| \log r)$. Linearization translates the computation of an ε-net for the range space $(P, \mathcal{S}_P)$ into the computation of an ε-net for ranges induced on a point set by the union of $O(1)$ simplices in a higher-dimensional space $\mathbb{R}^d$, to which the algorithm in [25] directly applies.

2.3. *Conflict List Computation*

Let B be a set of balls and let S be a set of spheres, all congruent. We want to compute efficiently all the conflict lists $S_{|\Delta}$, $\Delta \in \mathcal{T}(B)$. This is done by computing the conflicts of each $s \in S$ as follows: First determine a point v in $s \cap \mathrm{bd}(\mathcal{B})$, if it exists, using a *Dobkin–Kirkpatrick hierarchy* [14], and then determine all the conflicts by walking on the triangulation of $\mathrm{bd}(\mathcal{B})$ while following the intersection of s and $\mathcal{B}$. If $s \cap \mathcal{B}$ is empty, then $\mathcal{B}$ is either in the outside or in the inside of s, in which case s conflicts with all or none of the spherical-simplices, respectively.

Next, we elaborate on the construction of the hierarchy and on the walk.

Dobkin–Kirkpatrick (DK) Hierarchy. Let $B = B(P, r)$. A hierarchy for $\mathcal{B}(B)$ similar to that for a three-dimensional convex polytope by Dobkin and Kirkpatrick [14] can be defined for $\mathcal{B}(B)$. The *DK hierarchy* is a sequence $\mathcal{B}_i = \mathcal{B}(B_i)$, $i = 0, \ldots, k = O(\log|B|)$, where $B_0 = B$, $B_{i+1} \subseteq B_i$, $|B_k| = O(1)$, $|B_{i+1}| \leq \alpha|B_i|$ for some constant $0 < \alpha < 1$, and $D_i = B_i - B_{i+1}$ consists of a set of balls such that: (i) the facets of $b, b' \in D_i$ in $\mathcal{B}_i$ are not adjacent, (ii) the facet of $b \in D_i$ in $\mathcal{B}_i$ has $O(1)$ adjacent facets in $\mathcal{B}_i$. Thus, $\mathcal{B}_i$ is obtained from $\mathcal{B}_{i+1}$ by removing a number of disjoint constant-complexity "caps." The existence of such a set D_i follows from the planarity of the adjacency graph of $\mathrm{bd}(\mathcal{B}_i)$ [14]. The DK hierarchy can be computed in time $O(|B|)$.

Let $s = s(p, r)$ and $b = b(p, r)$. The DK hierarchy is used to determine a point v in the intersection of s and $\mathcal{B}(B)$, if it exists (this is also similar to the use of the DK hierarchy for convex polytopes). Start by locating a point v_k in $s \cap \mathcal{B}_k$ in time $O(1)$ if it exists, else halt with failure; from v_{i+1} in $\mathcal{B}_{i+1}$ one obtains v_i in $\mathcal{B}_i$ in time $O(1)$, if

[3] Alternatively, there is the concept of VC-exponent [19], which is simpler to state and implied by the VC-dimension property, and it is sufficient for our purposes: $(X, \mathcal{R})$ has VC-exponent at most c if for each $Y \subseteq X$, $|\mathcal{R}_{|Y}| = O(|Y|^c)$, where $\mathcal{R}_{|Y} = \{R \cap Y\colon\ R \in \mathcal{R}\}$.

