
Massively Parallel Computing with Cuda

- N-Body Problems -

Hendrik Lensch
Robert Strzodka

Today

- **N-Body Problem Introduction**
- **Excerpts from a Tutorial**
 - Fast N-body Algorithms for Massive Datasets
 - by Alexander Gray
 - presented at the 2008 SIAM Conference on Data Mining

Introduction

N-Body Problems

- **Ubiquitous N-Body Problems**

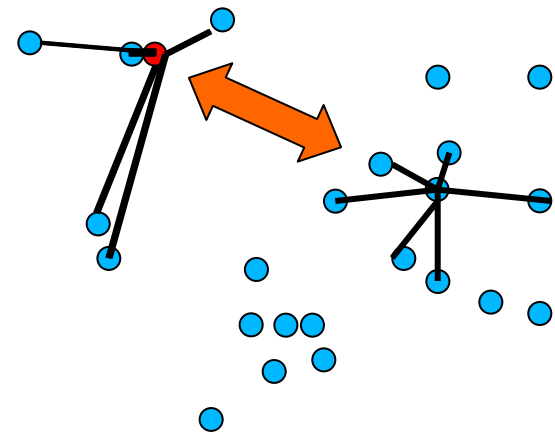
- Astrophysics
- Molecular Dynamics
- Particle discretizations for PDEs
- Data Mining
- Irregular Sampling in Graphics
- etc.

- **Simple Observations**

- Points have no intrinsic topology
- Metric/Kernel relations matter $K(x,y)$
- N^2 interactions for all to all

- **Typical Problems**

- Nearest neighbors
- Weighted interpolation
- Partition of unity
- Kernel density



Fast N-Body Algorithms for Massive Datasets

Alexander Gray

Georgia Institute of Technology

Two canonical problems

- **Nearest-neighbor search**

$$NN(x_q) = \arg \min_r \|x_q - x_r\|$$

- **Kernel density estimation**

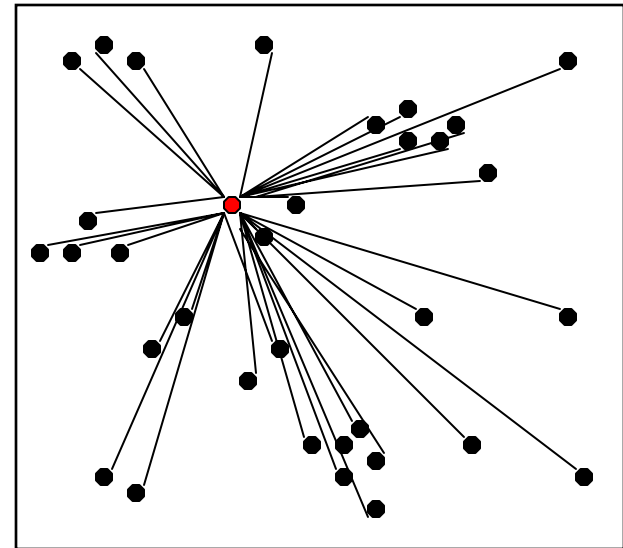
$$\hat{f}(x_q) = \frac{1}{N} \sum_{r \neq q}^N K_h(\|x_q - x_r\|)$$

Ideas

1. Data structures and how to use them
2. Monte Carlo
3. Series expansions
4. Problem/solution abstractions

Nearest Neighbor - Naïve Approach

- Given a query point X .
- Scan through each point Y :
 - Calculate the distance $d(X, Y)$
 - If $d(X, Y) < \text{best_seen}$ then Y is the new nearest neighbor.
- Takes $O(N)$ time for each query!

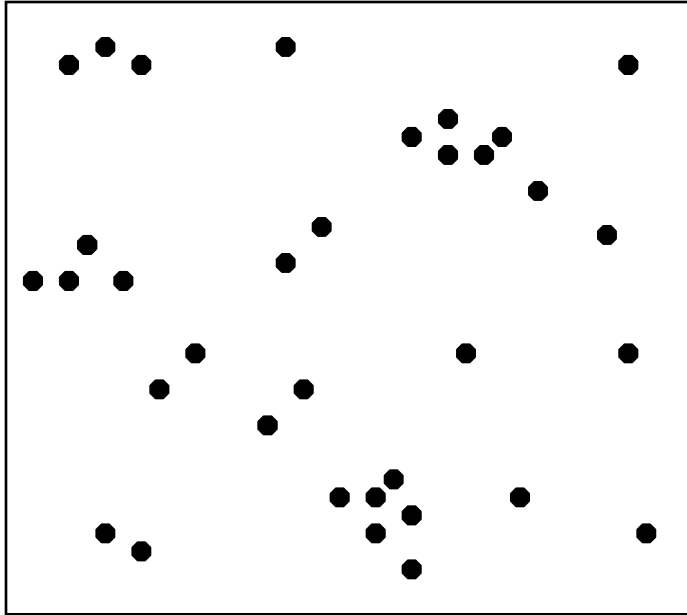


33 Distance Computations

Speeding Up Nearest Neighbor

- We can speed up the search for the nearest neighbor:
 - Examine nearby points first.
 - Ignore any points that are further than the nearest point found so far.
- Do this using a KD-tree:
 - Tree based data structure
 - Recursively partitions points into axis aligned boxes.

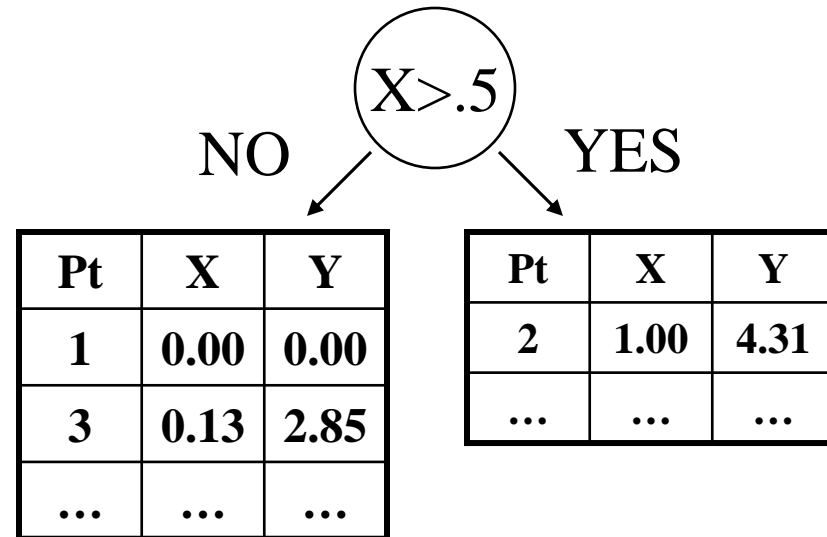
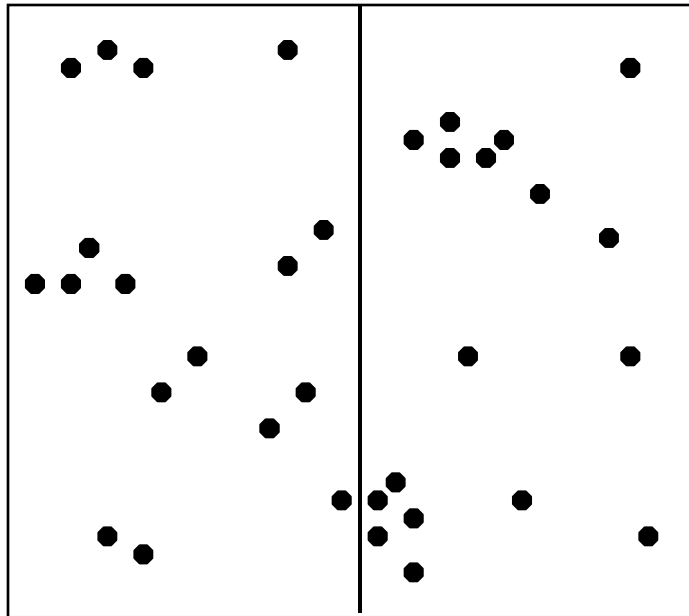
KD-Tree Construction



| Pt | X | Y |
|-----|------|------|
| 1 | 0.00 | 0.00 |
| 2 | 1.00 | 4.31 |
| 3 | 0.13 | 2.85 |
| ... | ... | ... |

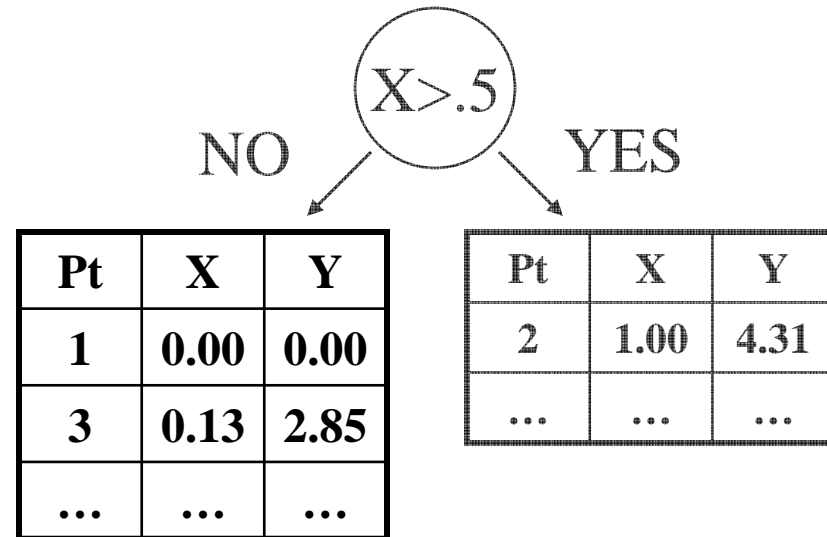
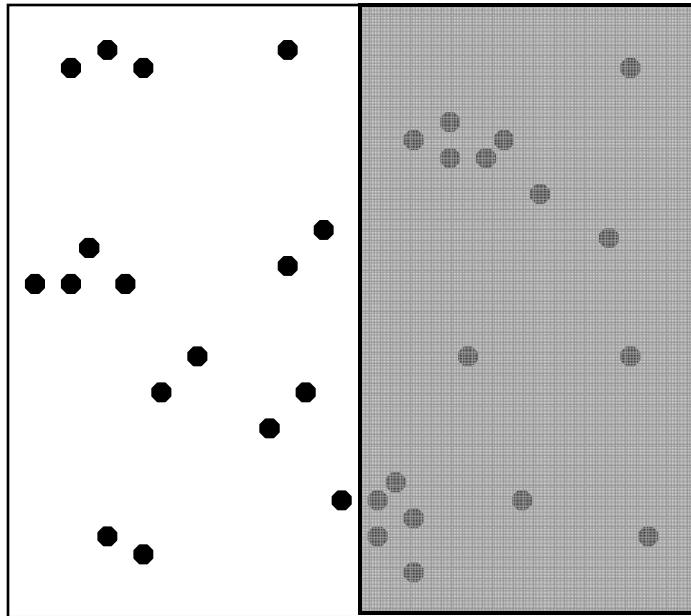
We start with a list of n-dimensional points.

KD-Tree Construction



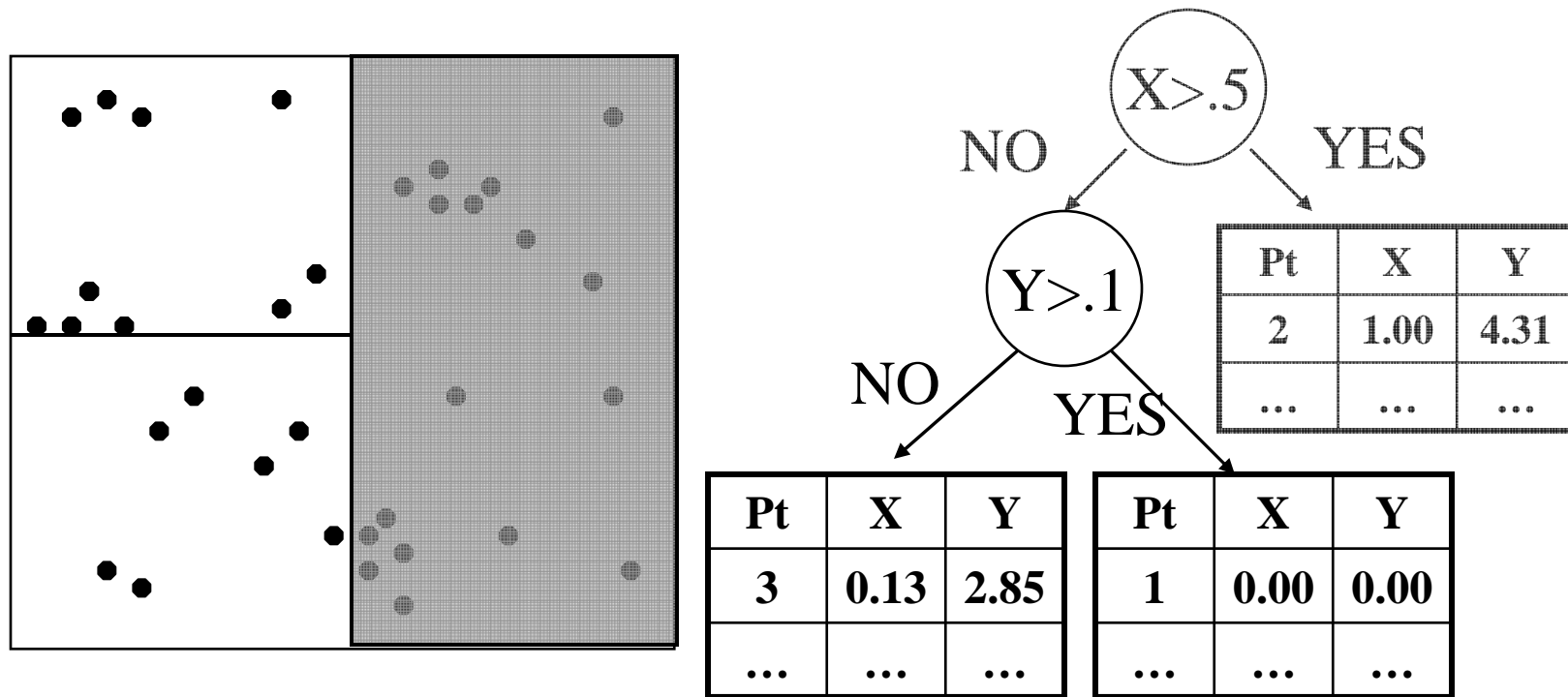
We can split the points into 2 groups by choosing a dimension X and value V and separating the points into $X > V$ and $X \leq V$.

KD-Tree Construction



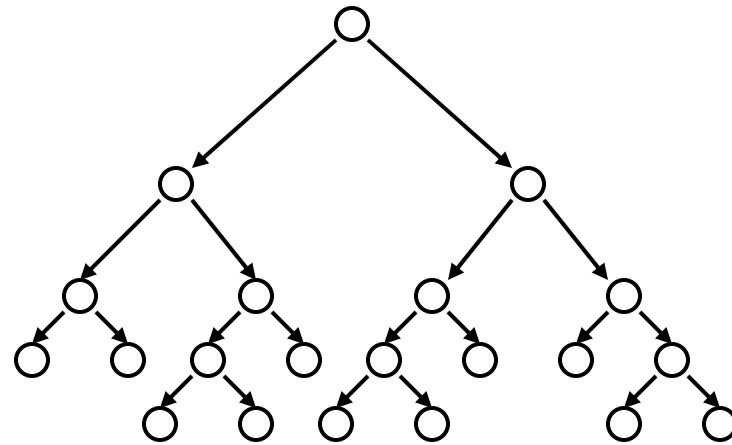
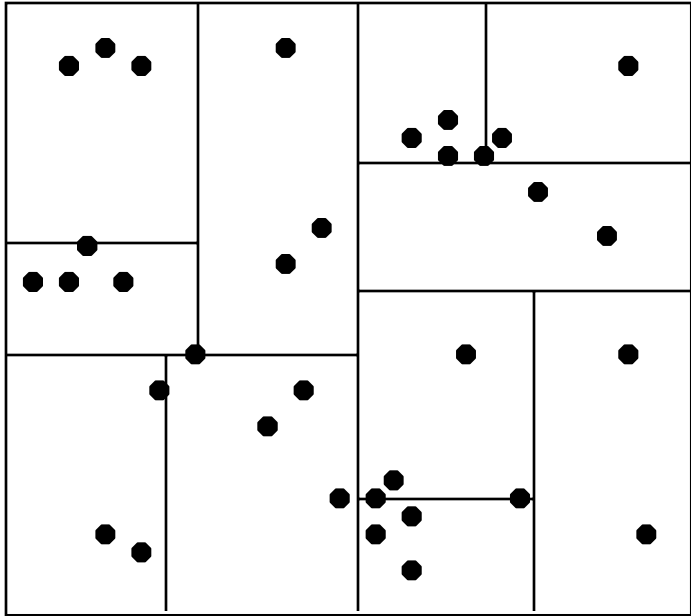
We can then consider each group separately and possibly split again (along same/different dimension).

KD-Tree Construction



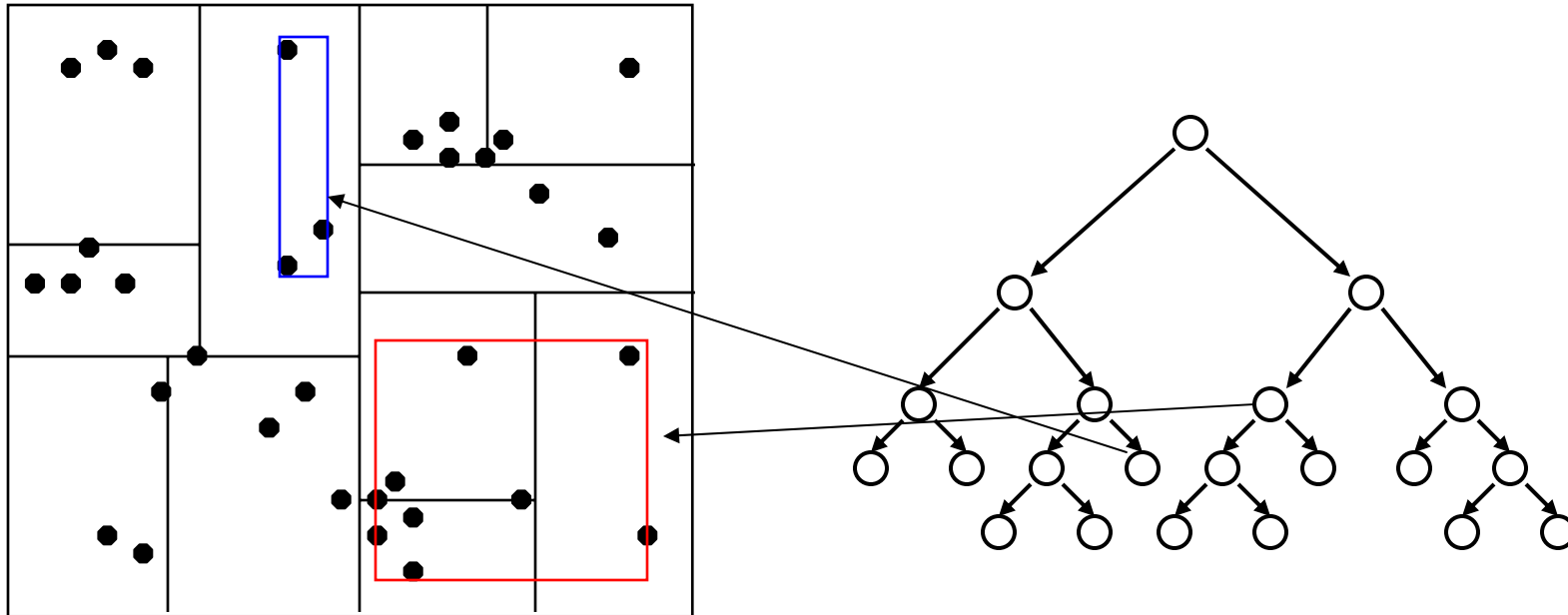
We can then consider each group separately and possibly split again (along same/different dimension).

KD-Tree Construction



We can keep splitting the points in each set to create a tree structure. Each node with no children (leaf node) contains a list of points.

KD-Tree Construction



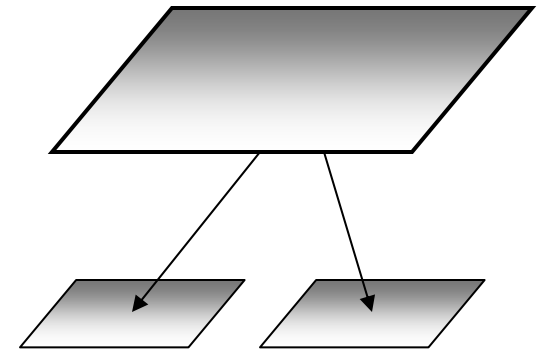
We will keep around one additional piece of information at each node. The (tight) bounds of the points at or below this node.

KD-Tree Construction

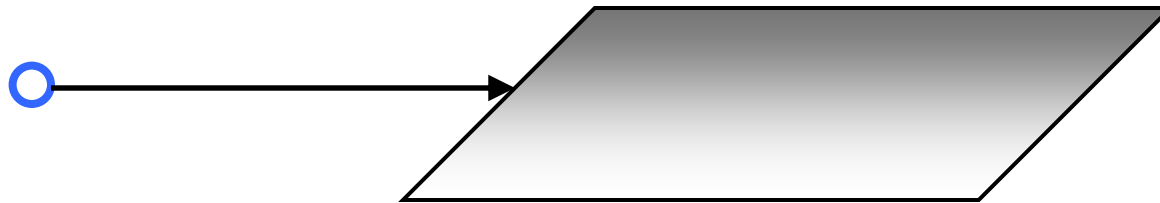
Use heuristics to make splitting decisions:

- Which dimension do we split along? **Widest**
- Which value do we split at? **Median of value of that split dimension for the points.**
- When do we stop? **When there are fewer than m points left OR the box has hit some minimum width.**

Exclusion and inclusion, using point-node *kd*-tree bounds.



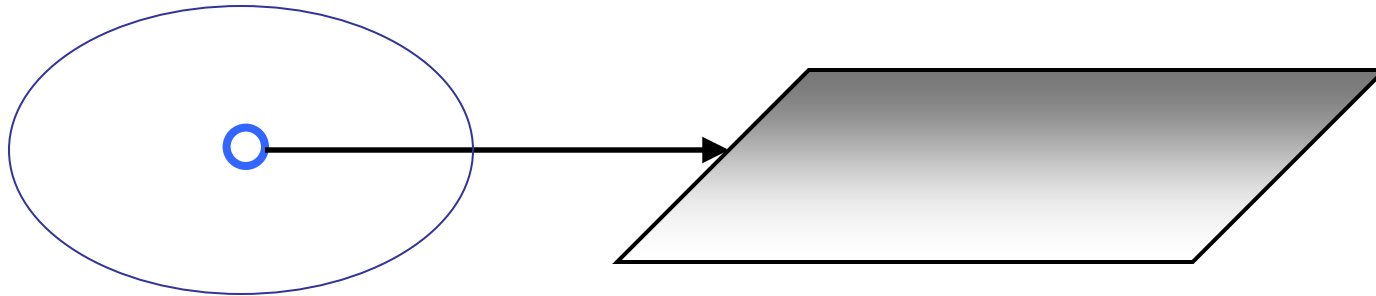
O(D) bounds on distance minima/maxima:



$$\min_i \|x - x_i\| \geq \sum^D \left[\max \{ (l_d - x_d)^2, 0 \} + \max \{ (x_d - u_d)^2, 0 \} \right]$$
$$\max_i \|x - x_i\| \leq \sum^D \max \{ (u_d - x_d)^2, (x_d - l_d)^2 \}$$

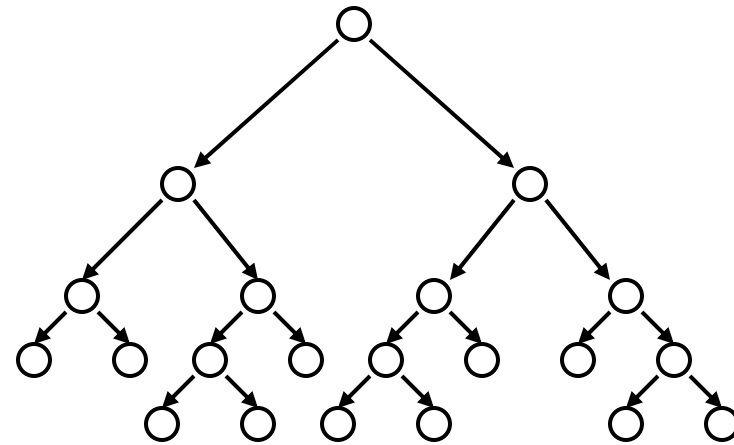
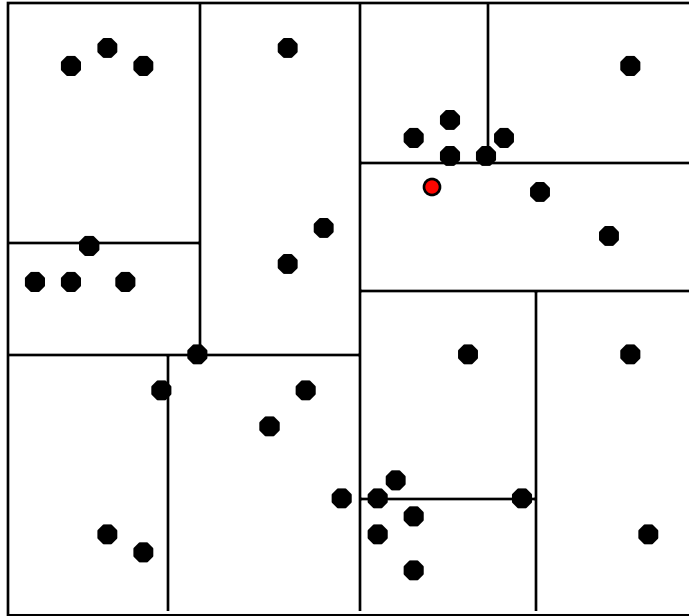
Exclusion and inclusion, using point-node *kd*-tree bounds.

