Staying in the Middle: Exact and Approximate Medians in $\mathbb{R}^1$ and $\mathbb{R}^2$ for Moving Points *

Pankaj K. Agarwal† Mark de Berg‡ Jie Gao§ Leonidas J. Guibas¶ Sariel Har-Peled‖

1 Introduction

We study the problem of "staying in the middle": we have a set of points moving in a geometric space and wish to maintain another point (possibly one of the given points, but not necessarily) that stays continuously "in the middle" of the moving set. More precisely, in $\mathbb{R}^1$ we wish to maintain the median or, more generally, a point of rank k. In $\mathbb{R}^2$ we wish to maintain suitable analogs of the median and point of rank k, defined as follows. Let P be a point set in $\mathbb{R}^d$, and define the *depth* $\Delta(p)$ of a point $p \in \mathbb{R}^d$ as the minimum number of points of P on either side of any hyperplane passing through p. A point with depth at least δn is called a *δ-center point*. A $1/(d+1)$-center point is called just a *center point*, and it generalizes the concept of median. It has been proven using Helly's Theorem that any point set has a center point [10]. For $\mathbb{R}^2$, given some $k \leq n/2$, the set of points with depth at least k is called k-th *depth contour*, denoted by D_k.

We study both exact and approximate algorithms for kinetic medians (in $\mathbb{R}^1$) and kinetic center points and depth contours (in $\mathbb{R}^2$). As we will see, in both cases the approximate algorithms offer far greater stability and maintenance efficiency, for a very modest loss in the quality of the partitions they can generate. Another advantage of approximation algorithms is that we can maintain not only the (median or) center point of the current configuration but also a compact representation of the trajectory of an approximate (median or) center point over the entire motion of points, so we can answer various queries on the center point of any future configuration of points. Such queries are central to spatial-temporal database systems. A main tool we develop for these approximate algorithms is the efficient maintenance of an ε-approximation of a range space under point insertion and deletion — which we believe to be of independent interest.

Related work. Finding the median (or more generally the point of rank k, for some given k) of a set of points in $\mathbb{R}^1$ can be done in linear time. If the points move on the real line, then maintaining the point of rank k is closely related to the concept of *levels* in the arrangement defined by the trajectories in the xt-plane, as defined next. The *level* of a point $p \in \mathbb{R}^2$ in an arrangement $\mathcal{A}(\Gamma)$ defined by a set Γ of x-monotone curves is the number of curves in Γ lying below p. Let $\Lambda_k(\Gamma)$ denote the closure of the edges of $\mathcal{A}(\Gamma)$ whose level is k; $\Lambda_k(\Gamma)$ is an x-monotone curve and it is called the k-level of the arrangement. Hence, the point of rank k at time t_0 is given by the trajectory that is on the k-level for $t = t_0$. Set $\lambda_k(\Gamma) = |\Lambda_k(\Gamma)|$, and define $\lambda_k(n) = \max\{\lambda_k(\Gamma)\}$, where the maximum is taken over all sets Γ of n curves of a given type (lines, for example, or algebraic curves of a certain maximum degree). Maintaining the median thus reduces to computing the $n/2$-level, and $\lambda_{n/2}(n)$ bounds the number of changes of the median if the trajectories in the xt-plane are of the given type. If Γ is a set of n lines, then $\lambda_k(\Gamma) = O(n^{4/3})$ [9]. For general curves, a recent result of Chan [4] implies that $\lambda_k(\Gamma) = O(n^{2-1/2s})$, where s is the maximum number of intersection points between two curves.

Maintaining an ε-approximate median reduces to computing an ε-approximate $n/2$-level, that is, an x-monotone curve lying between the $(1-\varepsilon)n/2$-level and

*P.A. was partially supported by NSF under grants CCR-00-86013 EIA-98-70724, EIA-99-72879, EIA-01-31905, and CCR-02-04118, and by a grant from the U.S.-Israeli Binational Science Foundation. M.d.B. was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301. Work by L.G. and J.G. was supported by NSF grant CCR-9910633 and grants from the Stanford Networking Research Center, the Okawa Foundation, and the Honda Corporation. S.H.-P. was supported by NSF CAREER award CCR-0132901.