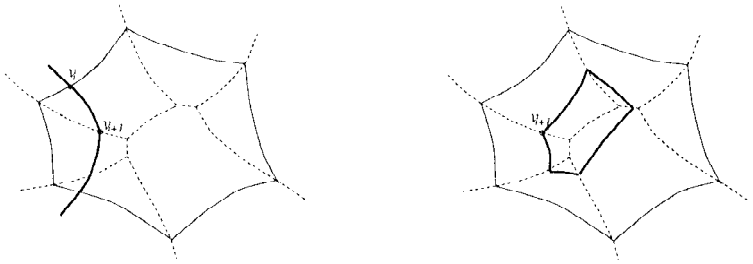

Fig. 1. Search in the DK hierarchy: The dashed lines show edges in $\mathcal{B}_{i+1}$; the thin continuous lines show the boundary of the bottom of the cap containing v_{i+1}; the thick continuous lines show $s \cap \mathcal{B}_{i+1}$. On the left, a new point $v_i \in s \cap \mathcal{B}_i$ is found by checking the boundary of the bottom of the cap. On the right, this fails because s intersects the cap but not its bottom; hence, $s \cap \mathcal{B}_i = \emptyset$.

it exists, as follows: either $v_i = v_{i+1}$ because v_{i+1} is already a point in $\mathcal{B}_i$, or only the $O(1)$ edges that bound the "bottom" of the cap containing v_{i+1} need to be checked (if s intersects one of the caps removed from $\mathcal{B}_{i+1}$ to obtain $\mathcal{B}_i$, but not its bottom, then it cannot intersect $\mathcal{B}_i$). If v_{i+1} does not exist, halt with failure. See Fig. 1. Thus, in time $O(\log|B|)$, a point $v = v_0$ in $s \cap \mathcal{B}$ is obtained, if it exists.

Walk. The walk for s starts at the point v, and follows the intersection of s and $\mathcal{B}$ through the triangulation (for which an appropriate adjacency data structure is available). This walk takes time linear in the number of conflicts found: This is the case because the walk for s only visits triangles that actually conflict with s and, on the other hand, a given triangle t can be visited at most $O(1)$ times (because the boundary of $s \cap \mathcal{B}$ intersects the boundary of t at most $O(1)$ times). Therefore, the total time for computing the conflict lists is $O(|S|\log|B|)$ plus an amount of time proportional to the total conflict list size.

2.4. *Clustering via Planar-Graph Separators*

The goal of the clustering is to minimize the size of the boundary between subproblems and, hence, to minimize the number of elements shared by subproblems. In this, we follow the work of Dehne et al. [12]. First, we recall the planar-graph separator theorem of Lipton and Tarjan [23] in the particular form we need.

Theorem 2.1. *Let $\mathcal{G}$ be a planar graph node set V and arc set E, whose node degree is bounded by a constant. Then V can be partitioned into sets V_1 and V_2 such that $|V_1|, |V_2| \leq \lceil |V|/2 \rceil$ and the number of arcs connecting V_1 and V_2 is $O(\sqrt{|V|})$. Furthermore, such sets can be determined in $O(|V|)$ time.*

The original separator theorem guarantees the existence of a node separator of size $O(\sqrt{|V|})$. Because of the bounded vertex degree, this implies an arc separator of the same size times a constant. Also the original theorem guarantees only a $\frac{2}{3}$ to $\frac{1}{3}$ balance, but by iterating it the $\frac{1}{2}$ to $\frac{1}{2}$ balance is obtained [23, Corollary 3]. Though this adds

some further unnecessary complication to the algorithm, it facilitates the argument and computation below.

Let $\eta = |V|$. As in [12], the clustering is performed by iterating the separator theorem, until *clusters* (groups) consisting of at most t nodes are obtained. Thus, with $l = \lceil \log(\eta/t) \rceil$ the total separator size is big-O of

$$\sum_{i=0}^{l} 2^i \left(\frac{\eta}{2^i}\right)^{1/2} = O\left(\frac{\eta}{t^{1/2}}\right).$$