O(D) bounds on distance minima/maxima:



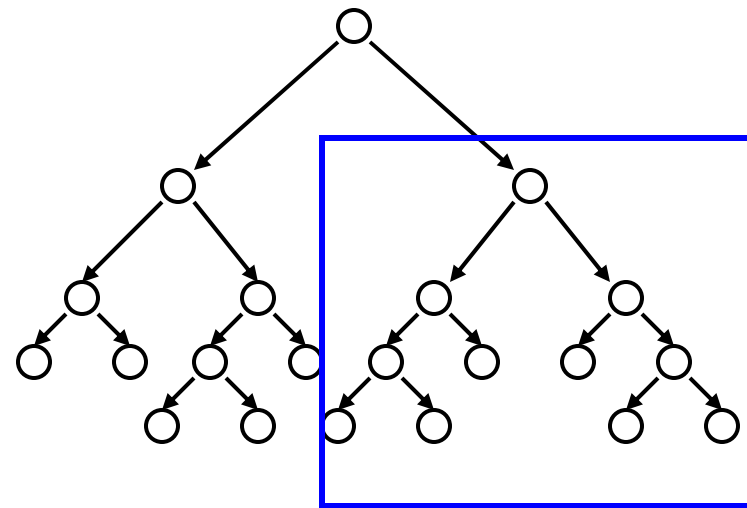
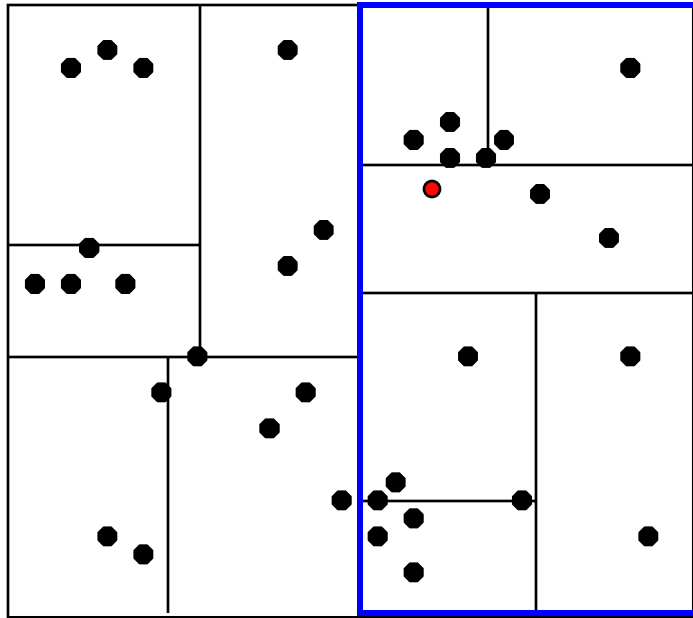
$$\min_i \|x - x_i\| \geq \sum_d^D \left[\max\{(l_d - x_d)^2, 0\} + \max\{(x_d - u_d)^2, 0\} \right]$$
$$\max_i \|x - x_i\| \leq \sum_d^D \max\{(u_d - x_d)^2, (x_d - l_d)^2\}$$

Nearest Neighbor with KD Trees



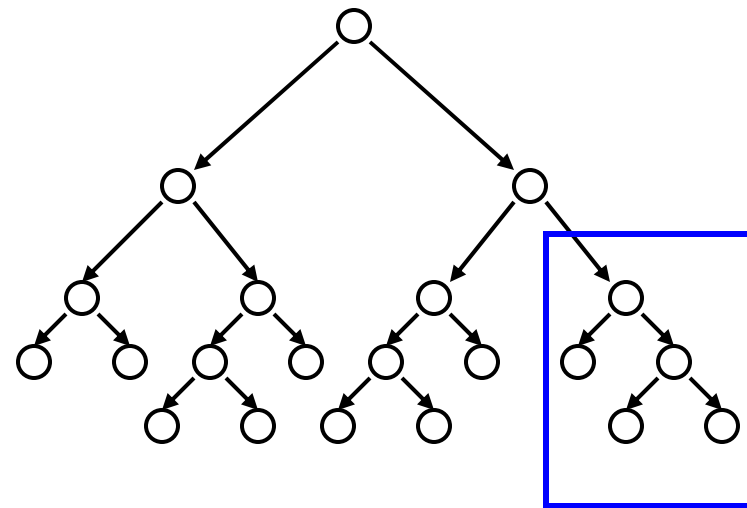
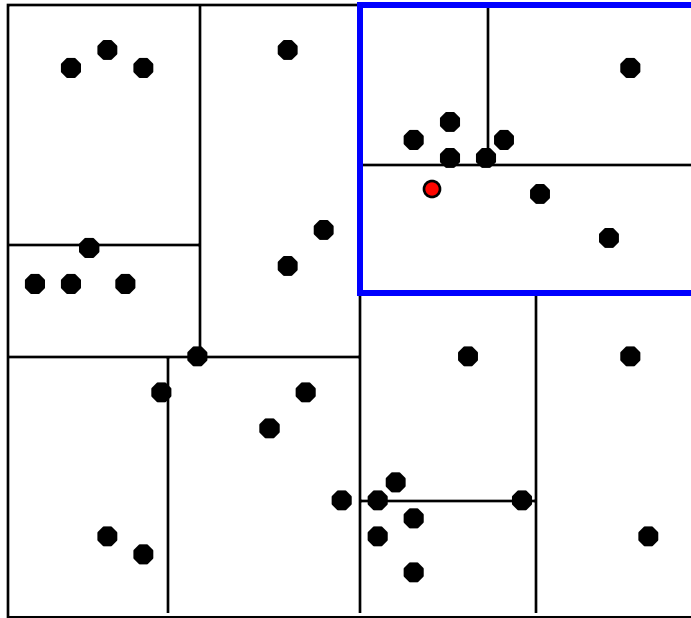
We traverse the tree looking for the nearest neighbor of the query point.

Nearest Neighbor with KD Trees



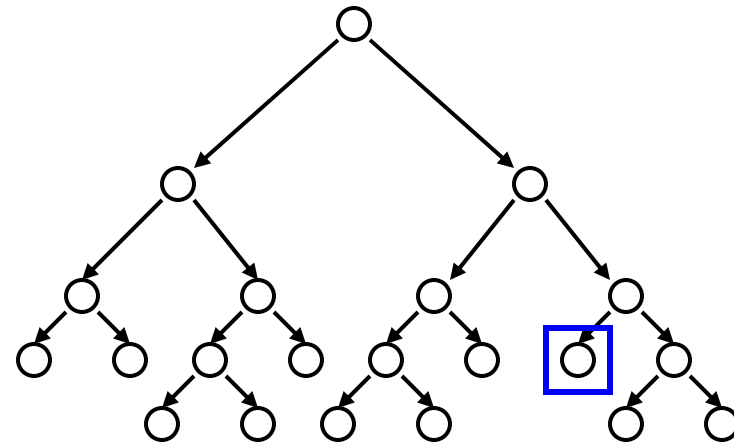
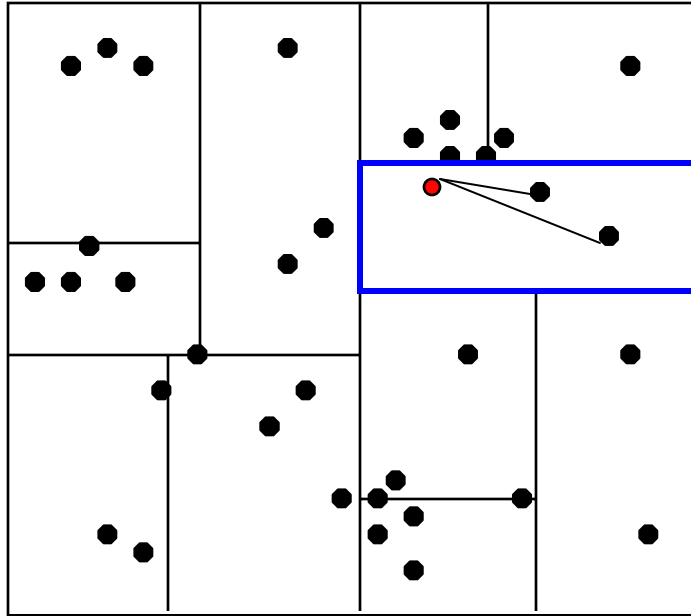
Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Nearest Neighbor with KD Trees



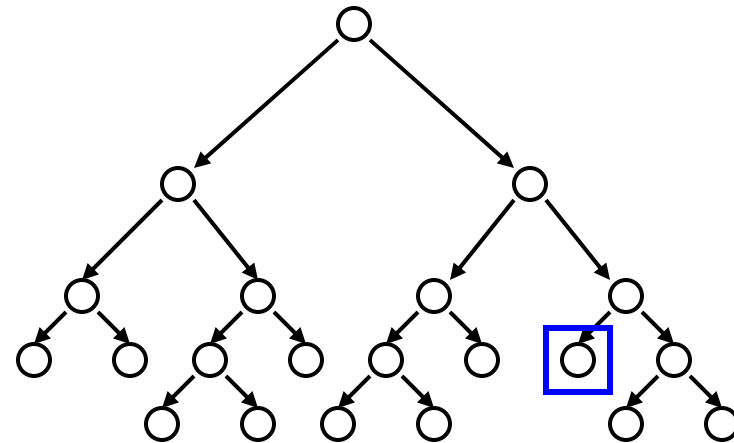
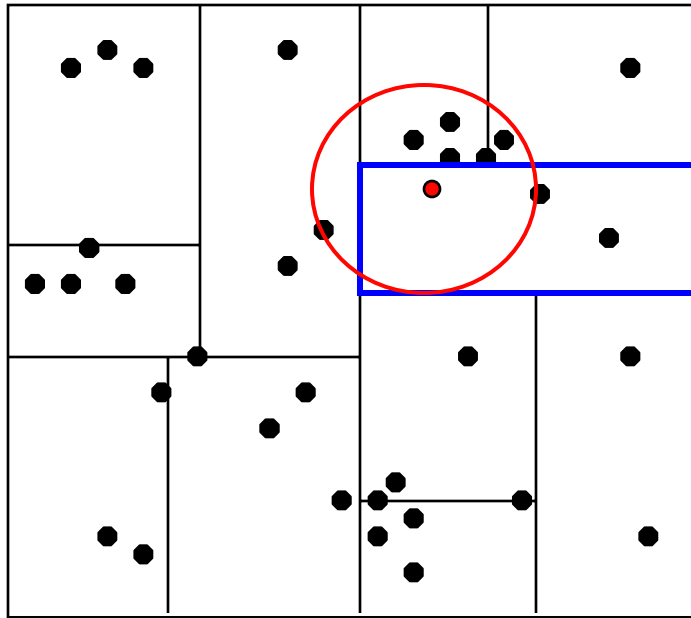
Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

Nearest Neighbor with KD Trees



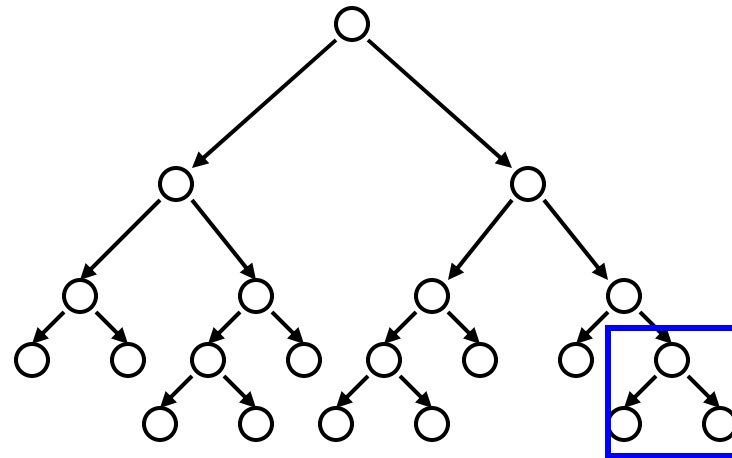
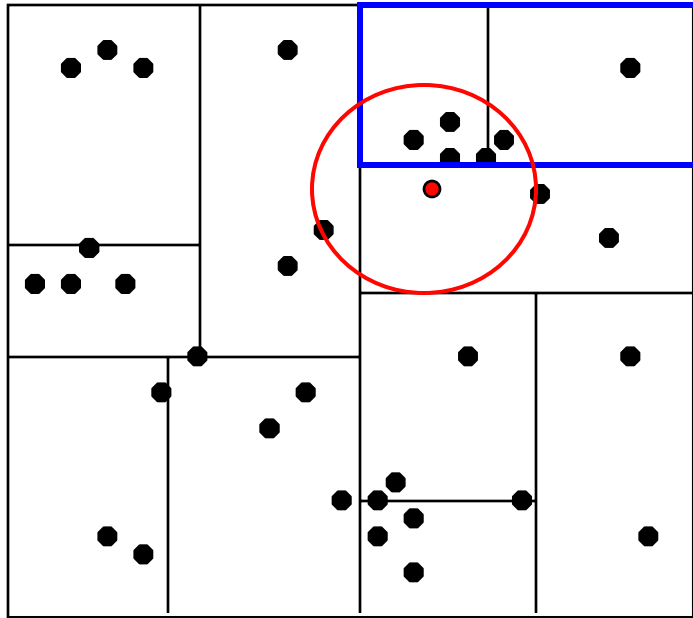
When we reach a leaf node: compute the distance to each point in the node.

Nearest Neighbor with KD Trees



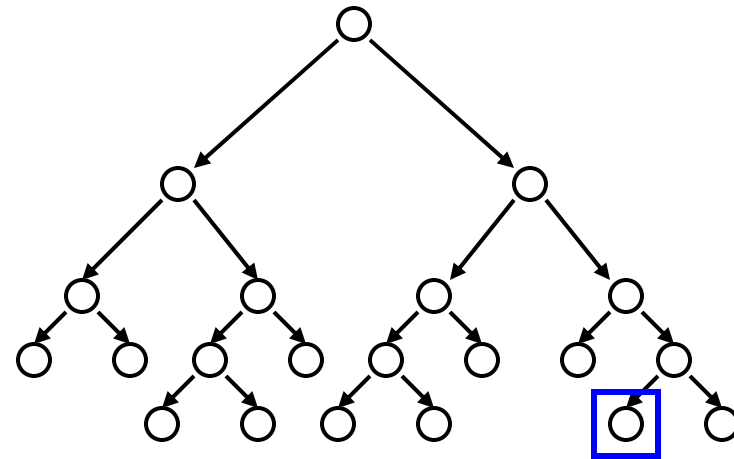
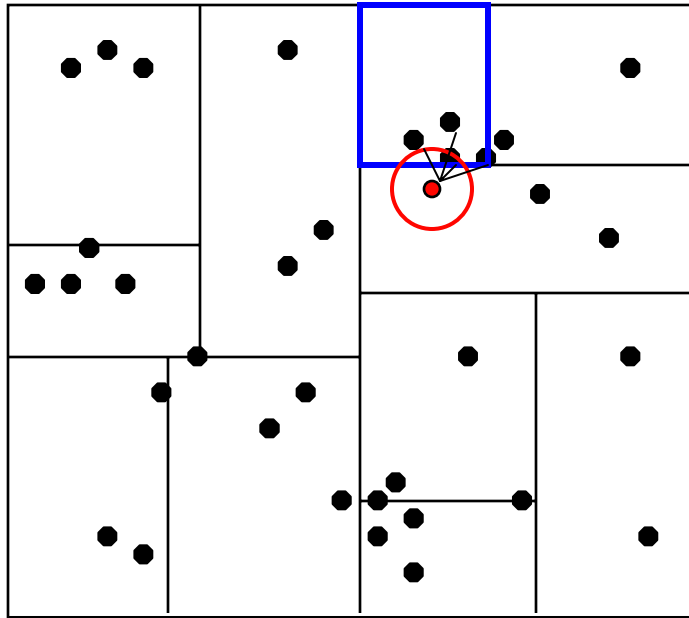
When we reach a leaf node: compute the distance to each point in the node.

Nearest Neighbor with KD Trees



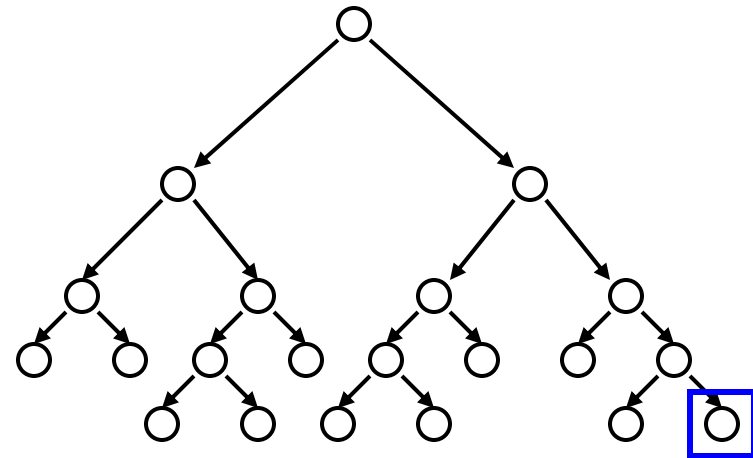
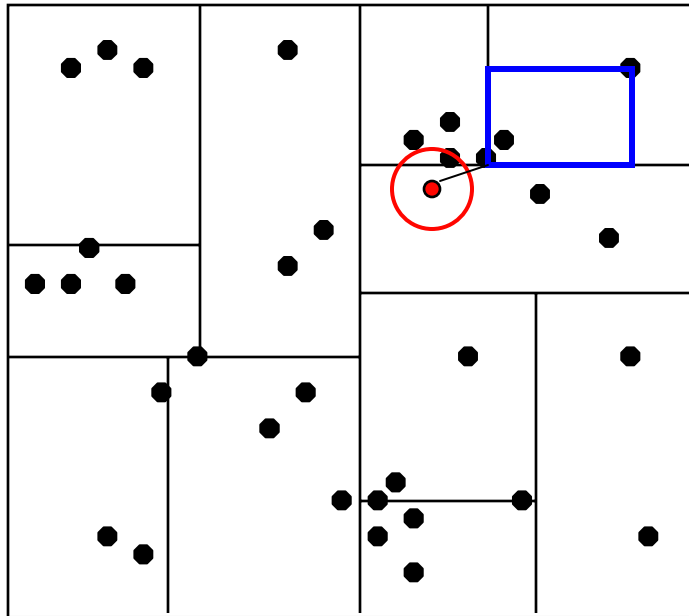
Then we can backtrack and try the other branch at each node visited.

Nearest Neighbor with KD Trees



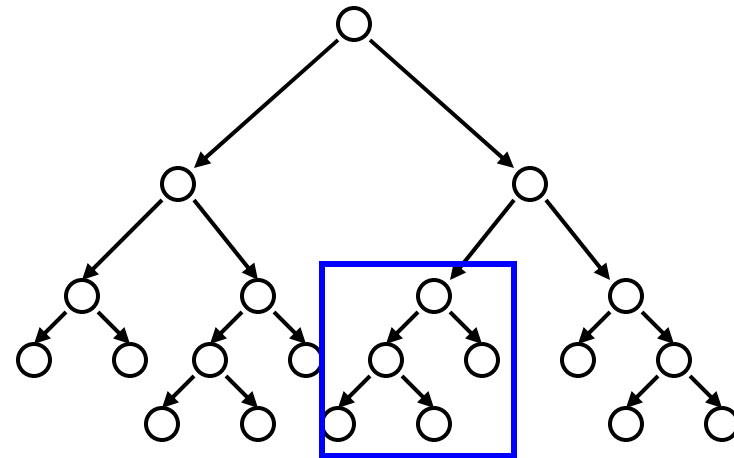
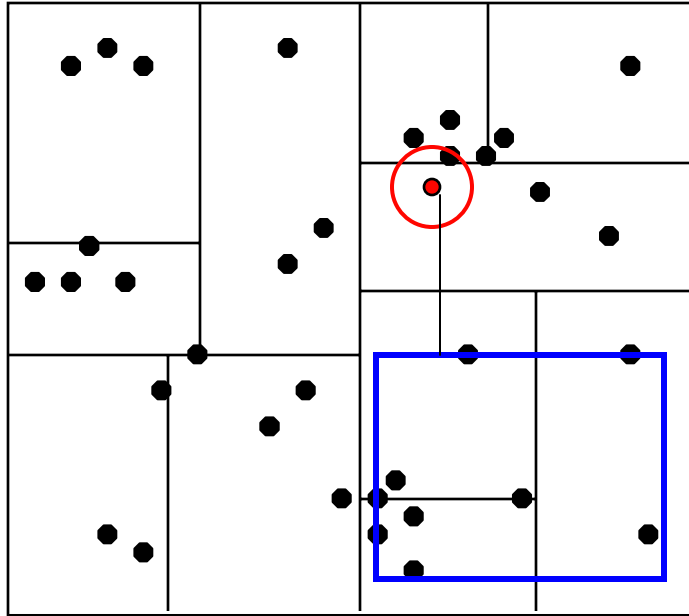
Each time a new closest node is found, we can update the distance bounds.

Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Nearest Neighbor with KD Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Simple recursive algorithm (k=1 case)

```
NN( $x_q, R, d_{lo}, x_{sofar}, d_{sofar}$ )  
{  
  if  $d_{lo} > d_{sofar}$ , return.  
  
  if leaf( $R$ ), [ $x_{sofar}, d_{sofar}$ ]=NNBase( $x_q, R, d_{sofar}$ ).  
  else,  
    [ $R1, d1, R2, d2$ ]=orderByDist( $x_q, R.l, R.r$ ).  
    NN( $x_q, R1, d1, x_{sofar}, d_{sofar}$ ).  
    NN( $x_q, R2, d2, x_{sofar}, d_{sofar}$ ).  
}
```

Nearest Neighbor with KD Trees

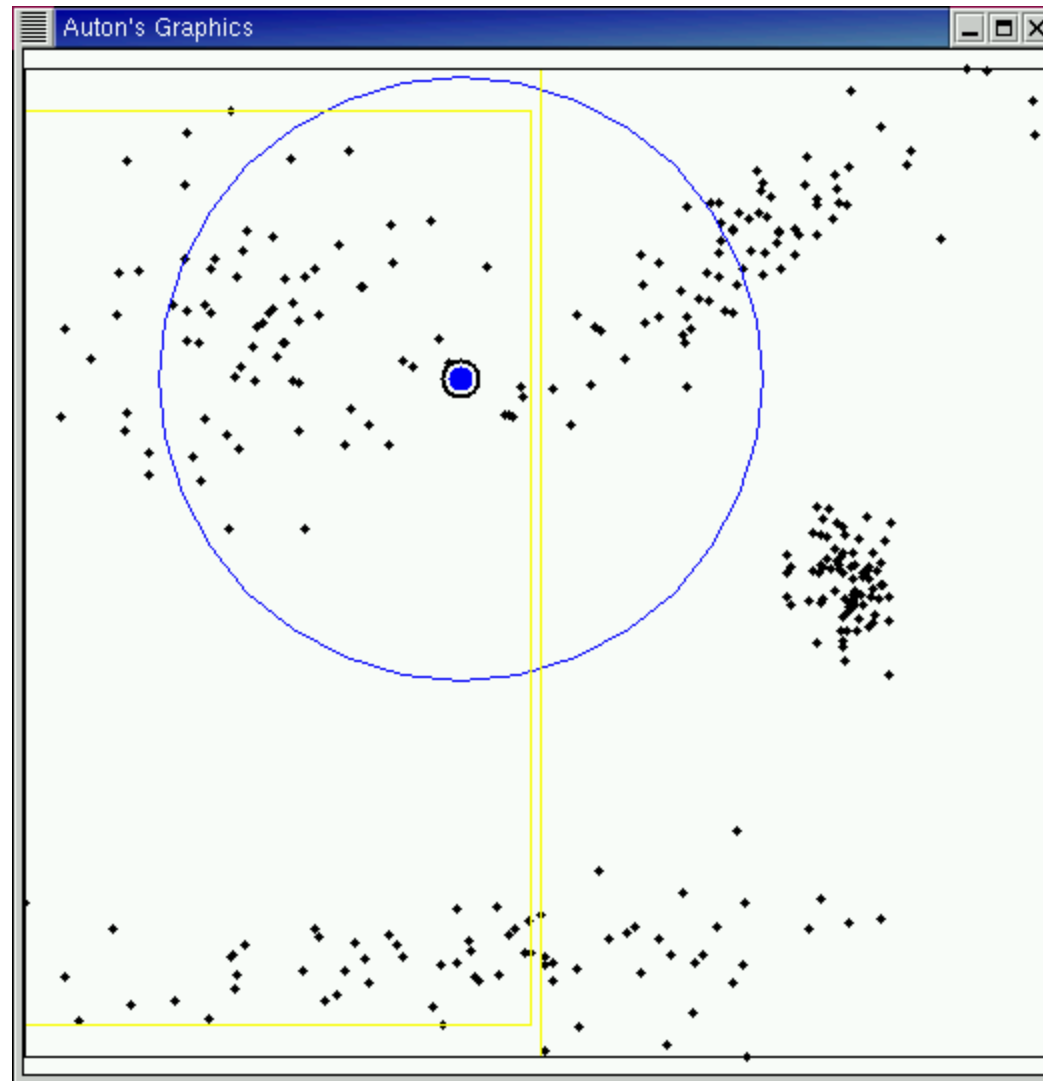
Instead, some animations showing real data...

1. kd-tree with cached sufficient statistics
2. nearest-neighbor with kd-trees
3. range-count with kd-trees

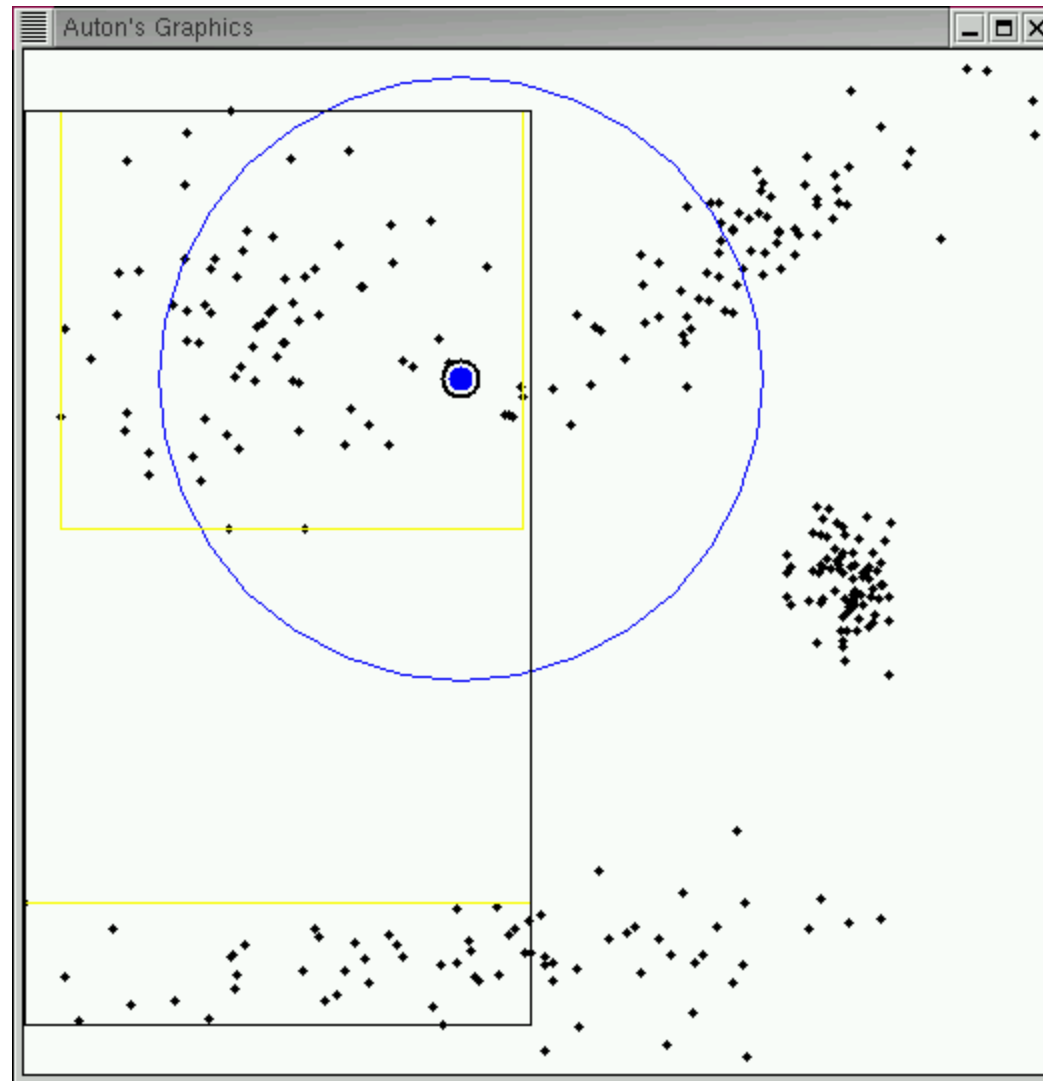
For animations, see:

<http://www.cs.cmu.edu/~awm/animations/kdtree>

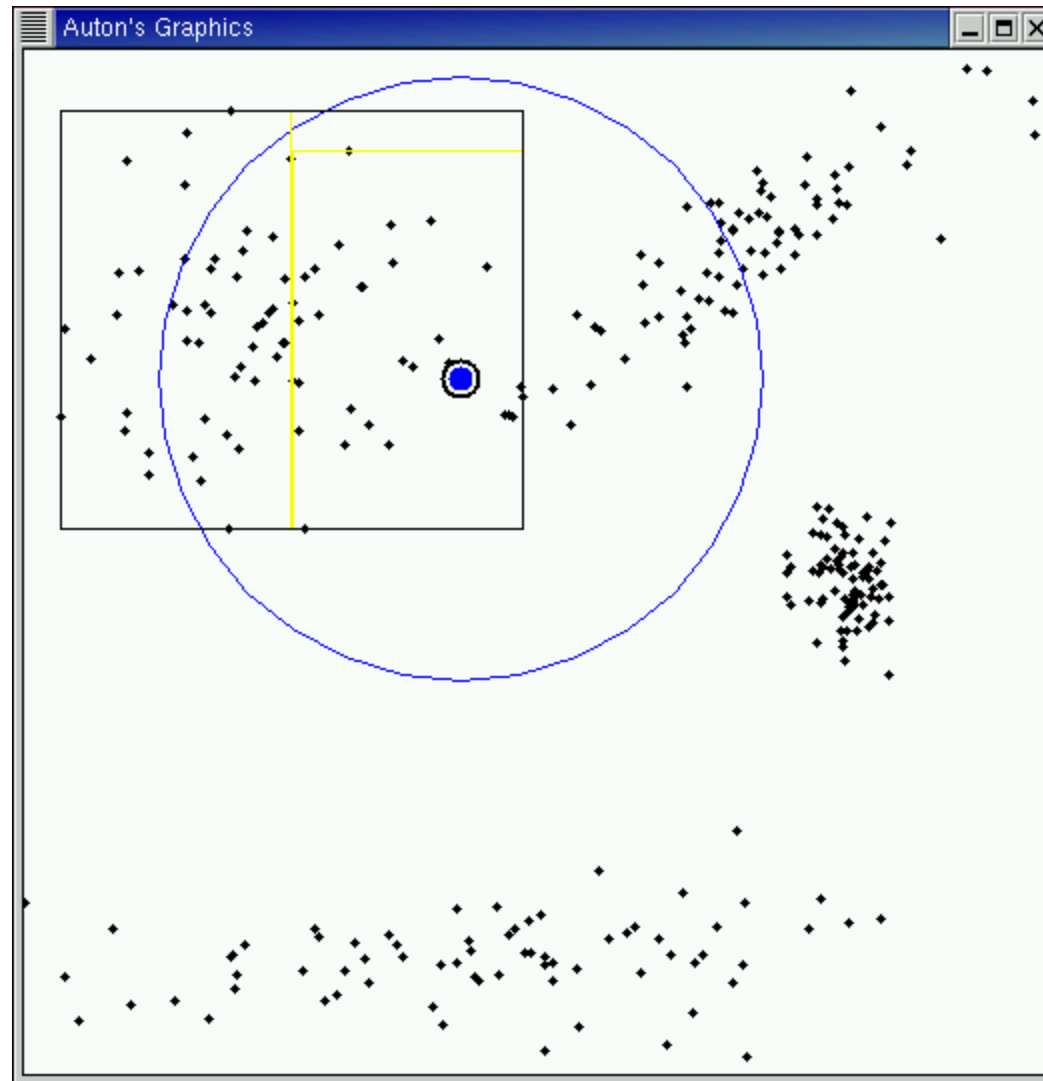
Range-count example



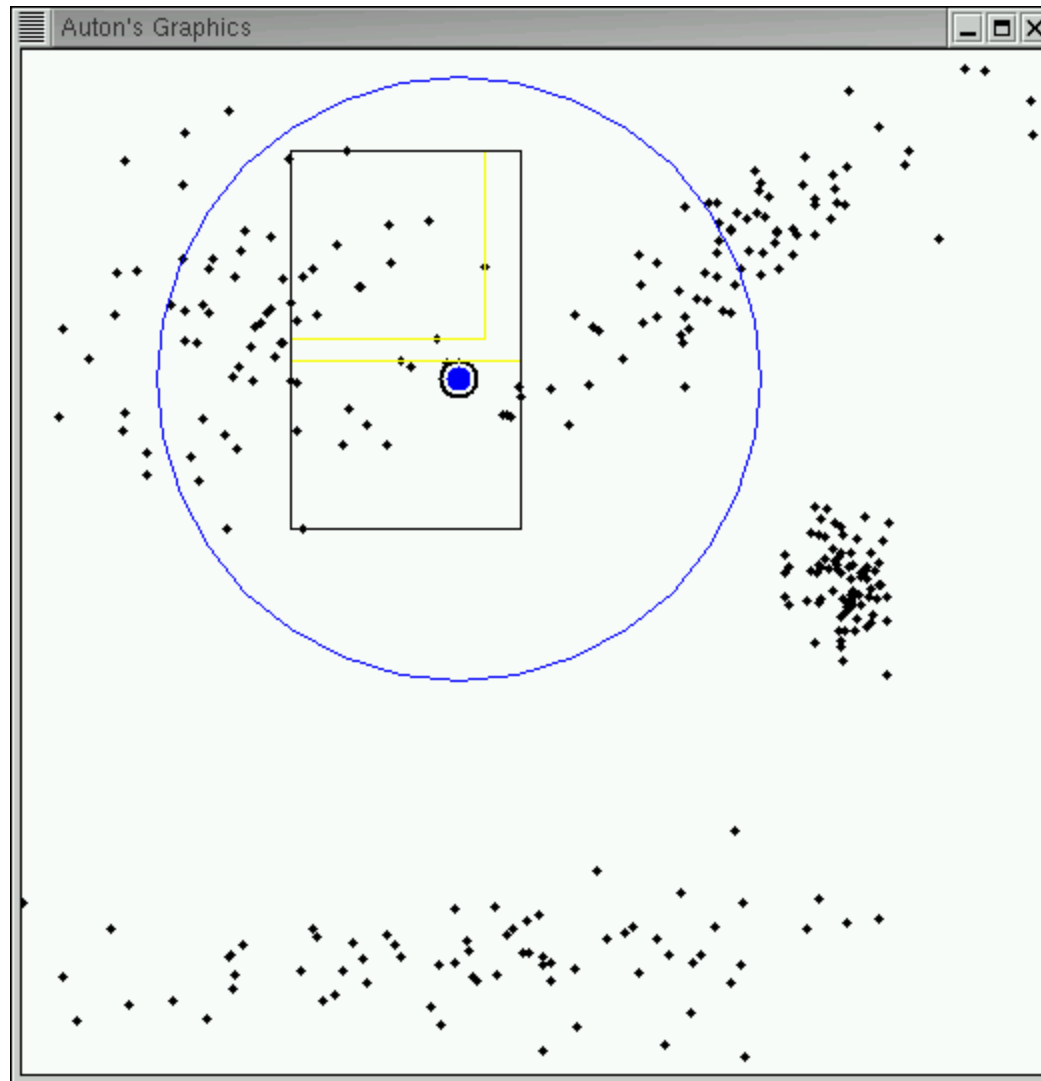
Range-count example



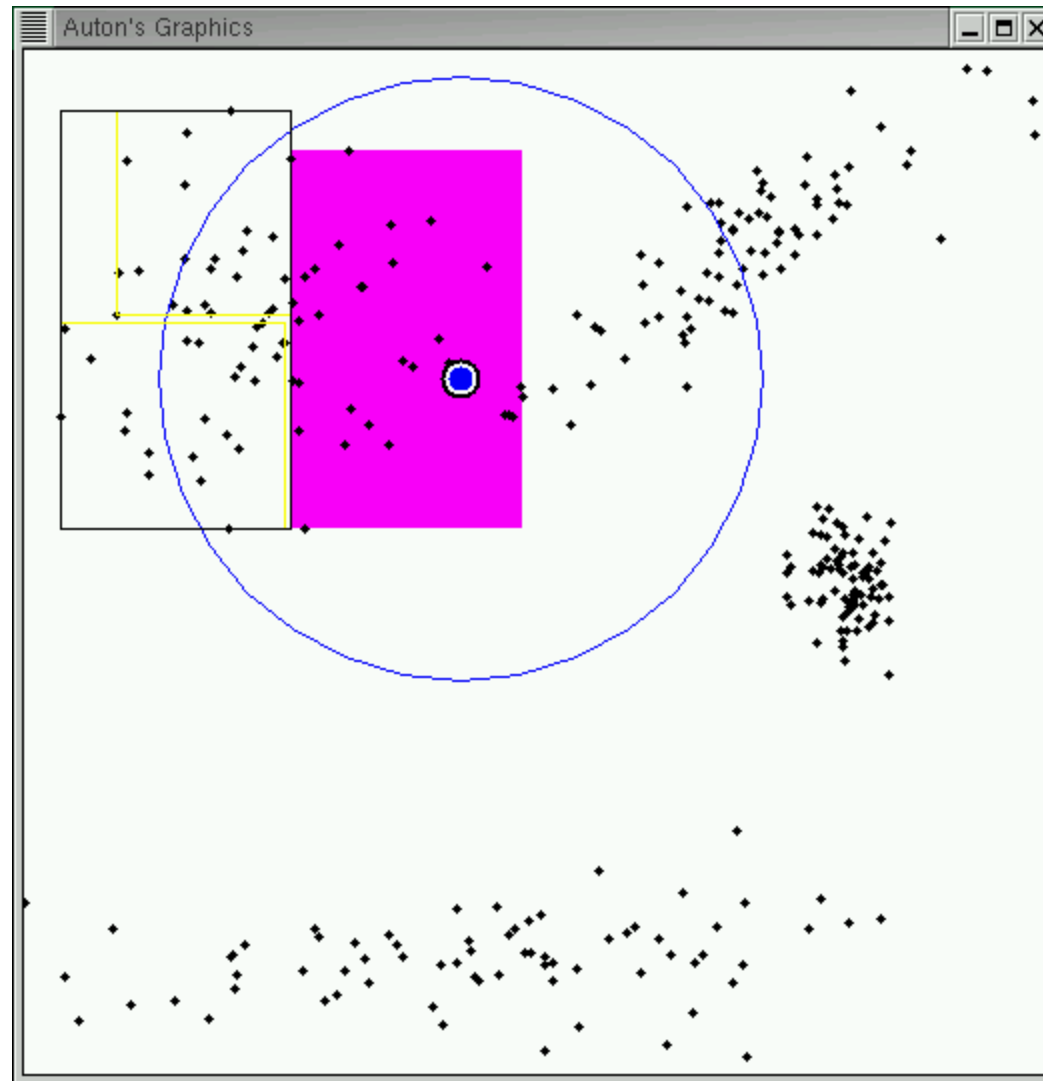
Range-count example



Range-count example

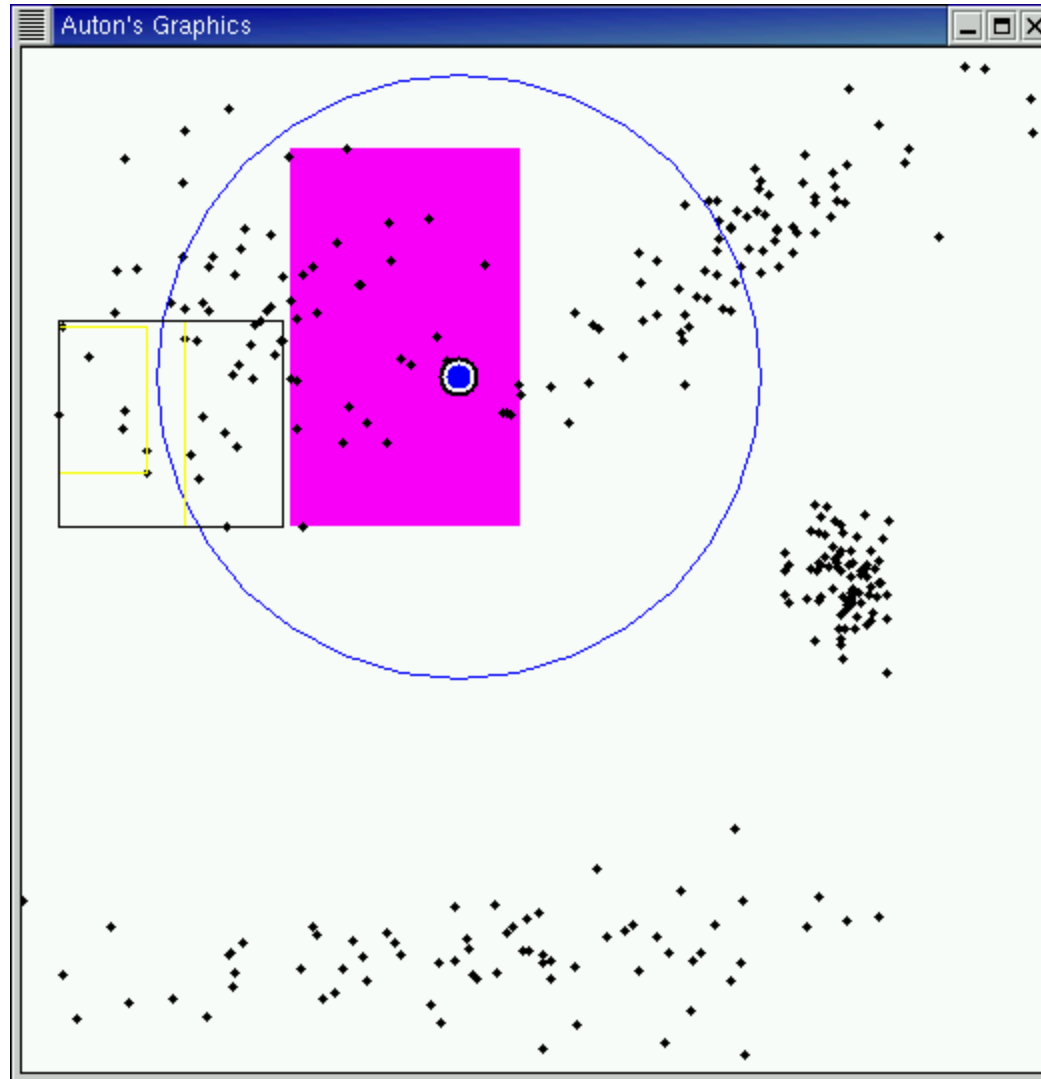


Range-count example

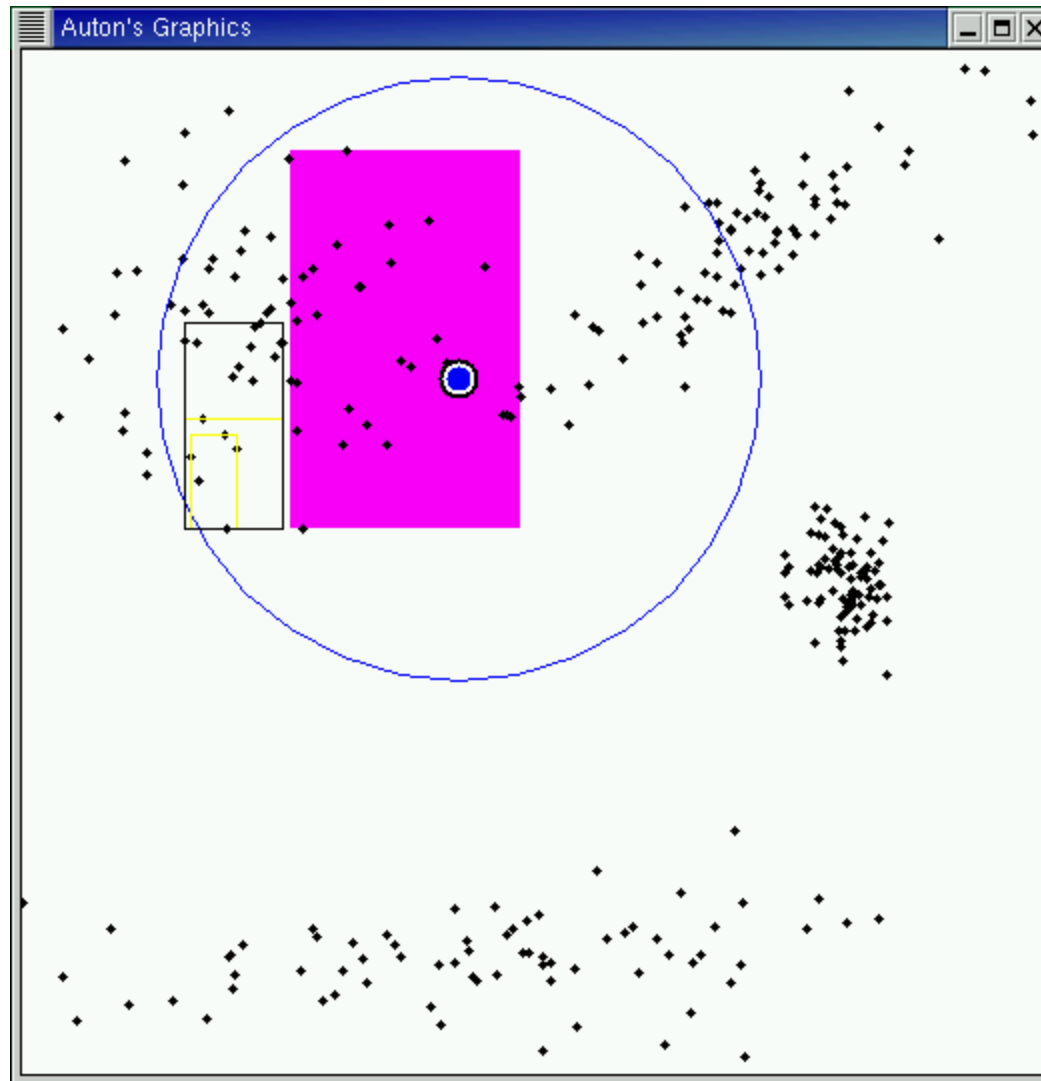


Pruned!
(inclusion)

