Data Structures for Moving Objects

Pankaj K. Agarwal

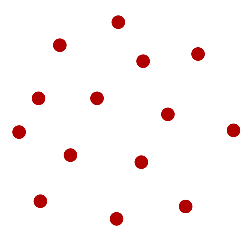
Department of Computer Science
Duke University

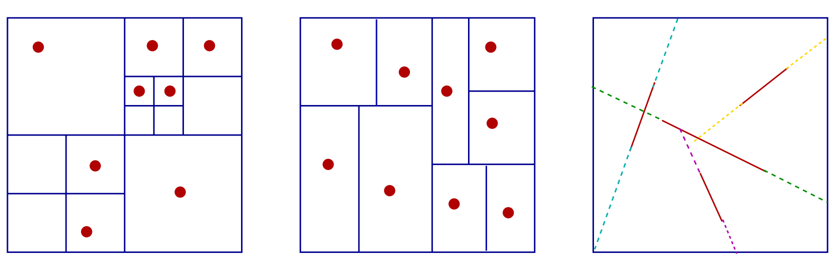
Geometric Data Structures

S: Set of geometric objects
Points, segments, polygons

☆ Ask several queries on S

- Range searching
- Nearest-neighbor searching





Quad Tree — kd−Tree — BSP

Moving Objects: Applications

- ☆ Traffic management
  - Location based services
  - Emergency services
  - Air traffic control
- ☆ Digital battlefields
- ☆ Molecular biology
- ☆ Deformable objects
- ☆ Adhoc networks

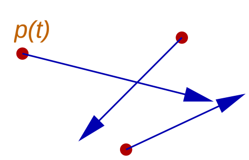
Need data structures for storing, analyzing, querying moving objects.

Modeling Motion

$p(t) = (x(t), y(t))$: Position of p at time t.

- $x(\cdot), y(\cdot)$: Polynomials
- *Degree* of motion: max degree of $x(\cdot), y(\cdot)$.
- *Linear motion:* Degree of motion is 1

  $p(t) = \mathbf{a}t + \mathbf{b}, \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^2$



- ☆ Mostly assume motion to be linear
- ☆ Trajectory of points can change
- ☆ Trajectory can be piecewise linear

Issues:

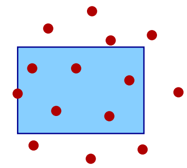
- ☆ Sampled motion
- ☆ Hierarchical motion
- ☆ Uncertainty

Range Searching

S: Set of points

Preprocess S into a data structure

Report all points of S lying inside a query rectangle



Data Structure	Space	Query
Range tree	$n\frac{\log n}{\log\log n}$	$\log n + k$
kd-tree	n	$\sqrt{n} + k$

External memory data structures also available

Example: R-tree

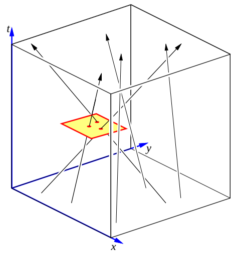
Kinetic Range Searching

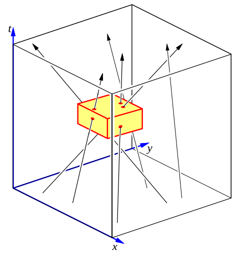
S: Set of points, each moving with fixed velocity in the plane

Preprocess S into a data structure:

Q1 Given a rectangle R and a time value t,
report all points of $S(t) \cap R$

Q2 Given a rectangle R and time values t_1, t_2,
report all points that pass through R during the time interval $[t_1, t_2]$.





