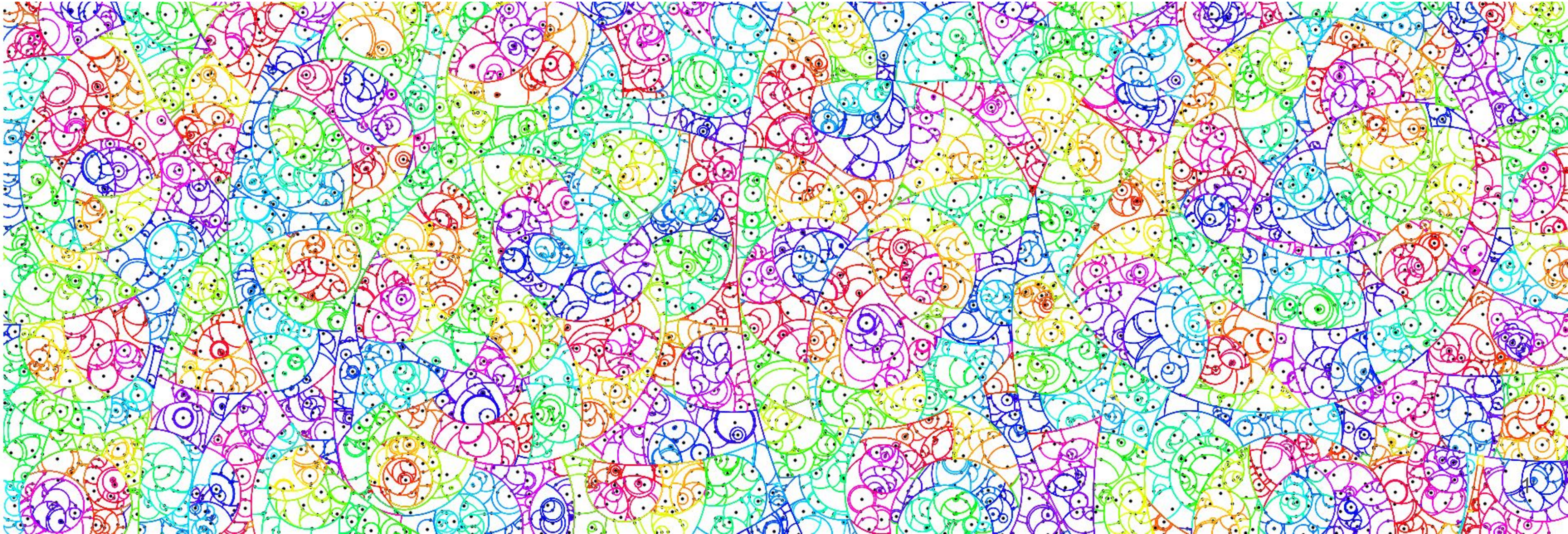
Vantage Point Trees

Based on “Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces”, Peter N. Yianilos (1993) - http://algorithmics.lsi.upc.edu/docs/practicas/p311-yianilos.pdf

(Image from https://fribbels.github.io/vptree/writeup)



Problem

- Nearest Neighbor Search
  - given X a dataset of N points and a query point q, find the point x_i minimizing the distance between itself and q.

Problem

- Nearest Neighbor Search
  - given X a dataset of N points and a query point q, find the point x_i minimizing the distance between itself and q.

- Set of possible solutions
  - Brute Force, $O(N^2)$

Problem

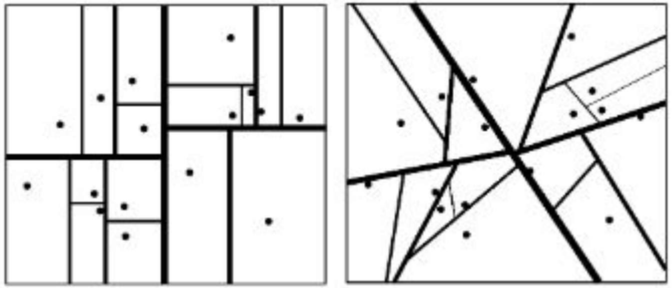
- Nearest Neighbor Search
  - given X a dataset of N points and a query point q, find the point x_i minimizing the distance between itself and q.

- Set of possible solutions
  - Brute Force, $O(N^2)$
  - Metric Trees
    - KD-trees

Problem

- Nearest Neighbor Search
  - given X a dataset of N points and a query point q, find the point x_i minimizing the distance between itself and q.

