
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

Staying in the middle

For a set of moving points, maintain another point (not necessarily in the set) that continuously ‘stays in the middle’.

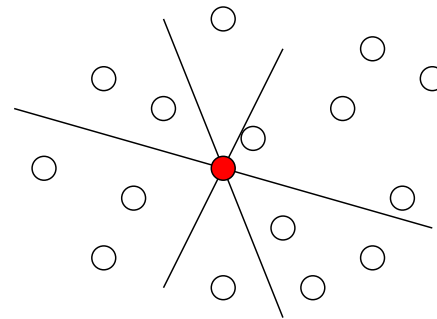
- find a point with roughly the same number of points on each side (in \mathbb{R}^1) or along each direction (in \mathbb{R}^2).

Many range query data structures, e.g., *kd*-trees, are based on balanced space partitions.

Medians and center points

A **median** of a set of points on a line (in \mathbb{R}^1) is the one with rank $\lfloor n/2 \rfloor$.

The **depth** $\Delta(p)$ of a point $p \in \mathbb{R}^2$ is the minimum number of points on either side of any line passing through p . A point with depth at least δn is called a **δ -center point**. By Helly's Theorem, a $1/3$ -center point exists for any point set and is called just a **center point**. For some $k \leq n/2$, the set of points with depth at least k is called **k -th depth contour**, denoted by D_k .



Kinetic medians and center points

Kinetic Data Structure (KDS) framework: the median/center point only changes at **discrete** times although the points move continuously.

- Maintain a number of combinatorial **certificates** whose validity certifies the correctness of the median/center point;
- When a certificate fails, i.e., an **event** happens, we update the certificate lists and the median/center point when necessary.

Kinetic medians and center points

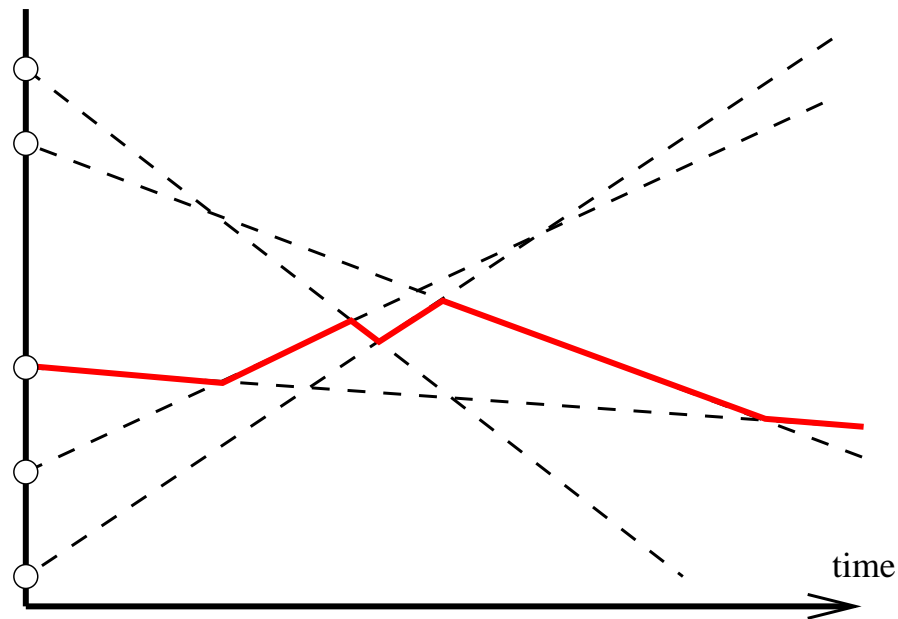
Evaluation criteria of a KDS:

- **compactness** = # certificates in the proof;
- **efficiency** = # events under pseudo-algebraic motion;
- **responsiveness** = update cost to fix an event;
- **locality** = # certificates involving any point;

Part I:
Maintaining the exact median and center points

A warm-up: maintaining the median in \mathbb{R}^1 , I

When the points move, the median follows $\lfloor n/2 \rfloor$ -level of the arrangement of the trajectories.



Maintaining the rank k point in \mathbb{R}^1 , II

Maintain three points:

- the rank k point, p_k ;
- the max among points with rank less than k , $p_{<k}$, by a kinetic tournament;
- the min among points with rank more than k , $p_{>k}$, by a kinetic tournament.

