Approximate Range Searching in the Absolute Error Model

Guilherme D. da Fonseca

CAPES BEX1319027

http://www.cs.umd.edu/~fonseca

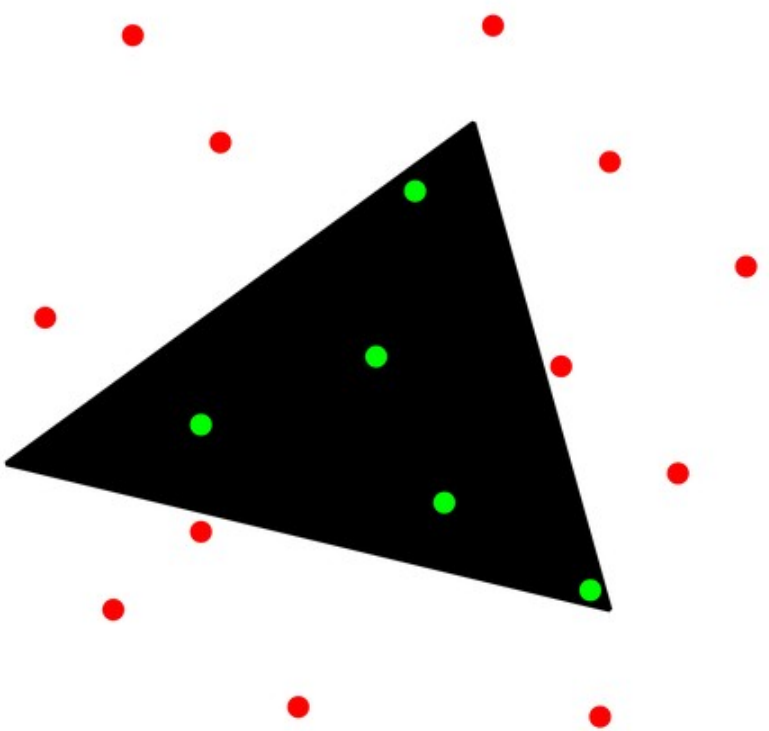
Advisor: David M. Mount

University of Maryland

Ph.D. defense

November 28th, 2007

Range Searching – Exact Version



P: Set of n points in d-dimensional space.

w: Weight function.

$\mathcal{R}$: Set of *ranges* (regions of the space).

- Preprocess P, in a way that, given $R \in \mathcal{R}$, we can efficiently compute:

$$\sum_{p \in P \cap R} w(p)$$

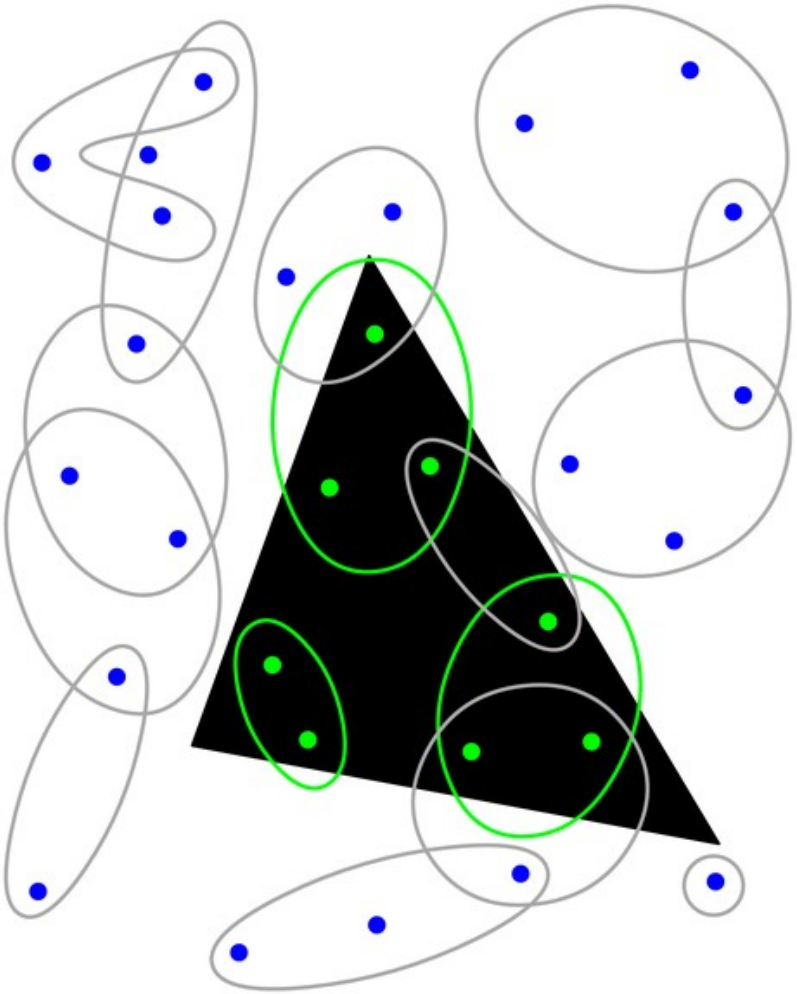
Semigroups, Groups and Idempotence

- In the most general version, the weights are drawn from a commutative **semigroup**.

Other properties may be useful:

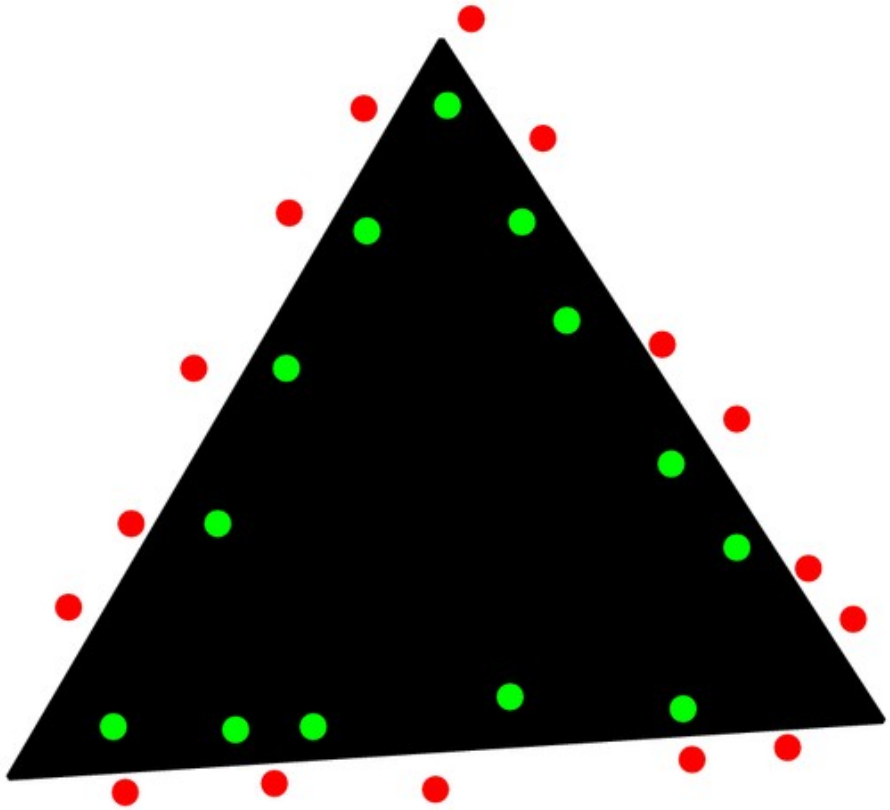
- **Group**: We can use subtraction.
- **Idempotence**: For every x, we have x+x=x:
  - *maximum,*
  - Boolean *or*.

Generators



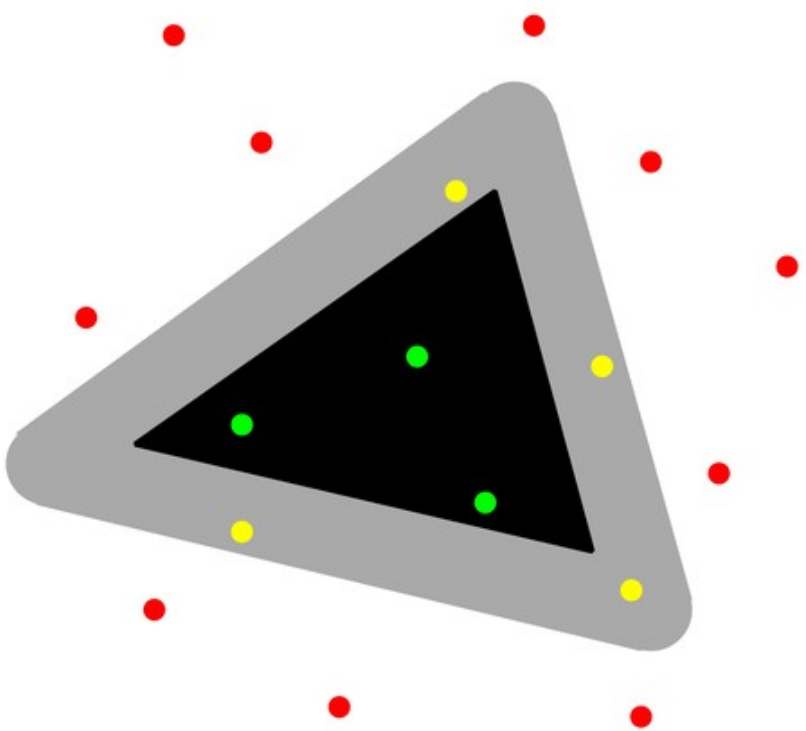
- Generators represent sets of points whose sum is precomputed.
- A query is processed by summing generators.
  - Large generators:
    - Low query time,
    - High storage.
  - Small generators:
    - High query time,
    - Low storage.

Why Approximate?



- Exact solutions are generally complicated and inefficient.
- Polylogarithmic query time requires about n^d space.
- With linear space, the query time approaches O(n) as d increases.
- Points near the range boundary require the use of many small generators.