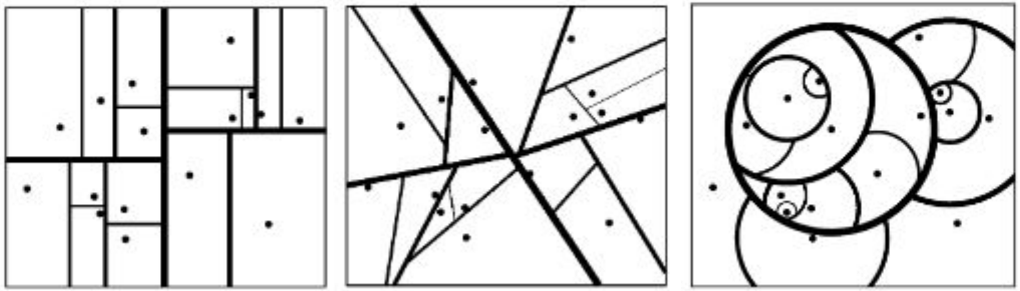
- Set of possible solutions
  - Brute Force, $O(N^2)$
  - Metric Trees
    - KD-trees
    - Ball Trees



Problem

- Nearest Neighbor Search
  - given X a dataset of N points and a query point q, find the point x_i minimizing the distance between itself and q.

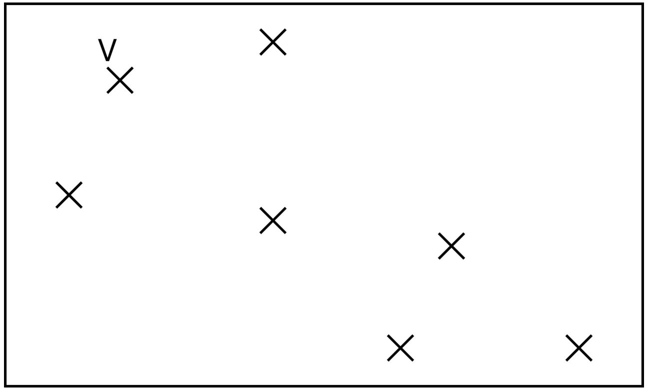
- Set of possible solutions
  - Brute Force, $O(N^2)$
  - Metric Trees
    - KD-trees
    - Ball Trees
    - **Vantage Point Trees**
      - O(N log(N)) to build it
      - O(log(N)) for a NN Search



(Images from https://link.springer.com/content/pdf/10.1007/978-3-540-88688-4_27.pdf)

Vantage Point Tree: Construction

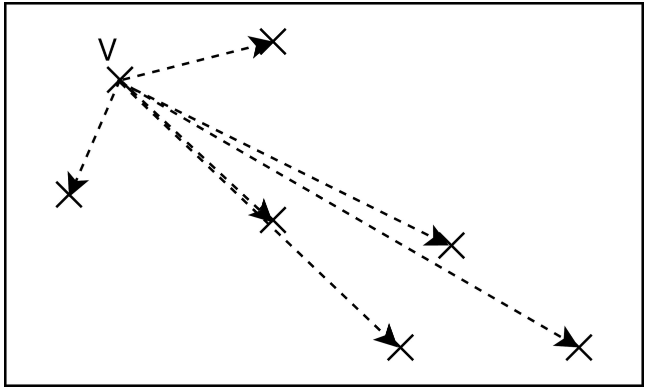
1. Select a vantage point v in X (eg. following a uniform distribution);



Vantage Point Tree: Construction

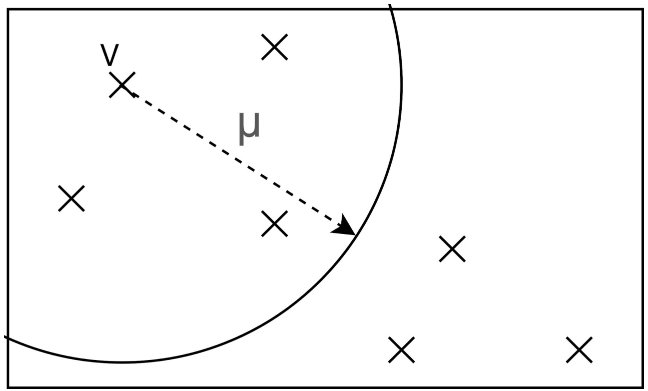
1. Select a vantage point v in X (eg. following a uniform distribution);
2. Compute the distances d(v, xi) between v and each point xi in X;

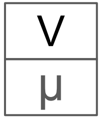
Euclidean in this example but works with any metric respecting the triangle inequality!