We now apply the clustering to our problem: Let $\mathcal{B} = \mathcal{B}(B)$ for some set of balls B, and let $\mathcal{T}(B)$ be its decomposition into spherical-simplices. Let $\mathcal{G}$ be the dual graph of $\mathcal{T}(B)$ (nodes in $\mathcal{G}$ correspond to spherical-simplices in $\mathcal{T}(B)$, and two nodes are connected by an arc if the corresponding spherical-simplices share an edge in $\mathrm{bd}(\mathcal{B})$). Let Λ be the set of clusters resulting from the procedure just described and let S be a set of n spheres (all balls in B and spheres in S are congruent). For $\lambda \in \Lambda$, let $S_{|\lambda}$ denote the conflict list of S in λ, that is, the set of those $s \in S$ for which the intersection of the outside of s and λ is nonempty. We want to bound $\sum_{\lambda\in\Lambda} |S_{|\lambda}|$, the total conflict list size for the clustering, under the assumption that for any $\Delta \in \mathcal{T}(B)$, $|S_{|\Delta}| \leq \kappa$. We observe that the *conflict region* of any $s \in S$, that is, the union of the spherical-triangles bounding the spherical-simplices with which it conflicts, is connected (because a sphere contributes a single facet). Thus, if the conflict region of a sphere does not intersect the boundaries between clusters, then the sphere conflicts with only one cluster. This is illustrated in Fig. 2 where the boundary of a cluster is shown with a continuous line, and the boundaries of the conflict regions for some spheres are shown with broken lines. Thus, we find that

$$\sum_{\lambda\in\Lambda} |S_{|\lambda}| = n + O\left(\frac{\eta}{t^{1/2}} \cdot \kappa\right).$$

In our application, we have that B is a $(1/r)$-net. So $\eta = O(r \log r)$ and, because of the epsilon-net property, $\kappa = n/r$. Thus, choosing $t = r^{1/2}$, we have that

$$\sum_{\lambda\in\Lambda} |S_{|\lambda}| = n \cdot \left(1 + O\left(\frac{\log r}{r^{1/4}}\right)\right).$$

We will verify that choosing $r = r(n)$ sufficiently large, specifically $r = \Omega(\log^9 n)$, the total subproblem size remains $O(n)$ through all recursive levels of the computation.

Note finally that for each $\lambda \in \Lambda$, $|S_{|\lambda}| \leq \kappa \cdot t = n/r^{1/2}$.

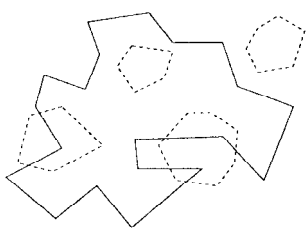

Fig. 2. Connectedness of conflict regions: if the conflict region of a sphere (each of the regions bounded by a dashed contour) does not intersect the boundary of a cluster (the continuous contour), then it lies completely inside or completely outside.

`Furthest`(P, Q)

1. If $|Q| = O(1)$ or $|P| = O(1)$, then compute by brute force and return $d_{\mathrm{F}}(P, Q)$.
2. Compute a $(1/r(|Q|))$-net $N \subseteq Q$ with respect to spherical-simplex ranges.
3. Compute $D = d_{\mathrm{F}}(P, N)$ by querying the furthest-point Voronoi diagram of N.
4. Compute $\mathcal{B}(B_N)$ and its decomposition $\mathcal{T}(B_N)$ where $B_N = B(N, D)$.
5. Cluster $\mathcal{T}(B_N)$ into a set Λ_N of clusters, each of size $\Theta(r(|Q|)^{1/2})$.
6. For each $\lambda \in \Lambda_N$, determine $P_{|\lambda}$ and $S_{Q|\lambda}$, where $S_Q = S(Q, D)$.
7. Return $\max(D, \max_{\lambda \in \Lambda_N} \texttt{Furthest}(P_{|\lambda}, \mathrm{Centers}(S_{Q|\lambda})))$.

Fig. 3. The procedure `Furthest`(P, Q) returns $d_F(P, Q)$, the furthest distance between a point in P and a point in Q.

3. Algorithm

Let $d_{\mathrm{F}}(P, Q)$ denote the furthest distance between the point sets P and Q. The procedure `Furthest`(P, Q) outlined in Fig. 3 determines $d_{\mathrm{F}}(P, Q)$. The diameter problem for a point set P is solved with a call to `Furthest`(P, P).