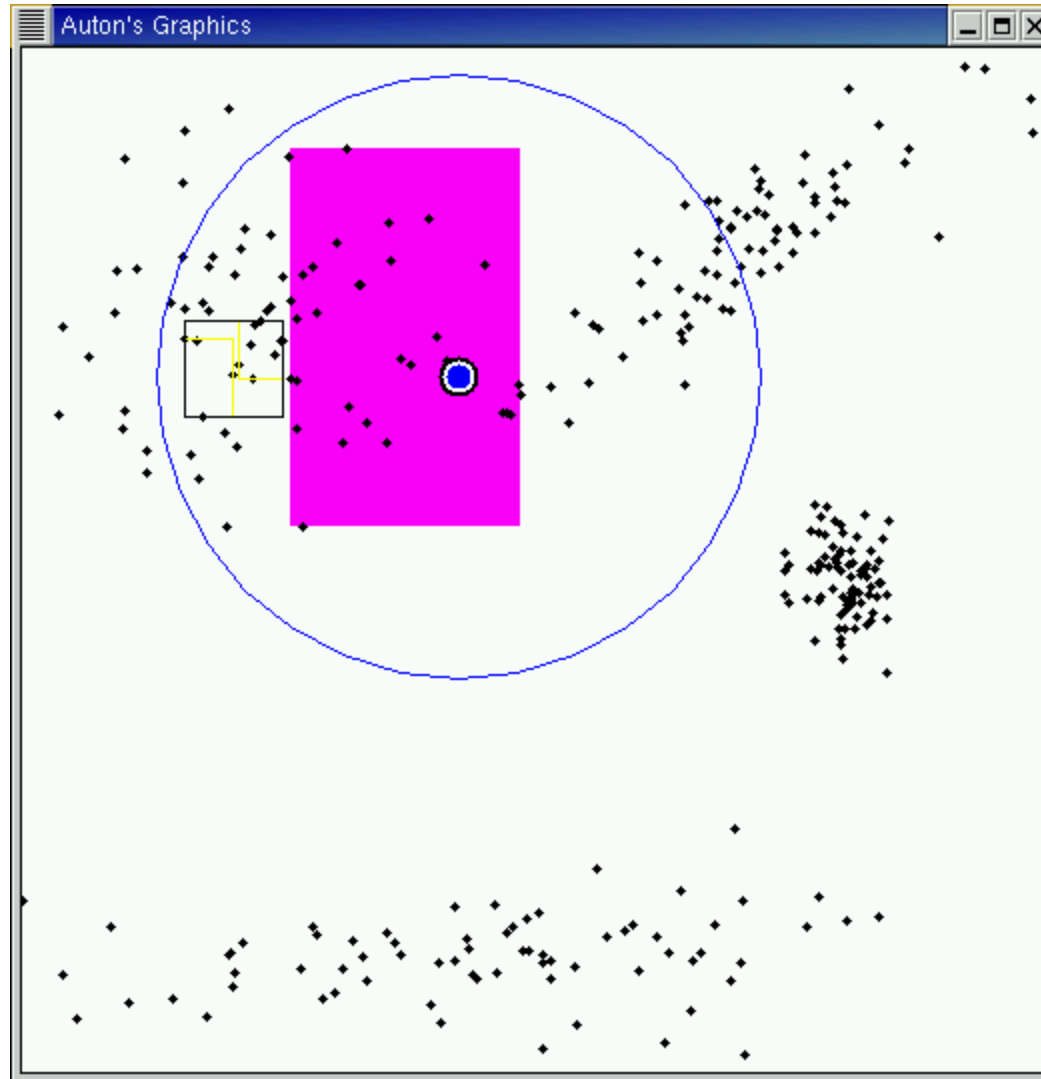
Range-count example



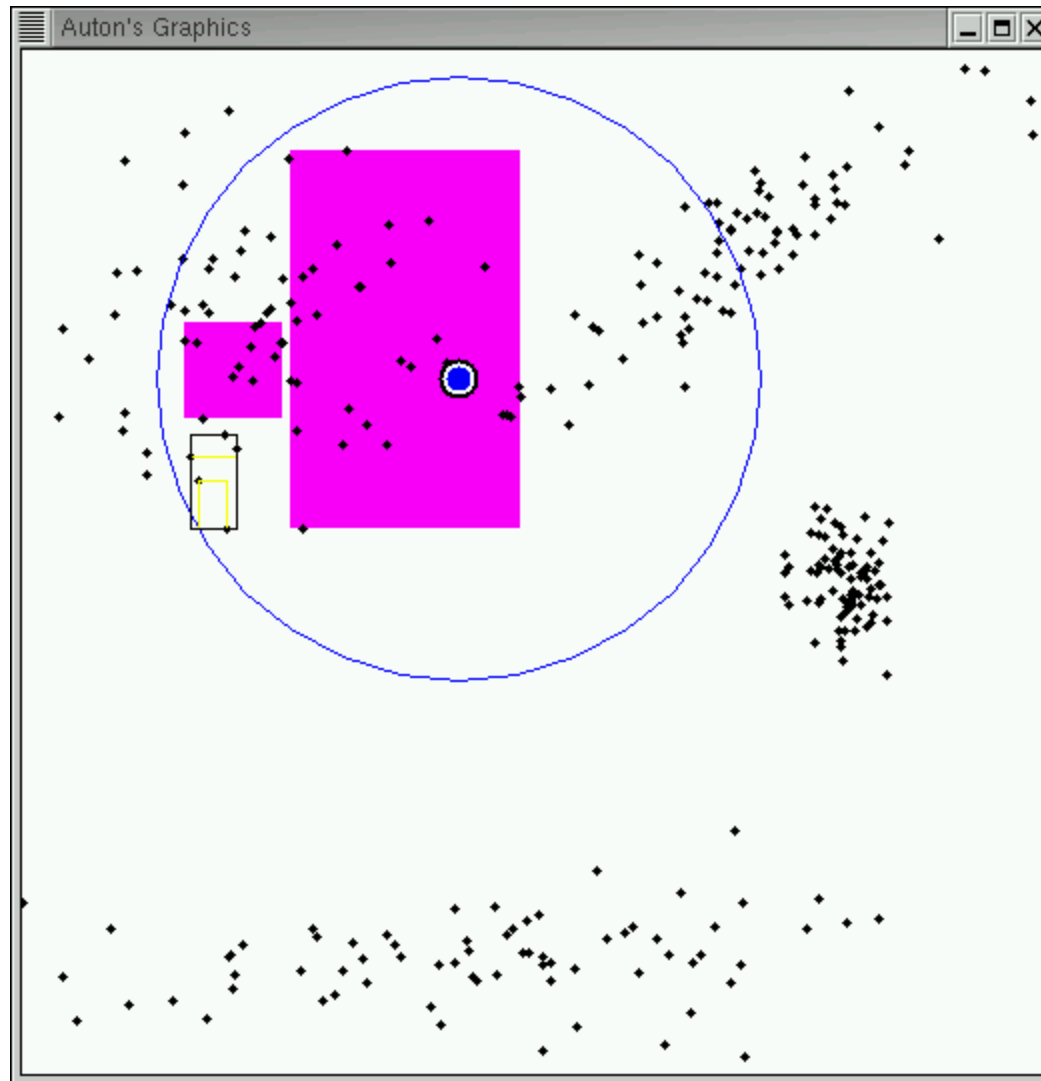
Range-count example



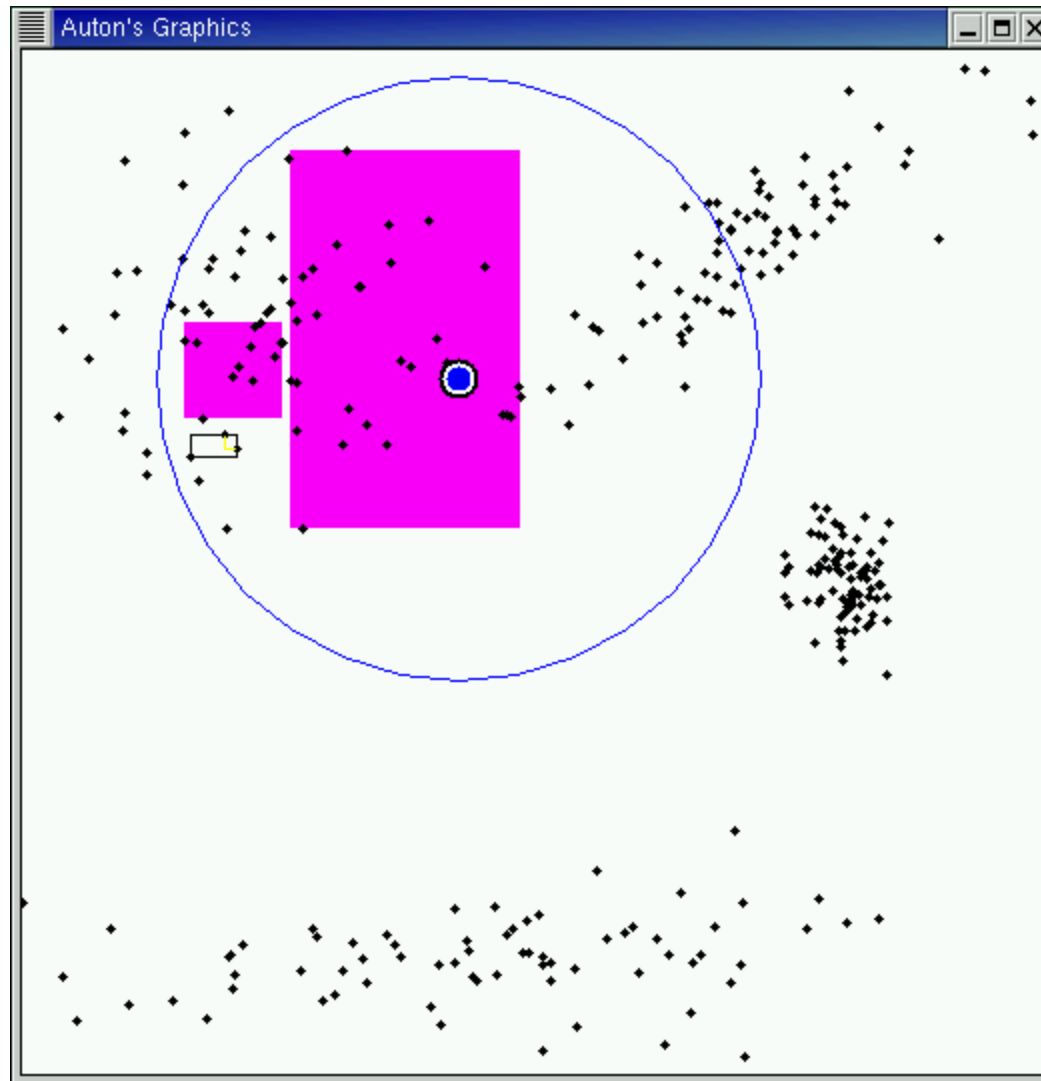
Range-count example



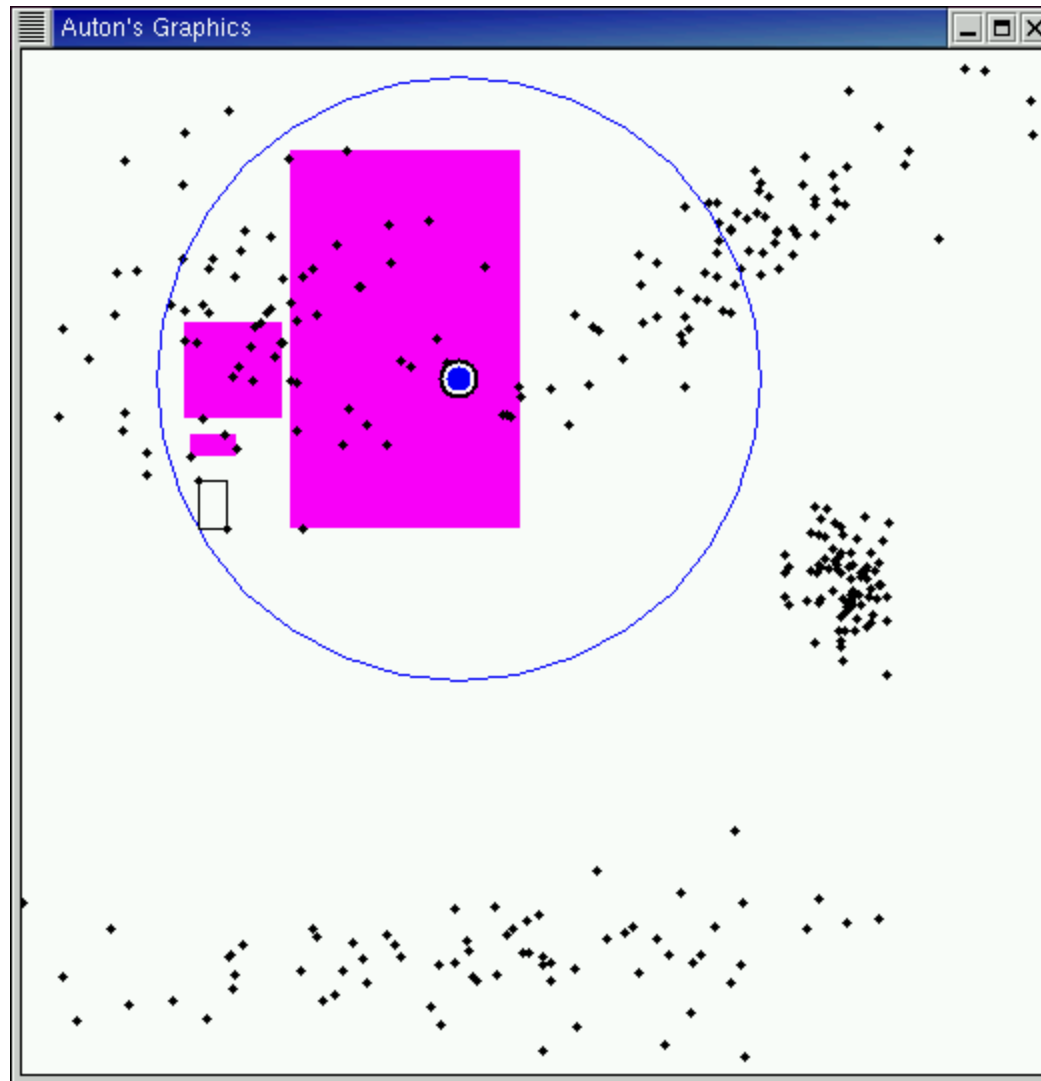
Range-count example



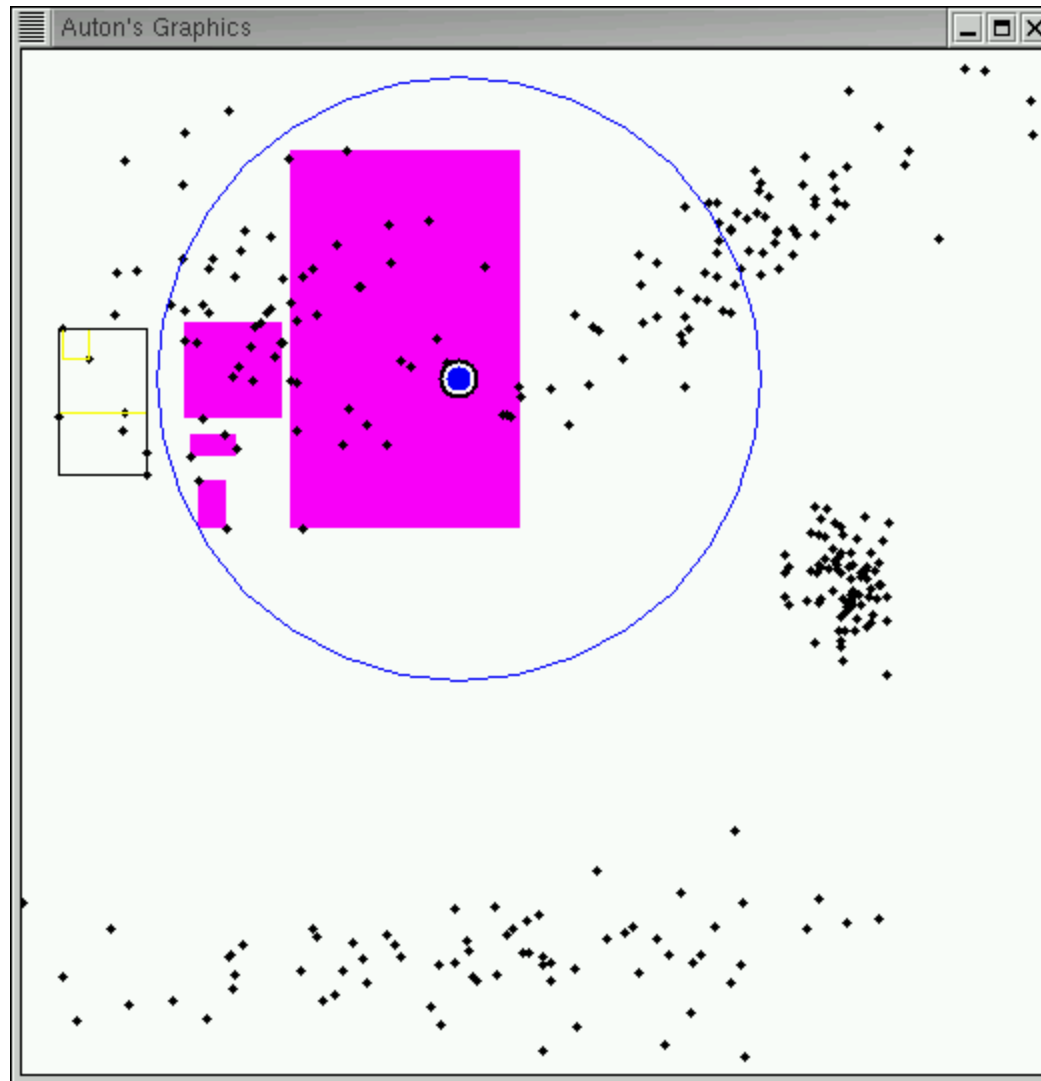
Range-count example



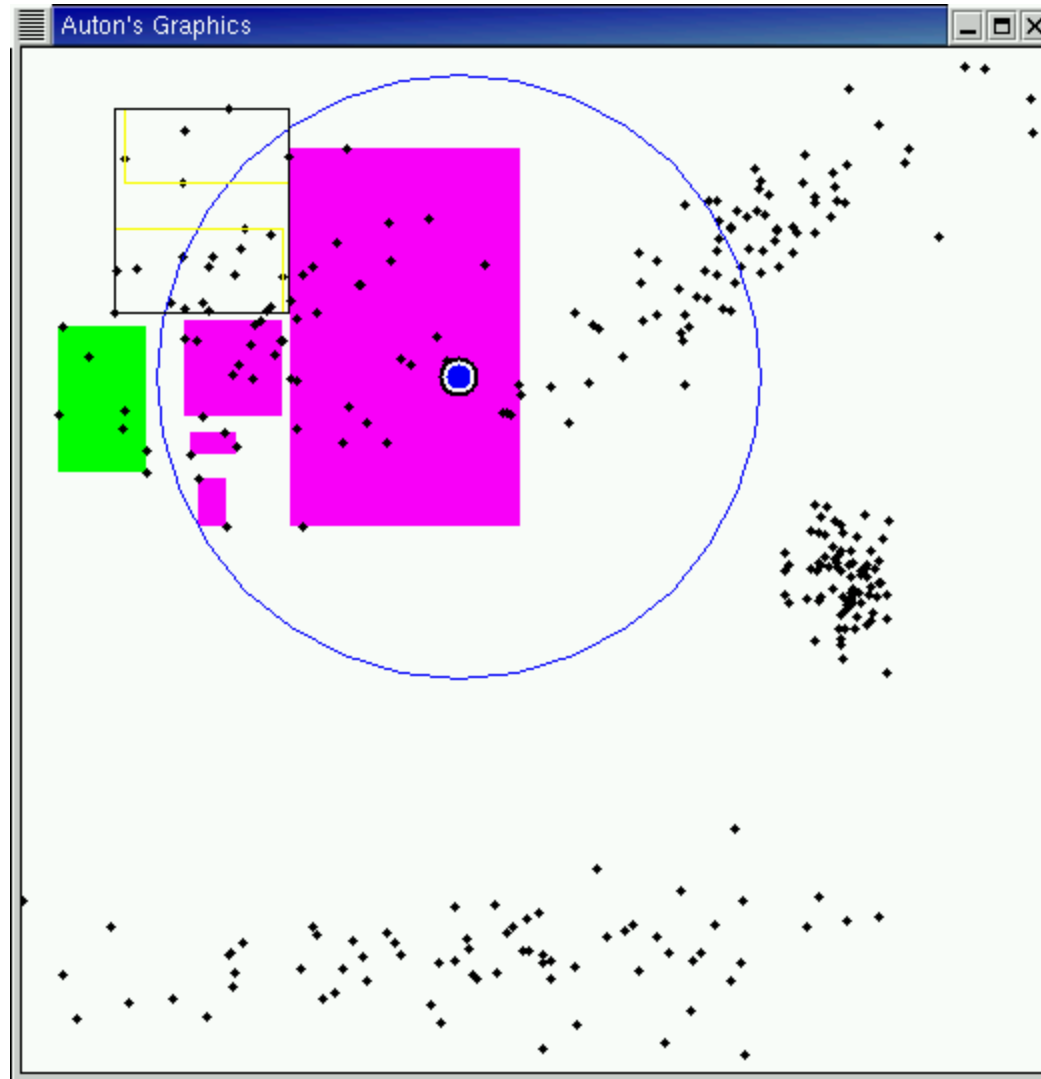
Range-count example



Range-count example

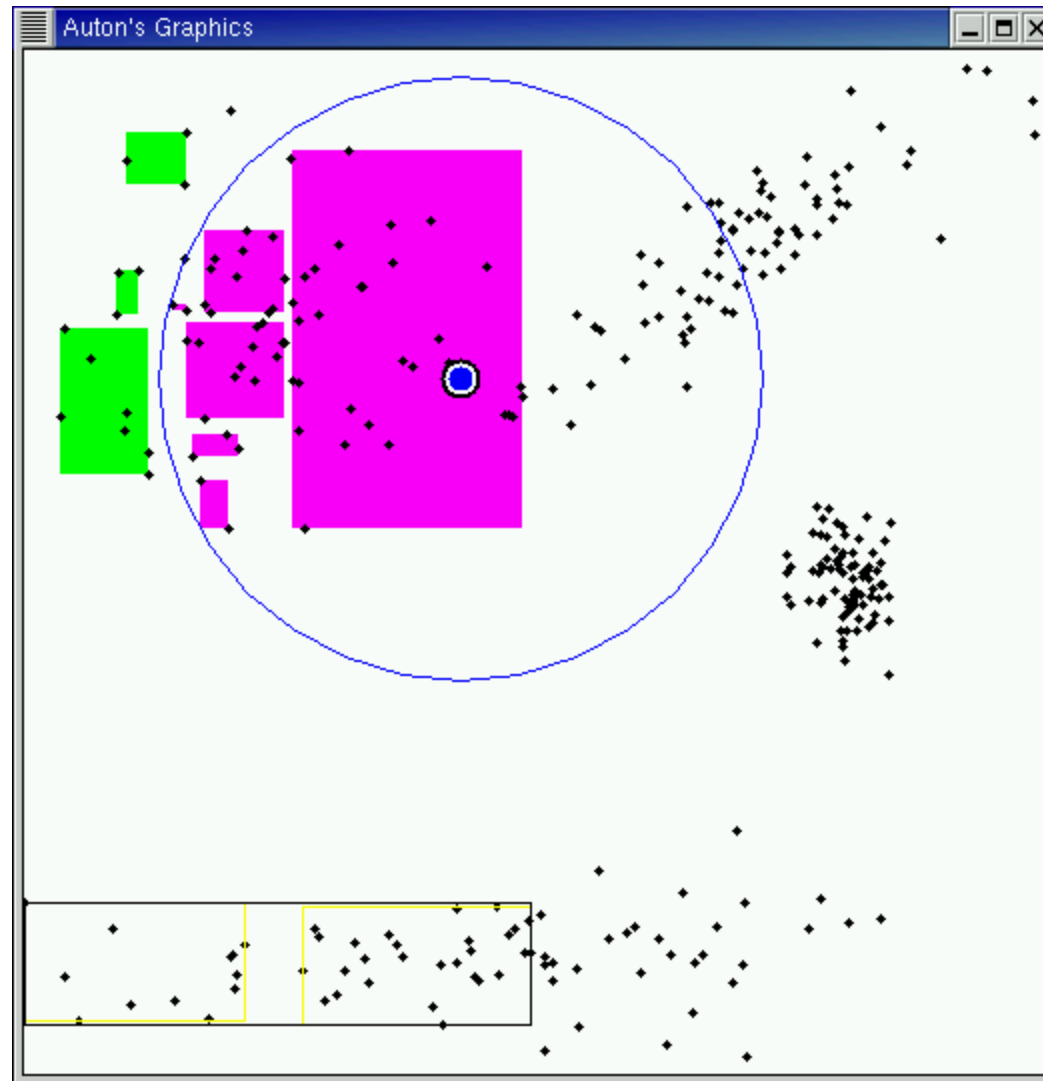


Range-count example

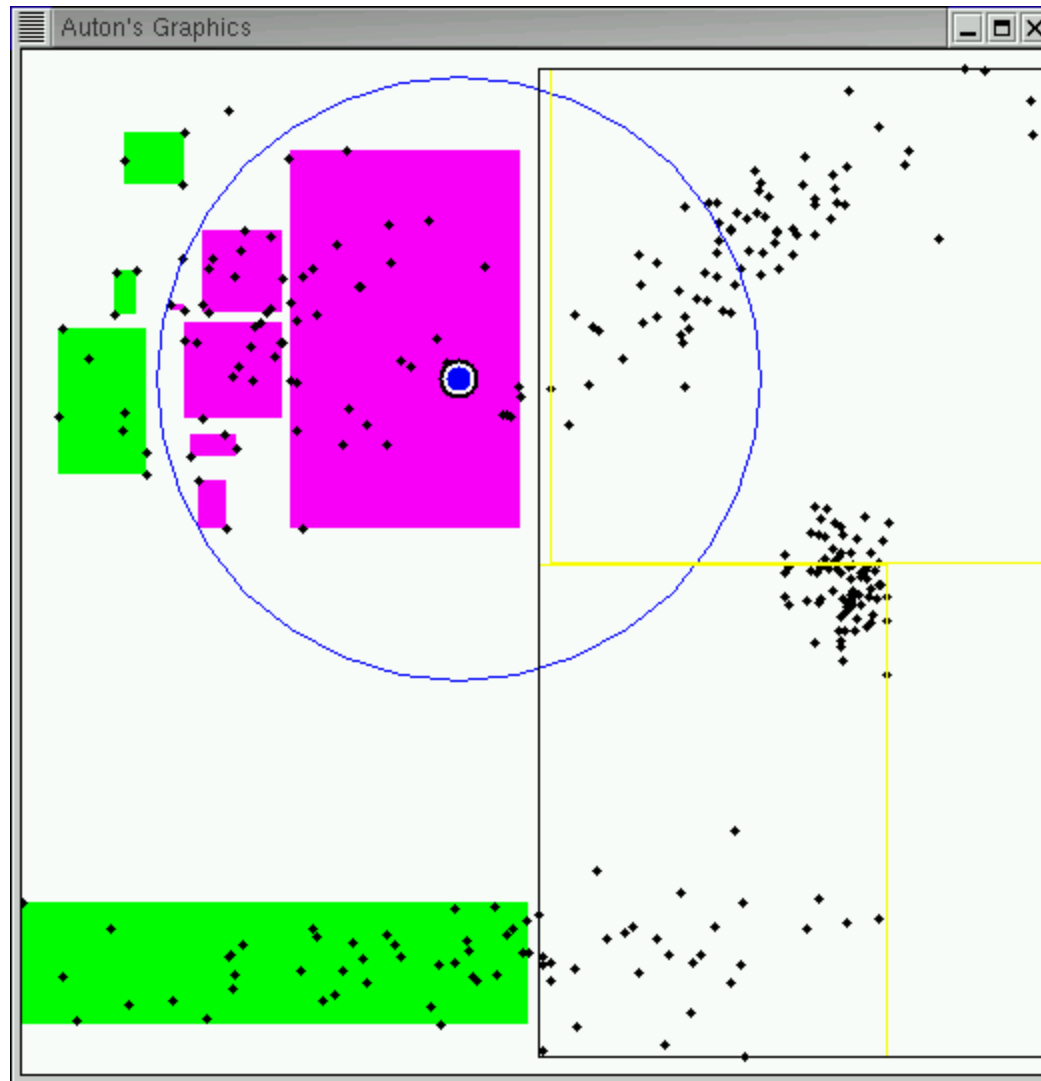


Pruned!
(exclusion)

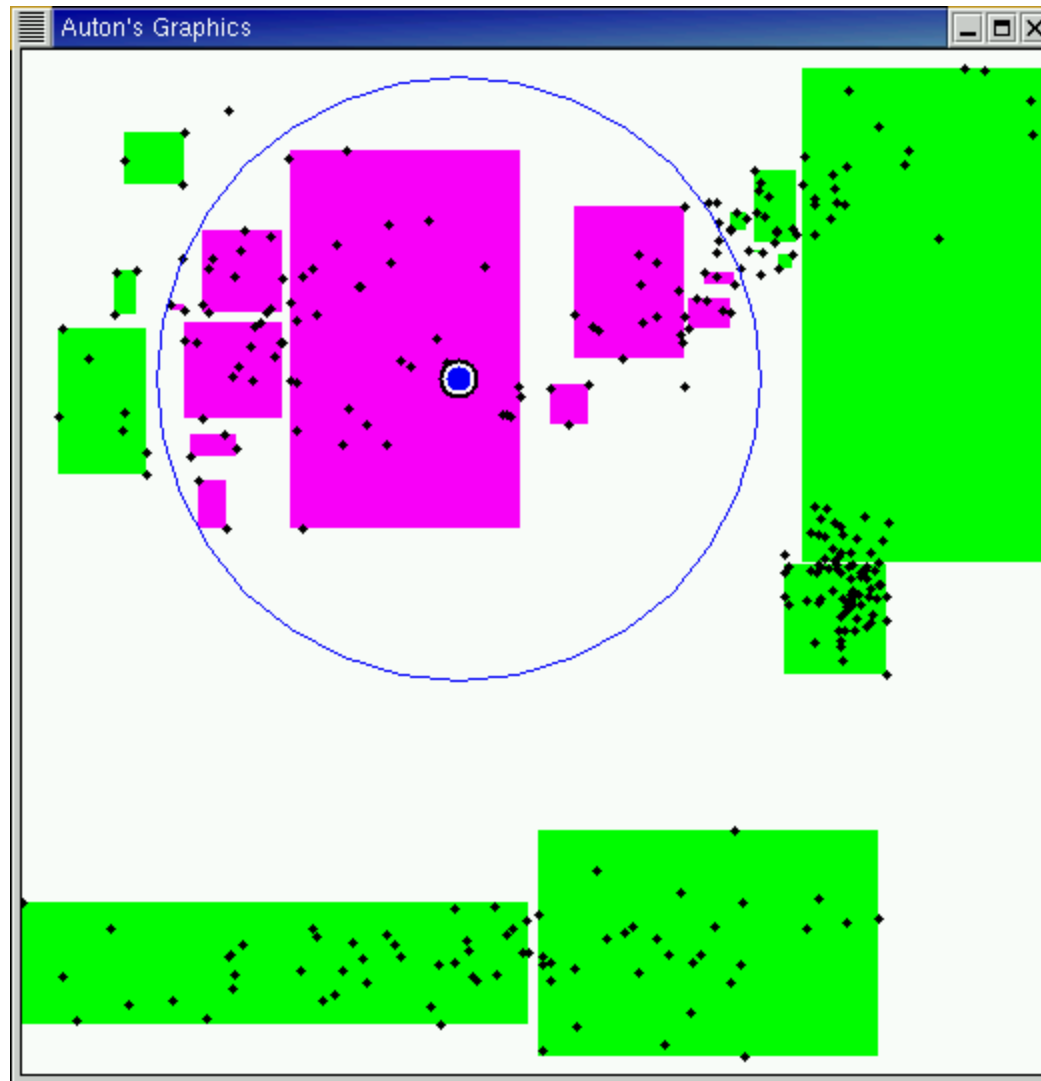
Range-count example



Range-count example



Range-count example



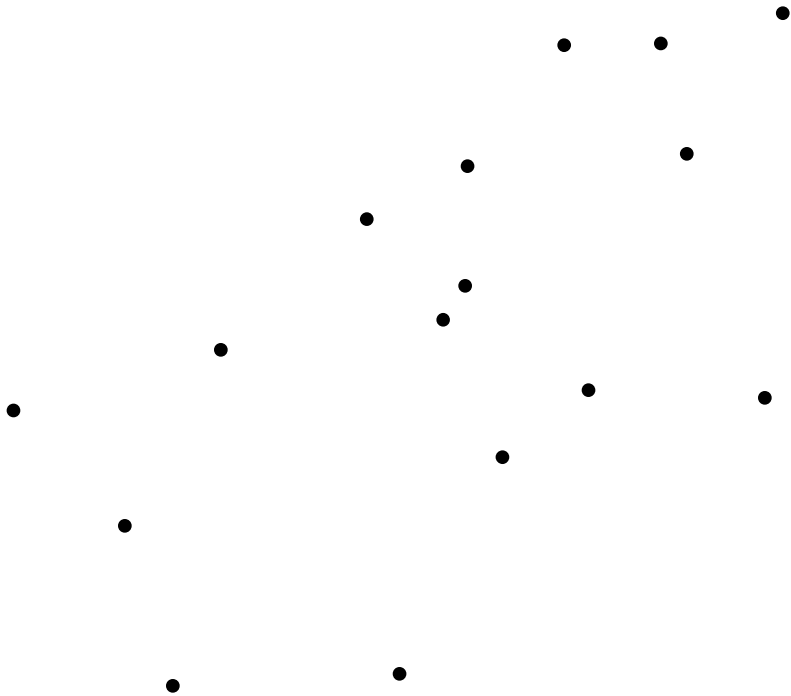
Some questions

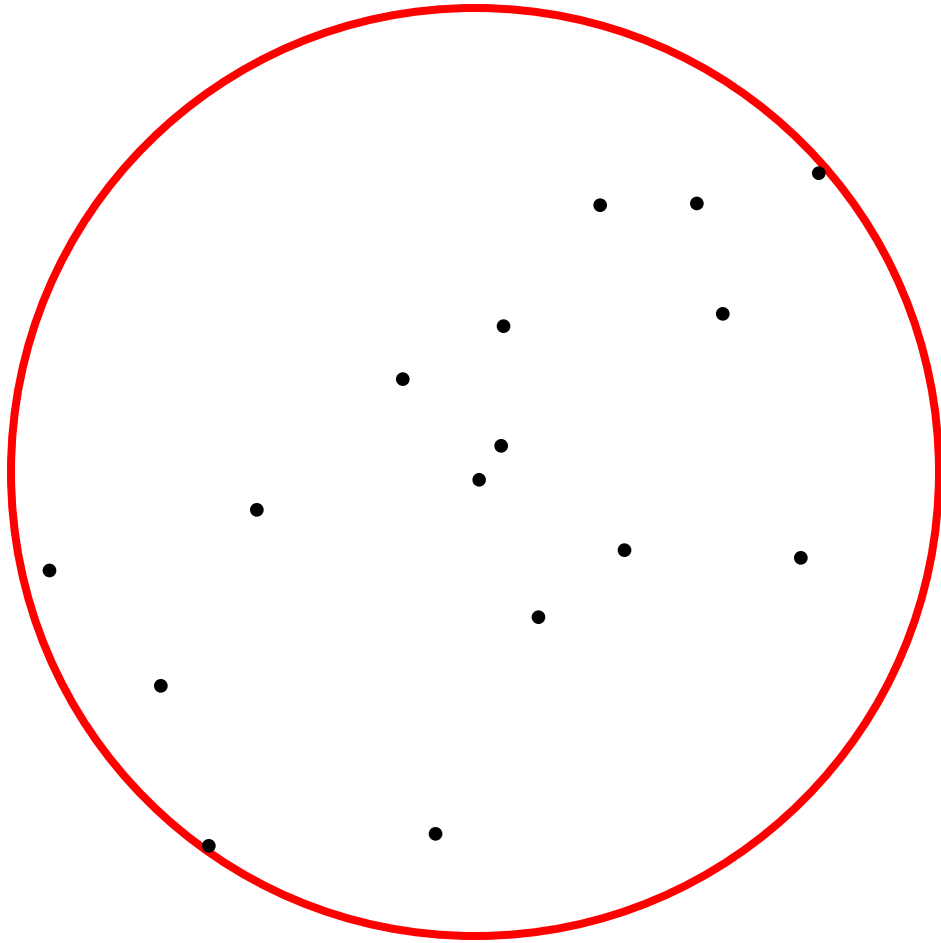
- Asymptotic runtime analysis?
 - In a rough sense, $O(\log N)$
 - But only under some regularity conditions
- How high in dimension can we go?
 - Roughly exponential in intrinsic dimension
 - In practice, in less than 100 dimensions, still big speedups

Another kind of tree

- Ball-trees, metric trees
 - Use balls instead of hyperrectangles
 - Can often be more efficient in high dimension (though not always)
 - Can work with non-Euclidean metric (you only need to respect the triangle inequality)
 - Many non-metric similarity measures can be bounded by metric quantities.