The Absolute Error Model

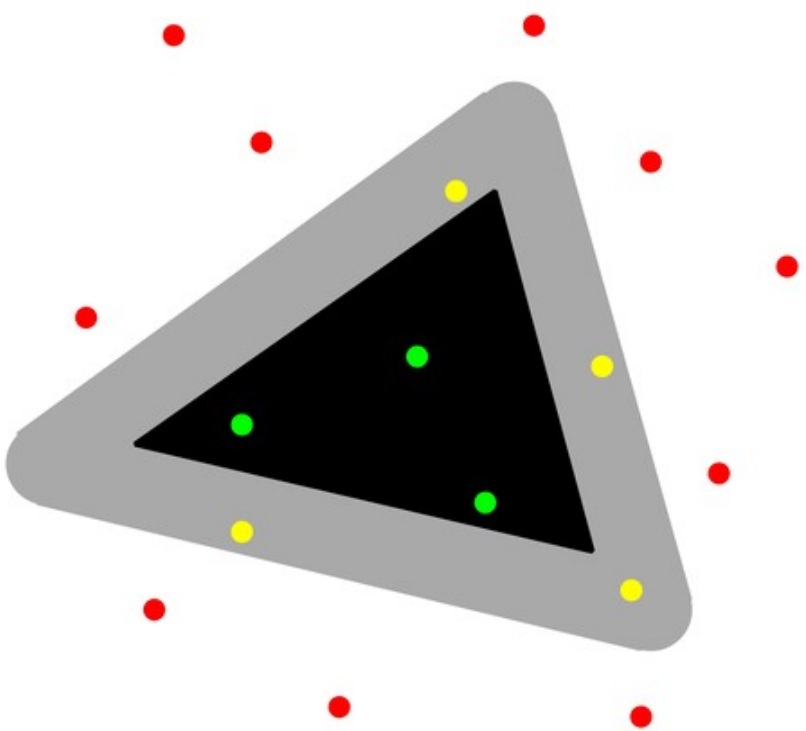


- **Absolute** error model: points within distance ε from the boundary may or may not be counted.
- All data points lie inside the unit hypercube $[0,1]^d$.
- **Relative** error model: fuzzy boundary is proportional to the diameter of the range (Arya, Mount, 2000).

Our Results

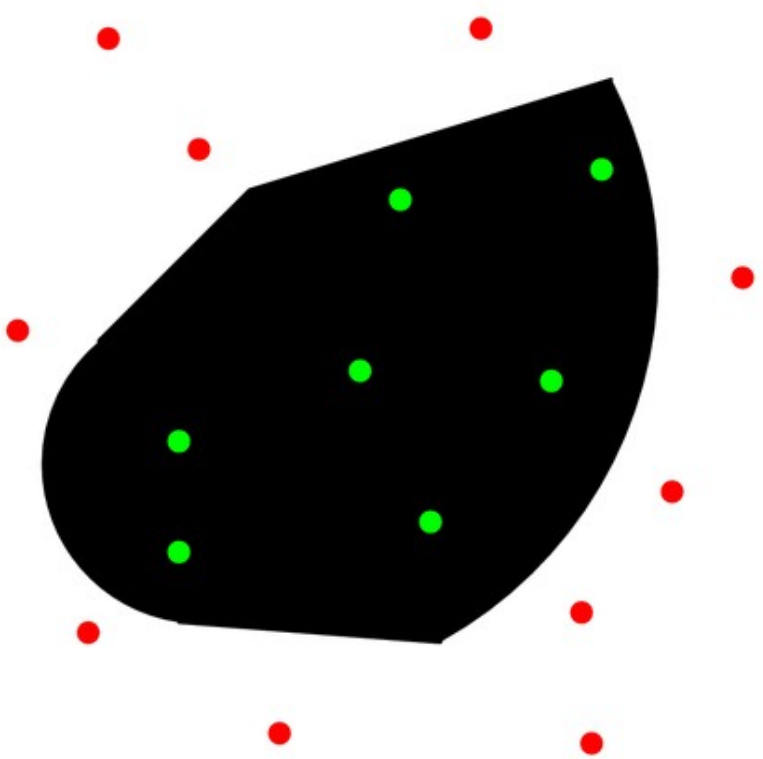
- The first work on approximate range searching in the **absolute error model** for fixed dimensions.
- Our data structures are **simple** and amenable to **efficient implementation**.
- We exploit **idempotence** to achieve better performance.
- Introduction of the versatile **halfbox quadtree**.
- We apply our results to several problems, including **exact** idempotent halfspace range searching, the **relative error model,** and the **data stream model**.

Contents



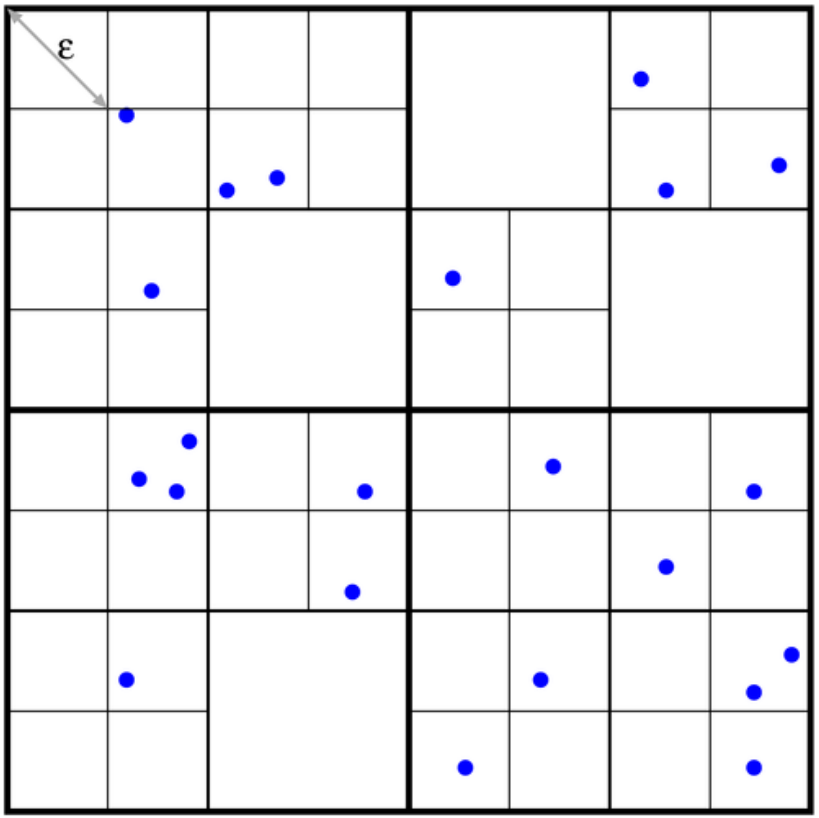
- Preliminaries
- Orthogonal & convex ranges
- Halfspace ranges (application to exact range searching)
- Halfbox quadtree
- Relative Model
- Conclusions

Convex Ranges



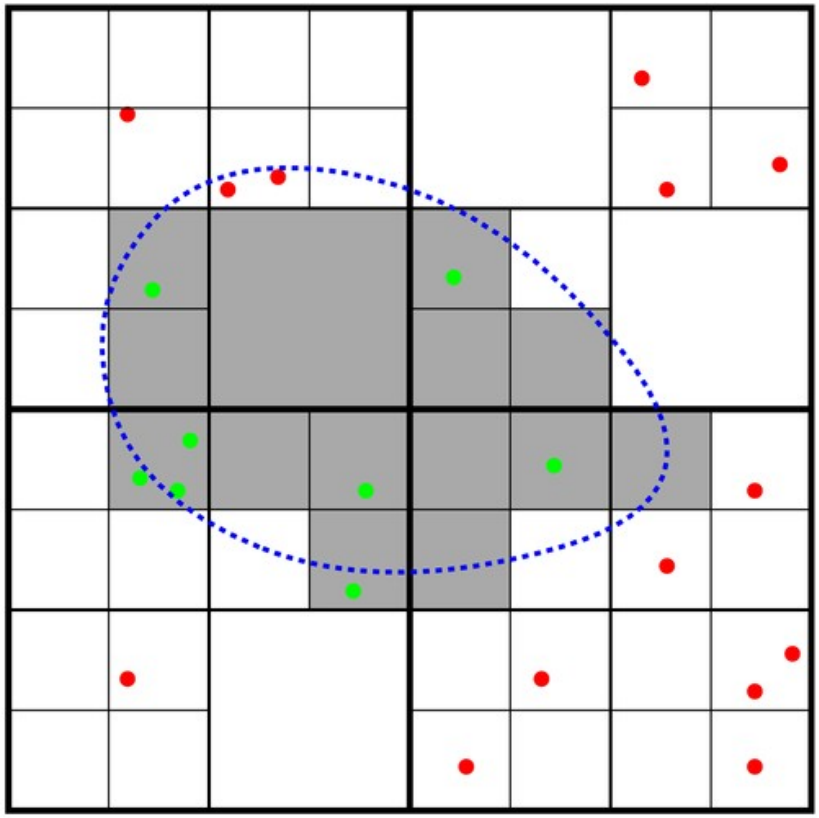
- Ranges are general convex shapes.
- *Unit cost test assumption:* we can determine whether the range intersects a given hypercube in constant time.
- No equivalent structure for the exact version.
- Semigroup in the absolute model:
  - Query time: $O(1/\varepsilon^{d-1})$.
  - Space: $O(\min(n, 1/\varepsilon^{d}))$.
- Relative error model structure by Arya and Mount, 2000.