Vantage Point Tree: Construction

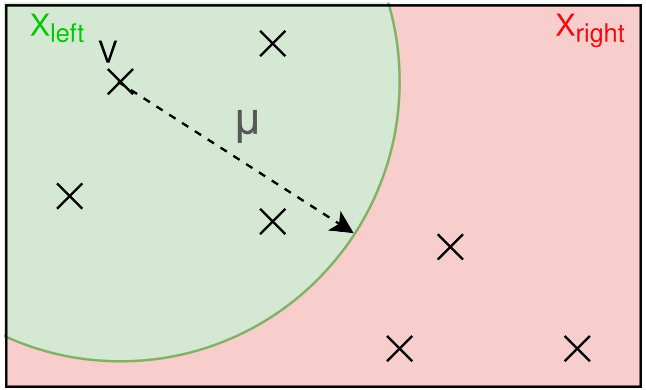
1. Select a vantage point v in X (eg. following a uniform distribution);
2. Compute the distances d(v, xi) between v and each point xi in X;
3. Take the median μ of these distances;

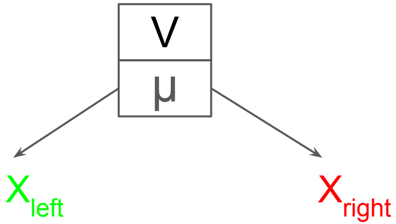




Vantage Point Tree: Construction

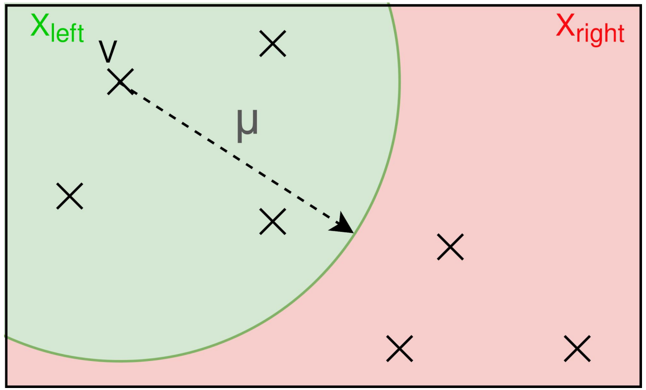
1. Select a vantage point v in X (eg. following a uniform distribution);
2. Compute the distances d(v, xi) between v and each point xi in X;
3. Take the median μ of these distances;
4. Divide X in 2 sets using μ as a threshold:
    a. X_{left} the set of points closest to v is put at the left
    b. X_{right} the set of points furthest from v is put at the right

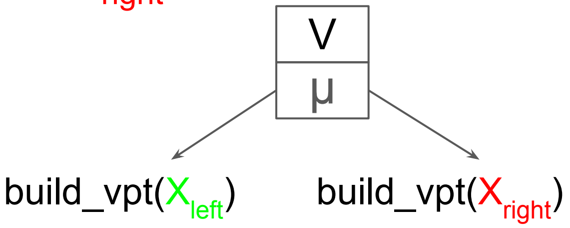




Vantage Point Tree: Construction

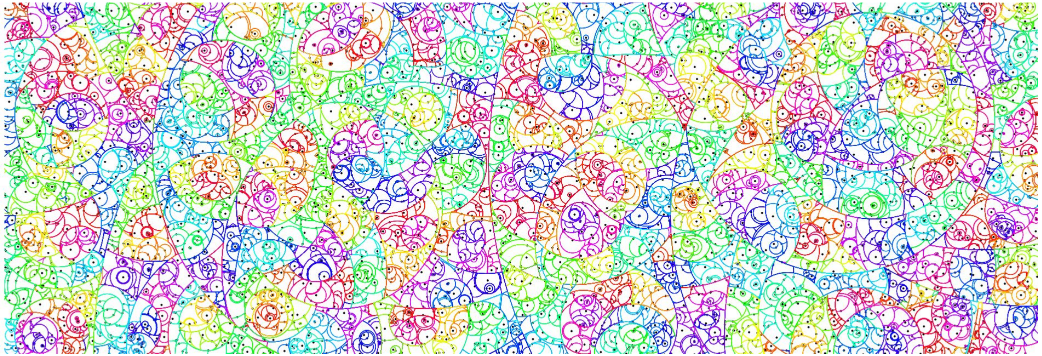
1. Select a vantage point v in X (eg. following a uniform distribution);
2. Compute the distances d(v, xi) between v and each point xi in X;
3. Take the median μ of these distances;
4. Divide X in 2 sets using μ as a threshold:
   a. X_{left} the set of points closest to v is put at the left
   b. X_{right} the set of points furthest from v is put at the right
5. Recursively apply the same algorithm for X_{left} and X_{right}