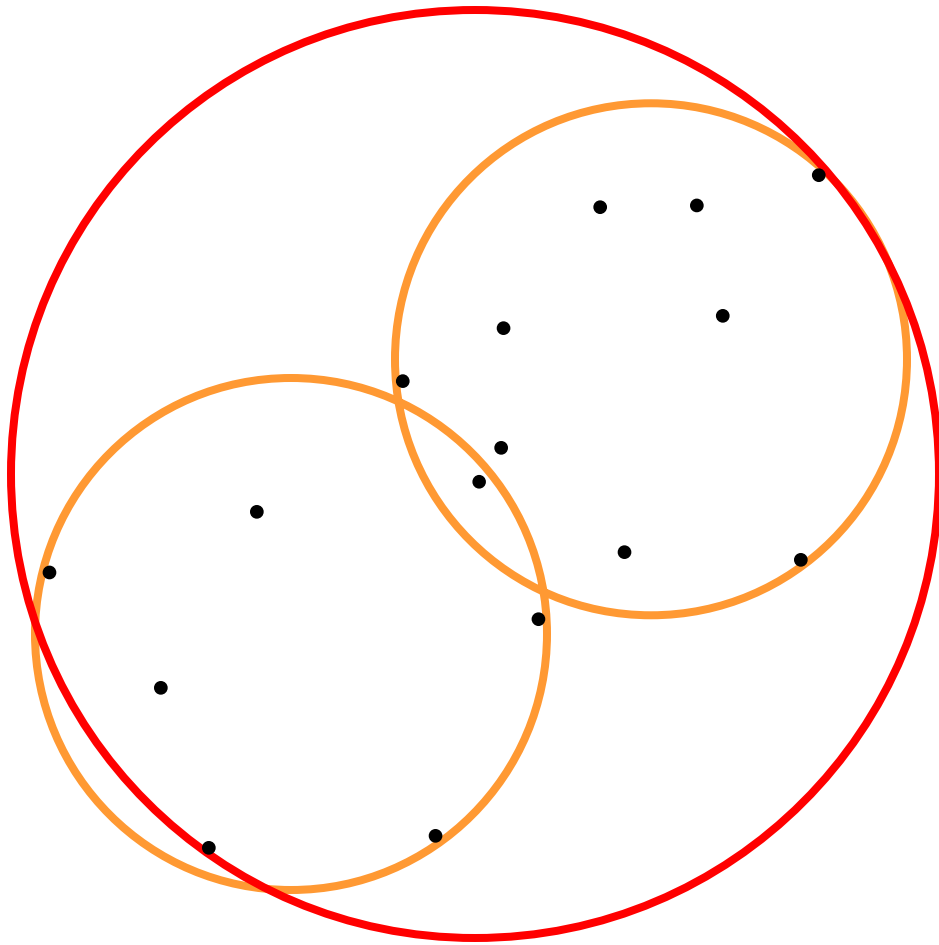
A Set of Points in a metric space



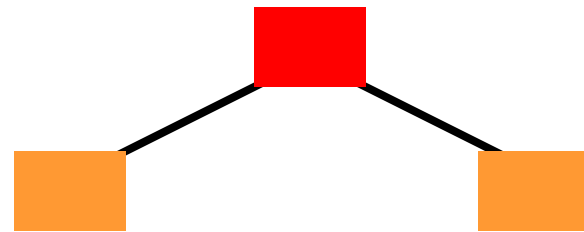


Ball Tree root
node

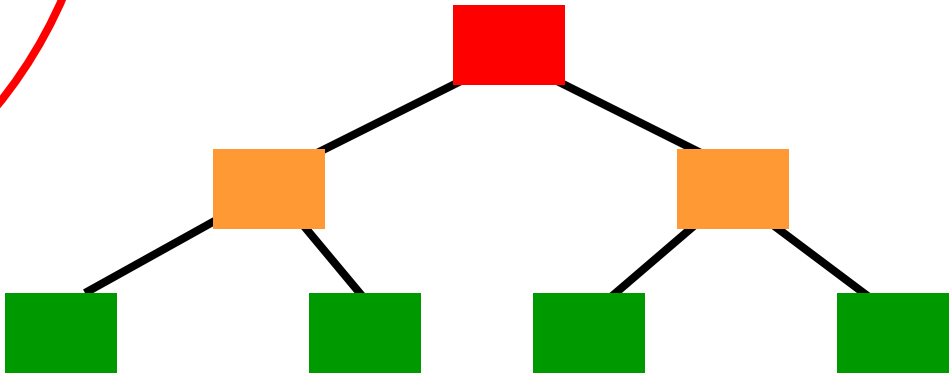
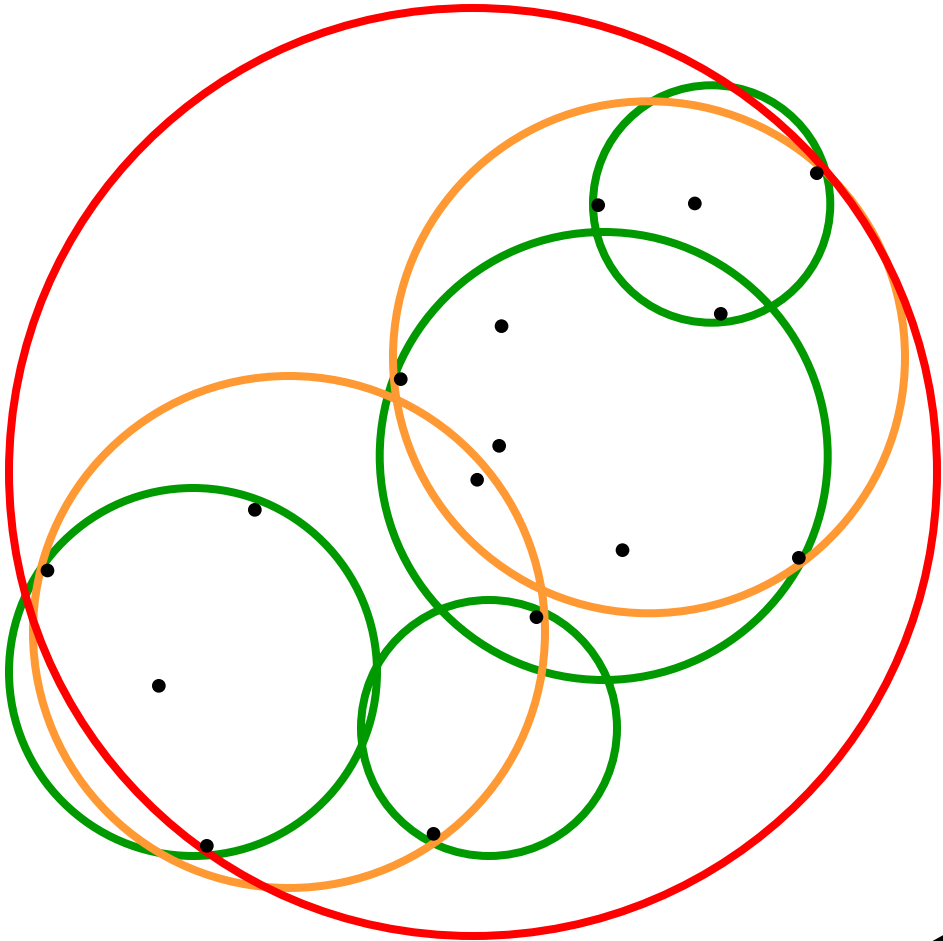




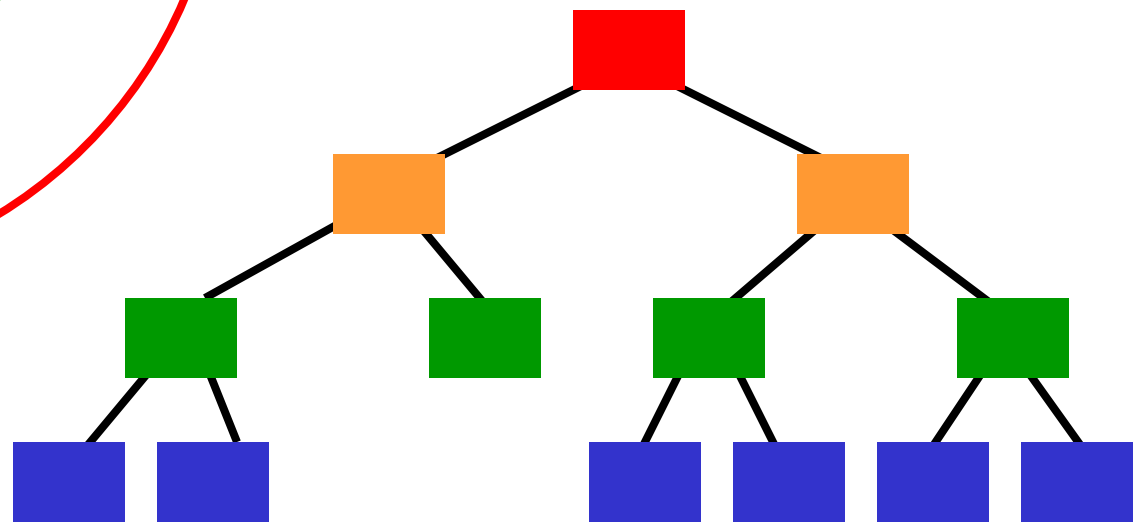
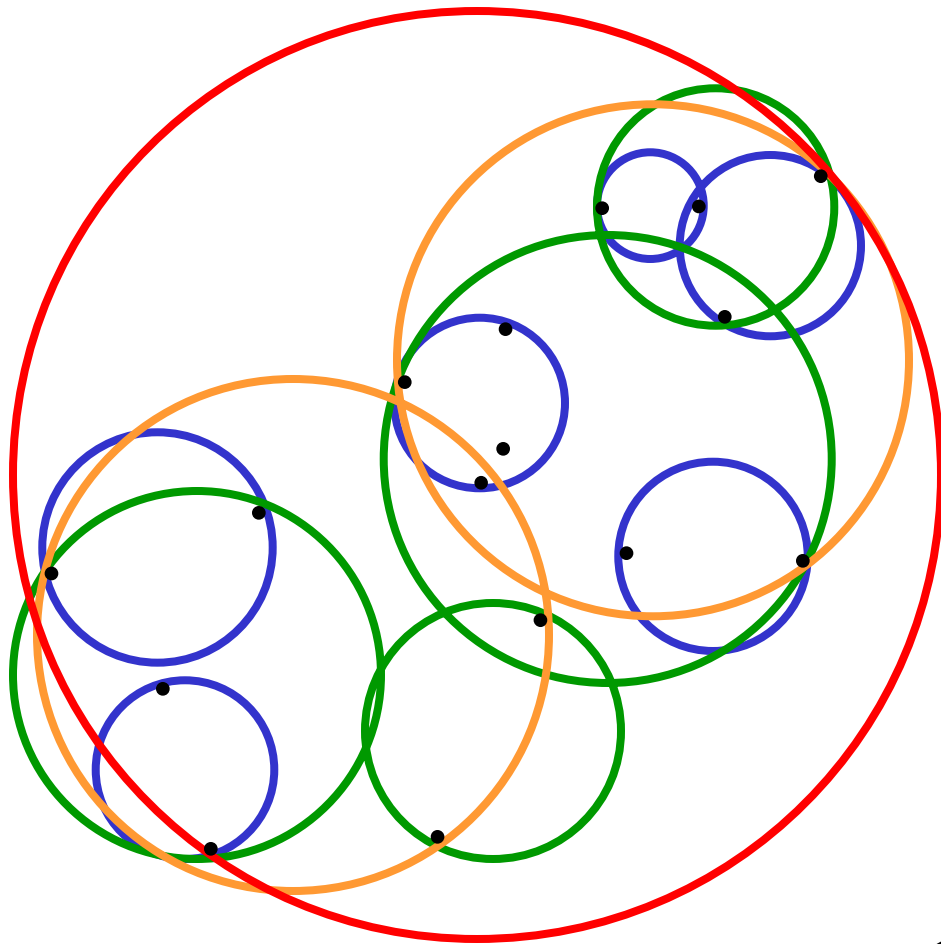
A Ball Tree



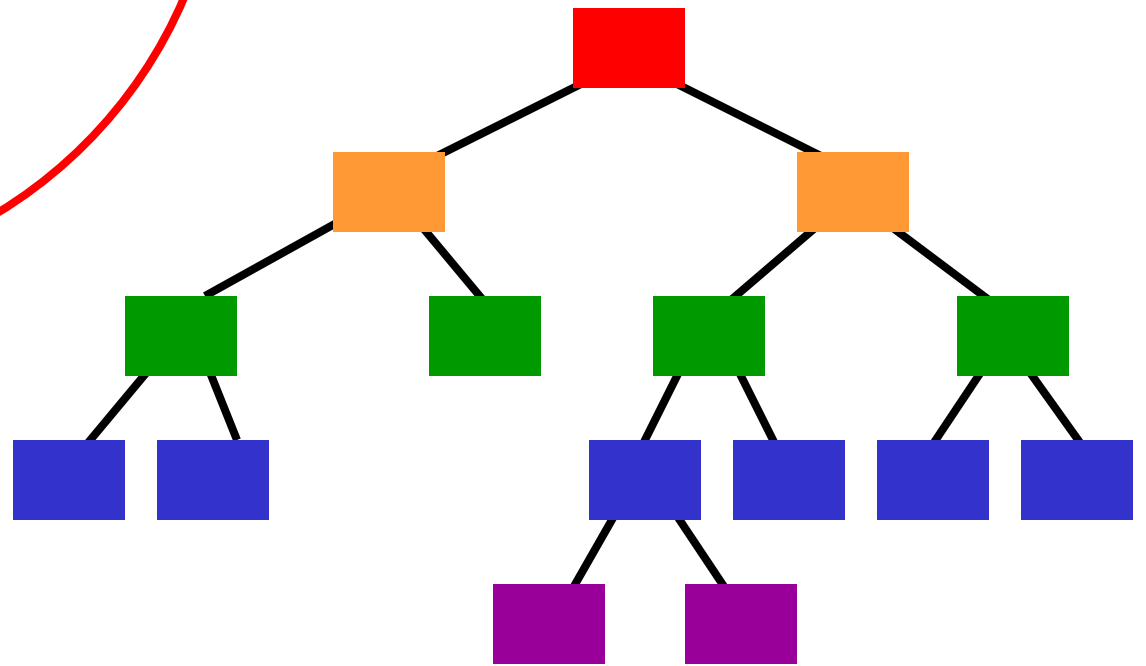
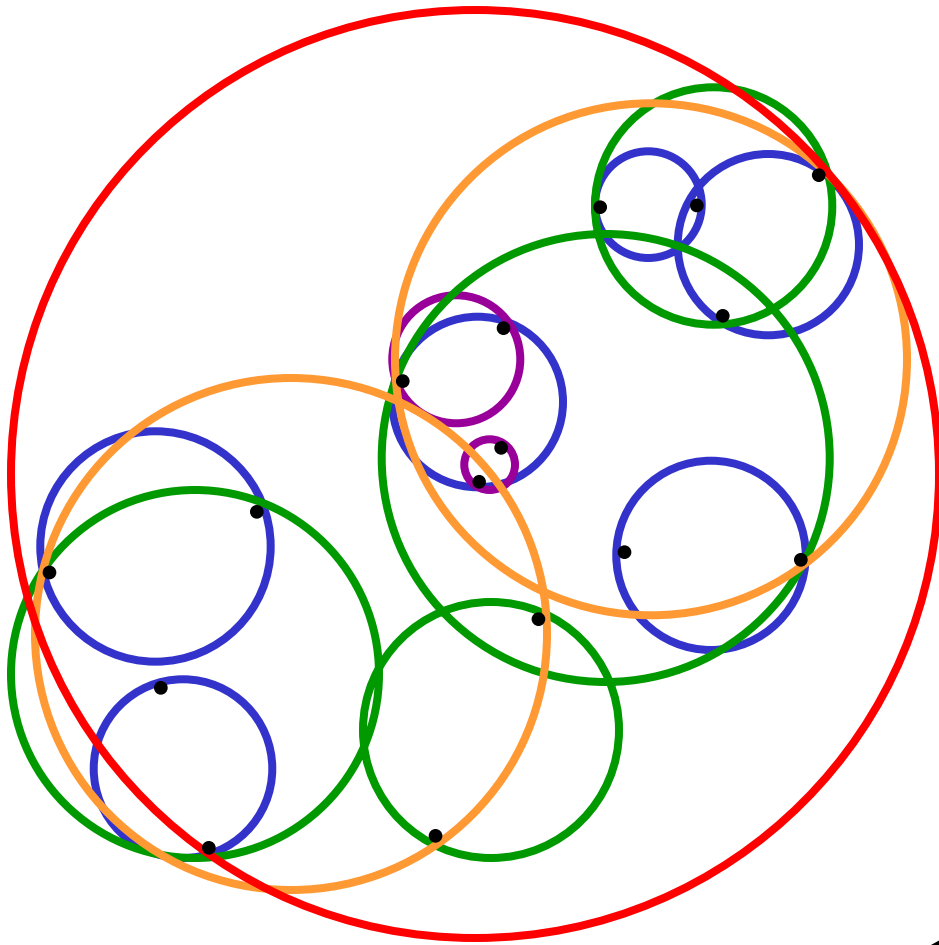
A Ball Tree



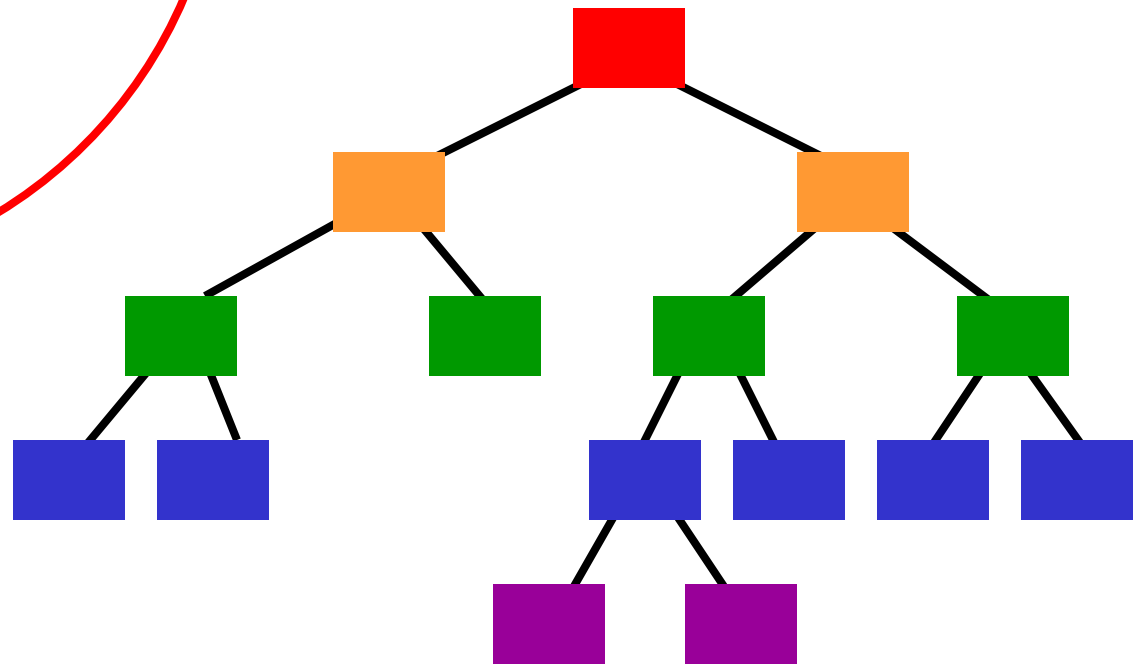
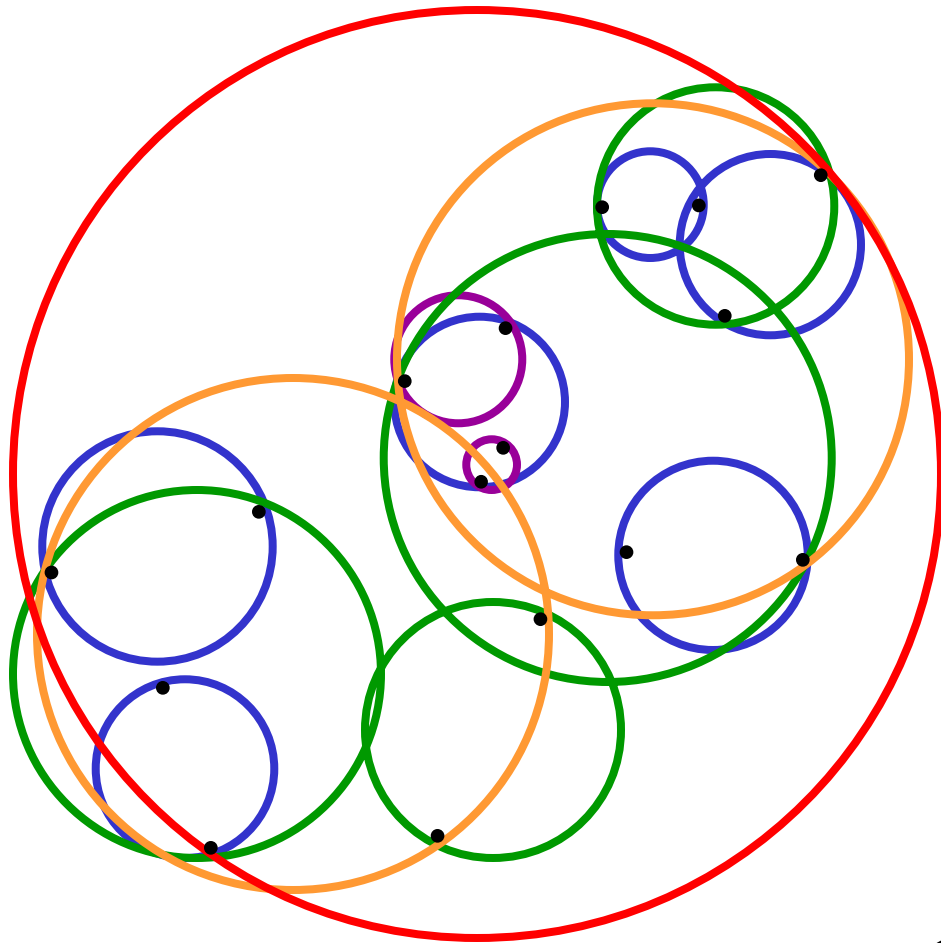
A Ball Tree



A Ball Tree



A Ball Tree



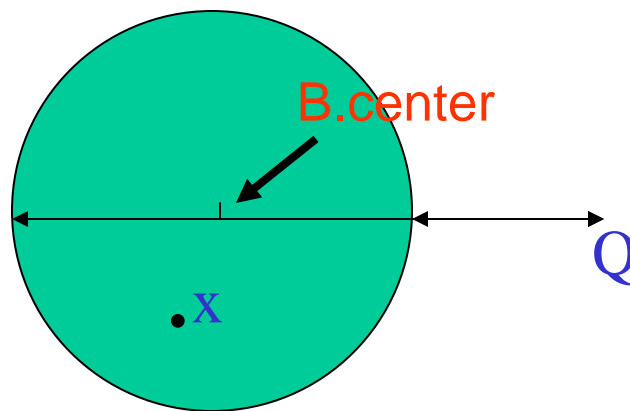
- J. Uhlmann, 1991
- S. Omohundro, NIPS 1991

Ball-trees: properties

Let Q be any query point and let x be a point inside ball B

$$|x - Q| \geq |Q - B.\text{center}| - B.\text{radius}$$

$$|x - Q| \leq |Q - B.\text{center}| + B.\text{radius}$$



How to build a metric tree, exactly?

- Must balance quality vs. build-time
- ‘Anchors hierarchy’ (farthest-points heuristic, 2-approx used in OR)
- Omohundro: ‘Five ways to build a ball-tree’
- Which is the best? A research topic...

Some other trees

- Cover-tree
 - Provable worst-case $O(\log N)$ under an assumption (bounded expansion constant)
 - Like a non-binary ball-tree
- Learning trees
 - In this conference

'All'-type problems

- **Nearest-neighbor search**

$$NN(x_q) = \arg \min_r \|x_q - x_r\|$$

All-nearest neighbor
(bichromatic):

$$\forall x_q : NN(x_q) = \arg \min_r \|x_q - x_r\|$$

- **Kernel density estimation**

$$\hat{f}(x_q) = \frac{1}{N} \sum_{r \neq q}^N K_h(\|x_q - x_r\|)$$

'All' version
(bichromatic):

$$\forall x_q : \hat{f}(x_q) = \frac{1}{N} \sum_{r \neq q}^N K_h(\|x_q - x_r\|)$$

Almost always 'all'-type problems

- Kernel density estimation
- Nadaraya-Watson & locally-wgtd regression
- Gaussian process prediction
- Radial basis function networks

Always 'all'-type problems

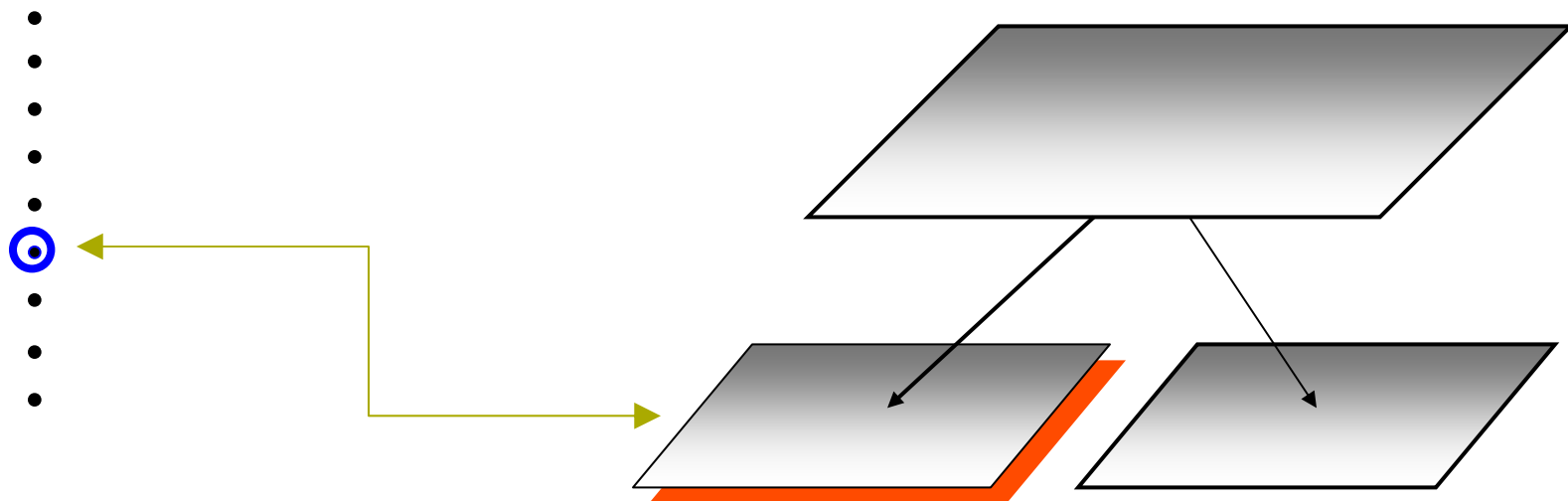
- Monochromatic all-nearest neighbor (e.g. LLE)
- n -point correlation (n -tuples)

Dual-tree idea

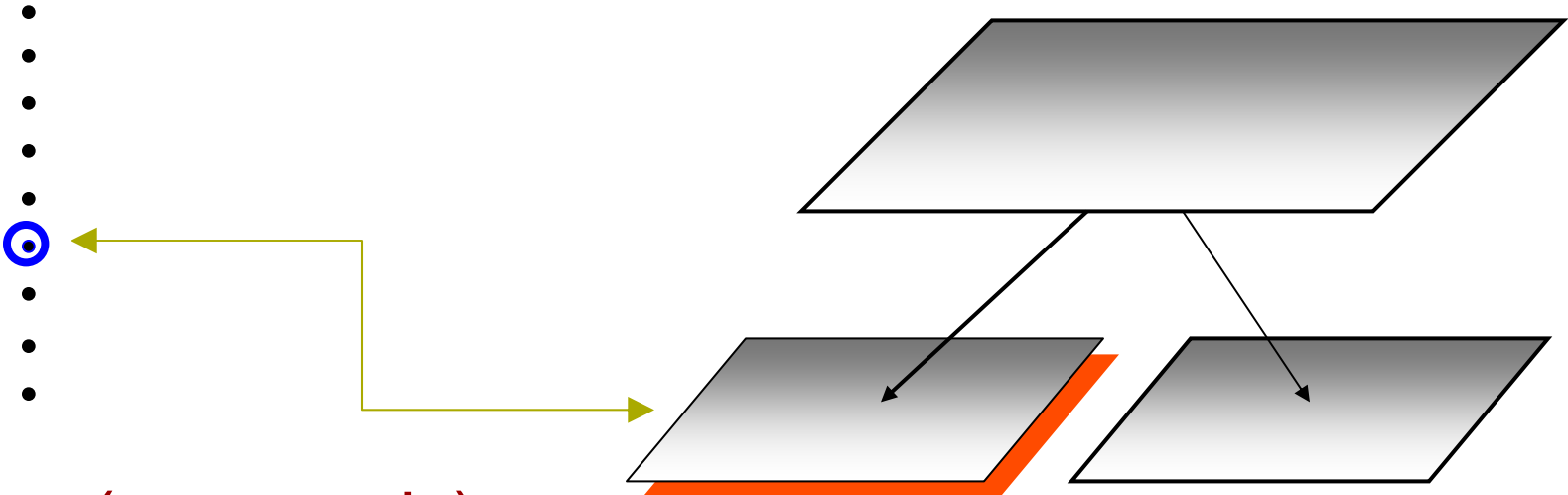
If all the queries are available simultaneously, then it is faster to:

1. Build a tree on the queries as well
2. Effectively process the queries in chunks rather than individually → *work is shared between similar query points*

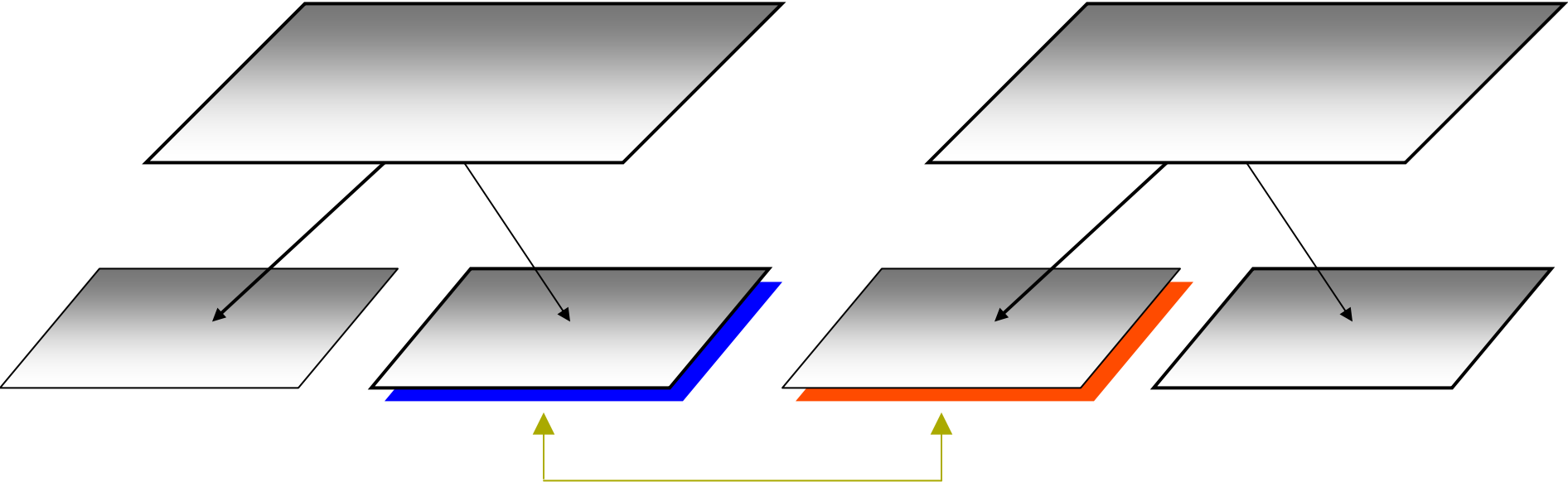
Single-tree:



Single-tree:

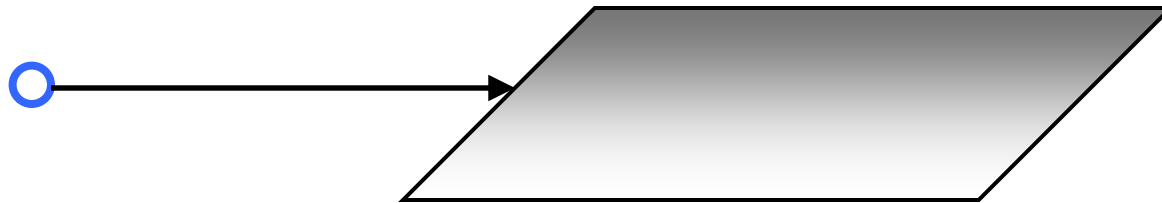


Dual-tree (symmetric):



Exclusion and inclusion, using point-node *kd*-tree bounds.