Convex Ranges Structure



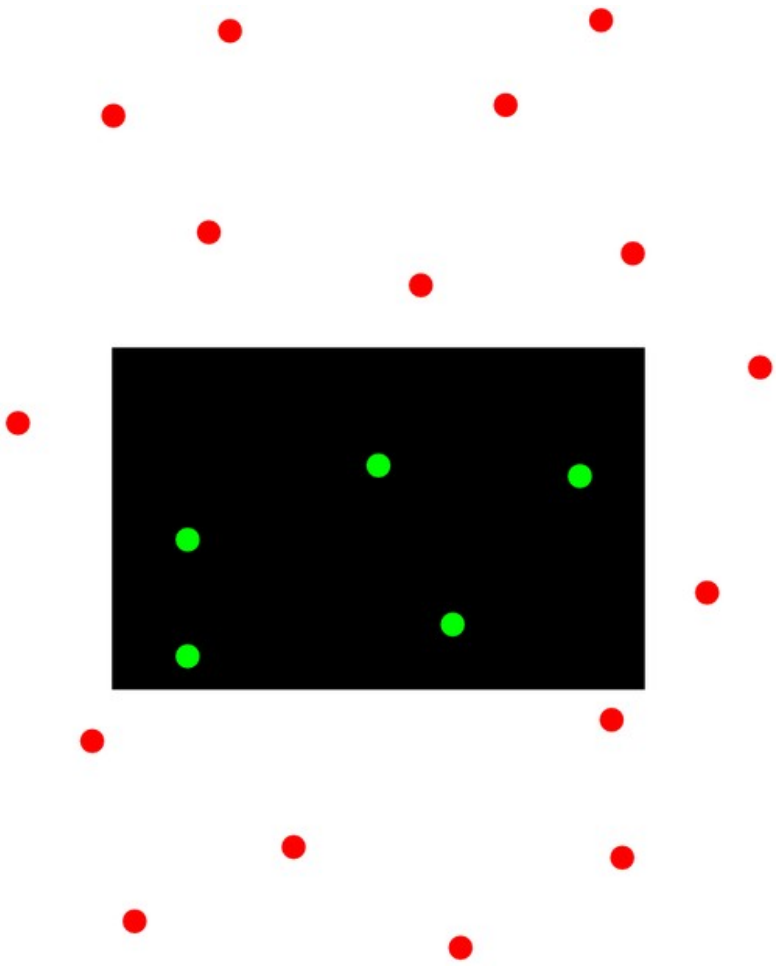
- Build a quadtree with the points.
- Stop subdividing when a box has diameter $\leq \varepsilon$ or there are no internal points.
- Each node stores a partial sum of the points inside.
- Space: $O(1/\varepsilon^d)$.
  Depth: $O(\log 1/\varepsilon)$.
- Compression can be used to reduce storage to $O(\min(n, 1/\varepsilon^d))$.

Convex Range Queries



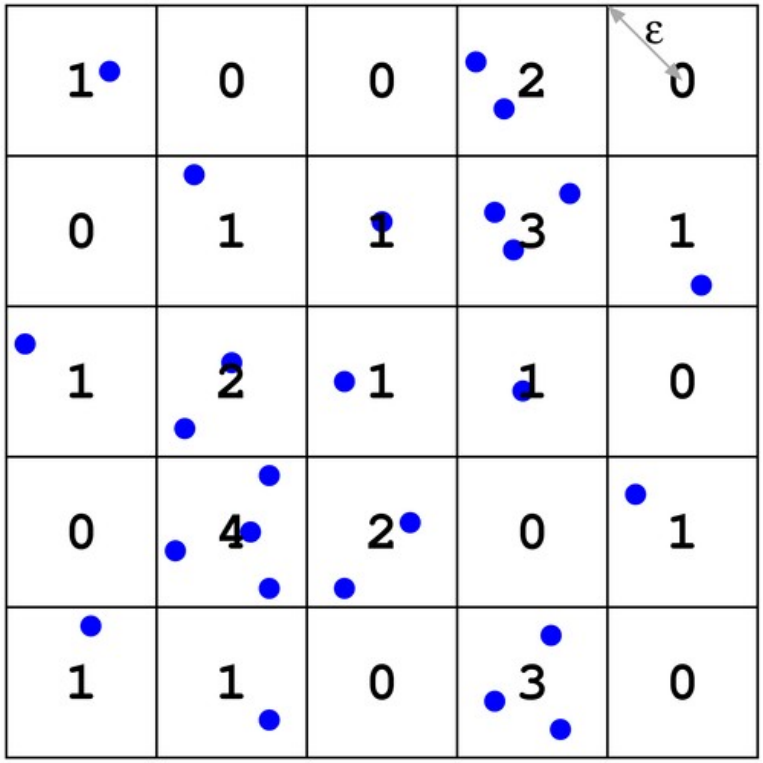
- Do not do anything if the box is empty.
- If the box is completely inside the range, count points as inside.
- If the box is completely outside the range, do not count points.
- If the box has minimum size, count the points only if the box center is inside the range.
- Otherwise, make a recursive call for each box subdivision.

Orthogonal Ranges



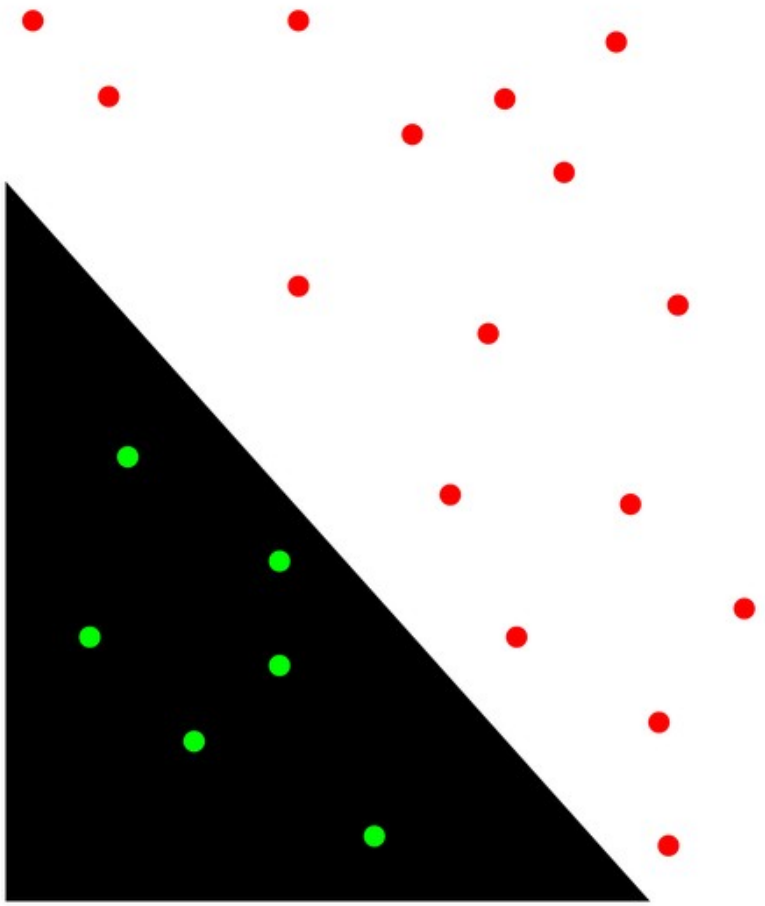
- Ranges are axis-aligned rectangles.
- Exact (Chazelle, 1988):
  - Query time: $O(\log^{d-1} n)$.
  - Space: $O(n \log^{d-2} n)$.
- Approximate group:
  - Query time: $O(1)$.
  - Space: $O(1/\varepsilon^{d})$.
- Slightly larger complexity for the semigroup version.

Orthogonal Ranges



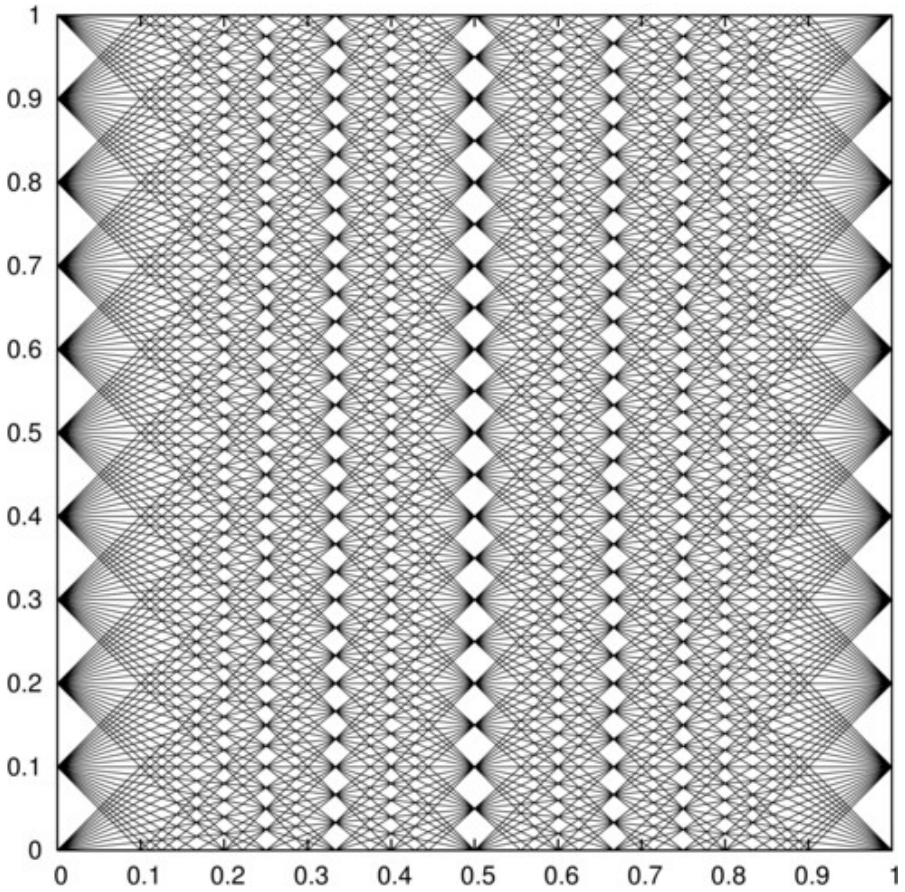
- Reduction to partial sums.
- *Partial sum*: orthogonal query in multidimensional array.
- Build a grid with cells of diameter 2ε.
- Build a d-dimensional array with the sum for each cell.
- Use data structures from Chazelle and Rosenberg, 1989 or Yao, 1982.