General Description. Step 1 takes care of the basis case by determining the distance for each pair of points. Step 2 computes a $(1/r(|Q|))$-net N from Q, with respect to spherical-simplex ranges, to be used as the sample for partitioning the problem. Here, $r(n)$ is an increasing function to be specified. Step 3 computes $D = d_{\mathrm{F}}(P, N)$, the furthest distance between the set P and the sample N. This is done by first constructing a data structure for furthest-point queries with respect to N, and then performing a query for each point in P on this data structure. Note that D is a lower bound D for $d_{\mathrm{F}}(P, Q)$. Step 4 computes the spherical-polytope for $B(N, D)$ and its decomposition into spherical-simplices. Step 5 clusters these spherical-simplices as described in the previous section. Finally, Step 6 computes the conflict lists of the clusters with respect to P and $S(Q, D)$, and Step 7 recurses on the resulting subproblems.

Correctness. We only need to argue the correctness of Step 7. First note that, because of the choice of D, P is contained in $\mathcal{B}_N$ and so the partitioning of P among the spherical-simplices of $\mathcal{T}(B_N)$ does not leave out any point from P. Now, consider a cluster $\lambda \in \Lambda_N$, and its two conflict lists $P_{|\lambda}$ and $S_{Q|\lambda}$. According to the definition of the conflict list of λ, $q \in Q \setminus \mathrm{Centers}(S_{Q|\lambda})$ implies that $b(q, D)$ contains λ, and so for any $p \in P_{|\lambda}$, the distance between p and q is at most D. Hence, any such (p, q) is not a candidate for furthest pair. This justifies recursing in Step 7 on the pairs $(P_{|\lambda}, \mathrm{Centers}(S_{Q|\lambda}))$, for each $\lambda \in \Lambda_N$.

Further Step Details. Let $m = |P|$ and $n = |Q|$. Step 1 requires time $O(m + n)$. Step 2 is performed using an algorithm by Matoušek [25]. For $r(n) = O(n^{\delta})$ where δ is

a fixed positive fraction, the algorithm requires time $O(n \log r(n))$ and the resulting N has size $O(r(n) \log r(n))$. Step 3 computes the furthest-point Voronoi diagram of N and a corresponding point location data structure so that a query can be answered in time $O(\log |N|)$. Using an algorithm by Clarkson [10], this takes time $O(|N|^{2+\varepsilon}) = O(r^3(n))$ where $\varepsilon > 0$ is a small fraction.[4] So, including the queries for the points in P, the total time required by Step 3 is $O(m \log r(n) + r^3(n))$. Step 4 can be performed in time $O(|N|^2) = O(r^3(n))$ by a simple incremental algorithm as the randomized algorithm in [11], but for a fixed permutation of N rather than for a random one. Step 5 can be performed using time linear in $|\mathcal{T}(B_N)|$ [18]; however, a simpler and less efficient algorithm suffices: Use the linear time graph separator algorithm [23] independently in each level of the iteration of the clustering procedure; this results in a running time $O(|\mathcal{T}(B_N)| \log |\mathcal{T}(B_N)|) = O(r(n) \log^2 r(n))$. In Step 6 the conflict lists $S_{Q|\lambda}$ are computed by first computing the conflict lists $S_{Q|\Delta}$ as indicated in Section 2.3, and then grouping them (eliminating duplicates in the process). The lists $P_{|\lambda}$ are computed by performing a point location in $\mathcal{T}(B_N)$ using a data structure with logarithmic query time. As is done in [11], using projection from the vertex used to obtain $\mathcal{T}(B_N)$, this is essentially a planar point location problem, so one of the known techniques could be adapted for the purpose [15], [29]. Alternatively, since $r(n)$ is relatively small, the point location could be handled directly in $\mathbb{R}^3$, with a data structure of possibly large size (say cubic) but with logarithmic query time. Thus, the computation of these conflict lists takes time $O(n \log r(n))$ plus the total conflict list size of $\mathcal{T}(B_N)$, and $O(m \log r(n))$, respectively. Note that the conflict list size of $\mathcal{T}(B_N)$ is $O((n/r(n))(r(n) \log r(n))) = O(n \log r(n))$. Thus, in summary, for $r(n) = O(n^{\min(\delta, 1/3)})$, the total time required, excluding the recursion in Step 7, is $O((m+n) \log r(n))$.