†Department of Computer Science, Duke University, Durham, NC 27708, USA. E-mail: pankaj@cs.duke.edu

‡Department of Computer Science, TU Eindhoven, P.O.Box 513, 5600 MB, the Netherlands. mdberg@win.tue.nl

§Center for the Mathematics of Information, Caltech, Pasadena, CA 91125, USA. E-mail: jgao@ist.caltech.edu

¶Department of Computer Science, Stanford University, Stanford, CA 94305, USA. E-mail: guibas@cs.stanford.edu

‖Department of Computer Science, University of Illinois, Urbana, IL 61801, USA. Email: sariel@uiuc.edu

the $(1+\varepsilon)n/2$-level. There has been a lot of work on computing approximate levels. For line arrangements, Edelsbrunner and Welzl [11] showed that an ε-approximate median level with at most $\lceil \lambda_{n/2}(n)/(\varepsilon n)\rceil$ edges exists, and Matoušek [13] gave an algorithm for obtaining an ε-approximate median level of constant complexity (depending on ε only). This idea was further explored by Agarwal *et al.* [1], who obtained an algorithm to compute in time $O(n\log n/\varepsilon^2)$ an ε-approximate median of size $O(1/\varepsilon^2)$ for arrangements of lines or line segments.

In 2-D, a center point can be computed in linear time [12]. Clarkson *et al.* [7] proposed a randomized algorithm to compute an ε-approximate center point in time polynomial in the dimension and $1/\varepsilon$. Miller *et al.* used $O(n^2)$ time and space to find all the depth contours, by computing the arrangement of the n lines in the dual plane [14]. A single k-th depth contour can be computed in $O(n\log^2 n)$ time [5]. To our knowledge, there are no prior results on maintaining the exact/approximate depth contours or center points under motion.

Our results. Most of our algorithms are based on the *kinetic data structure* (KDS) framework, originally proposed by Basch *et al.* [2]. We start by briefly looking at the problem of maintaining medians and points of a given rank k for points moving in $\mathbb{R}^1$. Then we present two KDS's for maintaining a center point of a set of n points moving in the plane. The first KDS actually maintains the k-th depth contour for a given $0 \le k < n$. As a byproduct, it gives a KDS for maintaining the entire center region (the set of all center points). It requires $O(n^2)$ certificates and processes $O(n^{4+\delta})$ events under pseudo-algebraic motion of points, each event requiring $O(\log^2 n)$ time. The second KDS maintains a subset of the center points. It needs only $O(n\log n)$ certificates but processes $O(n^{7+\delta})$ events in the worst case.

Since these exact algorithms are quite expensive, we then study approximation algorithms. We first describe an algorithm for maintaining an *ε-approximation* $A \subseteq S$ of a finite range space $(S,\mathcal{R})$ under insertions and deletions of points. We show that a center point of A for a suitably defined range space is an ε-approximate center point of S. Whenever we compute A, we also compute in $O(1/\varepsilon^{O(1)})$ time a t-monotone polygonal curve γ so that $\gamma(t)$ is a center point of $A(t)$. Therefore for any t, we can compute in $O(\log(1/\varepsilon))$ time an ε-approximate center point of the points in $S(t)$ based on their current trajectories. The set A does not change as long as the trajectories of the points in S don't change. Whenever a point p changes its trajectory, we delete p from S and re-insert it with the new trajectory. Our algorithm can update A in $(\log(n)/\varepsilon)^{O(1)}$ time. The same idea can also be used to maintain an ε-approximate median of a set of points moving in $\mathbb{R}^1$.

The ε-approximation based algorithm maintains an ε-approximate median that changes $O(1/\varepsilon^4\log^2(1/\varepsilon))$ times, and the total time spent is $O(n/\varepsilon^{O(1)})$. If the trajectories of the points are known in advance, then we can improve this. For example, if the points have fixed velocities, we give a Las Vegas algorithm such that the ε-approximate median changes $O(1/\varepsilon^{4/3}\log^2(1/\varepsilon))$ times, by spending $O(n/\varepsilon^{1/3}\log(1/\varepsilon))$ expected time, thus improving the result of Agarwal et al. [1].

Due to lack of space, we omit most details from this extended abstract.

