

# Proximity Searching and the Quest for the Holy Grail

David M. Mount

Department of Computer Science  
University of Maryland, College Park

CG-APT 2012: Algorithms in the Field

# Proximity Searching

## Proximity searching:

A set of related geometric retrieval problems that involve finding the objects close to a given query object.

Given an  $n$ -element set  $P$  of points in a **metric space**. Will assume that the space is a **vector space** of **low-dimension** with a Minkowski norm.

- **Nearest neighbor searching**: Given a query point  $q$ , find the closest point of  $P$  to  $q$
- **(Bounded) Range searching**: Given a bounded query range  $Q$ , count/report the points of  $P \cap Q$

# Proximity Searching: Variants

Variations and issues:

- **Nearest-Neighbor Searching:**
  - $k$ -nearest neighbors
  - high dimensions (avoid exponential dependencies in dimension)
  - exploit properties of metric spaces (e.g., doubling dimension)
  - space-time tradeoffs
  - non-metric distances (e.g., Bregman Divergence)
- **Range Searching:**
  - range emptiness
  - more space-time tradeoffs
  - semigroup properties (integral:  $x + y$ , idempotent:  $\max(x, y)$ )

# Proximity Searching: Applications

## Applications:

- Pattern recognition and classification
- Object recognition in images (SIFT descriptors [Lowe 1999, 2004])
- Content-based retrieval:
  - Shape matching
  - Image retrieval
  - Document retrieval
  - Biometric identification (face/fingerprint/voice recognition)
- Clustering and phylogeny
- Data compression (vector quantization)
- Physical simulation (collision detection and response)
- Computer graphics: photon mapping and point-based modeling

... and many more

# The problem that launched a thousand data structures

- 2-dimensions
  - Voronoi diagram + point location
- Low dimensional vector spaces
  - grids, kd-trees, quadtrees, R-trees, ...and variants
  - approximate Voronoi diagrams (AVD) [Har-Peled 2001, Arya *et al.* 2009]
- High dimensional vector spaces
  - locality sensitive hashing (LSH) [Gionis *et al.* 1999, Andoni and Indyk, 2008]
- Metric spaces
  - metric trees and ring separator trees [Indyk and Motwani 1998, Krauthgamer and Lee 2005] (...and variants)
  - pivot-based methods (AESA, LAESA, and others) [Brin 1995] [Chavéz *et al.* 2001]

# Overview

- The Structureless Structure
- Enumerating Distances
- ANN via Polytope Membership

# Overview

- The Structureless Structure
- Enumerating Distances
- ANN via Polytope Membership

# The Structureless Structure

## Motivation

- “Constant factors” can play a big role in query times. For example, in  $O(\log n + (1/\epsilon)^d)$  the term  $(1/\epsilon)^d$  is dominant
- Constant factors are often hidden by the **memory model**
- Tree-based data structures (if naively implemented) have notoriously **poor** memory access patterns



# Morton Order

## Morton Order

- Consider a point set  $P$ , lying within the unit hypercube  $[0, 1]^d$
- For each  $p = (p_1, \dots, p_d) \in \mathbb{R}^d$ , assume its coordinates are given  $w$ -bit binary values  $p_j = \langle 0.b_{j,1} \dots b_{j,w} \rangle$
- Map  $p$  to an integer by **shuffling** the bits of its coordinates,

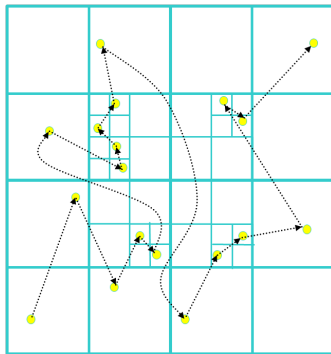
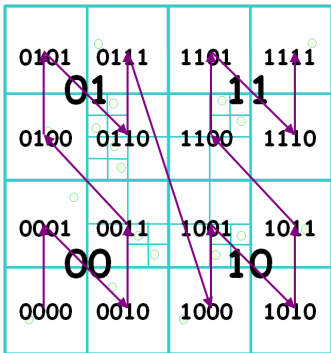
$$\sigma(p) = b_{1,1} \dots b_{d,1} | b_{1,2} \dots b_{d,2} | \dots | b_{1,w} \dots b_{d,w}$$

- This is called the **Morton order** or **Z order**.