Running Time. From the previous discussion, we obtain a recurrence for the total running time of the following form: if either $m = O(1)$ or $n = O(1)$, then $T(m, n) = O(m+n)$, otherwise, for appropriate constants C_0, C_1,

$$T(m, n) \leq \sum_{\lambda} T(m_\lambda, n_\lambda) + C_0(m+n) \log r(n),$$

with the constraints

$$\begin{aligned} \sum_{\lambda} m_\lambda &= m, \\ \sum_{\lambda} n_\lambda &\leq n \cdot \left(1 + C_1 \frac{\log r(n)}{r^{1/4}(n)}\right) \\ n_\lambda &\leq \frac{n}{r^{1/2}(n)} \qquad \text{for each } \lambda. \end{aligned}$$

From this, we obtain that

$$T(m, n) \leq 2C_0(m + n \cdot U(n)) \log n,$$

[4] The algorithm in [10] is actually for the nearest-point Voronoi diagram, and it is randomized. The adaptation to furthest-point is straightforward, and its derandomization is by now standard [9], [24].

where $U(n)$ is defined as follows: First define the sequence n_i as $n_0 = O(1)$ and, for $i \geq 1$, n_i so that $n_i / r^{1/2}(n_i) = n_{i-1}$; for n let $[n]$ denote the largest n_i not greater than n; then define $U(n_0) = O(1)$ and for $i \geq 1$,

$$U(n_i) = \left(1 + C_1 \cdot \frac{\log r(n_i)}{r^{1/4}(n_i)}\right) \cdot U(n_{i-1});$$

and, finally, $U(n) = U([n])$ for arbitrary n. We can verify that $U(n)$ is nondecreasing and, as long as $r(n) = \Omega(\log^9 n)$ (we have made no effort to optimize this exponent), $U(n) = O(1)$. To verify the latter, it suffices to verify that

$$\sum_{i=0}^{\infty} \frac{\log r(n_i)}{r^{1/4}(n_i)} = O(1).$$

This is the case for $r(n) = \log^9 n$ because then $\log n_i = \log n_{i-1} + \log r^{1/2}(n_i) = \Omega(i)$, with appropriate choice of constants, and

$$\frac{\log r(n_i)}{r^{1/4}(n_i)} \leq \frac{1}{\log^2 n_i} = O\left(\frac{1}{i^2}\right).$$

Since $\sum_{i \geq 1} 1/i^2 = O(1)$, then the sum above is also $O(1)$. To complete the analysis, we verify by induction the bound for $T(m, n)$ indicated above:

$$\begin{aligned}
T(m,n) &\leq \sum_{\lambda} T(m_\lambda, n_\lambda) + C_0(m+n)\log r(n) \\
&\leq \sum_{\lambda} 2C_0(m_\lambda + n_\lambda \cdot U(n_\lambda))\log n_\lambda + C_0(m+n)\log r(n) \\
&\leq 2C_0\left(\sum_{\lambda} m_\lambda\right)\log\left(\frac{n}{r^{1/2}(n)}\right) \\
&\quad + 2C_0\left(\sum_{\lambda} n_\lambda\right) U\left(\frac{n}{r^{1/2}(n)}\right)\log\left(\frac{n}{r^{1/2}(n)}\right) \\
&\quad + 2C_0(m+n)\log r^{1/2}(n) \\
&\leq 2C_0 m \log n + 2C_0 n\left(1 + C_1\frac{\log r(n)}{r^{1/4}(n)}\right) U\left(\frac{n}{r^{1/2}}\right)\log n \\
&\leq 2C_0(m + n \cdot U(n))\log n.
\end{aligned}$$

Thus, we obtain that $T(m, n) = O((m+n)\log n)$ and, hence, that the diameter of a set of n points in $\mathbb{R}^3$ can be computed in time $O(n \log n)$.