2 Exact Algorithms

Maintaining the median. Maintaining the exact median of a set S of n points moving in $\mathbb{R}^1$, or more generally, maintaining the point of rank k for a fixed k, can be done by adapting the HeapSweep algorithm [3]. We maintain (i) the point of rank k, (ii) the maximum of $S_{<k}(t)$, the points of rank less then k, in a kinetic tournament, and (iii) the minimum of $S_{>k}$, the points with rank greater than k, in a kinetic tournament. Following the analysis of [3], we prove the following.

Theorem 1 *Let S be a set of n points moving in $\mathbb{R}^1$, and let k be an integer. We can maintain the point of S of rank k using a KDS that uses $O(n)$ certificates. Assuming pseudo-algebraic motion, the number of events processed by the KDS is $O(\lambda_k(n)\log^2 n)$, and each event can be handled in $O(\log n)$ time.*

Maintaining the depth contour and center region. Given a set S of n points moving in $\mathbb{R}^2$ and an integer $0 \le k < \lceil n/2\rceil$, we describe a KDS for maintaining the k-th depth contour D_k of S as the points in S move. The center region of S is simply $D_{\lfloor n/3\rfloor}$. Note that the k-th depth contour of a set P of n stationary points in the plane is also closely related to the k-th and $(n-k)$-th levels in arrangements: if we dualize [8] P, we obtain a set P^* of n lines, and a point at depth k dualizes to a line lying between $\Lambda_k(P^*)$ and $\Lambda_{n-k}(P^*)$. Hence, the dual of points whose depth is at least k is the region lying between the lower hull of $\Lambda_{n-k}(S^*)$ and the upper hull of $\Lambda_k(S^*)$. Thus the problem at hand reduces to the following. Let $L = \{\ell_1,\ldots,\ell_n\}$ be a set of n lines in the plane, each moving independently, i.e., $\ell_i : y =$

$a_i(t)+b_i(t)x$, where $a_i(\cdot)$ and $b_i(\cdot)$ are polynomials in t. We wish to maintain the upper (or lower) hull of $\Lambda_k(L)$, the k-th level of $\mathcal{A}(L)$, as the lines in L move.

For each line $\ell \in L$, let $v^-(\ell)$ (resp. $v^+(\ell)$) be the leftmost (resp. rightmost) point of $\Lambda_k(L) \cap \ell$. Let $V = \{v^-(\ell), v^+(\ell) \mid \ell \in L\}$. The upper hull of $\Lambda_k(L)$ is the same as that of V. We maintain the upper hull of V using the kinetic data structure described in [2]. In more detail, we proceed as follows.

For a line $\ell \in L$, let $\Sigma(\ell)$ be the sequence of vertices of $\mathcal{A}(L) \cap \ell$, sorted from left to right. $\Sigma(\ell)$ partitions ℓ into the edges of $\mathcal{A}(L)$. For each edge e, let $\lambda(e)$ be the level of the points in e. Note that if e' lies immediately to the right of e, then $\lambda(e') = \lambda(e) \pm 1$ depending on the local geometric situation. We maintain the set $E(\ell)$ of all the edges on ℓ whose level is k. We add the left endpoint of the leftmost edge and the right endpoint of the rightmost edge of $E(\ell)$ (i.e., $v^-(\ell), v^+(\ell)$) to V. For each ℓ, we maintain: (i) the sequence of vertices and edges of the arrangement on ℓ, (ii) $E(\ell)$, and (iii) $v^-(\ell)$, $v^+(\ell)$. In addition, we also use a KDS to maintain the upper hull of V, as the lines in L move [2]. We will also need it to handle insertion and deletion of points.

There are three types of events that the algorithm has to handle to maintain the above structures:

(E1) Two lines ℓ_1, ℓ_2 become parallel.
(E2) Three lines ℓ_1, ℓ_2, ℓ_3 become concurrent.
(E3) An event of KDS for maintaining $\mathcal{UH}(V)$.

The KDS for maintaining $\mathcal{UH}(V)$ maintains $O(n \log n)$ certificates. We need additional $O(n^2)$ certificates to detect events of type (E1) and (E2): (i) leftmost and right vertices along each line of L, and (ii) adjacent pairs of vertices of $\mathcal{A}(L)$ along each line of L. We can show that the total number of events of all types is $O(n^{4+\delta})$, and that each event can be handled in $O(\log^2 n)$ time, leading to the following result.