Halfspace Ranges



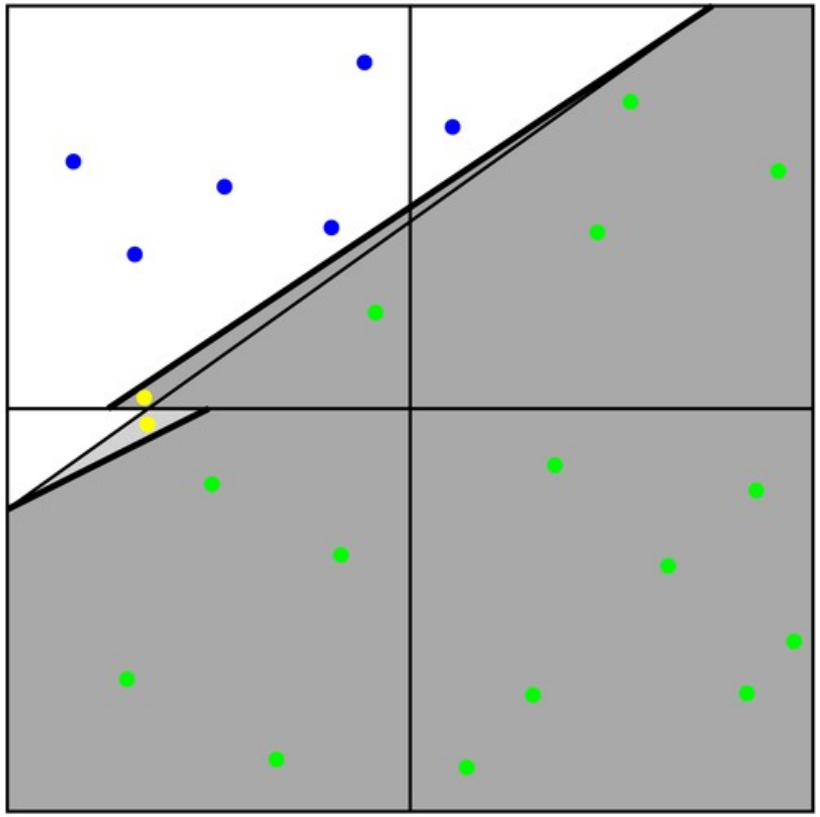
- Ranges are d-dimensional halfspaces.
- Exact (Matoušek, 1993):
  - Query time: $O(n^{1-1/d})$.
  - Space: $O(n)$.
  - Polylogarithmic query time takes n^d space.
- Approximate:
  - Query time: $O(1)$.
  - Space: $O(1/\varepsilon^d)$.

Halfspace Ranges Structure



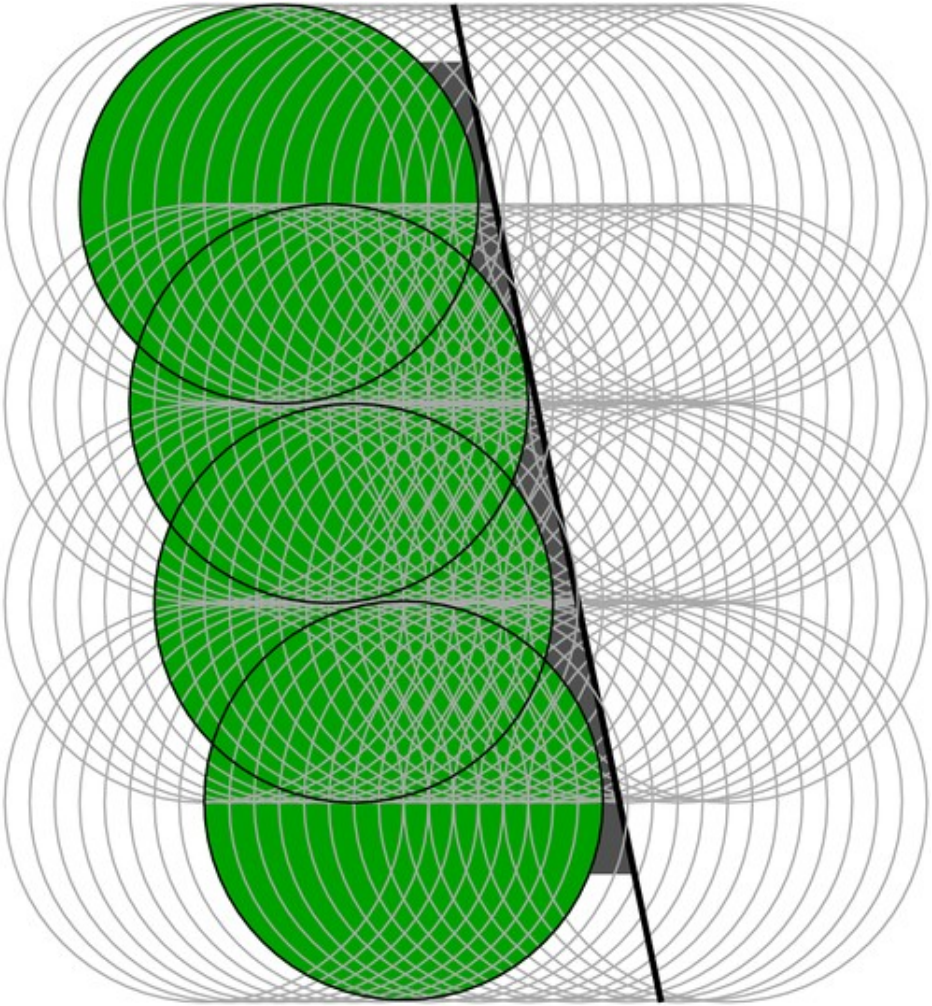
- We can ε-approximate all halfspaces inside the unit cube with $O(1/\varepsilon^d)$ halfspaces.
- Store the results in a table.
- Answer queries by rounding halfspace parameters and returning the proper table entry.

Preprocessing



- Naive preprocessing takes $O(n/\varepsilon^d)$ or $O(n+1/\varepsilon^{2d})$ time.
- Instead, we perform 2^d approximate queries in the quadtree subdivisions.
- Problem: Error accumulates through $O(\log 1/\varepsilon)$ levels.
- Solution: Scale ε to become $\varepsilon / \log(1/\varepsilon)$, and remove extra entries afterwards.
- Time: $O(\log^{d+1}(1/\varepsilon)/\varepsilon^d)$.

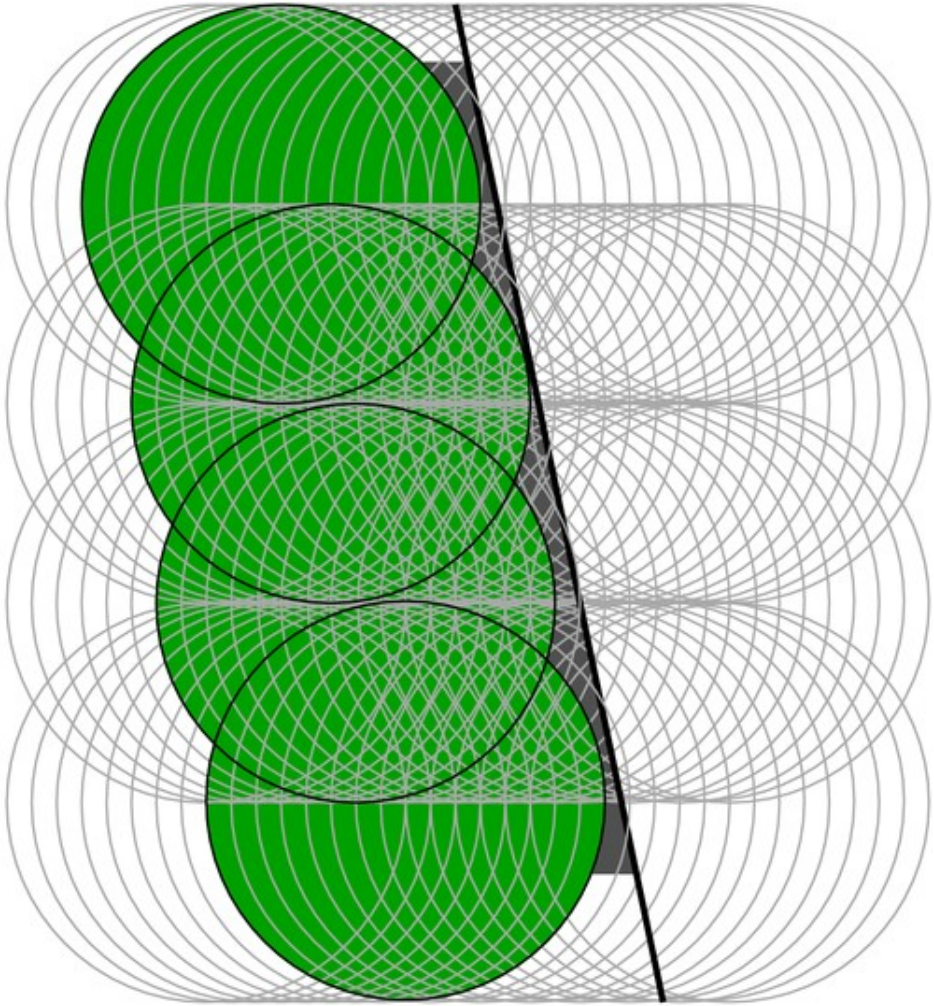
Idempotent Version



- Idempotent semigroup: $x+x=x$, for all x.
- Generators can overlap.
- Use large spherical generators.
  - Space: $O(1/\varepsilon^{(d+1)/2})$.
  - Query time: $O(1/\varepsilon^{(d-1)/2})$.
  - Trade-off: $O(m)$ space, $O(1/m\varepsilon^{d})$ query time.