Remarks. It would be interesting if $r(n) = \Theta(1)$ could be used, as in this case a considerably simpler algorithm, also due to Matoušek [24], could be used for the construction of the epsilon-net. Unfortunately, the analysis above does not support this possibility. In contrast with our restriction $r(n) = \Omega(\log^9 n)$, for the algorithm in [31] it is essential that $r(n) = \Theta(n^\delta)$, with an appropriate $\delta > 0$. In a randomized version of the algorithm, Step 2 finds a net by simply taking a random sample, and other steps can be implemented with simpler randomized procedures. Thus, except for the use of planar-graph separators, a relatively simple randomized algorithm is obtained.

4. Concluding Remarks

We have presented an optimal deterministic algorithm for the diameter problem in $\mathbb{R}^3$. We can view the new algorithm as a converging point for the two previous approaches leading to $O(n \log^2 n)$ time algorithms: the approach in [31] that constructs the ball intersection for a radius equal to the diameter via parametric search, on one hand, and the approach in [7] that considers the furthest point Voronoi diagram of the point set restricted to the boundary of the convex hull of the point set (but avoids computing it explicitly) on the other hand. Compared with the first approach, by having a more effective way of splitting into subproblems, we essentially do away with computing the ball intersection for a radius equal to the diameter; instead, the algorithm iteratively approaches the correct value of the diameter without actually computing the ball intersection for that radius. The resulting divide-and-conquer scheme is akin to that in the second approach but with more efficient partitioning. It remains as an open problem whether there is an optimal deterministic algorithm for the diameter problem that does not use derandomization tools, but rather exploits some further geometric facts, like in the work by Bespamyatnikh. In the same vein, in our algorithm, there might be a more geometric approach to the clustering than our use of planar-graph separators.

The idea of clustering to reduce the usual size "blow-up" of divide-and-conquer algorithms based on random sampling seems quite useful and we expect further applications. In fact, some of our previous work on computing segment intersections and abstract Voronoi diagrams [4], [6] can be simplified or improved using this approach. The connectedness of conflict regions of an object is not really necessary if there is another way to account for the influence of an object in different subproblems. We have applied the approach to the segment intersection problem [5] and thus obtained a new optimal deterministic divide-and-conquer algorithm, that is, using work $O(n \log n + k)$ where n is the number of segments and k is the number of pairwise intersections. The new algorithm works for curve segments, uses space $O(n + k)$, and runs in time $O(\log^{3/2} n)$ in the EREW PRAM model, in contrast to a previous algorithm in [4] which worked only for line segments, used $O(n \log \log n + k)$ space, and ran in time $O(\log^2 n)$. It remains to investigate further applications, and the possibility of applying the technique to higher-dimensional problems. For the latter, a major difficulty is the lack of results guaranteeing the existence of small separators.

References

1. P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete Comput. Geom.* **6** (1991), 407–422.
2. P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.* **11** (1994), 393–418.
3. N. M. Amato, M. T. Goodrich and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.* (*FOCS* 94), pp. 683–694, 1994.
4. N. M. Amato, M. T. Goodrich and E. A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 26th Annu. ACM Sympos. Theory Comput.* (*STOC* 95), pp. 672–682, 1995.
5. N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing the Arrangement of Curve Segments: Divide-and-Conquer Algorithms via Sampling. Manuscript, 1999. Short abstract in *Proc. 9th Annu.*