Theorem 2 *Let S be a set of n points moving in the plane. We can maintain the center region of S or, more generally, the k-th depth contour of S, using $O(n^2)$ certificates. The algorithm processes $O(n^{4+\delta})$ events under algebraic motion of S, and each event can be processed in $O(\log^2 n)$ time.*

A space-efficient algorithm for maintaining a center point. We now describe a KDS for maintaining a center point of a set S of $n = 3m$ points moving in the plane that uses only $O(n \log n)$ certificates. First, let us assume that the points in S are stationary. A result by Tverberg [16] implies that S can be partitioned into m triples $S_1, \ldots, S_m$ so that the intersection of triangles $\Delta_i = \mathrm{conv}(S_i)$, $1 \le i \le m$, is nonempty. Such a partition is called a *Tverberg partition*, and any point in $\Delta = \bigcap_{i=1}^m \Delta_i$ is called a *Tverberg point*. Any point in Δ is also a center point. Moreover, a Tverberg partition in the plane can be computed in $O(n \log n)$ time [15]. Since each Tverberg triangle Δ_i can be regarded as the intersection of three halfplanes, each bounded by a line passing through a pair of points in Δ_i, the region Δ is the intersection of a set $\mathcal{H}$ of $3m = n$ halfplanes. In the dual plane, a halfplane $h \in \mathcal{H}$ is mapped to a point h^* and the intersection of the halfplanes maps to the convex hull of the set $\mathcal{H}^*$ of points. Therefore we use a KDS to maintain the convex hull $\mathcal{CH}(\mathcal{H}^*)$ [2]; it uses $O(n \log n)$ certificates.

Now as the points in S move, we wish to maintain Δ. More precisely, we compute an initial Tverberg partition of S in $O(n \log n)$ time and maintain Δ using the kinetic convex hull structure described by Basch *et al.* [2], with the following twist. Whenever Δ shrinks to a point, we update the underlying Tverberg partition so that the interior of Δ after the modification becomes nonempty. The idea is similar to the one used by Tverberg [17]. We omit the details of the events and the analysis.

Theorem 3 *One can maintain a center point of a set S of moving points in the plane using $O(n \log n)$ certificates. The algorithm processes $O(n^{7+\delta})$ events under algebraic motion of S, and each event can be processed in $O(\log^2 n)$ time.*

3 Maintaining ε-approximations

A range space is a pair $X = (S, \mathcal{R})$, where S is a set and $\mathcal{R} \subset 2^S$. S is called the set of *points*, and $\mathcal{R}$ is called the set of *ranges*. In the sequel we deal with finite range spaces, where S is finite. A subset $A \subseteq S$ is called an *ε-approximation* for X if, for every range $R \in \mathcal{R}$, $||A \cap R|/|A| - |R|/|S|| < \varepsilon$. We give a deterministic algorithm, based on [6], for maintaining an ε-approximation as points are inserted or deleted.

Theorem 4 *Given a range space $X = (S, \mathcal{R})$ of VC-dimension d, one can (deterministically) maintain an ε-approximation of X of size $O(1/\varepsilon^2 \log(1/\varepsilon))$, in $O\Big(\frac{\log^{2d+3} n}{\varepsilon^{2d+2}} (\log(\log(n)/\varepsilon))^{2d+2}\Big)$ time per insertion or deletion, for a parameter $\varepsilon > 0$.*

4 Approximation Algorithms

The ε-approximation based algorithms. Let S be a set of n points moving in $\mathbb{R}^1$, and R the set of vertically downward rays in the tx-plane. For $\rho \in R$, let $S_\rho = \{p \mid$ there is a t with $p(t) \in \rho\}$. We define the range space $\mathcal{M} = (S, \{S_\rho \mid \rho \in R\})$. Let $A \subseteq S$ be an ε-approximation of S for the range space $\mathcal{M}$. Then for any t, the median of $A(t)$ is an ε-approximate median of $S(t)$.

If the trajectories of S are algebraic then the VC-dimension of $\mathcal{M}$ is finite. Therefore we can use the algorithms of the previous section to maintain an ε-approximation of S. We can compute the median level of the trajectories of A. Since we compute the entire median level, there are no events unless the trajectory of a point in S changes. When the trajectory of a point changes, A is recomputed in $(\log(n)/\varepsilon)^{O(1)}$ time using Theorem 4, and we recompute the median level of A in $(1/\varepsilon)^{O(1)}$ additional time. We thus obtain the following.