Exact Uniformly Distributed Idempotent Version

- Approximate version:
  - Space: $O(1/\varepsilon^{(d+1)/2})$.
  - Query time: $O(1/\varepsilon^{(d-1)/2})$.
- Set $\varepsilon = 1/n^{2/(d+1)}$:
  - Space: $O(n)$.
  - Query: $O(n^{1-2/(d+1)})$.
- The $\varepsilon n = O(n^{1-2/(d+1)})$ remaining points are counted one by one.

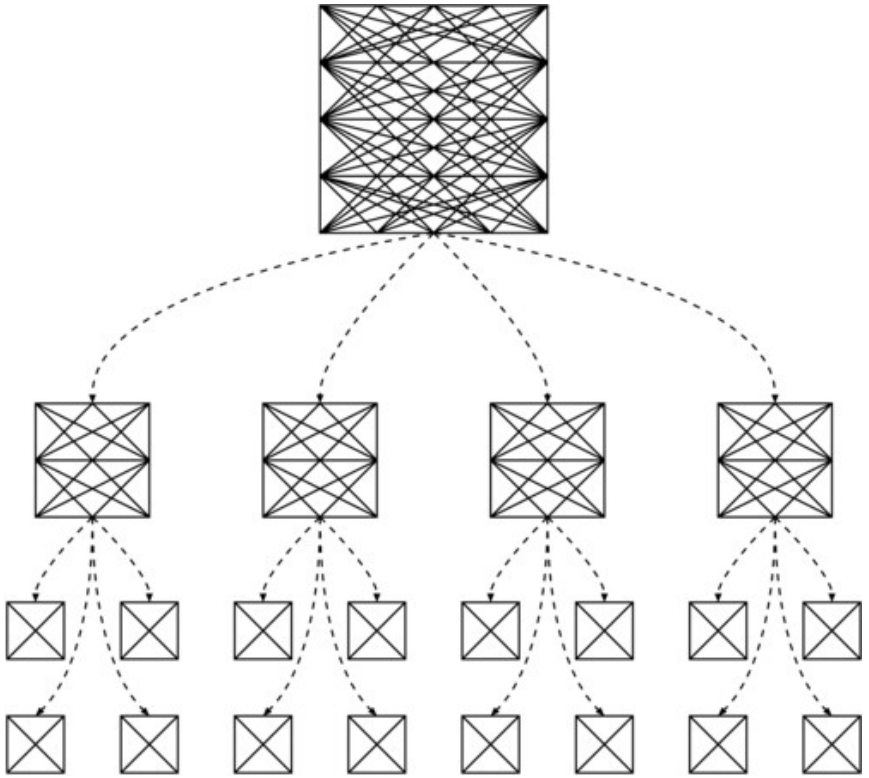


Exact Uniformly Distributed Idempotent Version

- Approximate version:
  - Space: $O(1/\varepsilon^{(d+1)/2})$.
  - Query time: $O(1/\varepsilon^{(d-1)/2})$.
- Set $\varepsilon = 1/n^{2/(d+1)}$:
  - Space: $O(n)$.
  - Query: $O(n^{1-2/(d+1)})$.
- The $\varepsilon n = O(n^{1-2/(d+1)})$ remaining points are counted one by one.

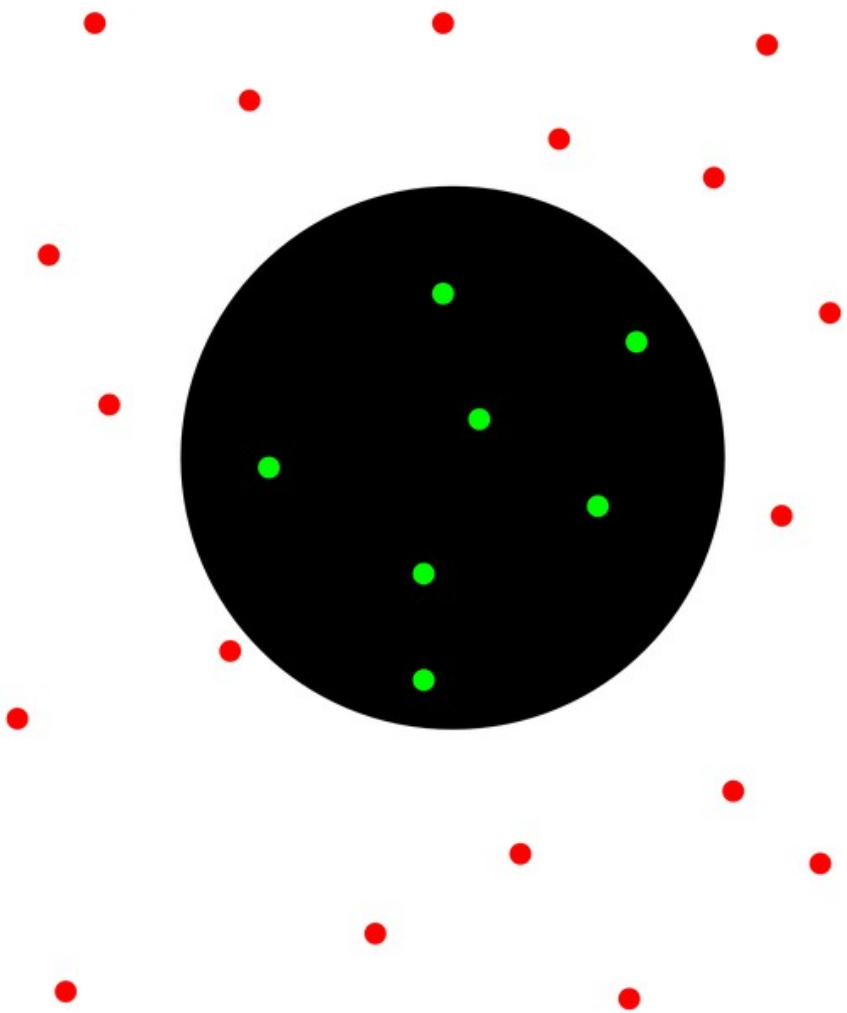
- Works in the *semigroup arithmetic model*.
- Uniform distribution.
- Matches the best lower bound up to logarithmic terms (Brönnimann, Chazelle, Pach, 1993).
- Same assumptions as the lower bound.

Halfbox Quadtree



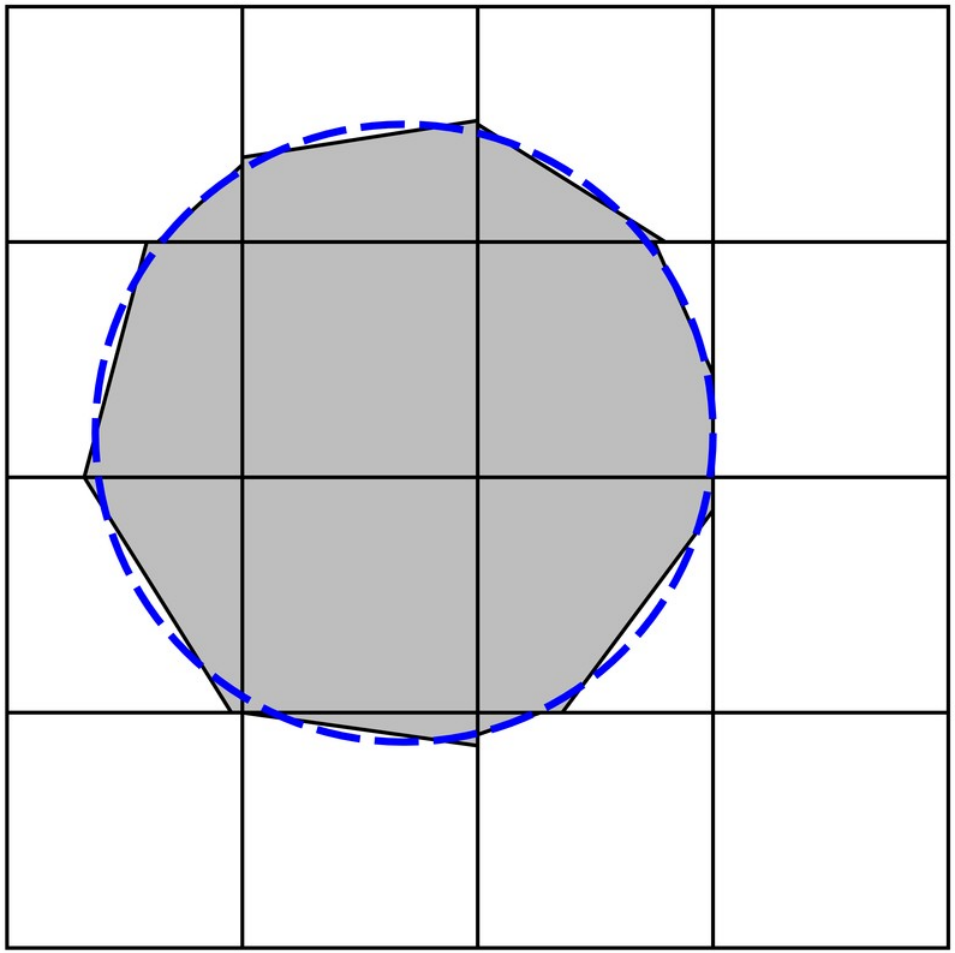
- One halfspace data structure for each quadtree box.
- Generators: intersection of quadtree boxes and halfspaces (*halfboxes*).
- Powerful building blocks!
- Smaller boxes take less space, as ε is constant.
  - Space: $O(\log(1/\varepsilon)/\varepsilon^d)$.
  - Prepro.: $O(\log^{d+1}(1/\varepsilon)/\varepsilon^d)$.