ACM–SIAM Sympos. Discrete Algorithms (*SODA* 00), pp. 705–706, 2000. Available from `http://www.mpi-sb.mpg.de/~ramos`
6. N. M. Amato and E. A. Ramos. On computing Voronoi diagrams by divide-prune-and-conquer. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.* (*SoCG* 96), pp. 672–682, 1996.
7. S. Bespamyatnikh. An efficient algorithm for the three-dimensional diameter problem. In *Proc. 9th Annu. ACM–SIAM Sympos. Discrete Algorithms* (*SODA* 98), pp. 137–146, 1998.
8. B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir. Diameter, width, closest line pair, and parametric searching, *Discrete Comput. Geom.* **10** (1993), 183–196.
9. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, **10** (1990) 229–249.
10. K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.* **17** (1988), 830–847.
11. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.* **4** (1989), 387–421.
12. F. Dehne, X. Deng, P. Dymond, A. Fabri and A. Khokbar. A randomized parallel 3D convex hull algorithm for coarse grained multicomputers. In *Proc. ACM Sympos. Parallel Algorithms Architectures* (*SPAA* 95), pp. 27–33, 1995.
13. X. Deng and B. Zhu. A randomized algorithm for the Voronoi diagram of line segments on coarse grained multiprocessors. *Algorithmica* **24** (1999), 270–286.
14. D. P. Dobkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoret. Comput. Sci.* **27** (1983), 241–253.
15. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg, 1987.
16. H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.* **9** (1990), 66–104.
17. I. Z. Emiris, J. F. Canny and R. Seidel. Efficient perturbations for handling geometric degeneracies. *Algorithmica* **19** (1997), 219–242.
18. M. T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. System Sci.* **51** (1995), 374–389.
19. M. T. Goodrich and E. A. Ramos. Bounded independence derandomization of geometric partitioning with applications to parallel fixed-dimensional linear programming. *Discrete Comput. Geom.* **18** (1997), 397–420.
20. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.* **2** (1987), 127–151.
21. A. Heppes. Beweis einer Vermutung von A. Vázsonyi. *Acta Math. Acad. Sci. Hungar.* **7** (1956), 463–466.
22. U. Kühn. Lokale Eigenschaften in der algorithmischen Geometrie mit Anwendungen in der Parallelverarbeitung. Inaugural-Dissertation, Fachbereich Mathematik und Informatik, Westfälische Wilhelms-Universität Münster, 1998.
23. R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Comput.* **36** (1979), 177–189.
24. J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.* (*STOC* 91), pp. 505–511, 1991. Also in *J. Comput. System Sci.* **50**, 203–208 (1995).
25. J. Matoušek. Efficient partition trees *Discrete Comput. Geom.* **8** (1992), 315–334.
26. J. Matoušek. Derandomization in computational geometry. *J. Algorithms* **20** (1996), 545–580.
27. J. Matoušek and O. Schwarzkopf. A deterministic algorithm for the three-dimensional diameter problem. In *Proc. 25th Annu. ACM Sympos. Theory Comput.* (*STOC* 93), pp. 478–484, 1993. Revised version in *Comput. Geom. Theory Appl.* **6** (1996), 253–262.
28. K. Mulmuley. *Computational Geometry: An Introduction through Randomized Algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1994.
29. F. Preparata and M. I. Shamos, *Computational Geometry – An Introduction*, Springer-Verlag, New York, 1985.
30. E. A. Ramos. An algorithm for intersecting equal radius balls in $\mathbb{R}^3$. *Comput. Geom. Theory Appl.* **8** (1995), 57–65.
31. E. A. Ramos Constructing 1-d lower envelopes and applications. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.* (*SoCG* 97), pp. 57–66, 1997.

32. V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl*. **16** (1971), 264–280.
33. A. C. Yao. On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput*. **11** (1982), 721–736.
34. A. C. Yao and F. F. Yao. A general approach to D-dimensional geometric queries. In *Proc*. 17*th Annu*. *ACM Sympos. Theory Comput*. (*STOC* 85), pp. 163–168, 1985.
35. C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. System Sci*. **39** (1989), 236–246.

Received May 10, 2000, *and in revised form November* 3, 2000. *Online publication June* 22, 2001.