# Linear Quadtree

## Linear Quadtree

- Sort  $P$  by Morton order
- Store the points in an array (or any 1-dimensional index)



# Linear Quadtree – Easy Shuffling

## Chan's Shuffle Trick [Chan 2002]

Compare Morton codes without bit manipulation, just exclusive-or!

```
// tests whether  $\lfloor \log_2 x \rfloor < \lfloor \log_2 y \rfloor$   
f(x, y) { return (x > y ? false : x < (x  $\oplus$  y)) }
```

```
// test whether  $\sigma(p) < \sigma(q)$   
compare(p, q) {  
    i  $\leftarrow$  1  
    for j  $\leftarrow$  2, ..., d do  
        if (f(pi  $\oplus$  qi, pj  $\oplus$  qj)) i  $\leftarrow$  j  
    return pi < qi  
}
```

# A Minimalist Approach to Nearest Neighbor Searching

Chan [Chan 2006] showed that it is possible to use a Morton-sorted array (no additional information) to answer **approximate nearest neighbor queries**

- Apply a **random shift** to the origin
- Query time is  $O(\log n + (1/\varepsilon)^d)$  in **expectation**
- Space is  $O(n)$ , in fact, it is an **in-place** algorithm
- Preprocessing time is  $O(n \log n)$
- Easily made dynamic (e.g., store in a skip list)

The program is **absurdly short** – less than 60 lines of C!

**Competitive** with ANN (my kd-tree implementation) in low dimensions

# Overview

- The Structureless Structure
- Enumerating Distances
- ANN via Polytope Membership

# Distance Enumeration

## Motivations:

- **Object-recognition:** Want a sufficiently large number of high quality features [Lowe 1999]
- **Global illumination:** Want to collect a sufficiently large number of sampled photons near a point [Jensen 2001]

Want the  $k$  nearest neighbors of  $q$ , but want to pick  $k$  on the fly

# Distance Enumeration

## Distance Enumerator:

- Visit the points  $P$  in **increasing order of distance** from a point  $q$
- Let  $\Pi(q) = \langle \pi_1, \dots, \pi_n \rangle$ , where  $p_{\pi_k}$  is  $q$ 's  $k$ th nearest neighbor
- Generate the elements of  $\Pi(q)$  efficiently, one at a time

## $(c, \varepsilon)$ -Enumerator

After preprocessing  $P$ , given a query point  $q$ , produces a generator for a  $\Pi'(q)$  such that:

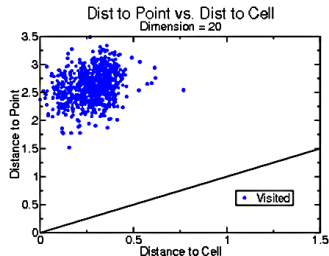
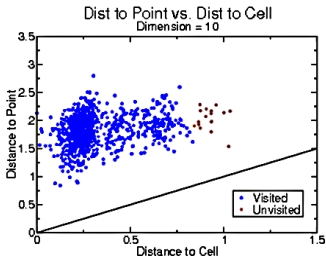
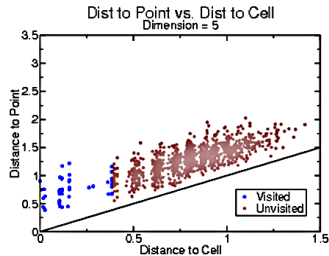
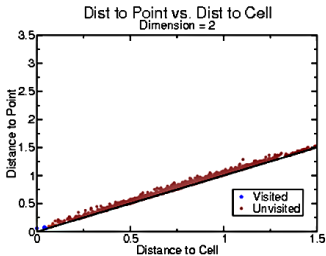
- Successive elements of  $\Pi'(q)$  generated rapidly, e.g.,  $O(\log n)$  time
- For  $1 \leq k \leq n$ , a  $(1 + \varepsilon)$  approximation to  $q$ 's  $k$ -th nearest neighbor appears among the first  $c \cdot k$  elements of  $\Pi'$