Spherical Range Searching



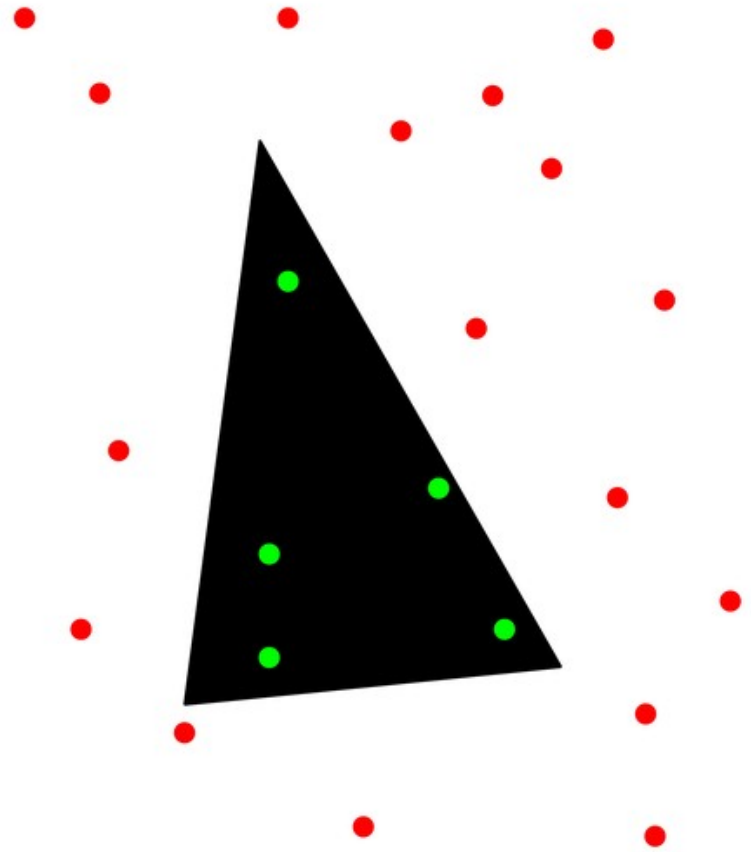
- Ranges are d-dimensional spheres.
- Exact version:
  - Project the points onto a (d+1)-dimensional paraboloid.
  - Use halfspace range searching.
- Approximate version:
  - Use the halfbox quadtree.

Spherical Range Searching



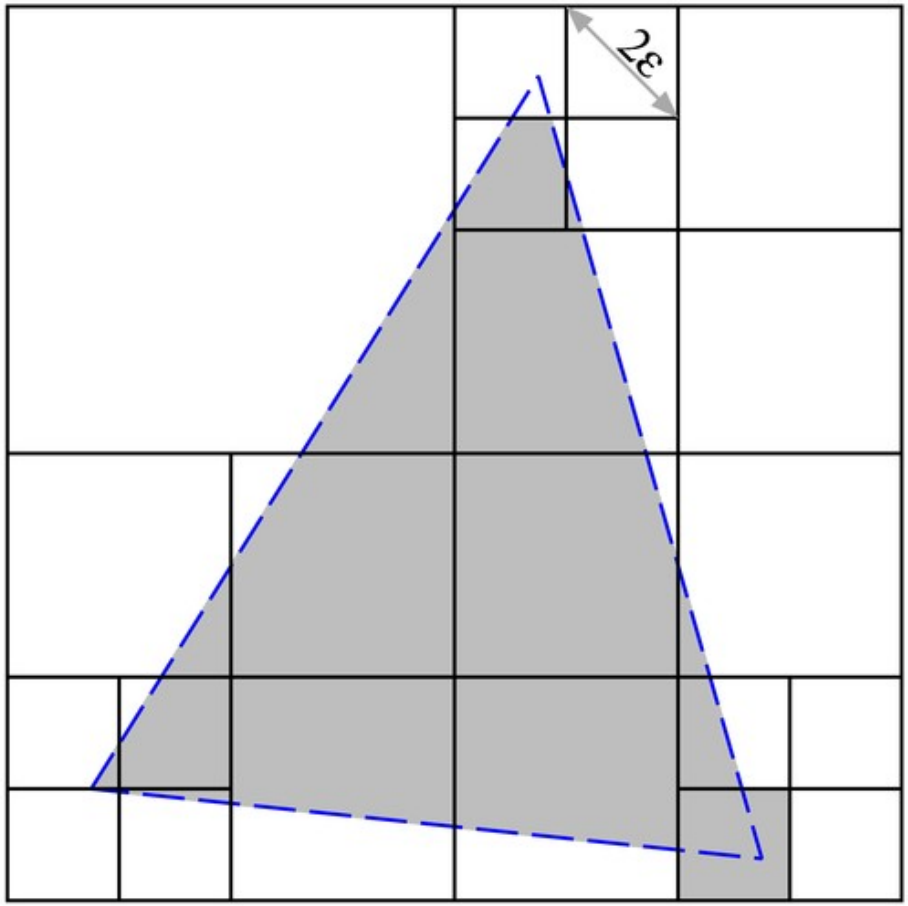
- Approximate the range with halfboxes.
- Only $O(1/\varepsilon^{(d-1)/2})$ halfboxes are necessary.
- Use the halfbox quadtree to query each halfbox in O(1) time.
  - Query time: $O(1/\varepsilon^{(d-1)/2})$.
  - Space: $O(\log(1/\varepsilon)/\varepsilon^{d})$.

Simplex Range Searching



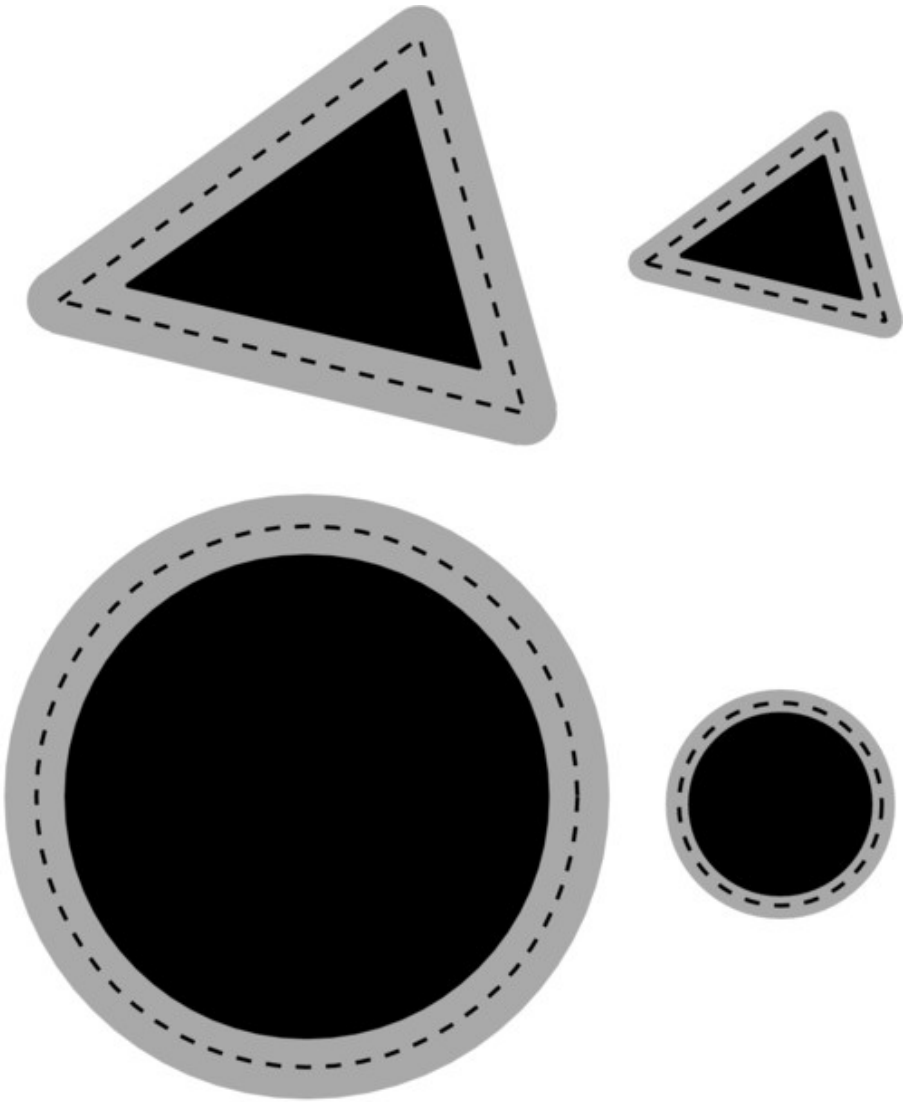
- Ranges are d-dimensional simplices: intersection of d+1 halfspaces.
- Exact version solved similarly to halfspace range searching: (Matoušek, 1993)
  - Query time: O($n^{1-1/d}$).
  - Space: O(n).