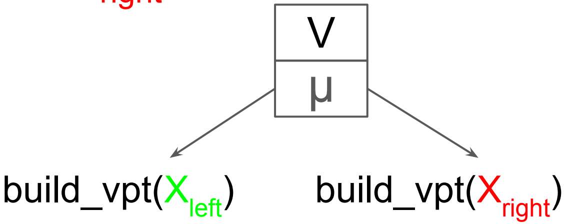




Vantage Point Tree: Construction

1. Select a vantage point v in X (eg. following a uniform distribution);
2. Compute the distances d(v, xi) between v and each point xi in X;
3. Take the median μ of these distances;
4. Divide X in 2 sets using μ as a threshold:
   a. X_{left} the set of points closest to v is put at the left
   b. X_{right} the set of points furthest from v is put at the right
5. Recursively apply the same algorithm for X_{left} and X_{right}

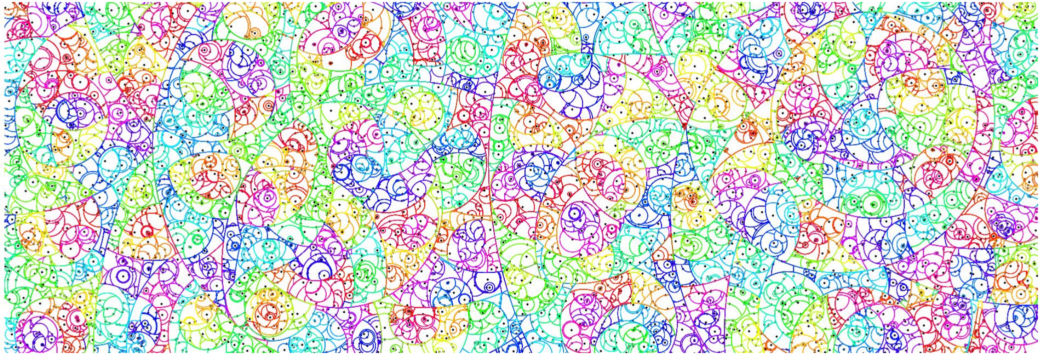


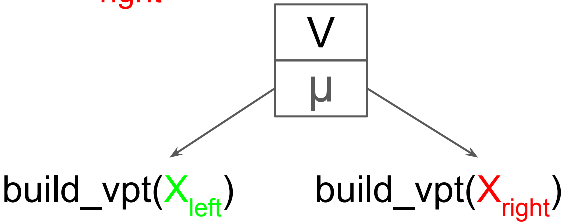


Vantage Point Tree: Construction

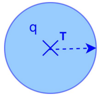
1. Select a vantage point v in X (eg. following a uniform distribution);
2. Compute the distances d(v, xi) between v and each point xi in X;
3. Take the media
4. Divide X in 2 se
    a. X_{left} the set of
    b. X_{right} the set o
5. Recursively apply the same algorithm for X_{left} and X_{right}

DEMO
https://fribbels.github.io/vptree/writeup

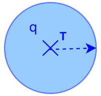




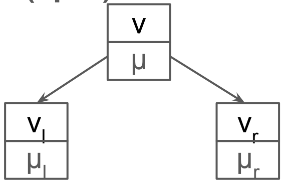
Vantage Point Tree: NN Search

- Given a query point q, we define **τ** the radius of the sphere representing the search space in which we search the NN of q 

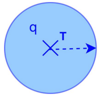
Vantage Point Tree: NN Search

- Given a query point q, we define τ the radius of the sphere representing the search space in which we search the NN of q 