# Priority Search

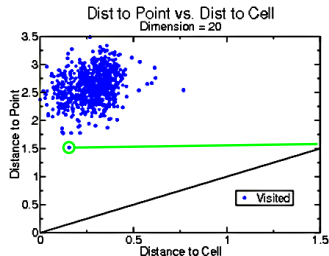
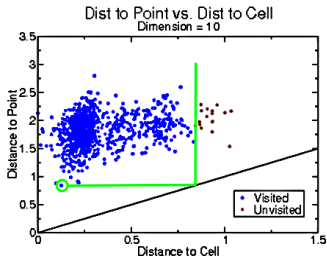
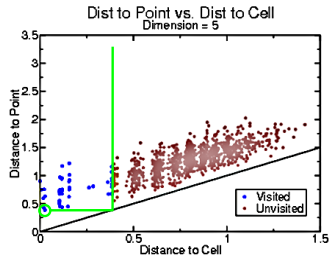
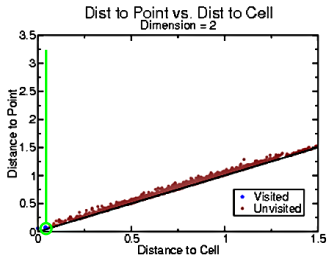
- Build a kd-tree  $T$  for  $P$
- For each node  $u$ , let  $C(u)$  be the cell associated with  $u$
- **Priority Search:**
  - Store the root  $u$  of  $T$  in a priority queue based on  $\text{dist}(q, C(u))$
  - Repeat until queue is empty:
    - Extract closest node  $u$  from the queue
    - If  $u$  is a leaf then output the associate point
    - Otherwise, enqueue  $u$ 's two children
- A  $(c, \varepsilon)$ -distance enumerator for  $c = O(1/\varepsilon^d)$  [Arya *et al.* 1998]



# Priority Search



# Priority Search



# Enhancing Robustness

Generate multiple “randomized” trees [Silpa-Anan and Hartley, 2008]

- Select splitting axis at random (after PCA)
- Rotate the points randomly:  $O(d^2n)$
- Project the points through a random hyperplane:  $O(dn)$
- Generate  $m$  such trees and enumerate  $c$  points from each
- Total time  $O(c \cdot m \log n)$

Cluster-based method [Muja and Lowe 2009]

- Preprocessing:
  - Perform  $k$ -means clustering for some  $k$  (depending on dimension)
  - Partition points into subtrees based on these clusters
  - Recurse
- Enumerate by visiting subtrees in order of distance of cluster center to query point

# Overview

- The Structureless Structure
- Enumerating Distances
- ANN via Polytope Membership

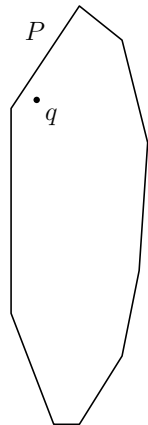
# Polytope Membership Queries

## Polytope Membership Queries

Given a polytope  $P$  in  $d$ -dimensional space, preprocess  $P$  to answer membership queries:

Given a point  $q$ , is  $q \in P$ ?

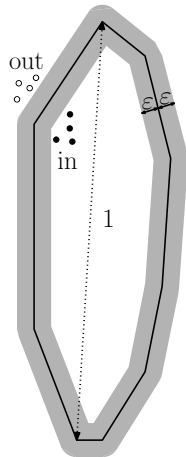
- Assume that dimension  $d$  is a constant and  $P$  is given as intersection of  $n$  halfspaces



# Approximate Polytope Membership Queries

## Approximate Version

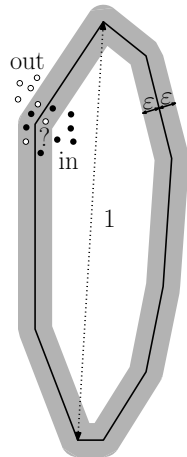
- An **approximation parameter**  $\epsilon$  is given (at preprocessing time)
- Assume the polytope has **diameter 1**
- If the query point's distance from  $P$ 's boundary:
  - $> \epsilon$ : answer must be **correct**
  - $\leq \epsilon$ : **either** answer is acceptable



# Approximate Polytope Membership Queries

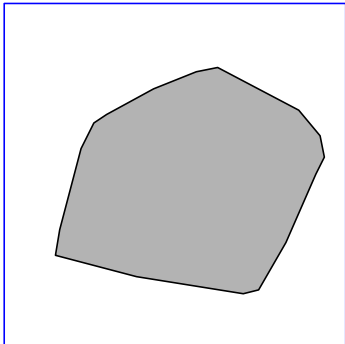
## Approximate Version

- An **approximation parameter**  $\epsilon$  is given (at preprocessing time)
- Assume the polytope has **diameter 1**
- If the query point's distance from  $P$ 's boundary:
  - $> \epsilon$ : answer must be **correct**
  - $\leq \epsilon$ : **either** answer is acceptable