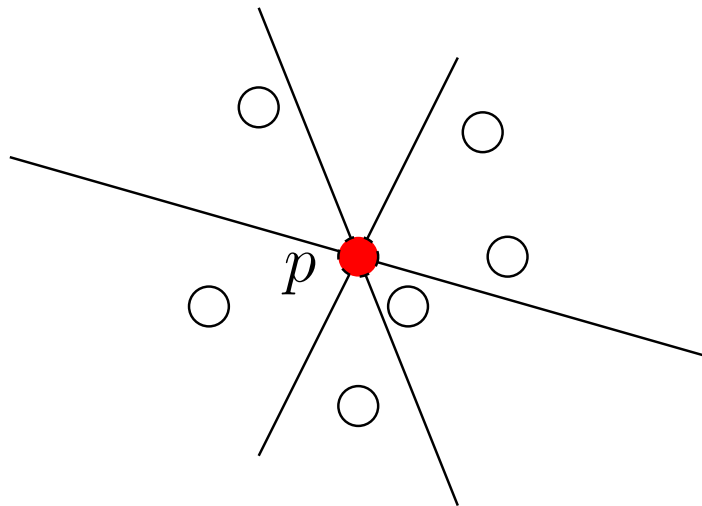
Certificates: $p_{<k} \leq p_k \leq p_{>k}$, plus the certificates of the two kinetic tournaments.

The total number of events is $O(\lambda_k(n) \log^2 n)$, where $\lambda_k(n)$ is the complexity of the k -level. Each event can be handled in $O(\log n)$ time.

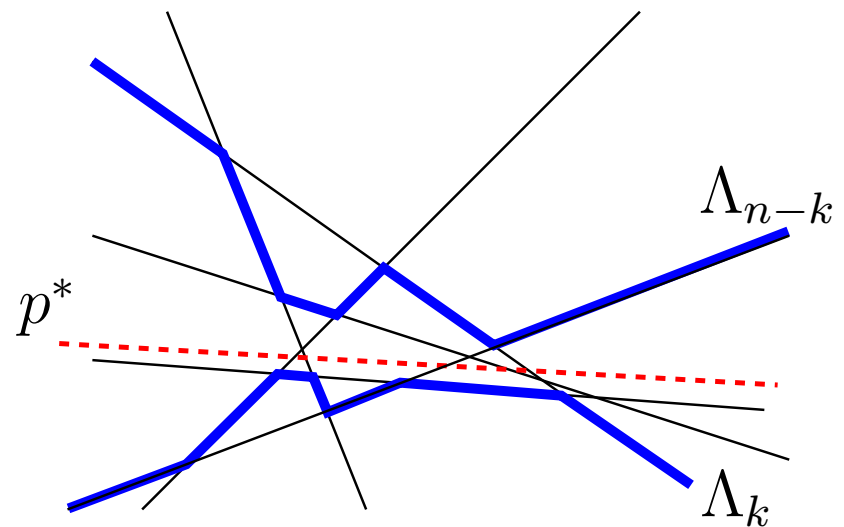
Maintaining depth contour and center region in \mathbb{R}^2 , I

Consider the set of lines P^* in the dual plane, a point p with depth k dualizes to a line between the k -level, Λ_k , and $(n - k)$ -level, Λ_{n-k} .

primal



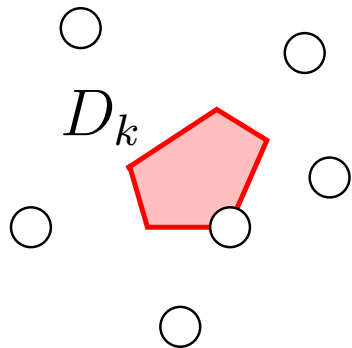
dual



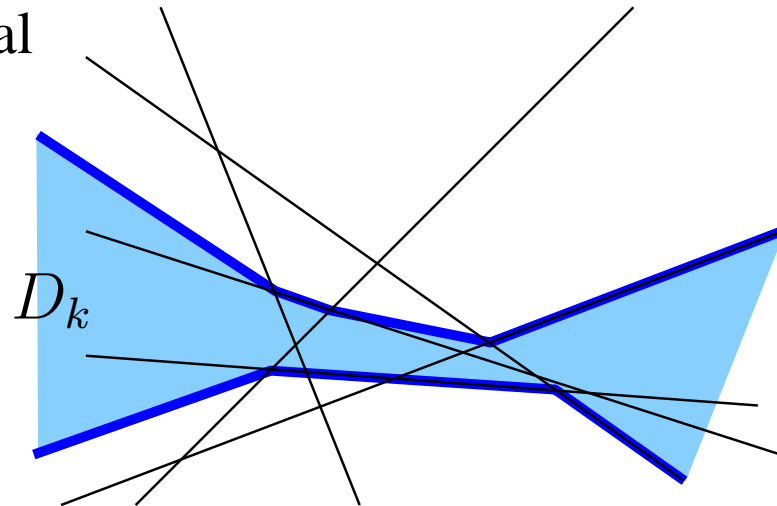
Maintaining depth contour and center region in \mathbb{R}^2 , II