Early Approaches

☆ One-dimensional data structures
[Wolfson et al. '98-'99, Tayeb et al. '98, Kollios et al. '99]

☆ Two-dimensional data structures

- Map trajectories to higher dimensional points [Kollios et al.]
- Build index on trajectories [Pfoser et al.]
- Parametric R-trees [Saltenis et al.]
  Assumes frequent updates on trajectories

Kinetic Range Searching

(A., Arge, Erickson, 2001)

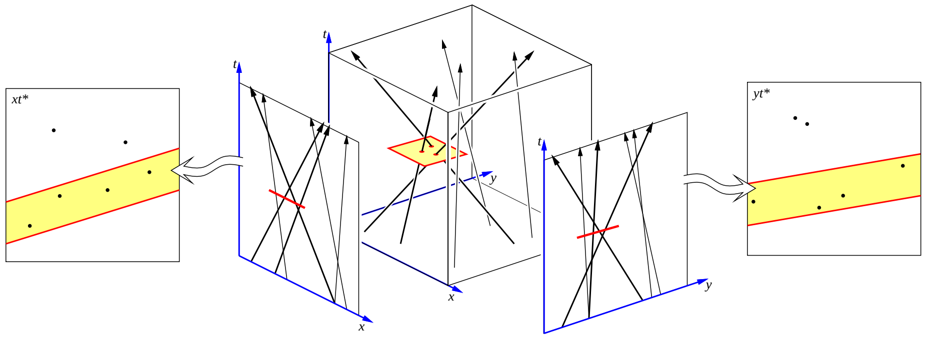
☆ Partition-tree based approach

- $O(n)$ space, $\sim \sqrt{n} + k$ query time
- $\log^2 n$ insertion/deletion/trajectory-change
- Time oblivious scheme

☆ Kinetic range trees

- $n \log n / \log \log n$ space, $\log n + k$ query
- Events: x- or y-coordinates of two points become equal
- $\Theta(n^2)$ events, each requiring $\log^2 n$ time
- Tradeoff between # events and query time
- Queries have to arrive in a chronological order

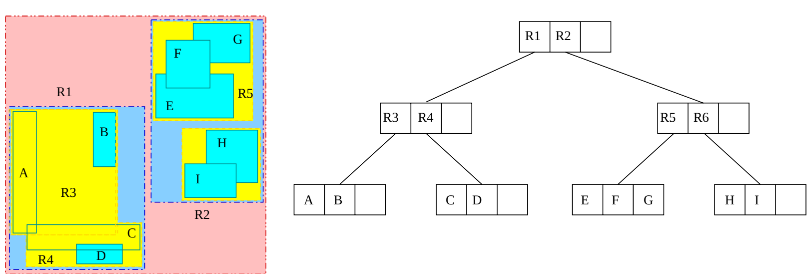
Partition Tree Based Approach



- ☆ Trajectory of a point p_i is a line ℓ_i in $\mathbb{R}^3$
- ☆ $p_i(t) \in R \iff \ell_i$ *intersects* (R, t)
- ☆ ℓ_x, ℓ_y: Projection of ℓ onto the xt- & yt-planes
- ☆ ℓ intersects $(R, t) \Leftrightarrow \ell_x$ intersects (R_x, t) & ℓ_y intersects (R_y, t)
- ☆ Use duality and partition trees

R-Trees

- ☆ Bounding box hierarchy, B-tree
- ☆ Each node v is associated with a subset S_v of points and the smallest rectangle R_v containing S_v
- ☆ Partition S_v into B clusters, each associated with a child of v
- ☆ Several heuristics are proposed for partitioning S_v into B clusters

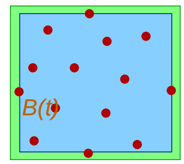


STAR-Tree

☆ Kinetic R-tree

☆ Maintain the smallest box enclosing the set of moving points

- Box is defined by four points
- The combinatorial structure can change $\Omega(n)$ times
- Maintain an approximation of the smallest enclosing box



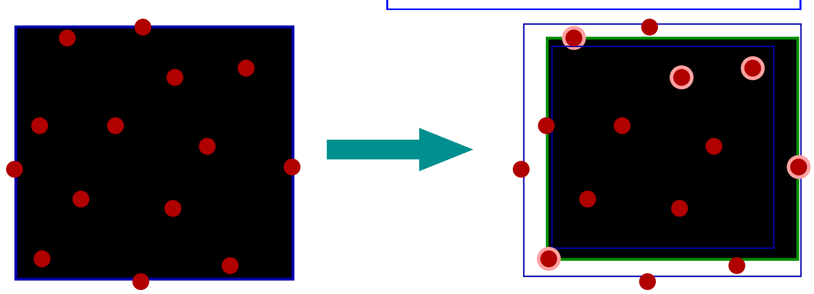
☆ Maintaining the clustering kinetically