# Split-Reduce

$t = 2$



## Preprocess:

- Input  $P$ ,  $\varepsilon$ , and desired query time  $t$
- $Q \leftarrow$  unit hypercube
- Split-Reduce( $Q$ )

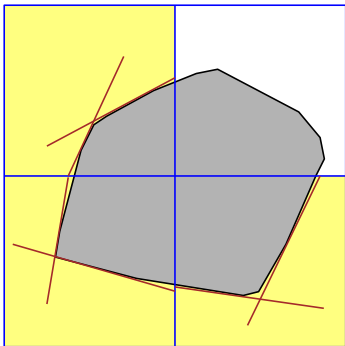
## Split-Reduce( $Q$ )

- Find an  $\varepsilon$ -approximation of  $Q \cap P$
- If **at most  $t$  facets**, then  $Q$  stores them
- Otherwise, **subdivide**  $Q$  and recurse



# Split-Reduce

$t = 2$



## Preprocess:

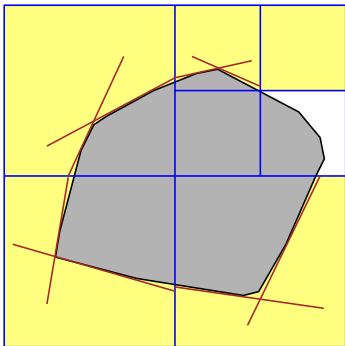
- Input  $P$ ,  $\epsilon$ , and desired query time  $t$
- $Q \leftarrow$  unit hypercube
- Split-Reduce( $Q$ )

## Split-Reduce( $Q$ )

- Find an  $\epsilon$ -approximation of  $Q \cap P$
- If **at most  $t$  facets**, then  $Q$  stores them
- Otherwise, **subdivide**  $Q$  and recurse

# Split-Reduce

$t = 2$



## Preprocess:

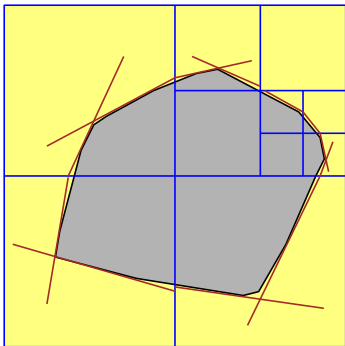
- Input  $P$ ,  $\epsilon$ , and desired query time  $t$
- $Q \leftarrow$  unit hypercube
- Split-Reduce( $Q$ )

## Split-Reduce( $Q$ )

- Find an  $\epsilon$ -approximation of  $Q \cap P$
- If **at most  $t$  facets**, then  $Q$  stores them
- Otherwise, **subdivide**  $Q$  and recurse

# Split-Reduce

$t = 2$



## Preprocess:

- Input  $P$ ,  $\epsilon$ , and desired query time  $t$
- $Q \leftarrow$  unit hypercube
- Split-Reduce( $Q$ )

## Split-Reduce( $Q$ )

- Find an  $\epsilon$ -approximation of  $Q \cap P$
- If **at most  $t$  facets**, then  $Q$  stores them
- Otherwise, **subdivide**  $Q$  and recurse

# Space-Time Tradeoff

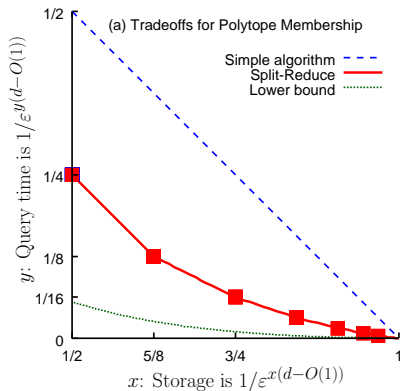
Arya *et al.* prove the following space-time tradeoff [Arya *et al.* 2011] for polytope membership queries

## Theorem:

Split-Reduce can answer  $\varepsilon$ -approximate polytope membership queries with

**Storage:**  $O(1/\varepsilon^{(d-1)/(1-k/2^k)})$

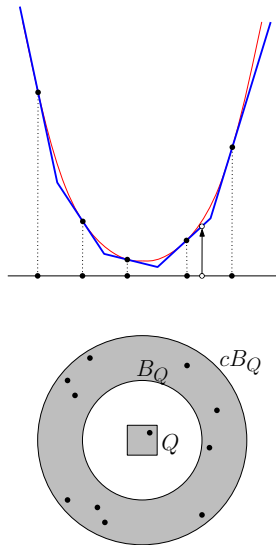
**Query time:**  $O(1/\varepsilon^{(d-1)/2^k})$