The k -depth contour in the primal plane, D_k , dualizes to the region bounded between the upper envelope of the k -level and lower envelope of the $(n - k)$ -level.

primal



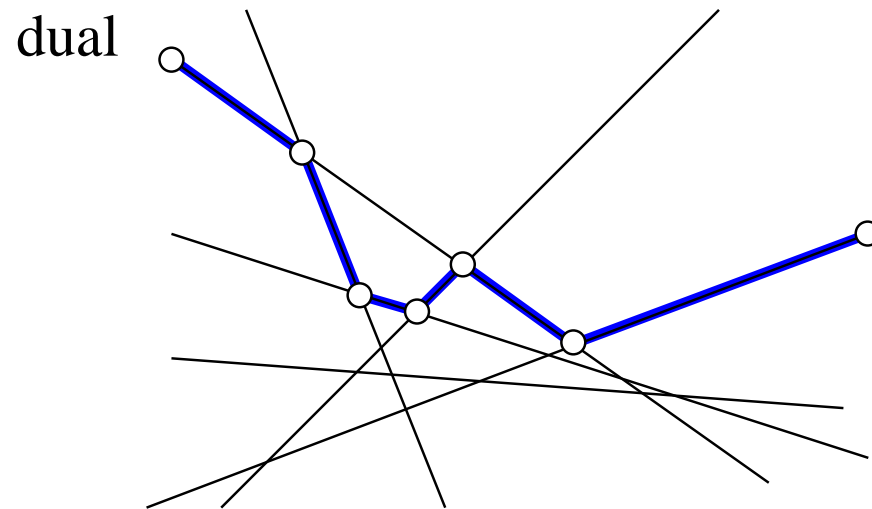
dual



Problem reduces to maintaining the upper hull of the k -level of a set of moving lines \mathcal{L} .

Maintaining depth contour and center region in \mathbb{R}^2 , III

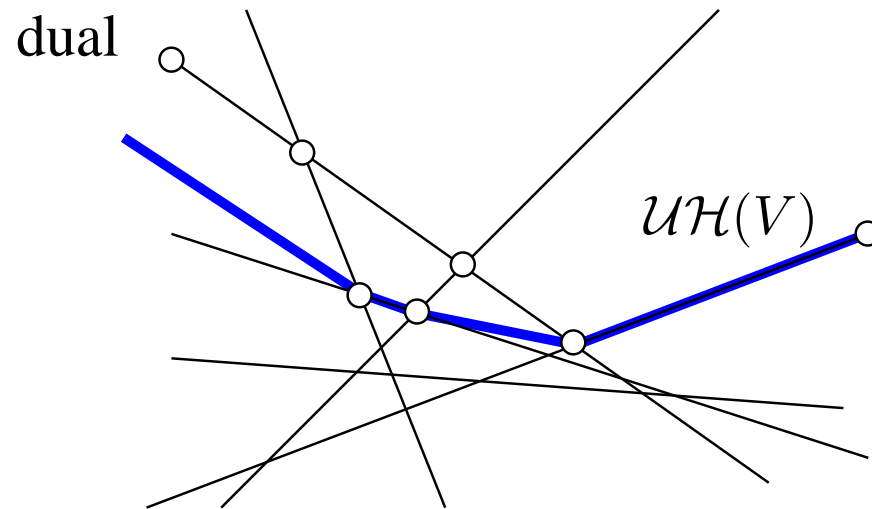
- For each line $\ell \in \mathcal{L}$, maintain the leftmost, $v^-(\ell)$, and rightmost, $v^+(\ell)$, intersections of ℓ with k -level Λ_k .
- Maintain the upper hull of those intersections V , $\mathcal{UH}(V)$, by existing kinetic convex hull algorithm.



This structure maintains $O(n^2)$ certificates, $O(n^{4+\delta})$ events such that each event can be updated in $O(\log^2 n)$ time.

Maintaining depth contour and center region in \mathbb{R}^2 , III

- For each line $\ell \in \mathcal{L}$, maintain the leftmost, $v^-(\ell)$, and rightmost, $v^+(\ell)$, intersections of ℓ with k -level Λ_k .
- Maintain the upper hull of those intersections V , $\mathcal{UH}(V)$, by existing kinetic convex hull algorithm.