Simplex Range Searching



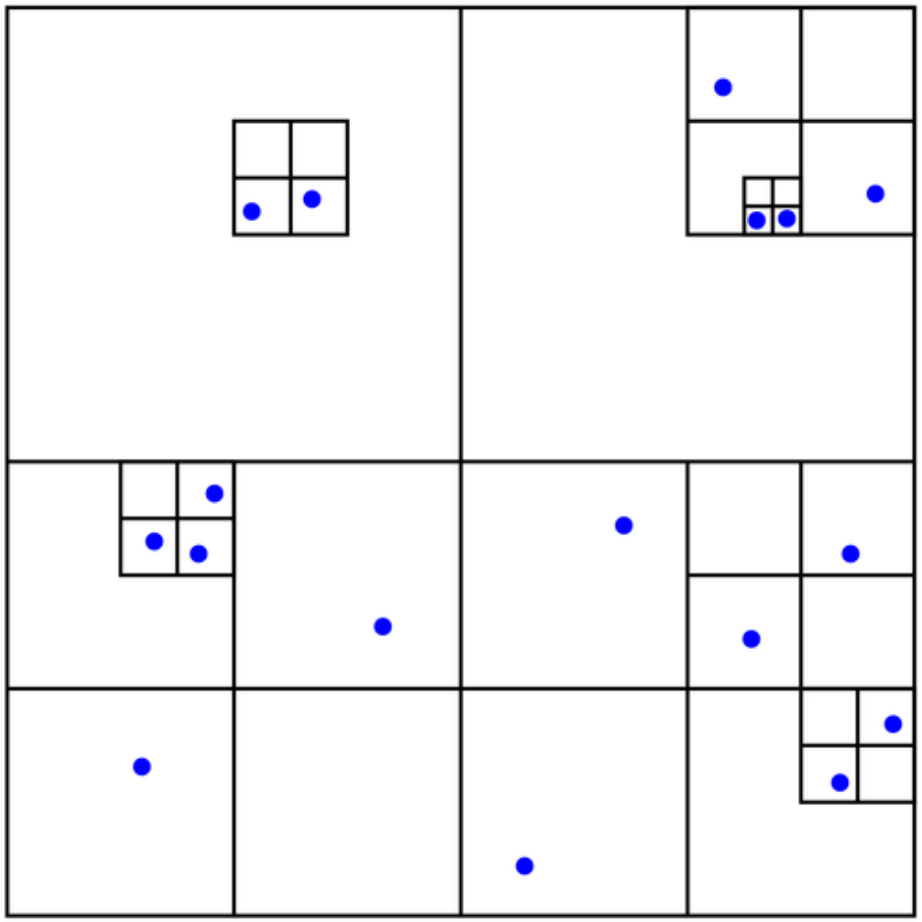
- Use the halfbox quadtree.
- Recurse when you hit a (d-2)-face.
- Otherwise, subtract all disjoint (d-1)-faces.
- Group version:
  - Query time: $O(1/\varepsilon^{d-2} + \log(1/\varepsilon))$.
  - Space: $O(\log(1/\varepsilon)/\varepsilon^{d})$.

Relative Error Model



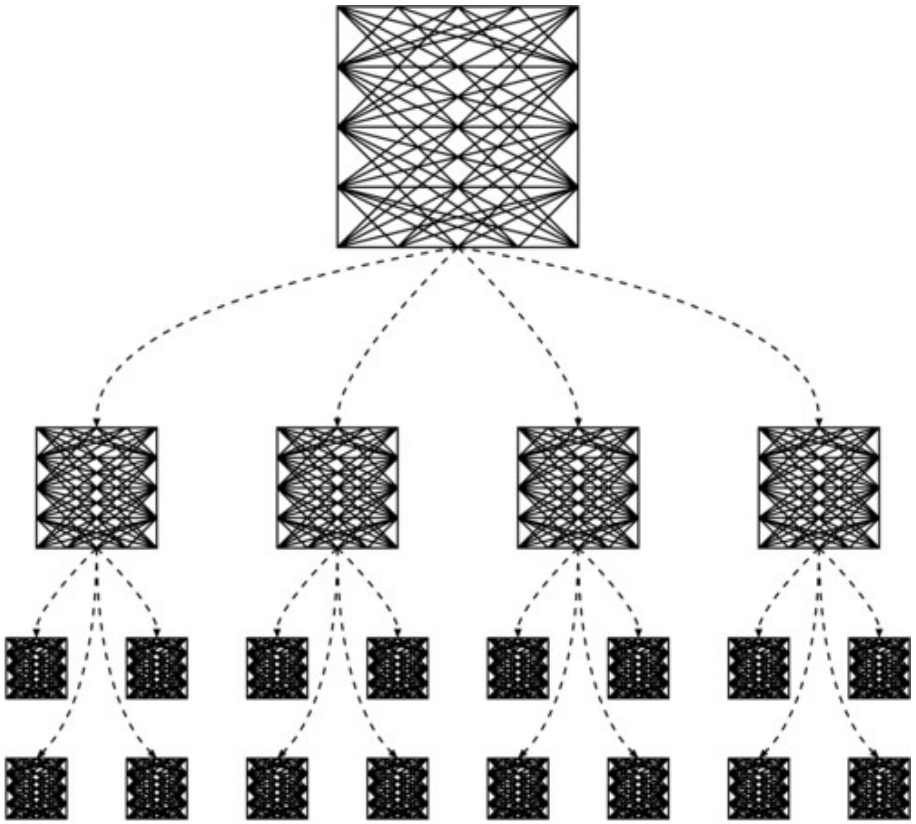
- Several data structures are known for the relative error model.
- Most use complicated properties from AVDs.
- We improve the query time, storage space, and preprocessing time with a simpler data structure.
- First data structure for simplex and smooth ranges.

Compressed Quadtree



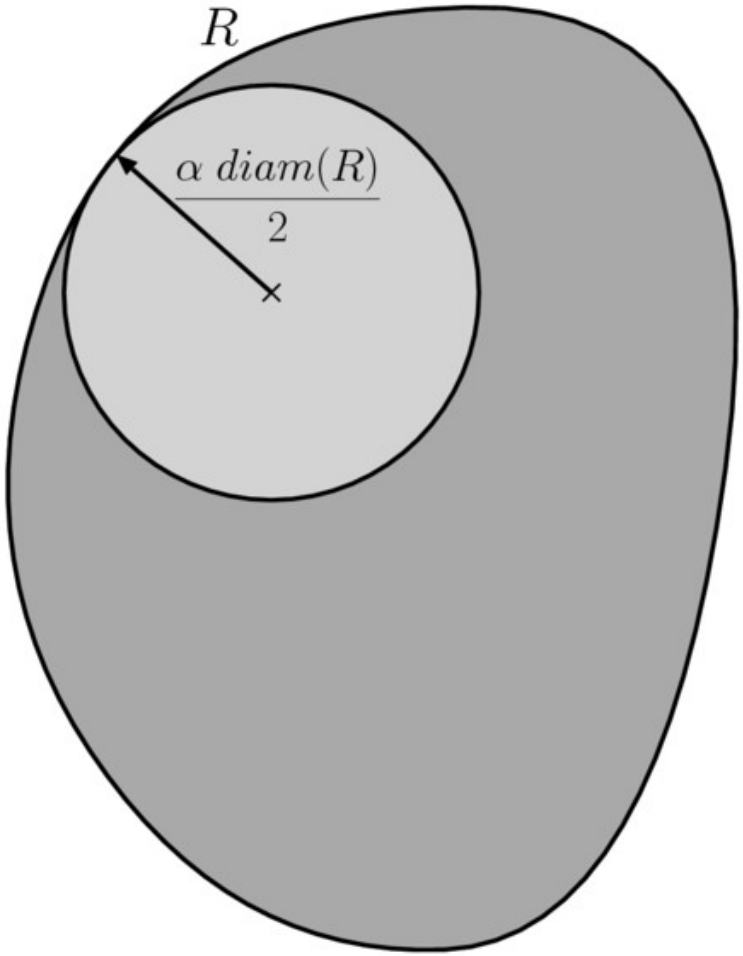
- The storage space of a quadtree is not bounded in terms of n.
- Compression can be used to make storage $O(n)$, but the height is still $\Theta(n)$.
- Fingers can be added to search the compressed quadtree in $O(\log n)$ time.
- Preprocessing takes $O(n \log n)$ time.

Relative Halfbox Quadtree



- Let γ be a parameter to control the tradeoff.
- Associate a (δ/γ)-approximate halfspace range searching data structure with each quadtree box of size δ.
- Storage: $O(n\gamma^d)$.
- Preprocessing: $O(n\gamma^d + n \log n)$.

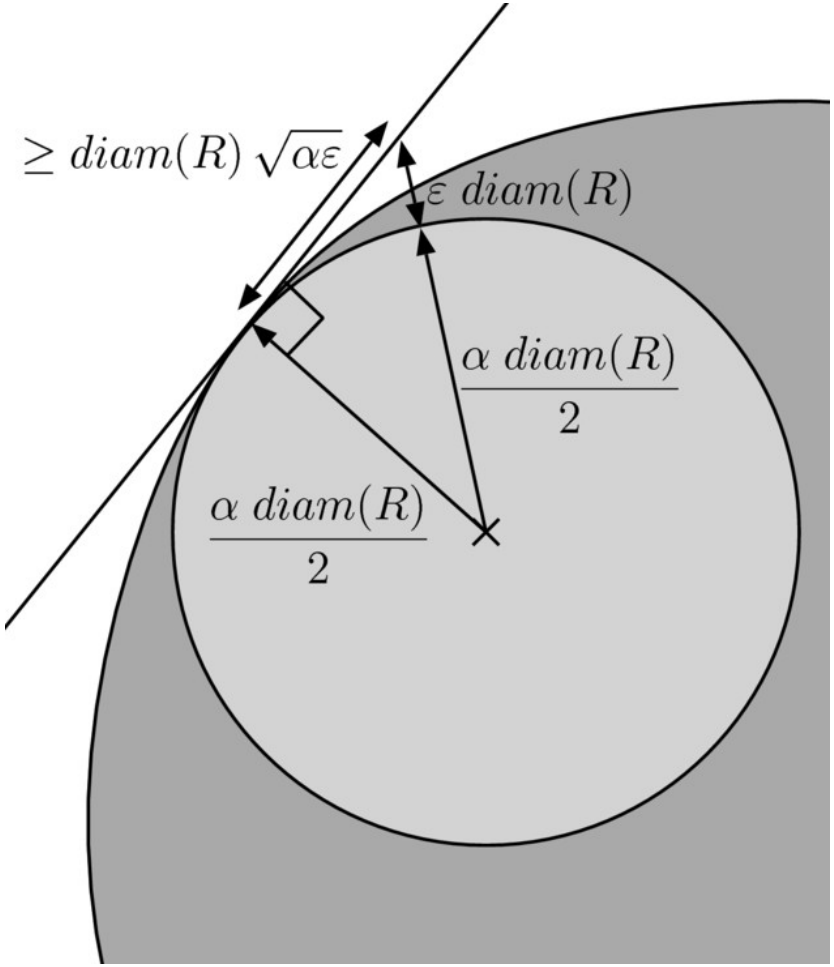
Smooth Range



- A convex range R is α-smooth if every point in the boundary of R can be touched by a ball of diameter α $diam(R)$ contained in R.
- A sphere is 1-smooth.
- A range is smooth if it is α-smooth for constant α.
- Idempotent data structure by Arya, Malamatos, and Mount, 2006.