- Extend the known heuristics
- No theoretical nontrivial results known on kinetic clustering

Smallest Enclosing Box

☆ $R(P(t))$: Smallest box enclosing P at time t

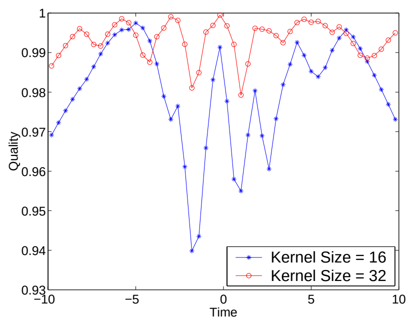
☆ ε-core-set: $C \subseteq S$ ε-coreset if $\boxed{\forall t\ (1-\varepsilon)R(S(t)) \subseteq R(C(t))}$

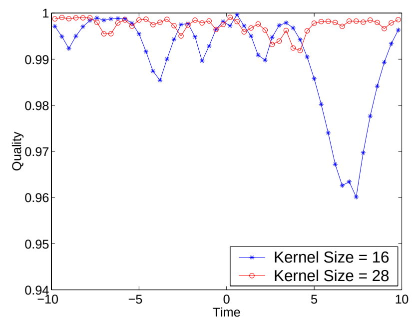


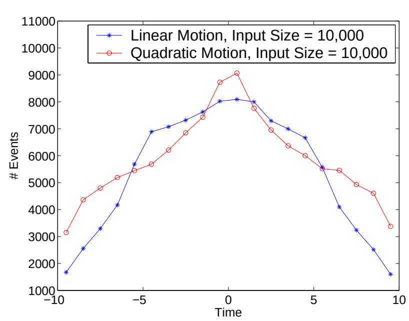
Theorem: *$\exists$ ε-core-set of size $1/\sqrt{\varepsilon}$; Computation time: $n + 1/\varepsilon$*

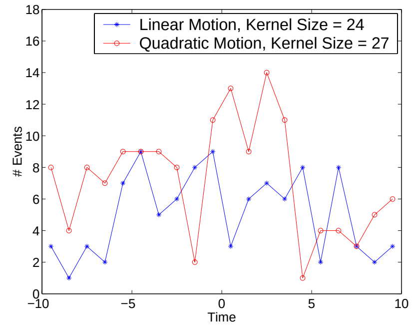
A more general result on core sets in [A., Har-Peled, Varadarajan]
Leads to approximatation algorithms for several problems

Smallest Enclosing Box





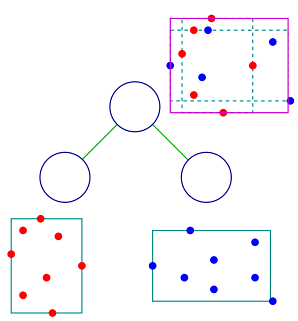




Smallest Enclosing Box

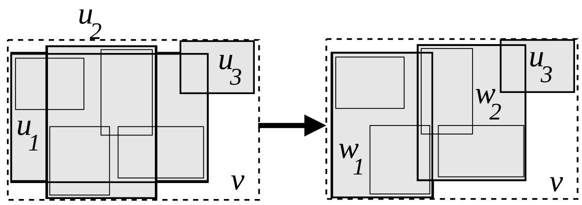
STAR-tree: Maintain a box enclosing S_v at each node v

- ☆ Compute $C_v \subset S_v$ for each node in a bottom-up manner
  - Merge the core sets computed at the children of v
  - Prune the merged set
- ☆ Maintain the smallest enclosing box $R(C_v)$

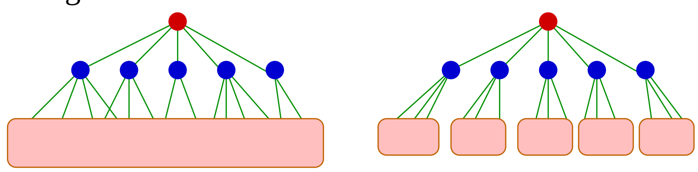


Re-Clustering

- ☆ Reorganize the children of a node if the rectangles of their children overlap a lot.