Theorem 5 *Let S be a set of points moving in $\mathbb{R}^1$ (or $\mathbb{R}^2$), and let $\varepsilon > 0$ be a parameter. Assuming that the motion of S is algebraic, we can construct a data structure that can be updated in $(\log(n)/\varepsilon)^{O(1)}$ time whenever the trajectory of a point in S changes, and that, for any time t, can return an ε-approximate median (or ε-approximate center point) of $S(t)$ in $O(\log(1/\varepsilon))$ time based on the current trajectories of S.*

Faster off-line algorithms. An ε-approximate median of a set S of n points moving in $\mathbb{R}^1$ can be computed more efficiently off-line. However, unlike the algorithm of Section 4, the ε-approximate median we maintain is not necessarily one of the input points, and we cannot change the trajectory of points on-line.

Theorem 6 *Let S be a set of points moving in $\mathbb{R}^1$, and $\varepsilon > 0$ a constant. Assuming that their trajectories are algebraic with degree of motion μ, we can compute in $O(n\log(1/\varepsilon) + 1/\varepsilon^{O(\mu^2)})$ time a t-monotone curve γ of size $O(1/\varepsilon^2)$ so that the rank of $\gamma(t)$, for any $t \in \mathbb{R}$, lies between $n/2(1 \pm \varepsilon)$.*

We can compute an ε-approximate median with fewer breakpoints if the points have fixed velocities.

Theorem 7 *Let S be a set of points moving in $\mathbb{R}^1$ moving with fixed velocity, and $\varepsilon > 0$ a parameter. We can compute in expected $O((n/\varepsilon^{1/3})\log(1/\varepsilon))$ time a t-monotone curve γ of size $O(1/\varepsilon^{4/3}\log^2(1/\varepsilon))$ so that the rank of $\gamma(t)$, for any $t \in \mathbb{R}$, lies between $n/2(1 \pm \varepsilon)$.*

References

[1] P. K. Agarwal, J. Gao, and L. J. Guibas. Kinetic medians and kd-trees. In *Proc. 10th Annu. European Sympos. Algorithms*, pages 5–16, 2002.

[2] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999.

[3] J. Basch, L. J. Guibas, and G. D. Ramkumar. Reporting red-blue intersections between two sets of connected line segments. *Algorithmica*, 35:1–20, 2003.

[4] T. M. Chan. On levels in arrangements of curves, II: a simple inequality and its consequence. In *Proc. 44th IEEE FOCS'03*, pages 544–550, 2003.

[5] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 423–429, 2004.

[6] B. Chazelle. *The Discrepancy Method*. Cambridge University Press, 2000.

[7] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterative Radon points. *Internat. J. Comput. Geom. Appl.*, 6:357–377, 1996.

[8] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000.

[9] T. K. Dey. Improved bounds for planar k-sets and related problems. *Discrete Comput. Geom.*, 19(3):373–382, 1998.

[10] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Heidelberg, 1987.

[11] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM J. Comput.*, 15:271–284, 1986.

[12] S. Jadhav and A. Mukhopadhyay. Computing a centerpoint of a finite planar set of points in linear time. *Discrete Comput. Geom.*, 12:291–312, 1994.

[13] J. Matoušek. Construction of ϵ-nets. *Discrete Comput. Geom.*, 5:427–448, 1990.

[14] K. Miller, S. Ramaswami, P. Rousseeuw, T. Sellares, D. L. Souvaine, I. Streinu, and A. Struyf. Fast implementation of depth contours using topological sweep. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 690–699, 2001.

[15] S. H. Teng. *Points, Spheres, and Separators: a unified geometric approach to graph partitioning*. PhD thesis, School Comp. Sci., Carnegie-Mellon Univ., 1990. Report CMU-CS-91-184.

[16] H. Tverberg. A generalization of Radon's theorem. *J. London Math. Soc.*, 41:123–128, 1966.

[17] H. Tverberg. A generalization of Radon's theorem II. *Bull. Australian Math. Soc.*, 24:321–325, 1981.