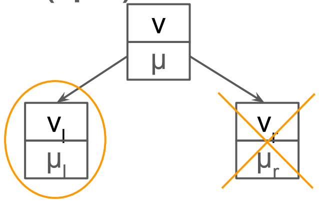
- Beginning at the root (v,μ) of the VP Tree:
  1. Check whether $d(q,v) < \tau$: if true then current NN = v and $\tau = d(q,v)$

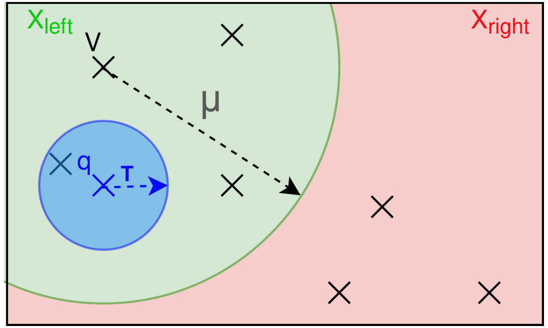


Vantage Point Tree: NN Search

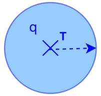
- Given a query point q, we define τ the radius of the sphere representing the search space in which we search the NN of q 

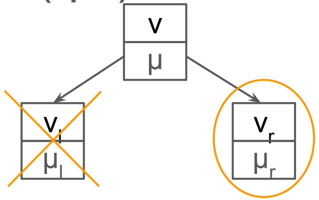
- Beginning at the root (v,μ) of the VP Tree:
  1. Check whether d(q,v) < τ : if true then current NN = v and τ = d(q,v)
  2. Then 3 cases:
     a. If the q-sphere is inside the v-sphere: explore left tree only

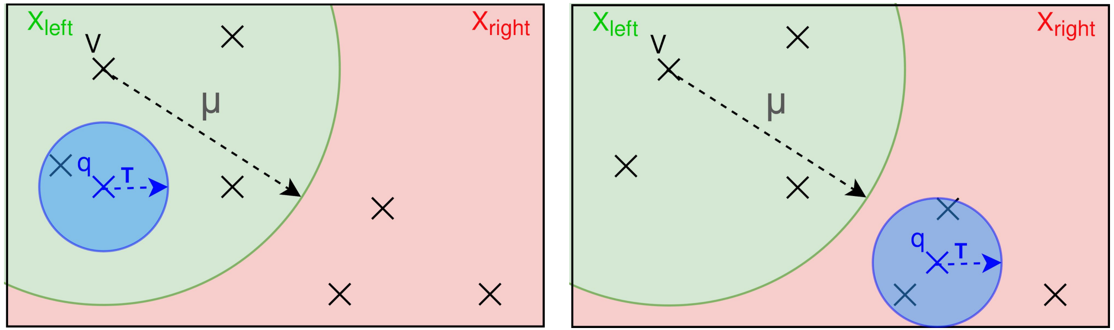




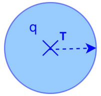
Vantage Point Tree: NN Search

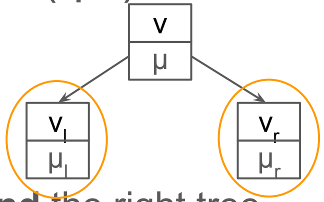
- Given a query point q, we define τ the radius of the sphere representing the search space in which we search the NN of q 
- Beginning at the root (v,μ) of the VP Tree:
  1. Check whether d(q,v) < τ : if true then current NN = v and τ = d(q,v)
  2. Then 3 cases:
     a. If the q-sphere is inside the v-sphere: explore left tree only
     b. If the q-sphere is not inside the v-sphere: explore right tree only