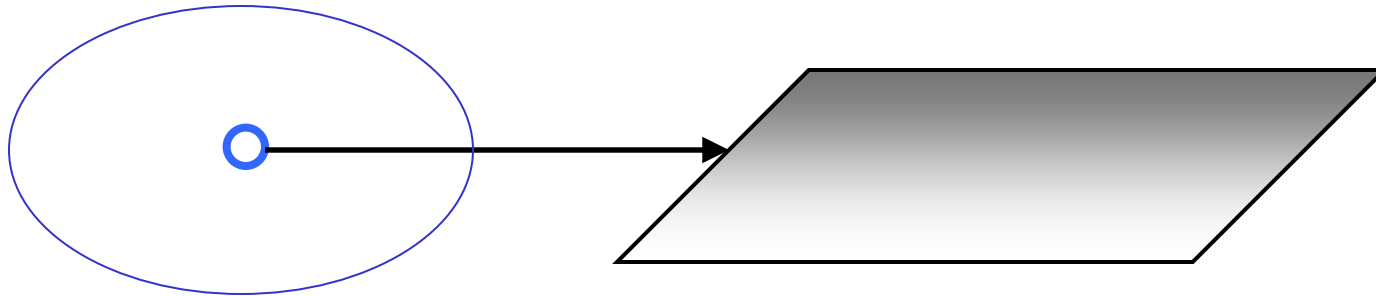
O(D) bounds on distance minima/maxima:



$$\min_i \|x - x_i\| \geq \sum_d^D \left[\max\{(l_d - x_d)^2, 0\} + \max\{(x_d - u_d)^2, 0\} \right]$$
$$\max_i \|x - x_i\| \leq \sum_d^D \max\{(u_d - x_d)^2, (x_d - l_d)^2\}$$

Exclusion and inclusion, using point-node *kd*-tree bounds.

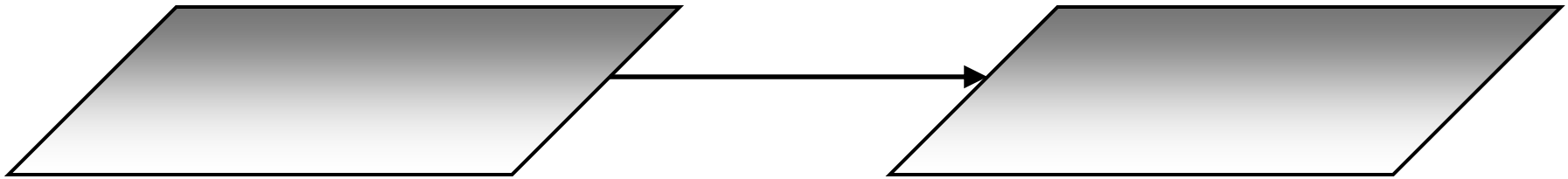
O(D) bounds on distance minima/maxima:



$$\min_i \|x - x_i\| \geq \sum_d^D \left[\max\{(l_d - x_d)^2, 0\} + \max\{(x_d - u_d)^2, 0\} \right]$$
$$\max_i \|x - x_i\| \leq \sum_d^D \max\{(u_d - x_d)^2, (x_d - l_d)^2\}$$

Exclusion and inclusion, using *kd*-tree node-node bounds.

$O(D)$ bounds on distance minima/maxima:



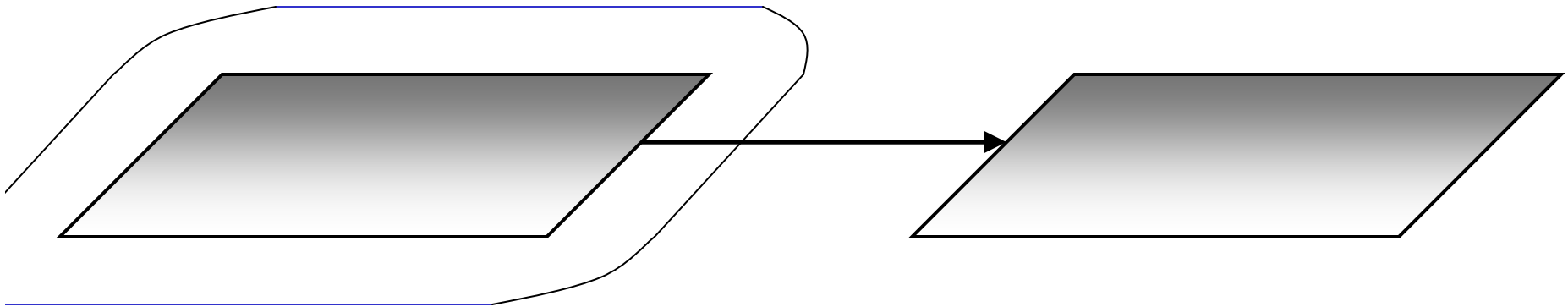
(Analogous to point-node bounds.)

Also needed:

Nodewise bounds.

Exclusion and inclusion, using *kd*-tree node-node bounds.

$O(D)$ bounds on distance minima/maxima:



(Analogous to point-node bounds.)

Also needed:

Nodewise bounds.

Single-tree: simple recursive algorithm (k=1 case)

```
NN( $x_q, R, d_{lo}, x_{sofar}, d_{sofar}$ )  
{  
  if  $d_{lo} > d_{sofar}$ , return.  
  
  if leaf( $R$ ), [ $x_{sofar}, d_{sofar}$ ]=NNBase( $x_q, R, d_{sofar}$ ).  
  else,  
    [ $R1, d1, R2, d2$ ]=orderByDist( $x_q, R.l, R.r$ ).  
    NN( $x_q, R1, d1, x_{sofar}, d_{sofar}$ ).  
    NN( $x_q, R2, d2, x_{sofar}, d_{sofar}$ ).  
}
```

Single-tree \rightarrow Dual-tree

- $x_q \rightarrow Q$
- $d_{lo}(x_q, R) \rightarrow d_{lo}(Q, R)$
- $x_{sofar} \rightarrow \underline{x}_{sofar}$, $d_{sofar} \rightarrow \underline{d}_{sofar}$
- store $Q.d_{sofar} = \max_Q \underline{d}_{sofar}$
- 2-way recursion \rightarrow 4-way recursion

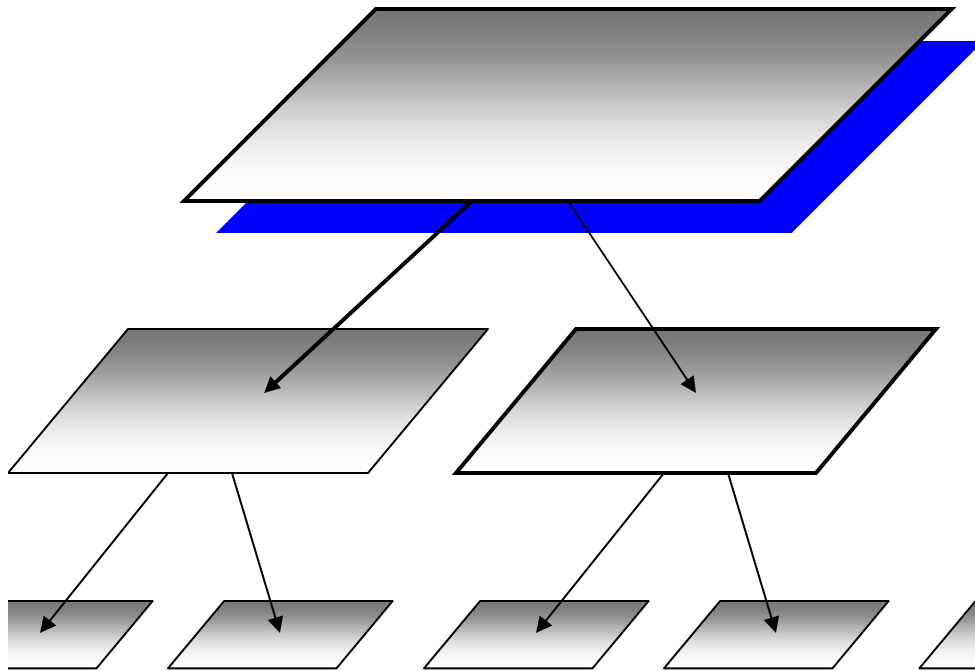
- $N \times O(\log N) \rightarrow O(N)$

Dual-tree: simple recursive algorithm (k=1)

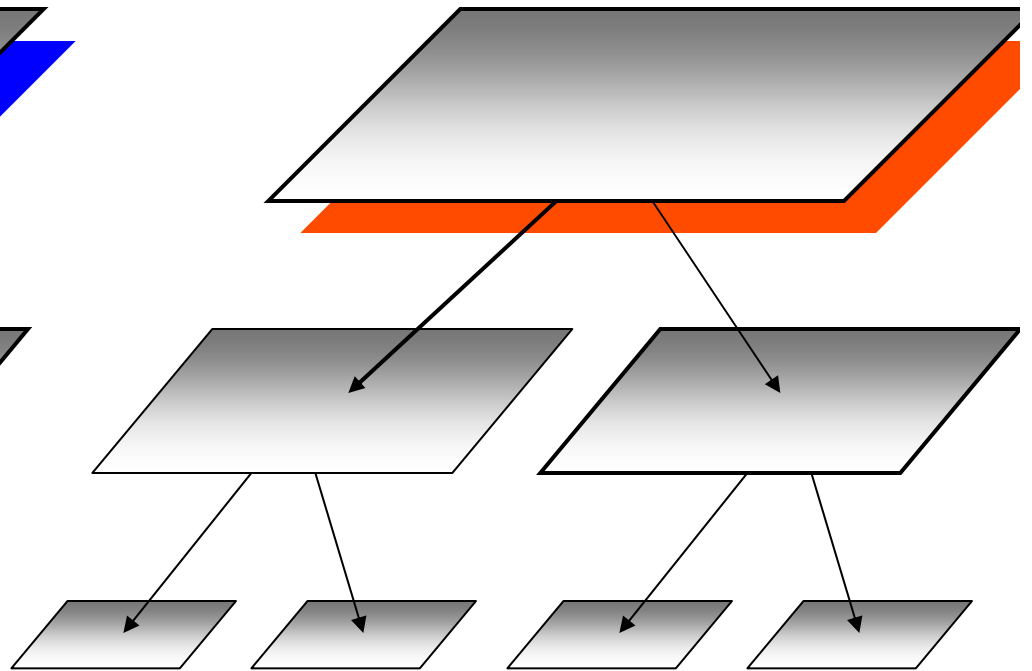
```
AIINN(Q,R,dlo,xsofar,dsofar)
{
  if dlo > Q.dsofar, return.
  if leaf(Q) & leaf(R),
    [xsofar,dsofar]=AIINNBase(Q,R,dsofar). Q.dsofar=maxQdsofar.
  else if !leaf(Q) & leaf(R), ...
  else if leaf(Q) & !leaf(R), ...
  else if !leaf(Q) & !leaf(R),
    [R1,d1,R2,d2]=orderByDist(Q.l,R.l,R.r).
    AIINN(Q.l,R1,d1,xsofar,dsofar).
    AIINN(Q.l,R2,d2,xsofar,dsofar).
    [R1,d1,R2,d2]=orderByDist(Q.r,R.l,R.r).
    AIINN(Q.r,R1,d1,xsofar,dsofar).
    AIINN(Q.r,R2,d2,xsofar,dsofar).
    Q.dsofar = max(Q.l.dsofar,Q.r.dsofar).
}
```

Dual-tree traversal (depth-first)