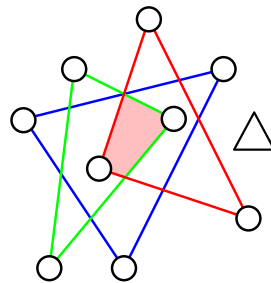


This structure maintains $O(n^2)$ certificates, $O(n^{4+\delta})$ events such that each event can be updated in $O(\log^2 n)$ time.

A space-efficient algorithm to maintain a center point, I

Tverberg partition: any $n = 3m$ points can be partitioned into m triples such that the intersections of these triangles Δ_i , $\Delta = \bigcap_{i=1}^m \Delta_i$, is non-empty (Tverberg 1966).

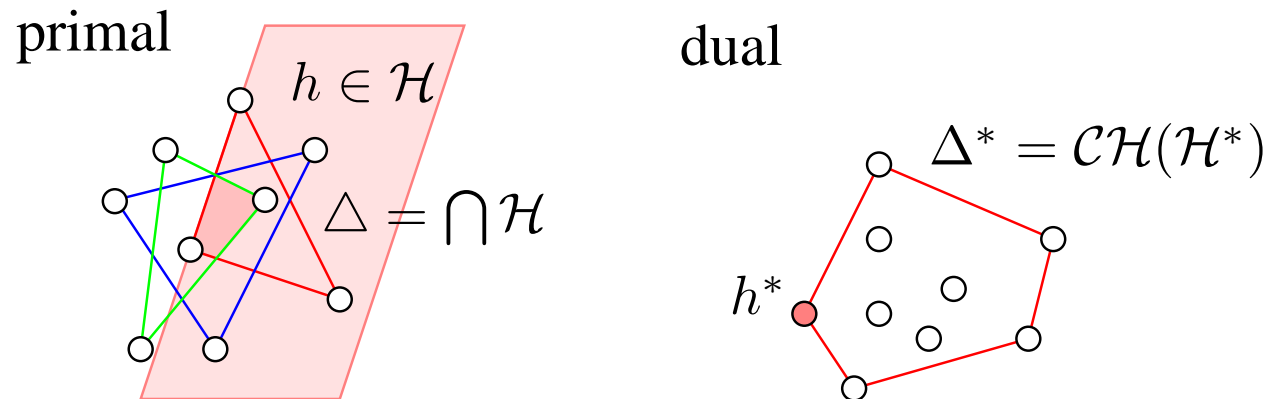
Observation: any point inside Δ is a center point.



So, we just maintain Δ under motion.

A space-efficient algorithm to maintain a center point, II

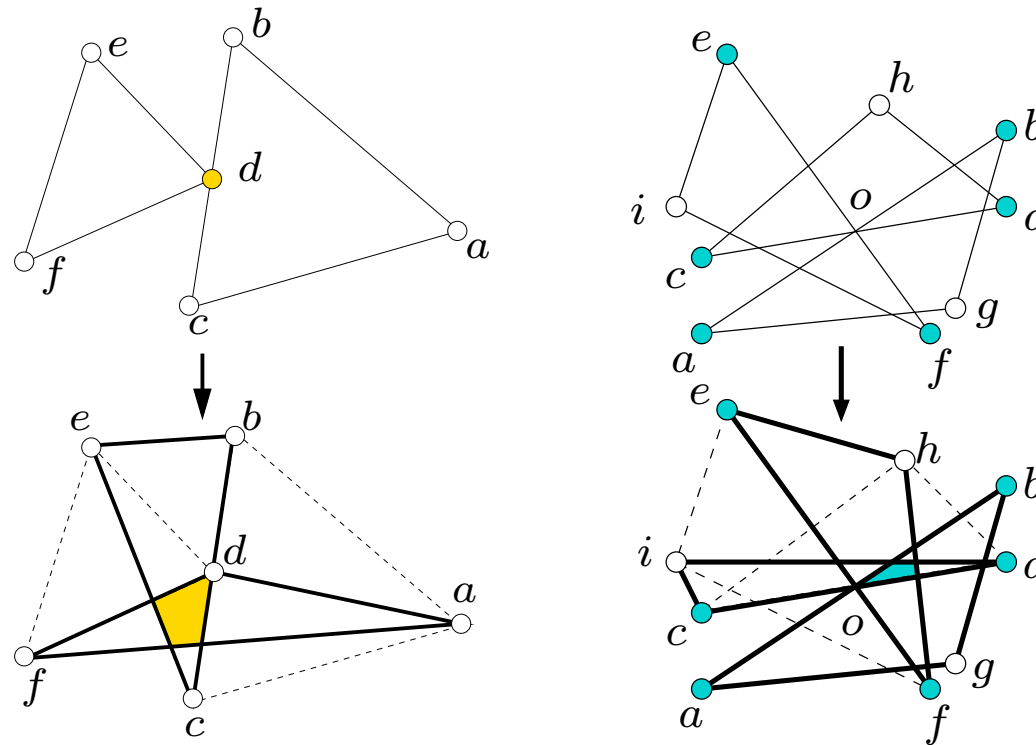
Δ is the intersection of n half planes \mathcal{H} . In the dual space, each half plane h maps to an oriented point h^* and Δ maps to the convex hull of \mathcal{H}^* . We use existing kinetic convex hull algorithm to maintain $\mathcal{CH}(H^*)$.