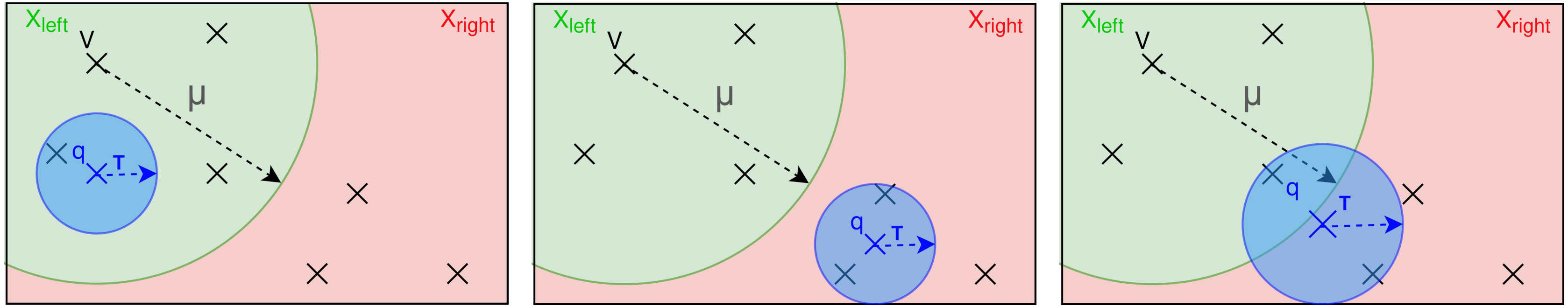




Vantage Point Tree: NN Search

- Given a query point q, we define τ the radius of the sphere representing the search space in which we search the NN of q 
- Beginning at the root (v,μ) of the VP Tree:
    1. Check whether d(q,v) < τ : if true then current NN = v and τ = d(q,v)
    2. Then 3 cases:
        a. If the q-sphere is inside the v-sphere: explore left tree only
        b. If the q-sphere is not inside the v-sphere: explore right tree only
        c. If the q-sphere is inside and outside the v-sphere: explore the left tree **and** the right tree





Vantage Point Tree: Extensions

- VP^S-Tree
  - Instead of retaining only the median, a node retains 2 values per ancestor corresponding to the boundaries of the corresponding subspace

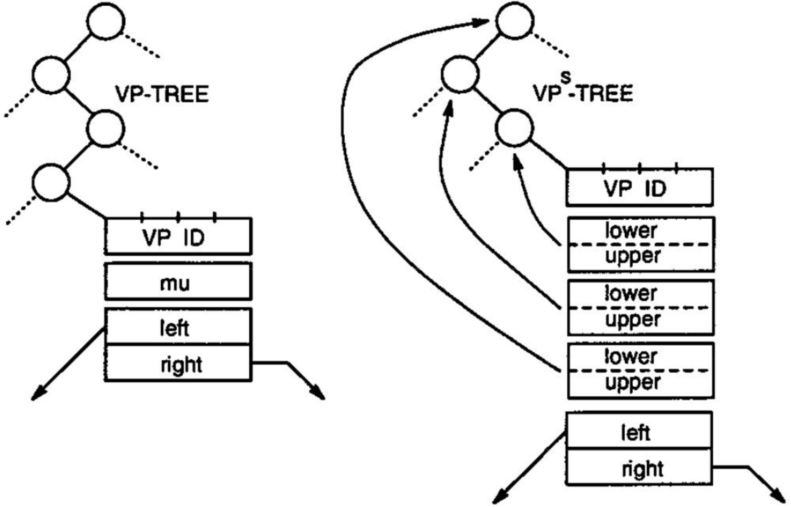


Figure 5: Sample 32-bit machine data structure implementations for the most basic vp-tree, the vp^{s}-tree, and the vp^{sb}-tree.

Vantage Point Tree: Extensions

- VP^S-Tree
  - Instead of retaining only the median, a node retains 2 values per ancestor corresponding to the boundaries of the corresponding subspace
- VP^{SB}-Tree
  - A VP^S-Tree in which each leaf contains B points resulting in a bucket structure
  - Reduced memory consumption