- ☆ Collect all the grandchildren of the node

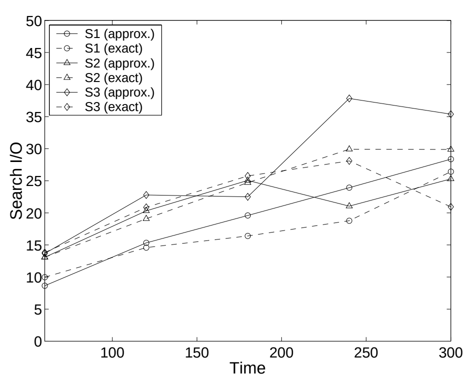


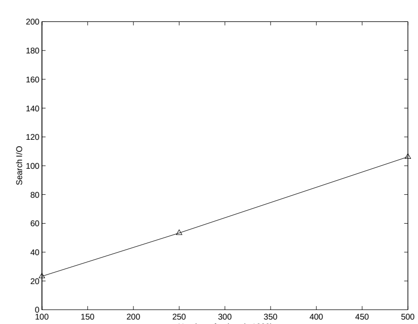
- ☆ Reconstruct a 2-level R-tree on them

Experimental Results

Synthetic Data

- ☆ 100,000–500,000 points inside $1000 \times 1000\ km^2$ area with different distributions
- ☆ Points are inserted/deleted dynamically, at any time at least 80% points present
- ☆ Three range of speed: 45 km/h, 75km/h, 180 km/h



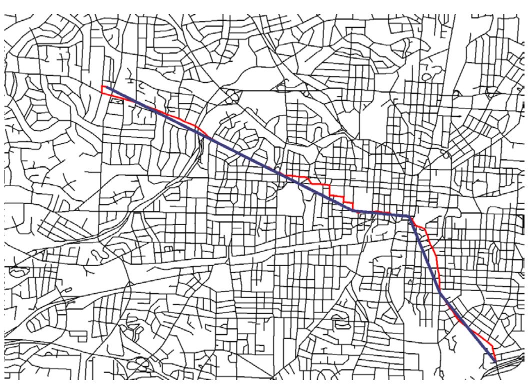


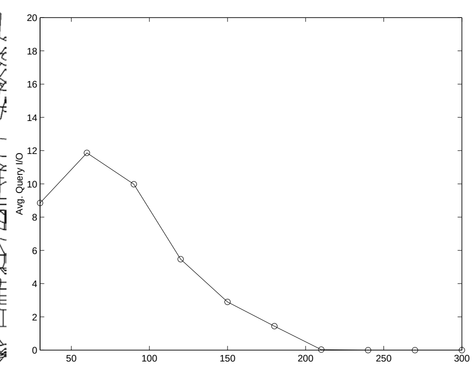
Experimental Results

Realistic Data

- ☆ Extracted the roads map around Durham, NC, within 120 miles centered at Durham ($\approx 250,000$ polygonal chains)
- ☆ Computed a planar map of the road network
- ☆ Chose source and destinations randomly with some distribution
- ☆ Computed a good path using Dijkstra's algorithm — minimize the length + number of turns
- ☆ Used Douglas Peucker algorithm to simplify the paths

Experimental Results





Tradeoffs in Performance

☆ **Accuracy vs efficiency**

- Maintain approximate structures

☆ **Query vs events**

- Combine KDS and time-oblivious approaches

☆ **Time Responsive Approach:**

- Near-future queries are more critical than far-future queries
- Fast query time for near-future queries
- Measure *future* by the number of events occurred
- # events: Δ, query: $\sqrt{\Delta/n} + k$

Concluding Remarks

☆ Incorporating more realistic motions

- Use dynamic systems, e.g., Kalman, particle filters, to model trajectories
- How does one perform geometric computation in this model?
- Geometric computation under uncertainty

☆ Hierarchical representation of motion

☆ Kinetic data structure for clustering, similarity searching