If Δ is non-empty, $\mathcal{CH}(H^*)$ is non-empty.

A space-efficient algorithm to maintain center point, III

When \triangle shrinks to empty, we re-arrange the two or three triangles to re-establish the Tverberg partition.



We use $O(n \log n)$ certificates and $O(n^{7+\delta})$ events.

Part II:
Maintaining the approximate median and center points

Approximate medians and center points

An ε -approximate median is a point in between the rank $(1 - \varepsilon)n/2$ and $(1 + \varepsilon)n/2$ points. An ε -approximate center point is a point with depth in between $(1 - \varepsilon)n/3$ and $(1 + \varepsilon)n/3$.

Approximate medians and center points can change much less often than the exact ones.

E.g., if the points follow linear motion, the best known bound on the number of changes of the exact median is $O(n^{4/3})$. But an ε -approximate median can change only $O(1/\varepsilon^2)$ times and can be computed in time $O(n \log n/\varepsilon^2)$ (Matoušek 1990 and Agarwal *et al.* 2002).

The ε -approximation based algorithm

For a range space $X = (S, \mathcal{R})$, where S is a set and $R \subset 2^S$, a subset $A \subseteq S$ is called an ε -approximation for X if, for every range $R \in \mathcal{R}$,

$$\left| \frac{|A \cap R|}{|A|} - \left| \frac{R}{S} \right| \right| < \varepsilon.$$

For a finite range space, there exists an ε -approximation with size $O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon}\right)$.

If the trajectories of the points are fixed, we can find an ε -approximation A of P in the standard way such that the median (or center point) of A is an ε -approximate median (or center point) of P , at any time.

It seems we are all set, but not really...

The ε -approximation based algorithm

If a point changes its trajectory, how to guarantee the ε -approximation is still good?

We need a deterministic algorithm for an ε -approximation that allows dynamic update in polylogarithmic time.

A dynamic algorithm for ε -approximation, I

Build a balanced binary tree over the points of S (assuming $|S| = 2^k$), and compute the ε -approximation in a bottom-up fashion, by using the followings (Chazelle's book 2000).

- **Merging**: Merge the ε -approximation of the children.
- **Halving**: If the ε -approximation gets too large, then halve it (and lose a small approximation factor).

Finally, one can get an ε -approximation with size $O(\frac{1}{\varepsilon^2} \log \frac{1}{\varepsilon})$.

Now, we make this structure dynamic.

A dynamic algorithm for ε -approximation, II

Invariant: Every internal node maintains an approximation to all the points in its subtree.

When a point is inserted, make it a subtree. When a point is deleted, the original tree is cut into $O(\log n)$ pieces.

Then we merge the trees and re-compute the approximation at each node (merge and halve the approximations of subtrees).

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

The ε -approximation based algorithm

Theorem For a set of points S moving in \mathbb{R}^2 , one constructs 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 center point of $S(t)$ in $O(\log(1/\varepsilon))$ time based on the current trajectories of S .

Faster off-line algorithms

If the trajectories are **fixed**, we can find ε -approximate medians of size $O(1/\varepsilon^2)$ in time $O(n \log(1/\varepsilon) + 1/\varepsilon^{O(\mu^2)})$, where μ is the degree of the algebraic trajectory.

If the trajectories are **lines**, we can find ε -approximate medians of size $O(1/\varepsilon^{4/3} \log^2(1/\varepsilon))$ in time $O((n/\varepsilon^{1/3}) \log(1/\varepsilon))$.

Please see the full paper for details.

Summary

- Approximate medians or center points change much less often than the exact ones.
- The deterministic dynamic algorithm for an ε -approximation is interesting on its own.