Query points

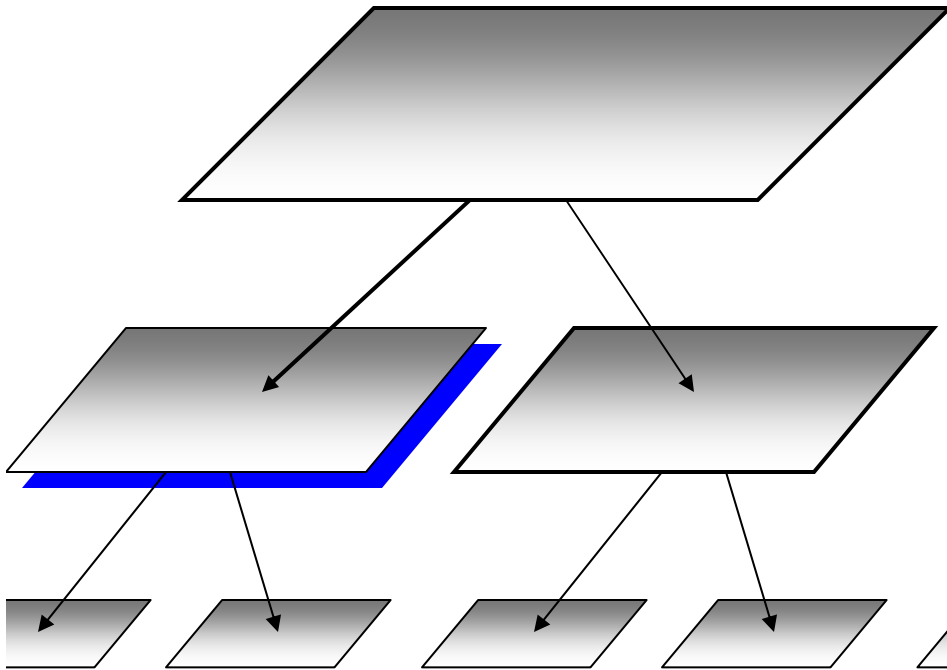


Reference points

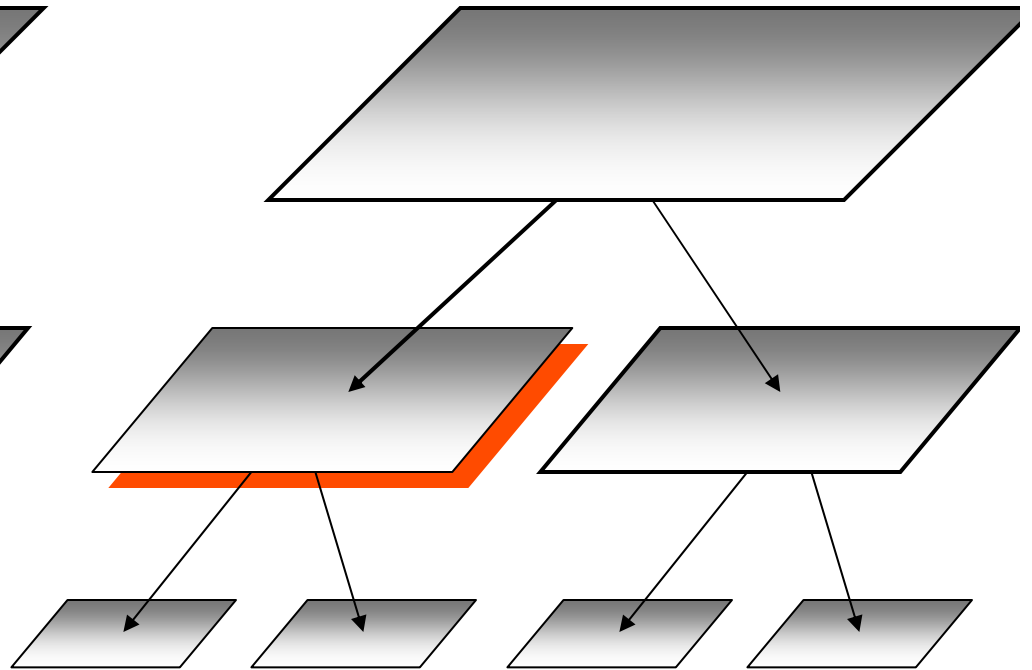


Dual-tree traversal

Query points

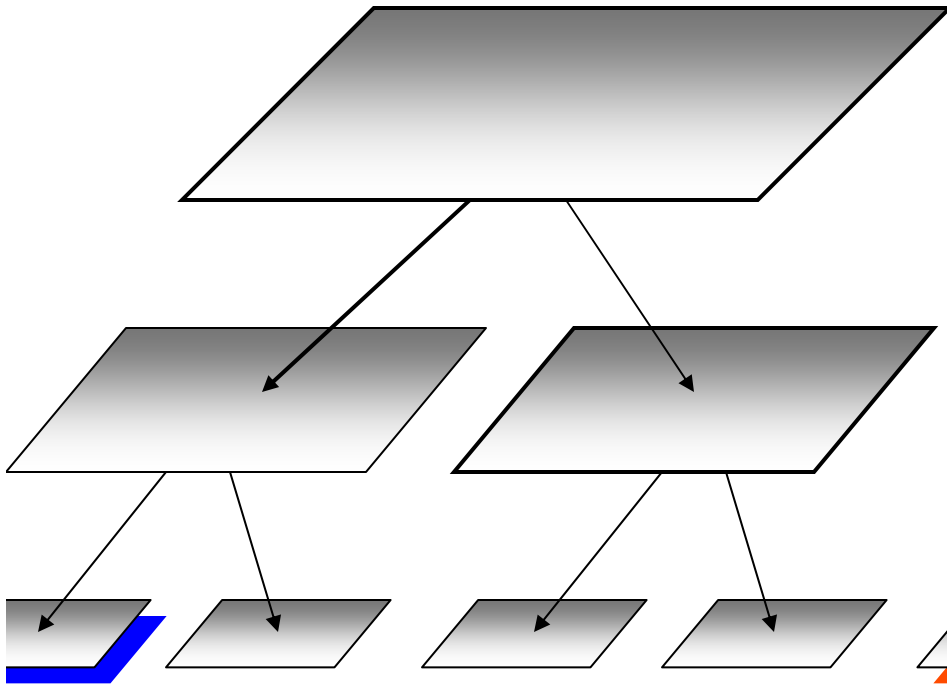


Reference points

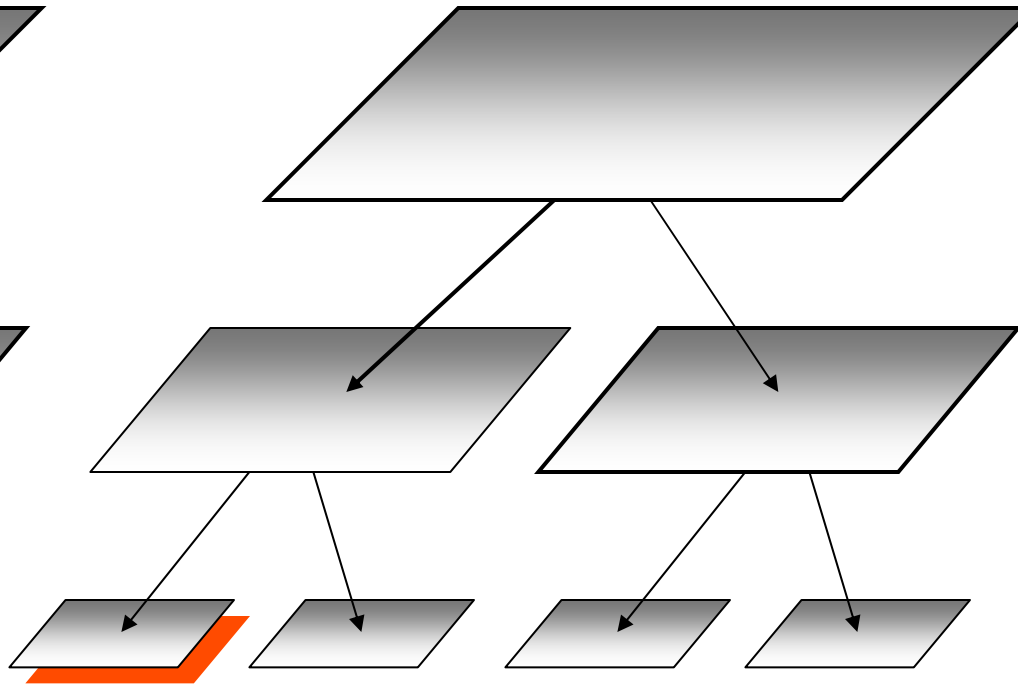


Dual-tree traversal

Query points

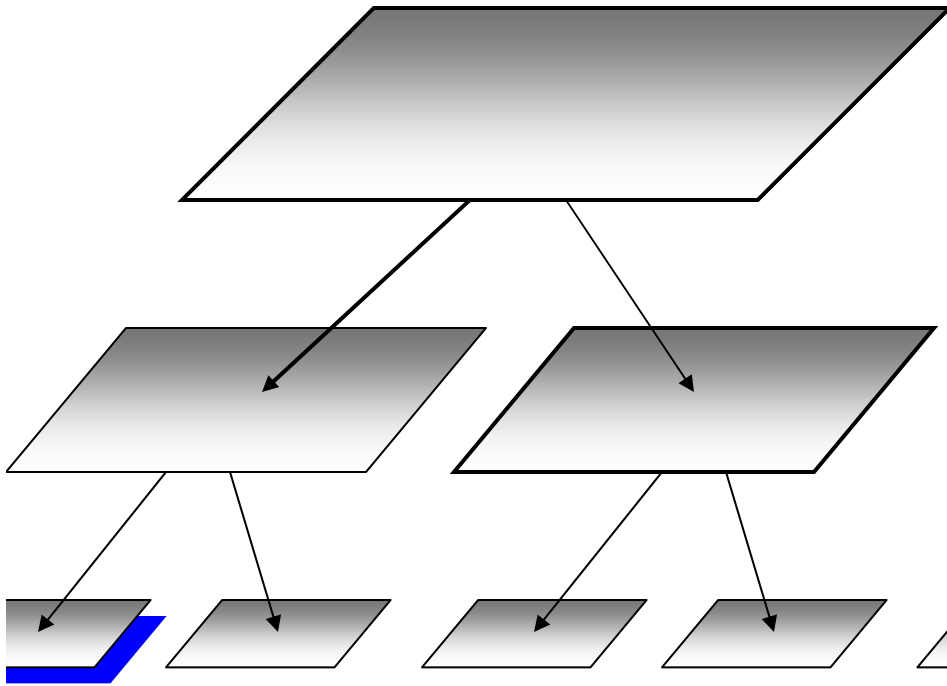


Reference points

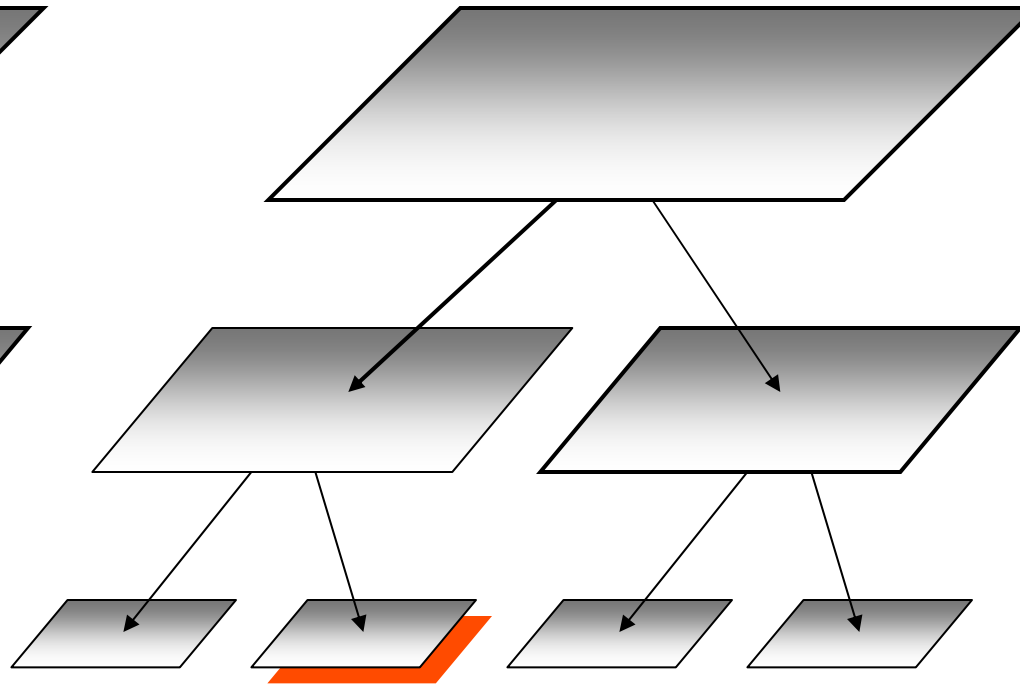


Dual-tree traversal

Query points

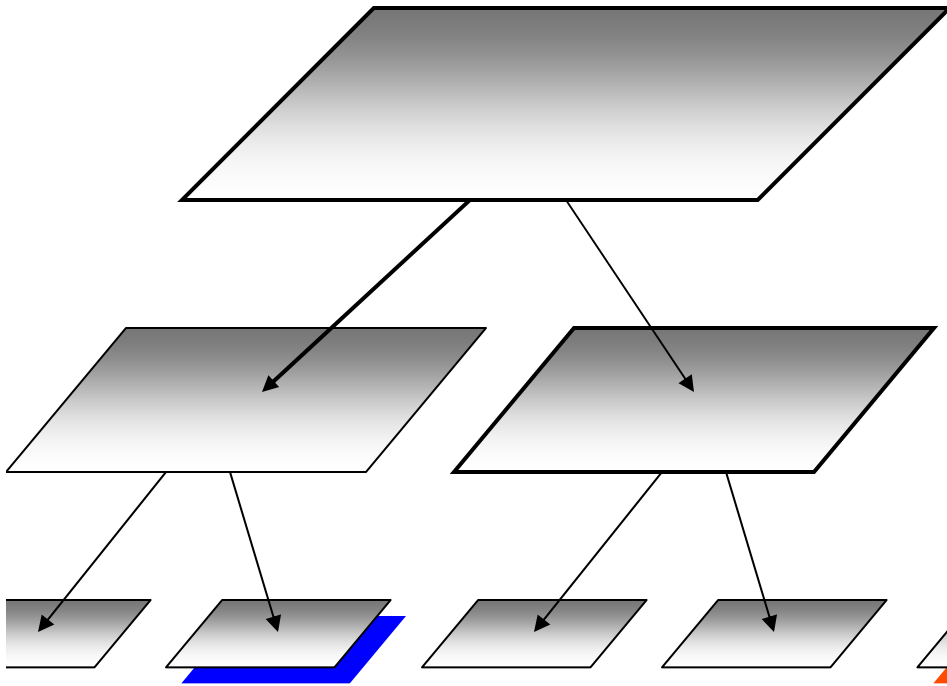


Reference points

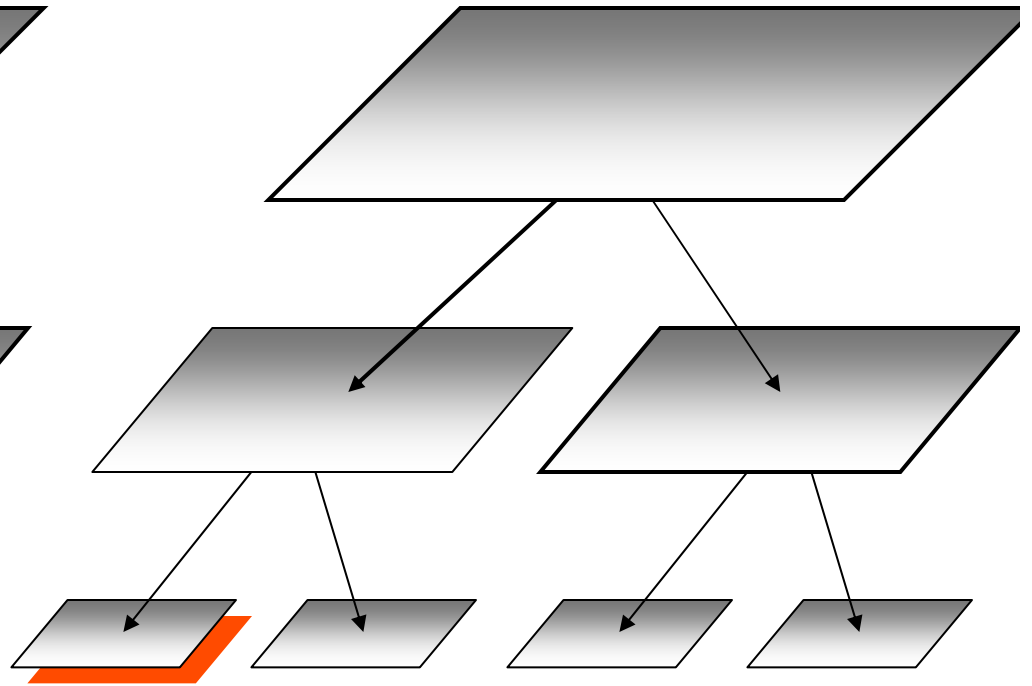


Dual-tree traversal

Query points

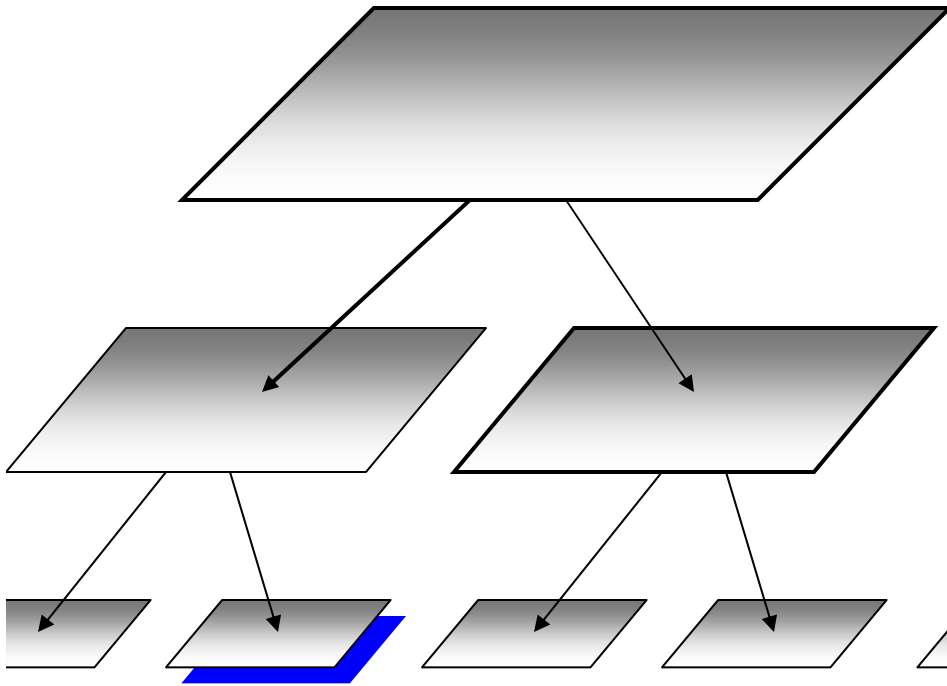


Reference points

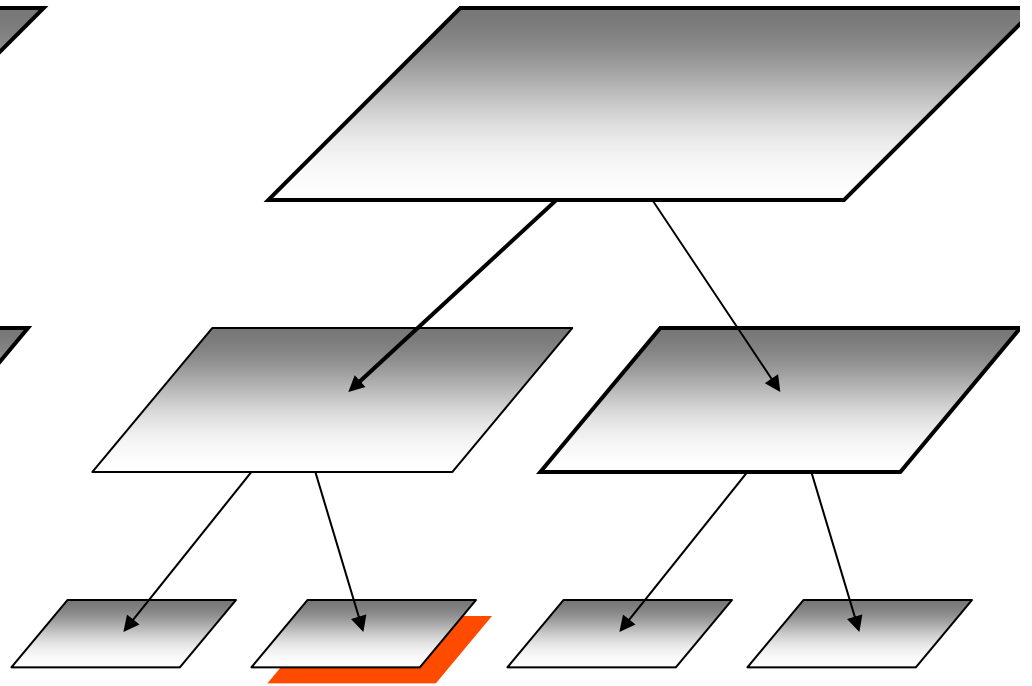


Dual-tree traversal

Query points

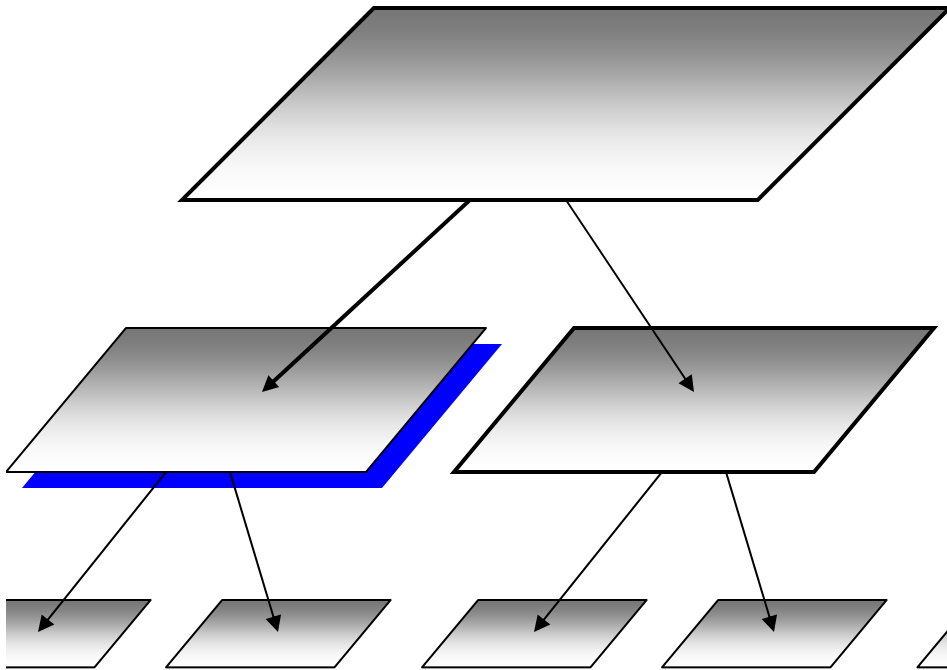


Reference points

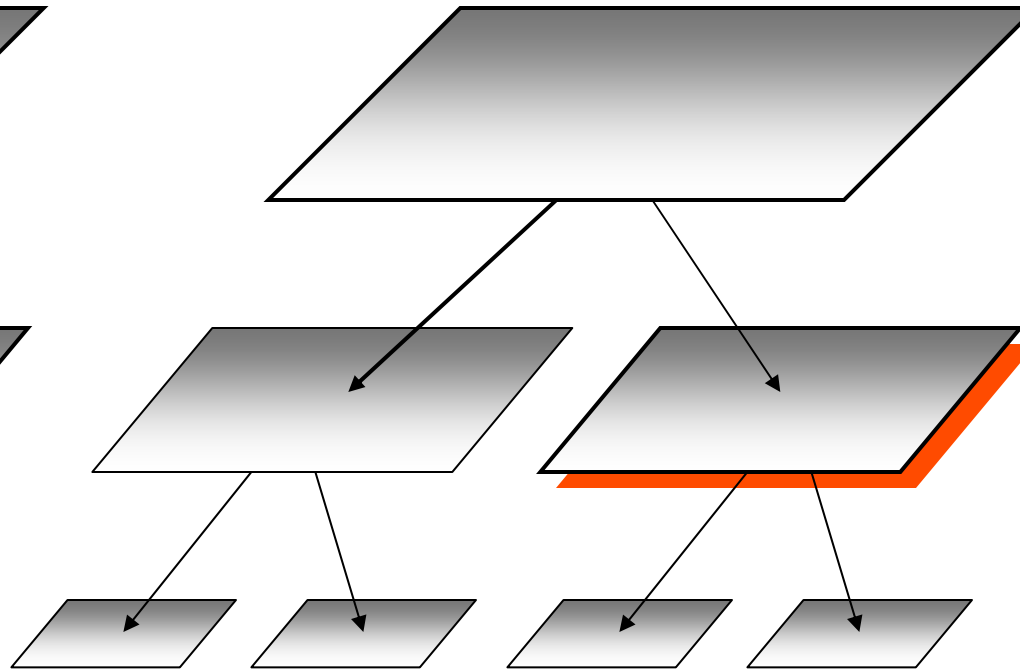


Dual-tree traversal

Query points

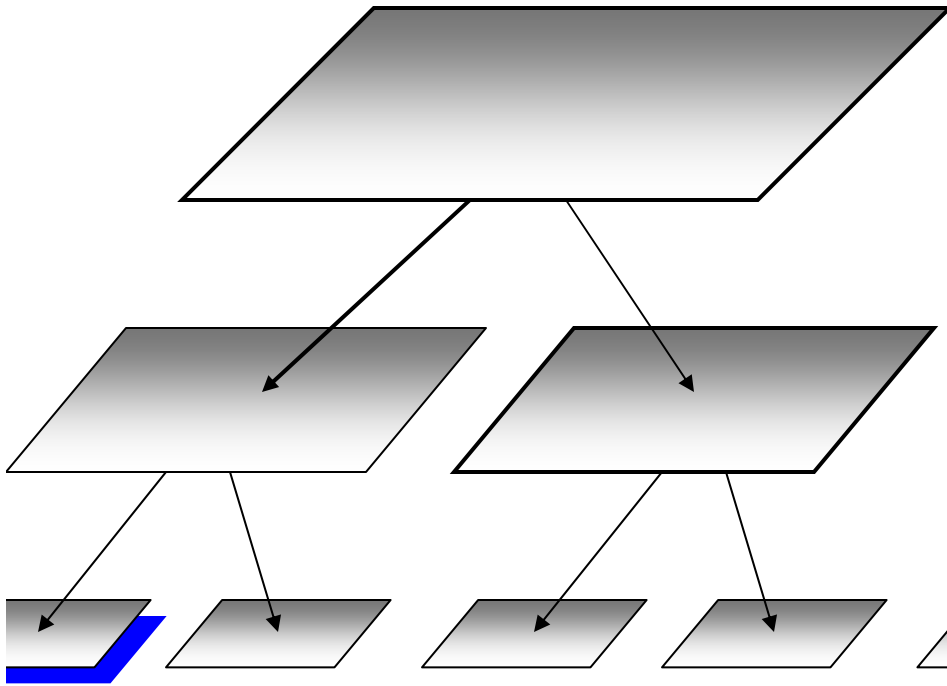


Reference points

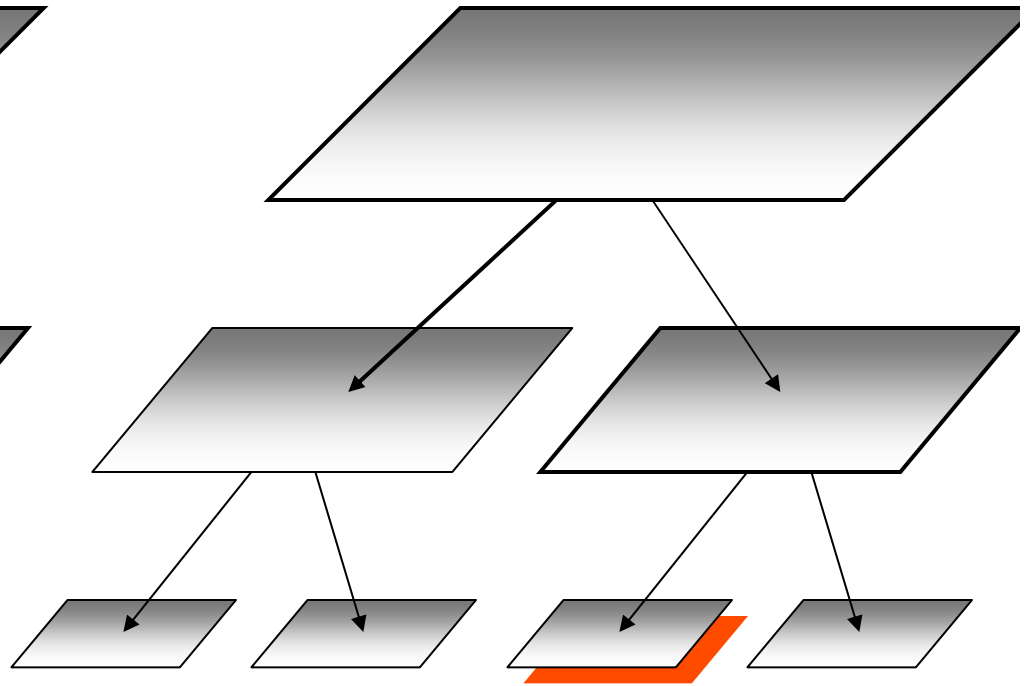


Dual-tree traversal

Query points

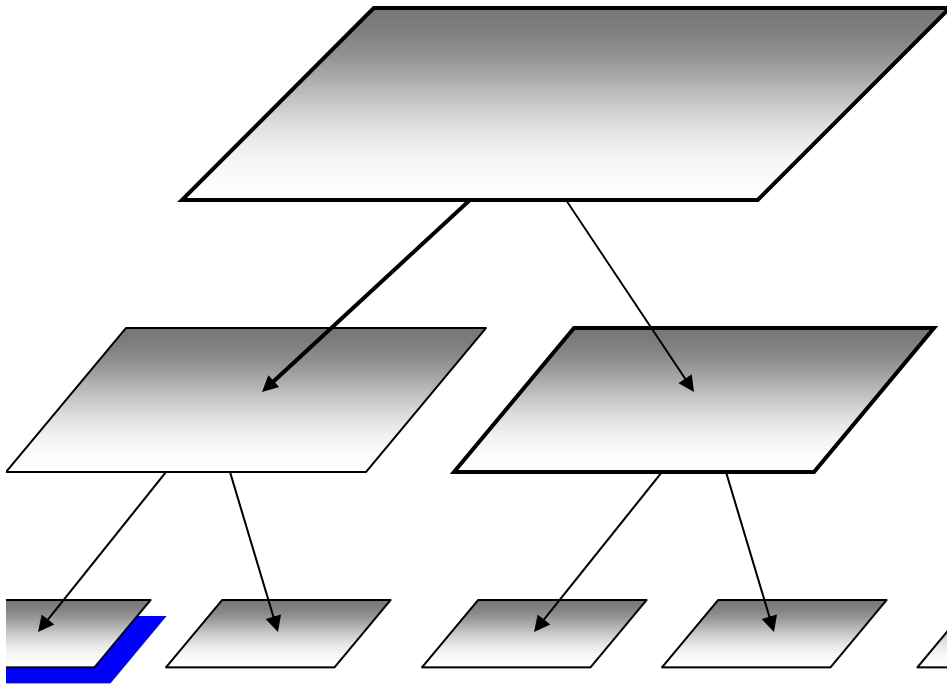


Reference points

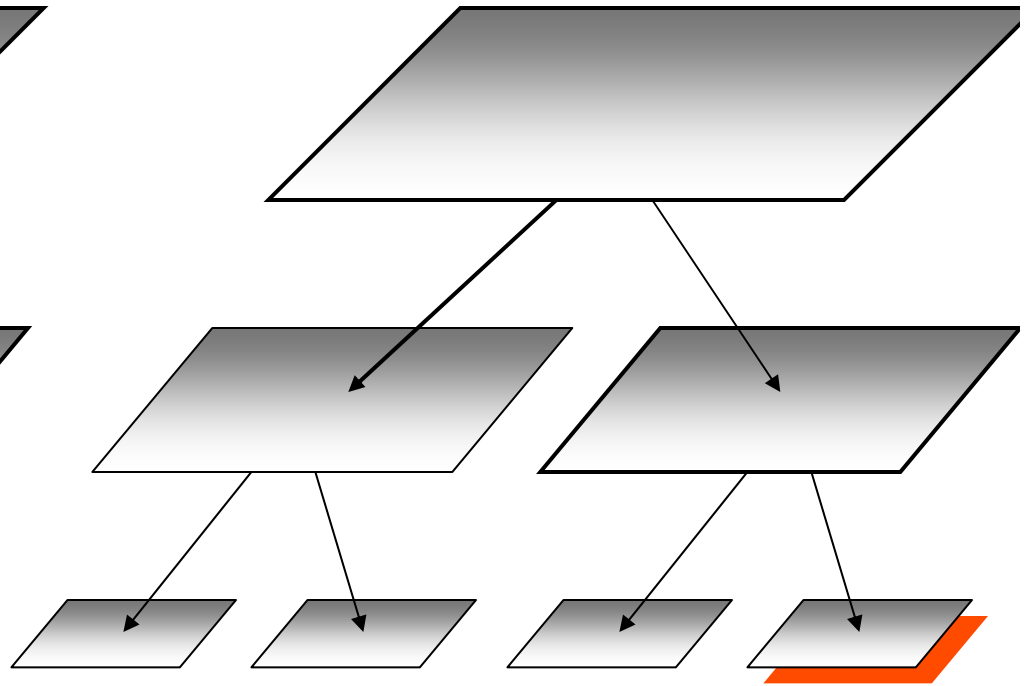


Dual-tree traversal