Smooth Range Searching

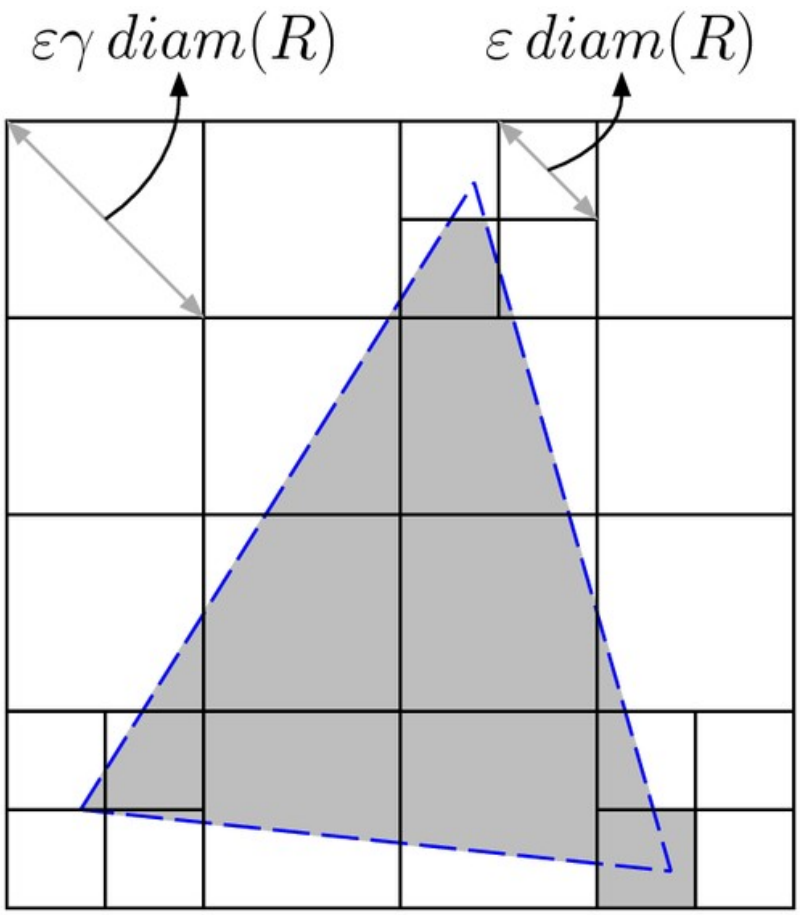
- Besides the unit-cost test assumption, we assume that a tangent hyperplane inside a quadtree box can be found in O(1) time.
- Use quadtree boxes of diameter at most $diam(R)\sqrt{\alpha\varepsilon}$ for the boundary.



Smooth Range Searching

- Besides the unit-cost test assumption, we assume that a tangent hyperplane inside a quadtree box can be found in O(1) time.
- Use quadtree boxes of diameter at most $diam(R)\sqrt{\alpha\varepsilon}$ for the boundary.
- Since each quadtree box of diameter δ contains a (δ/γ)-approximate data structure, use boxes of diameter at most εγ $diam(R)$ for the boundary.
- By packing lemma, the number of boxes is $O(1/\varepsilon^{(d-1)/2} + 1/(\varepsilon\gamma)^{d-1})$.
- Query time: $O(\log n + 1/\varepsilon^{(d-1)/2} + 1/(\varepsilon\gamma)^{d-1})$.

Simplex Range Searching (group version)



- Subtraction is used to handle the intersection with multiple disjoint halfspaces.
- Any box size can be used in the interior of the simplex.
- Only boxes of size at most ε $diam(R)$ can be used for the (d-2)-faces.
- Only boxes of size at most εγ $diam(R)$ can be used for the (d-1)-faces.

Query Time Analysis

- **Lemma:** If S is a set of pairwise disjoint quadtree boxes, each of diameter at least δ<1, that intersect the boundary of a convex region of diameter 1, then

$$|S| = O\left(\left(\frac{1}{\delta}\right)^{d-1}\right).$$

- **Lemma:** If S is a set of pairwise disjoint quadtree boxes, each of diameter at least δ<1, that intersect the (d-2)-faces of a simplex of diameter 1, then

$$|S| = O\left(\left(\frac{1}{\delta}\right)^{d-2}\right).$$

- Simplex query time:

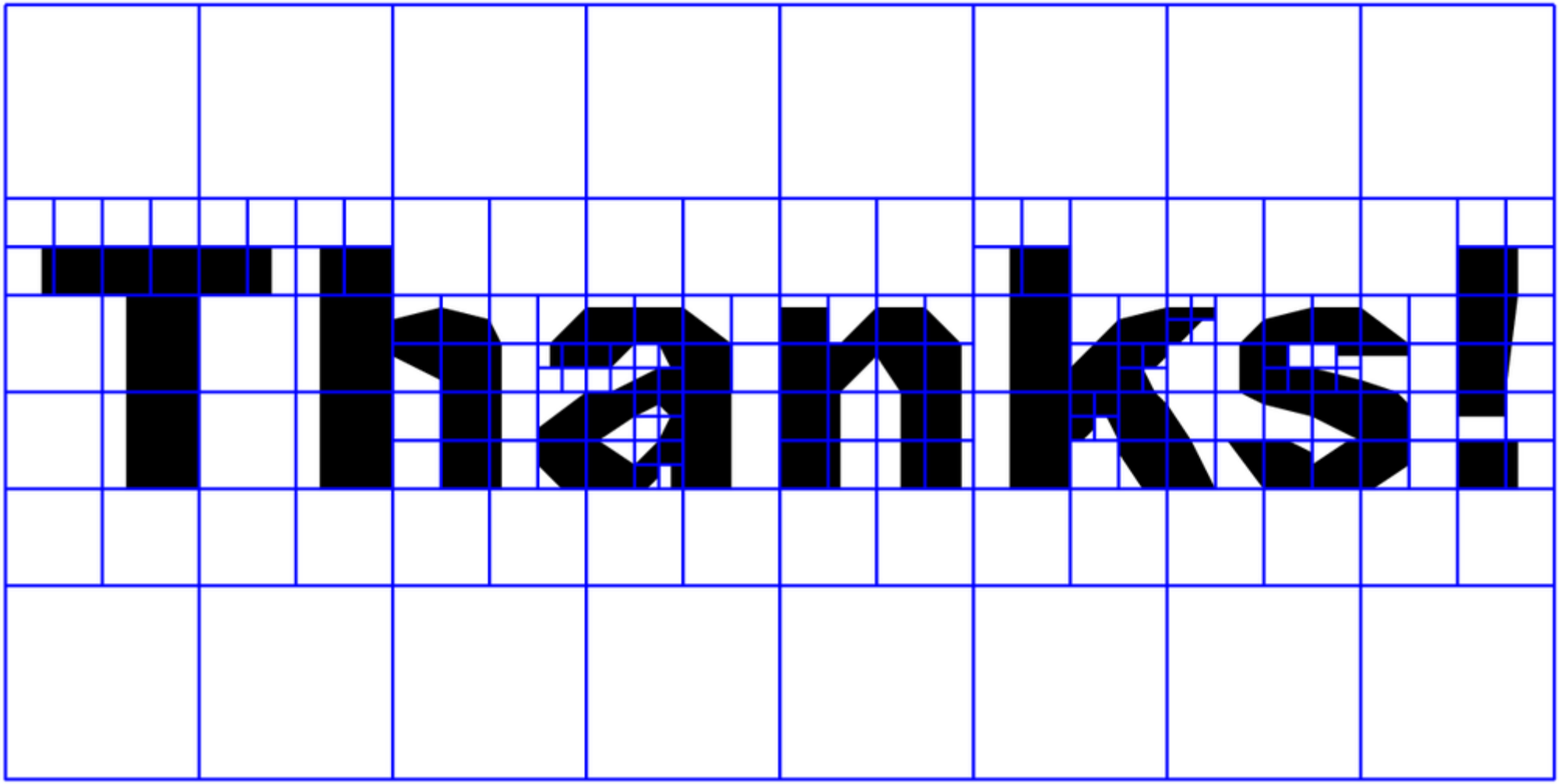
$$\sum_{i=0}^{\log O(1/\varepsilon)} O\left(\left(2^i\right)^{d-2}\right) + \sum_{i=0}^{\log O(1/\varepsilon\gamma)} O\left(\left(2^i\right)^{d-1}\right) = O\left(\log\left(\frac{1}{\varepsilon}\right) + \frac{1}{\varepsilon^{d-2}}\right) + O\left(\frac{1}{(\varepsilon\gamma)^{d-1}}\right).$$

Conclusions

- The absolute error model is better suited for several applications.
- Data structures are simpler than exact and other approximate data structures.
- Halfspace range searching is extremely fast.
- In the idempotent version, halfspace range searching requires little space (with slower query time).
- The halfbox quadtree is efficient for various shapes of ranges.
- The techniques extend to exact range searching and approximate range searching in the relative model.

Future Research

- Can we improve the data structures or obtain matching lower bounds?
- How much space is required to achieve O(1) query time for different shapes of ranges?
- Which other problems are interesting in the absolute error model?
- Can we construct an idempotent version of the relative halfbox quadtree?
- Can we obtain improved relative error model data structures?


Thanks!