# Approximate Nearest Neighbor (ANN) Searching

Approximate nearest neighbor searching can be **reduced** to approximate polytope membership:

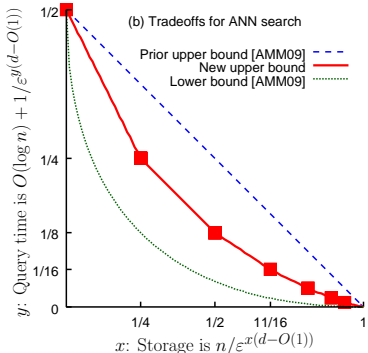
- **Lift:** By lifting, we can reduce nearest neighbor search to ray shooting to a polytope
- **Separate:** Partition space into roughly  $O(n)$  cells such that all nearest neighbors of each cell are of similar distance (AVD)



# Approximate Nearest Neighbor (ANN) Searching

## Space-Time Tradeoffs for ANN Search

- Arya, *et al.* [2009] showed that ANN queries could be answered in query time roughly  $O(1/\varepsilon^{d/\alpha})$  with storage roughly  $O(n/\varepsilon^{d(1-2/\alpha)})$ , for  $\alpha \geq 2$
- This is **optimal in the extremes**
- The reduction to polytope membership queries improves the tradeoff throughout the spectrum



# Conclusions

Next steps on the quest?

- Better analyses of the performance of methods on **realistic data sets**
- Emphasis on **general/flexible methods** (distance enumeration)
- More **repositories** of good test data
- How to **explore/visualize/analyze** the structure of multi-dimensional data sets



# Acknowledgements

- The work on polytope approximation is joint with Sunil Arya and Guilherme da Fonseca
- I would also like to acknowledge the support of NSF grant CCR-0635099 and ONR grant N00014-08-1-1015



# Bibliography

- M. Muja and D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration. *Int. Conf. on Computer Vision Theory and App.*, (VISSAPP'09), 2009, 331–340.
- C. Silpa-Anan and R. Hartley, Optimised KD-trees for fast image descriptor matching. *Proc. CVPR*, 2008, 1–8.
- A. Andoni and P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51, 2008, 117–122.
- S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. ACM*, 45, 1998, 891–923.
- S. Arya, T. Malamatos, and D. M. Mount, Space-Time Tradeoffs for Approximate Nearest Neighbor Searching. *J. ACM*, 57, 2009, 1–54.
- S. Arya, G. D. da Fonseca, and D. M. Mount, Approximate Polytope Membership Queries. *Proc. 43rd Symp. on Theory of Comput.*, 2011, 579–586.
- S. Brin, Near neighbor search in large metric spaces. *Proc. 21st VLDB Conf.*, 1995, 574–584.

# Bibliography

- T. M. Chan, Closest-point problems simplified on the RAM. *Proc. 13th ACM-SIAM SODA*, 2002, 472–473.
- T. M. Chan, A minimalist implementation of an approximate nearest neighbor algorithm in fixed dimensions. unpublished manuscript, 2006.
- E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, Searching in Metric Spaces. *ACM Comput. Surveys*, 33, 2001, 273–321.
- A. Gionis, P. Indyk, R. Motwani. Similarity search in high dimensions via hashing. *Proc. 25th VLDB Conf.*, 1999, 518–529.
- S. Har-Peled. A replacement for Voronoi diagrams of near linear size. *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, 2001, 94–103.
- P. Indyk, and R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality. *Proc. 30th Symp. on Theory of Comput.*, 1998, 604–613.
- H. W Jensen, *Realistic Image Synthesis Using Photon Mapping*. A.K. Peters, 2001

# Bibliography

- R. Krauthgamer and J. R. Lee, The black-box complexity of nearest neighbor search. *Theoret. Comput. Sci.*, 348, 2005, 129–366.
- D. G. Lowe, Object recognition from local scale-invariant features. *Proc. Int. Conf. on Computer Vision*, 2, 1999, 1150–1157.
- D. G. Lowe, Distinctive image features from scale-invariant keypoints *Int. J. of Computer Vision*, 60, 2004, 91-110.
- M. L. Mico, J. Oncina, and E. Vidal, A new version of the nearest-neighbour approximating and eliminating search algorithm (AESAs) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, 15, 1994, 9–17