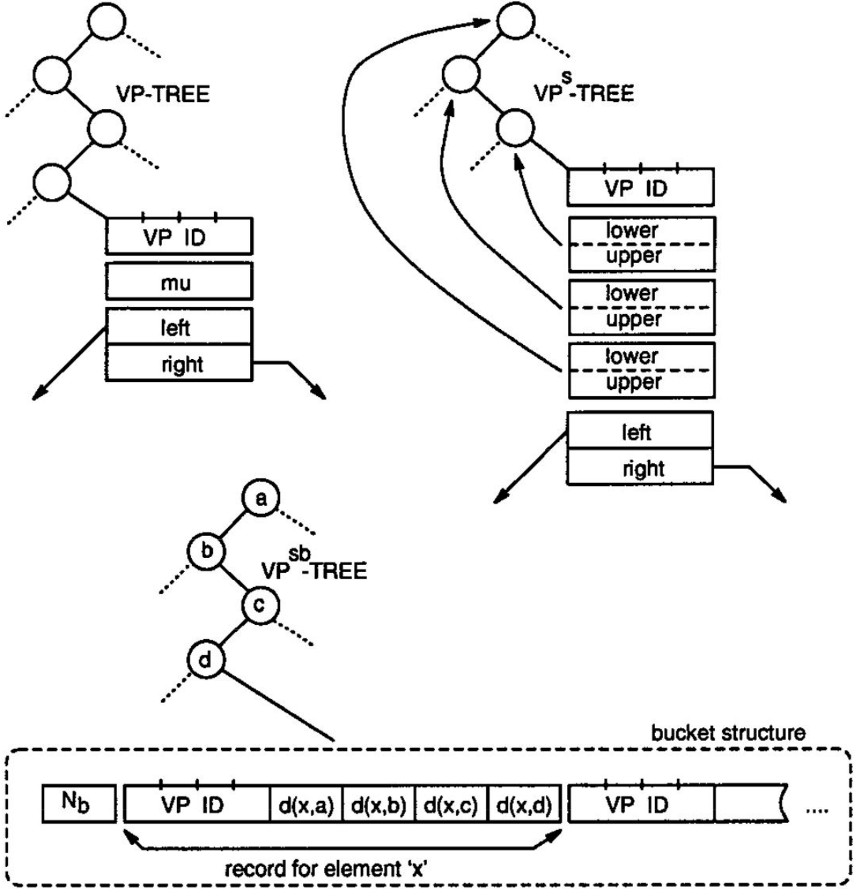


Figure 5: Sample 32-bit machine data structure implementations for the most basic vp-tree, the vp^s-tree, and the vp^{sb}-tree.

Vantage Point Tree: Extensions

- VP^S-Tree
  - Instead of retaining only the median, a node retains 2 values per ancestor corresponding to the boundaries of the corresponding subspace
- VP^{SB}-Tree
  - A VP^S-Tree in which each leaf contains B points resulting in a bucket structure
  - Reduced memory consumption
- VP Tree in non-metric space
  - With Bregman divergences: "Bregman Vantage Point Trees for Efficient Nearest Neighbor Queries" (Nielsen et al., 2009)

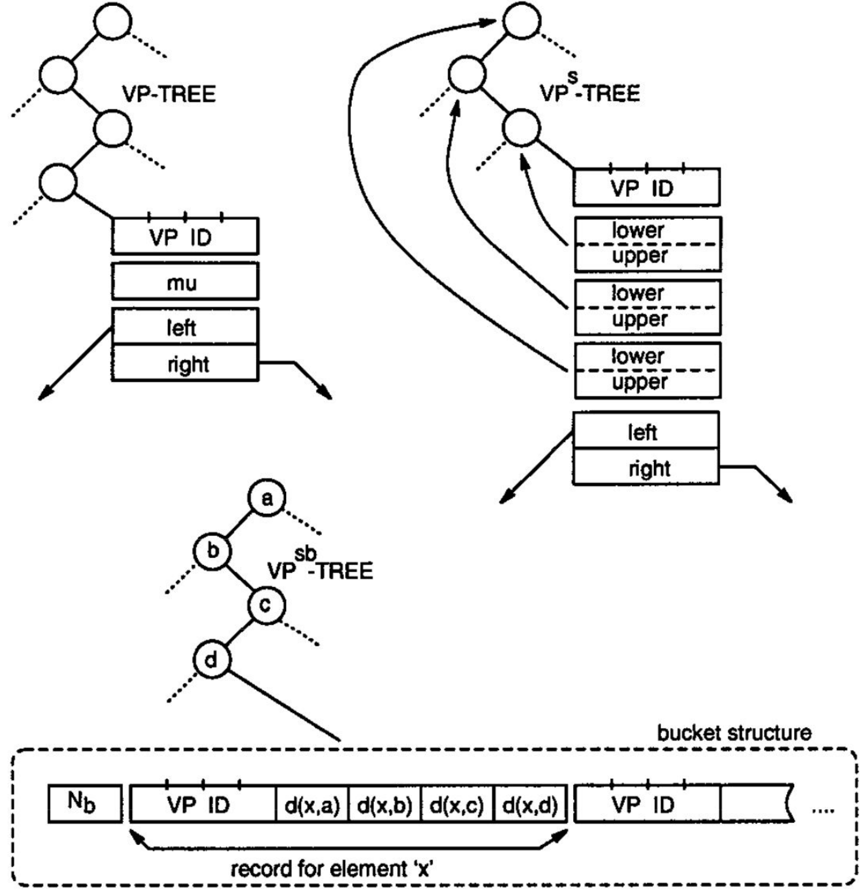


Figure 5: Sample 32-bit machine data structure implementations for the most basic vp-tree, the vp^s-tree, and the vp^{sb}-tree.

Performance comparison with classical metric trees

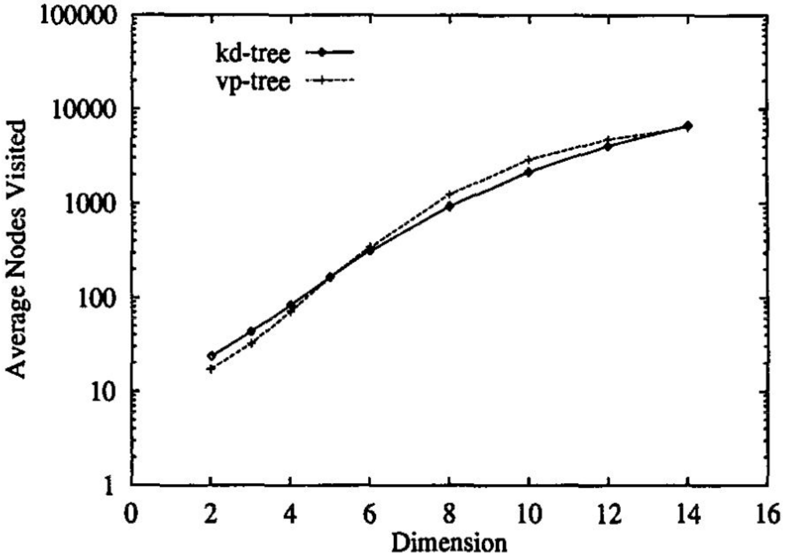


Figure 6: Search Cost vs. Dimension Comparison For L_2 Metric and Based on Nodes Visited

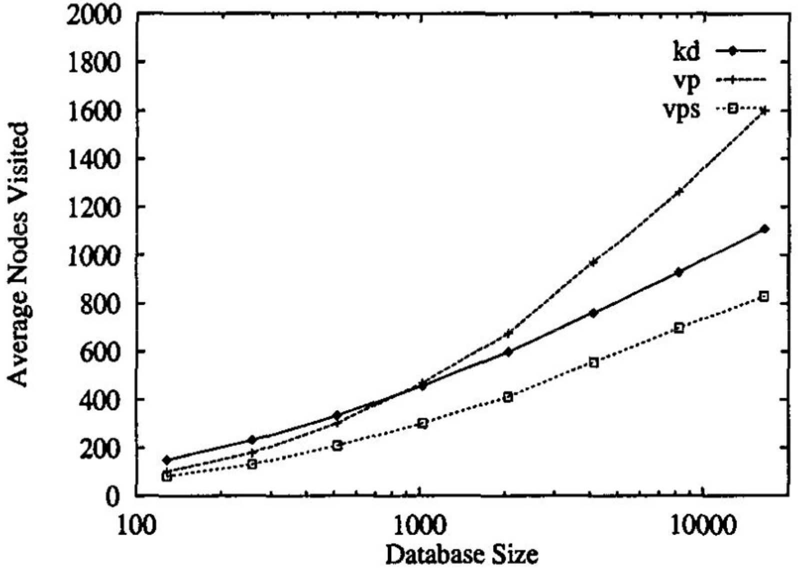


Figure 7: Search Cost vs. Database Size - Dimension 8

Performance comparison with classical metric trees

- "What Is a Good Nearest Neighbors Algorithm for Finding Similar Patches in Images?" (Neeraj et al., 2008)

Table 2. Summary of results. The *vp*-tree performs well in all respects.

Method	Construction Performance	ε-NN Search Performance	k-NN Search Performance
kd-Tree	Excellent	Poor	Poor
PCA Tree	Poor	Fair	Fair
Ball Tree	Fair	Excellent	Excellent
k-Means	Poor	Good	Good
vp-Tree	Excellent	Excellent	Excellent

Merci pour votre attention !