Query points

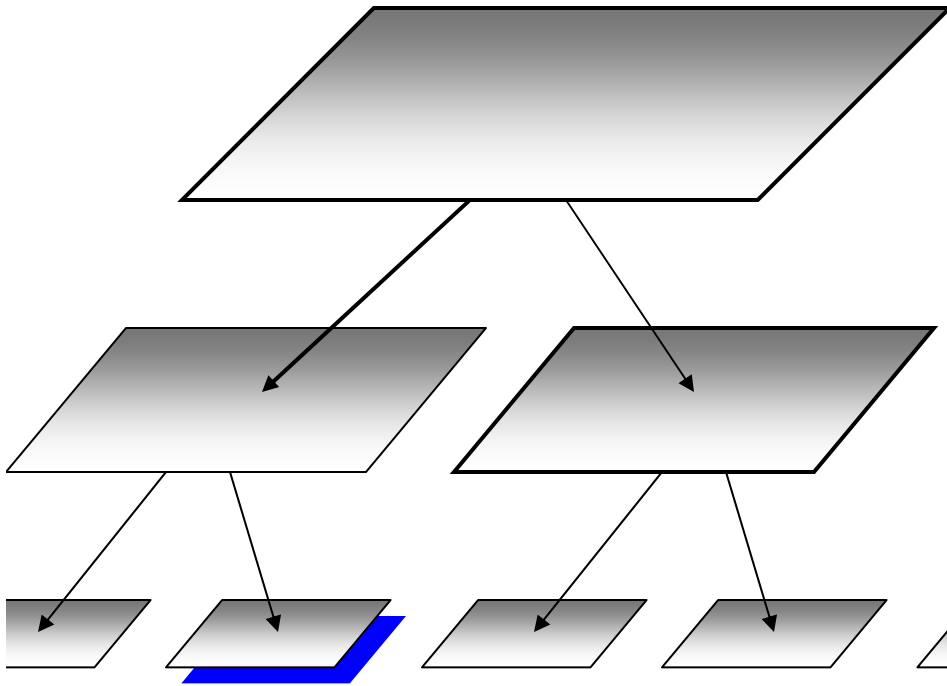


Reference points

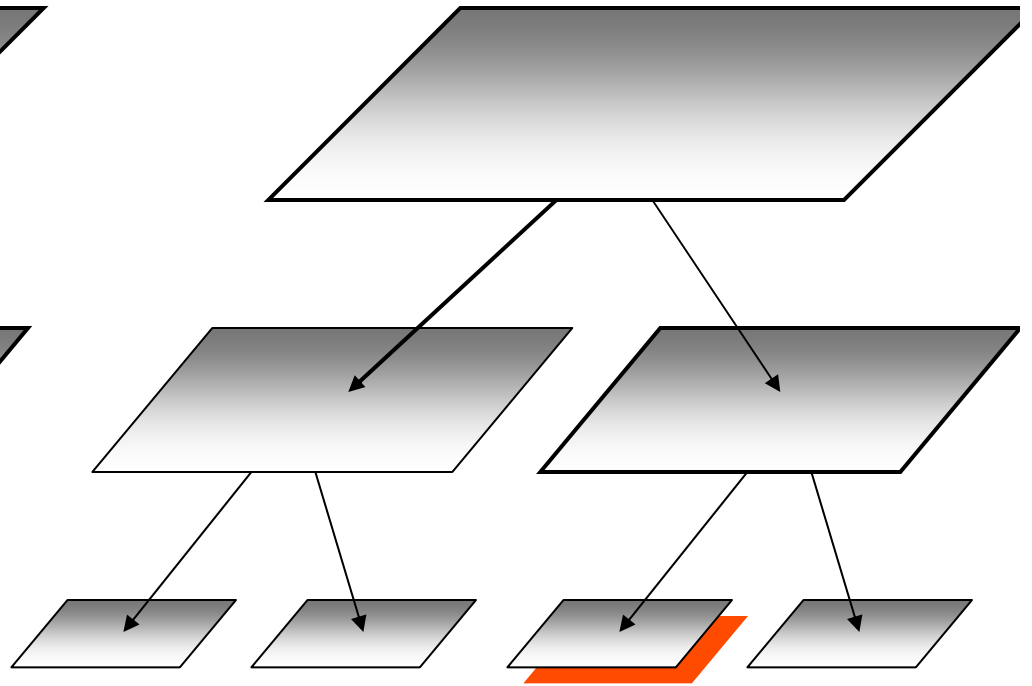


Dual-tree traversal

Query points

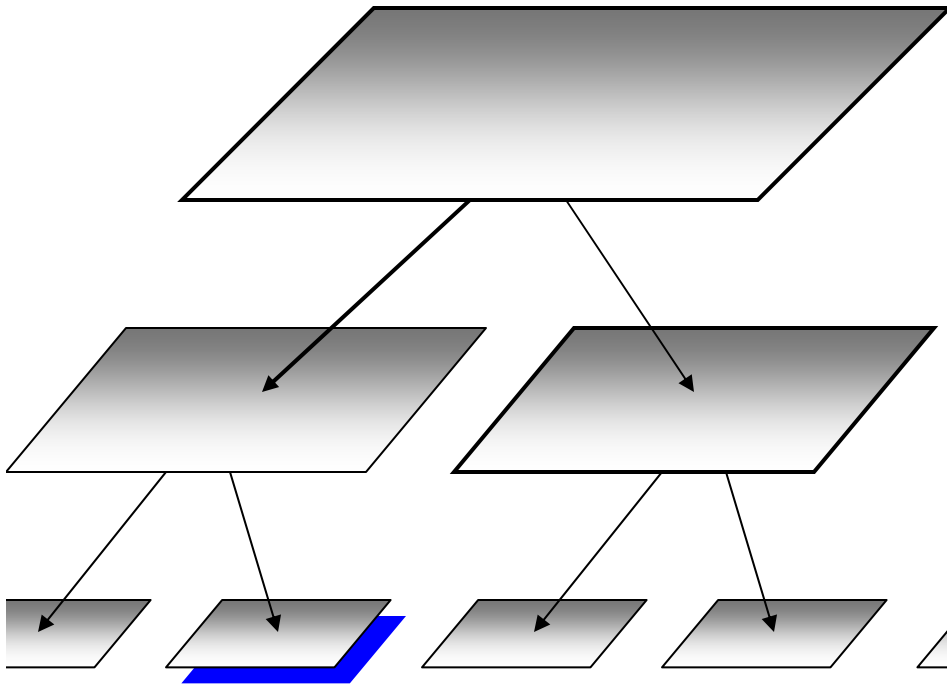


Reference points

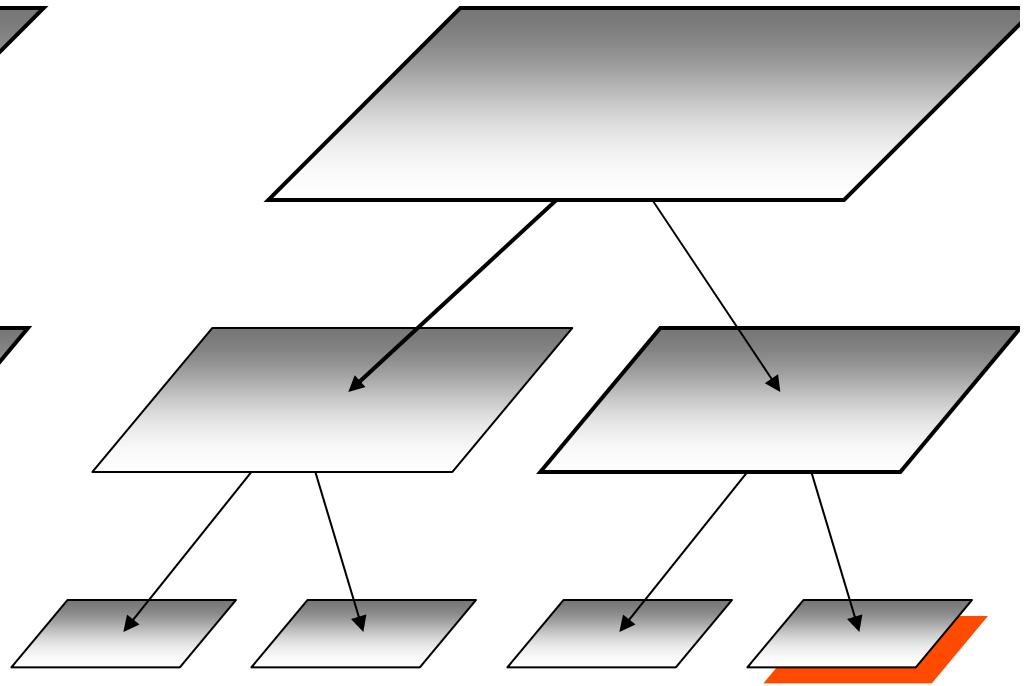


Dual-tree traversal

Query points

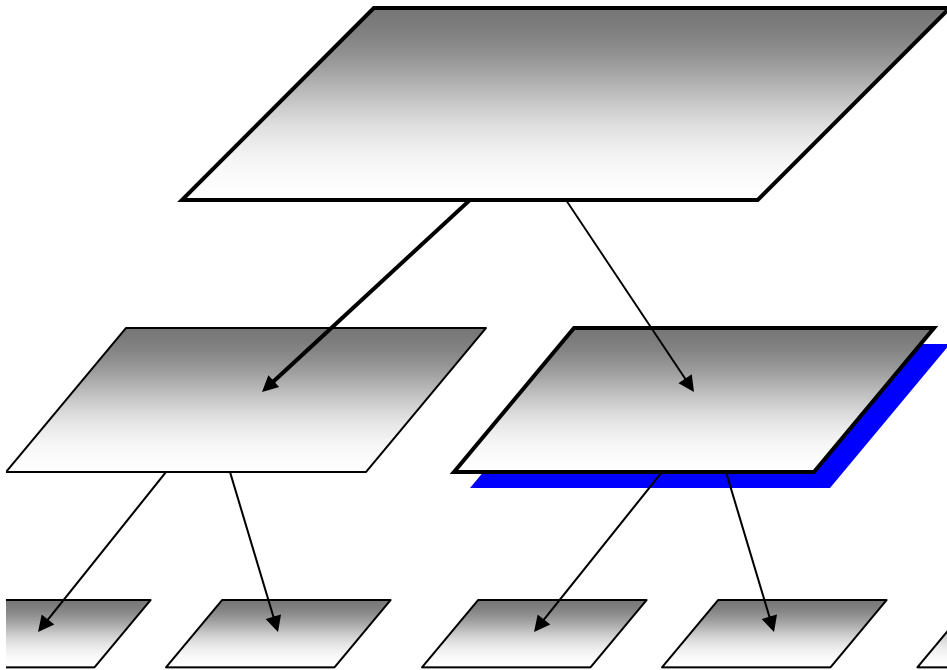


Reference points

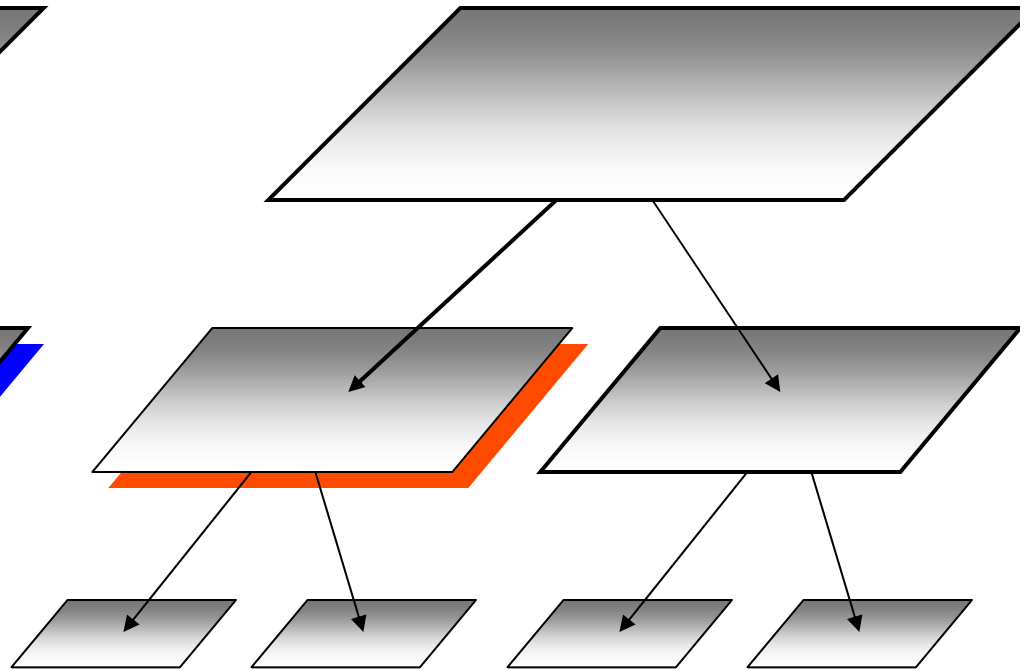


Dual-tree traversal

Query points

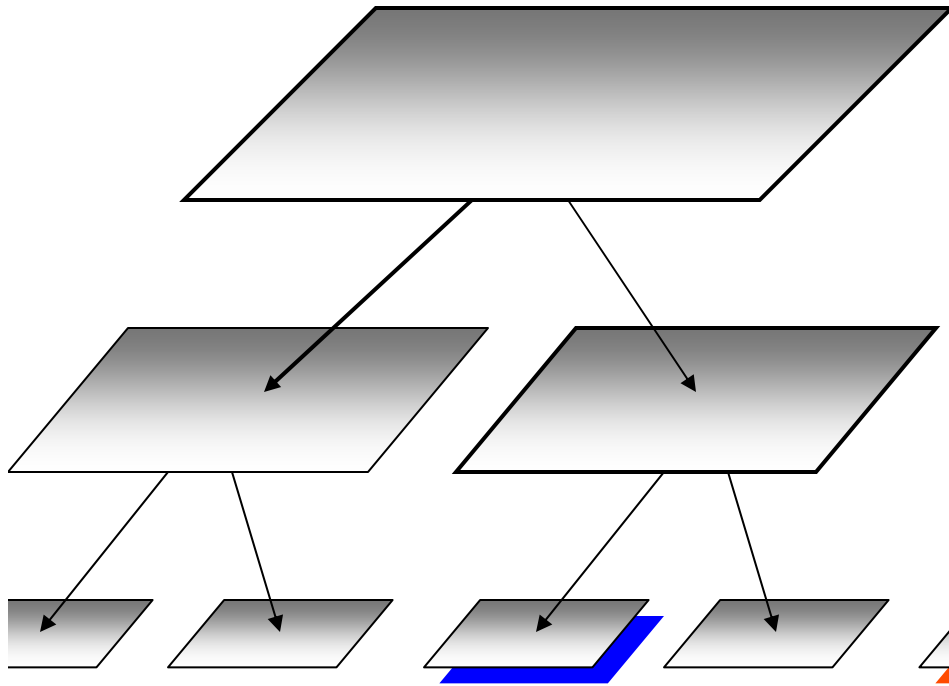


Reference points

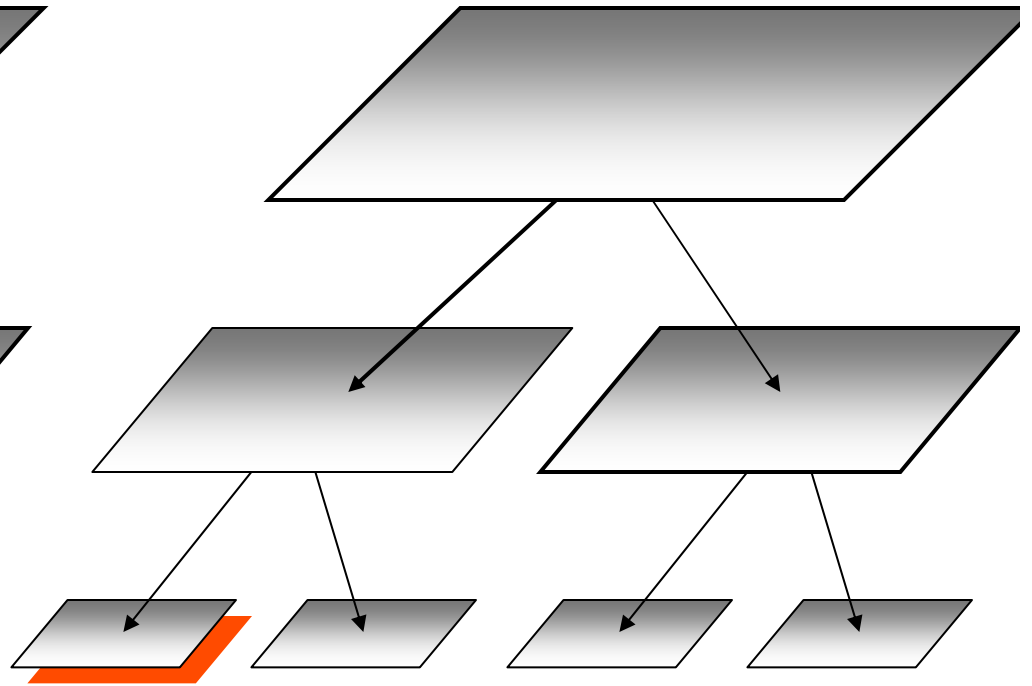


Dual-tree traversal

Query points

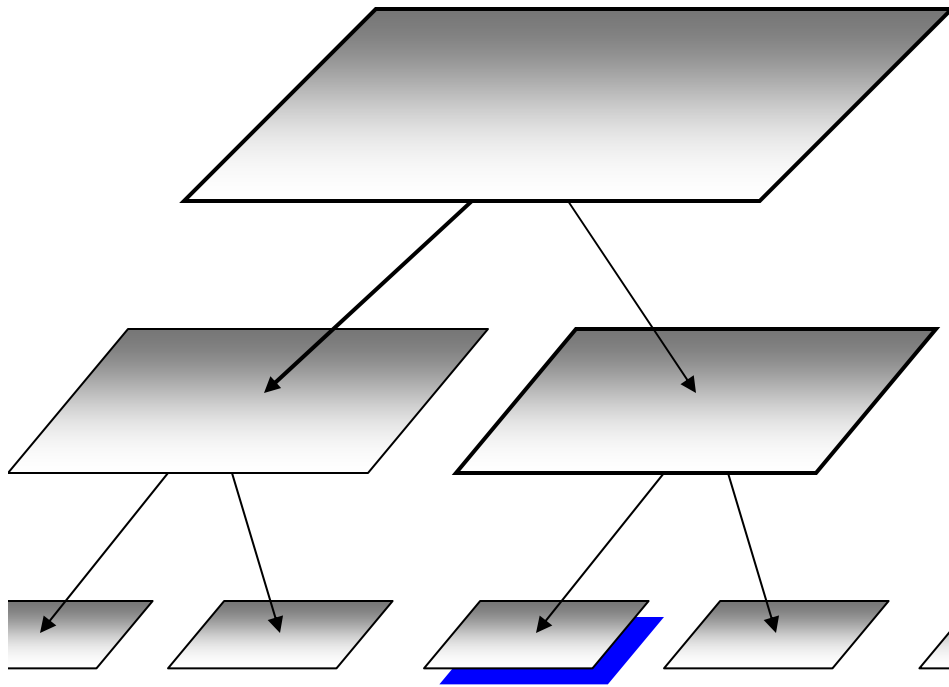


Reference points

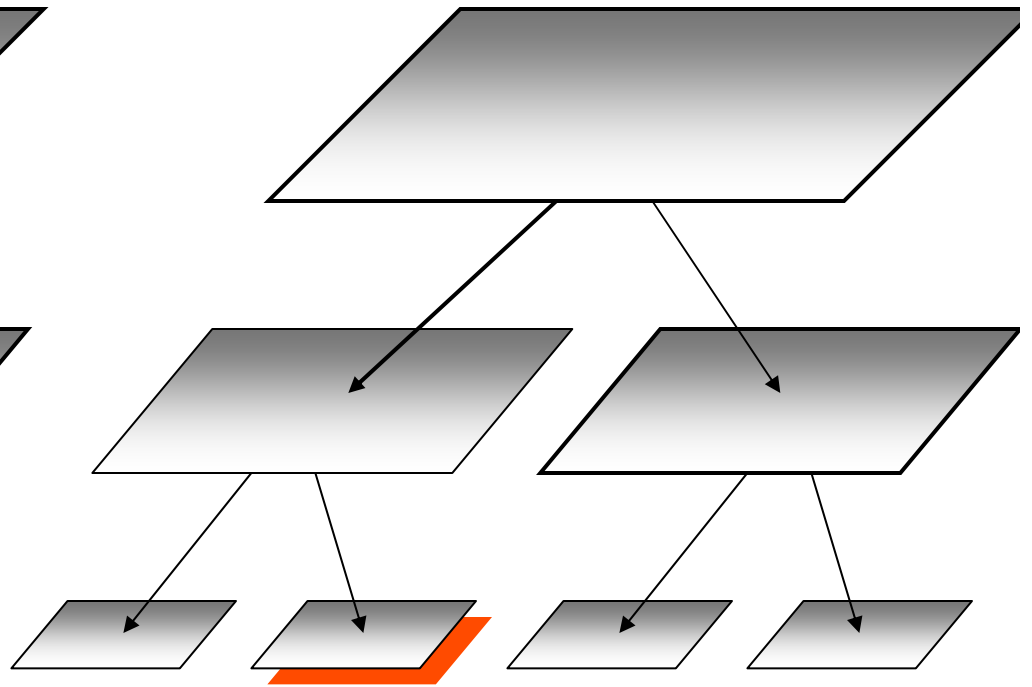


Dual-tree traversal

Query points

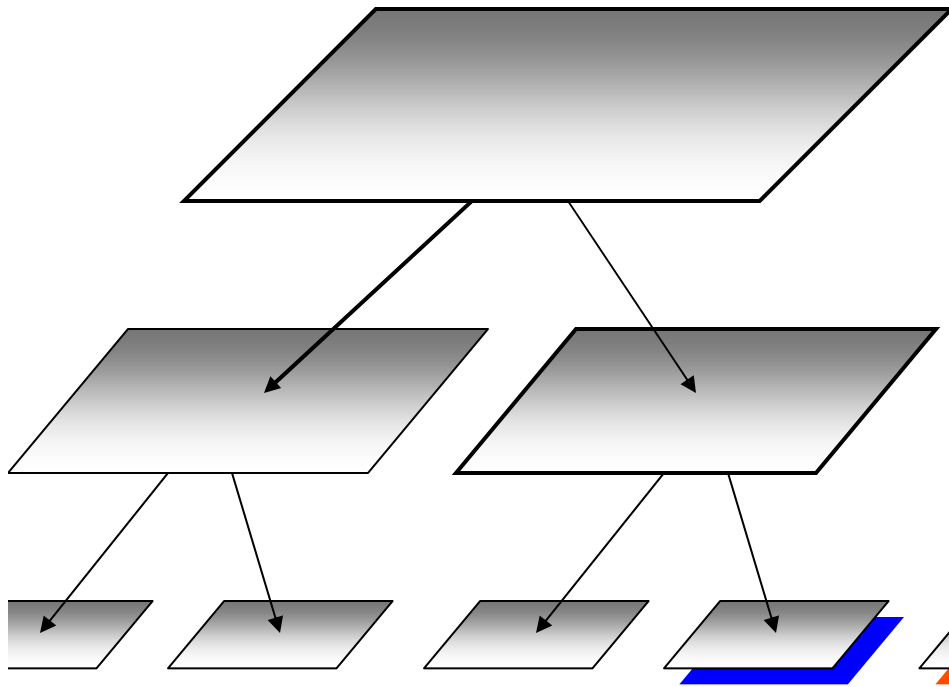


Reference points

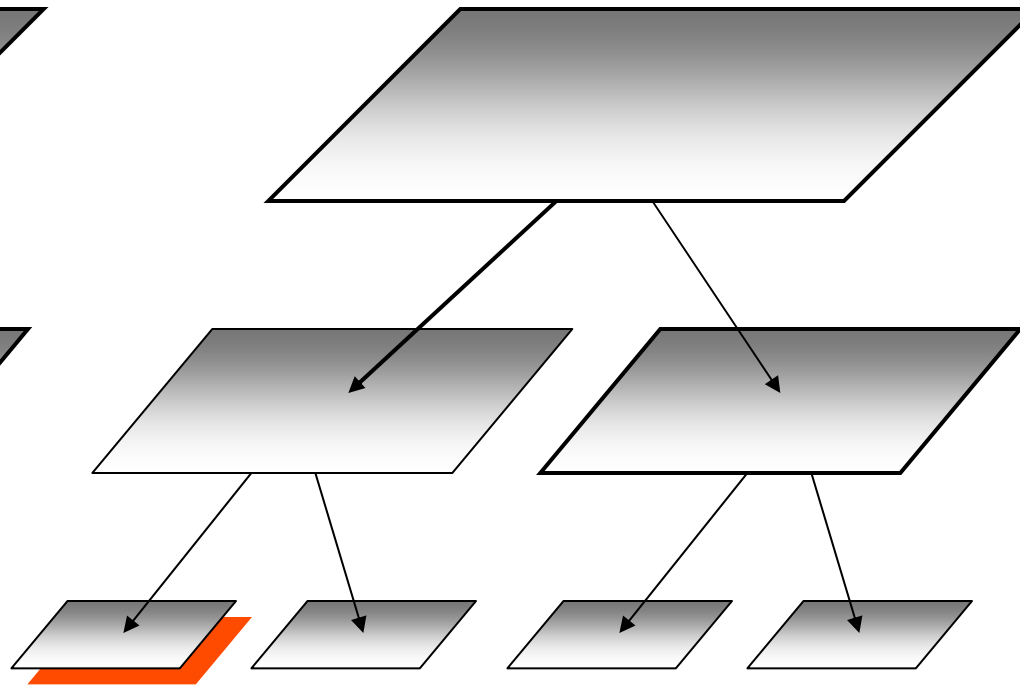


Dual-tree traversal

Query points

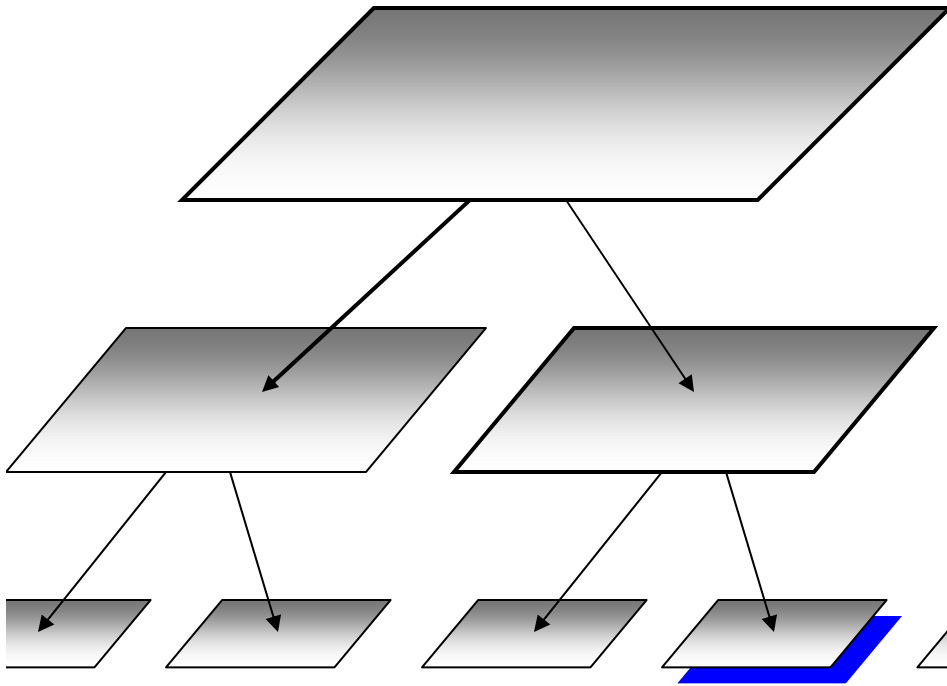


Reference points

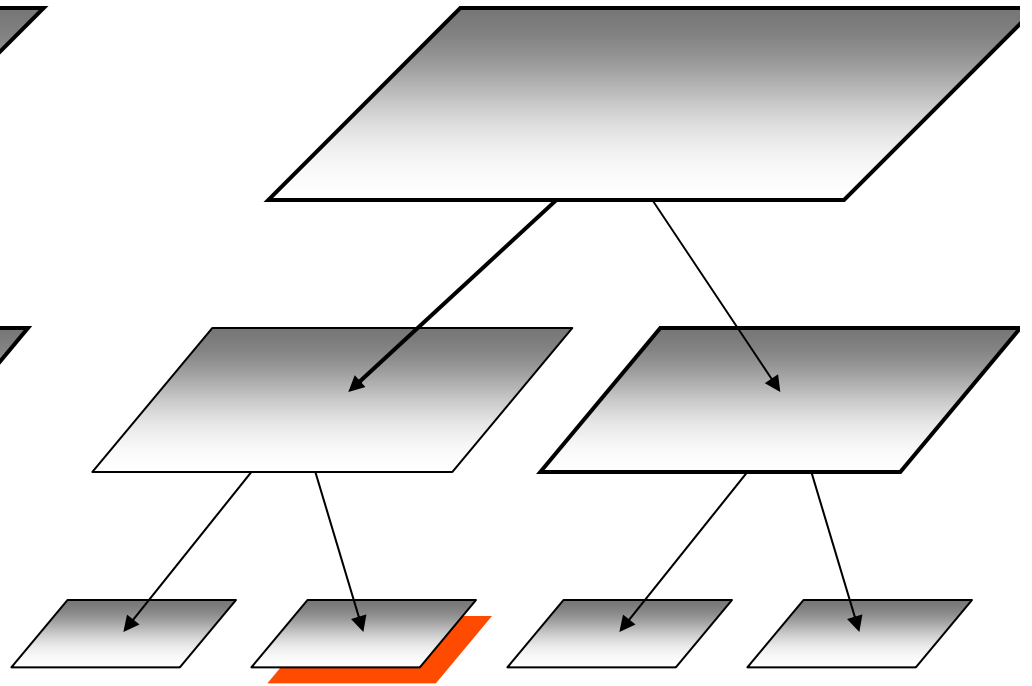


Dual-tree traversal

Query points

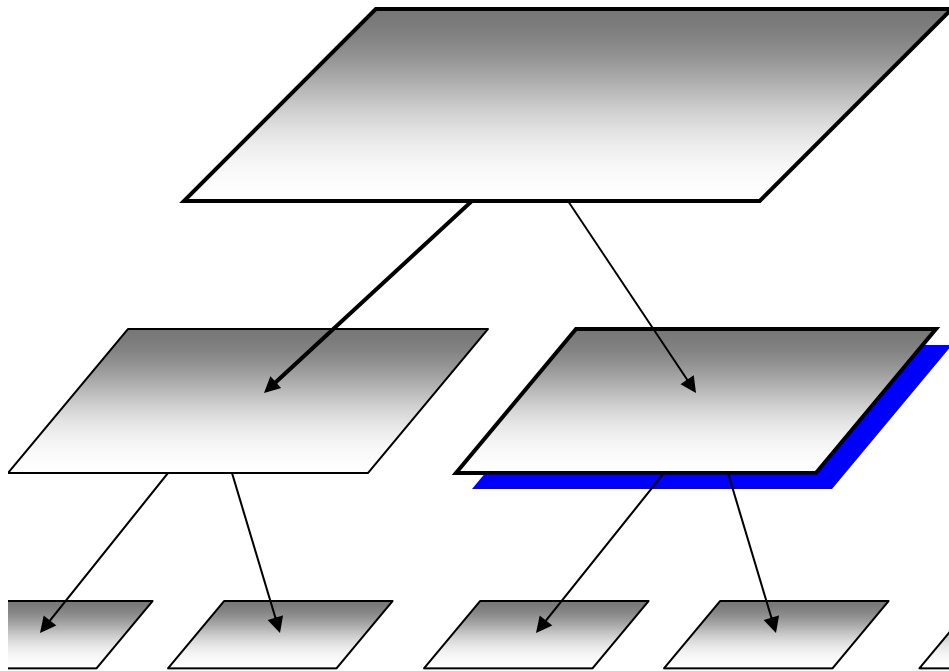


Reference points

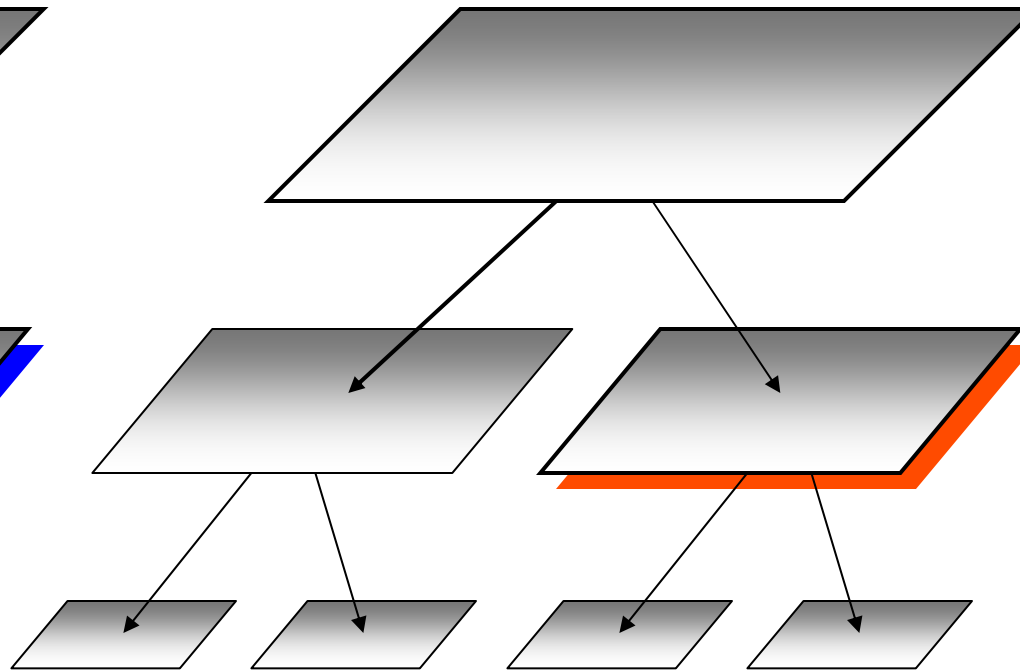


Dual-tree traversal

Query points

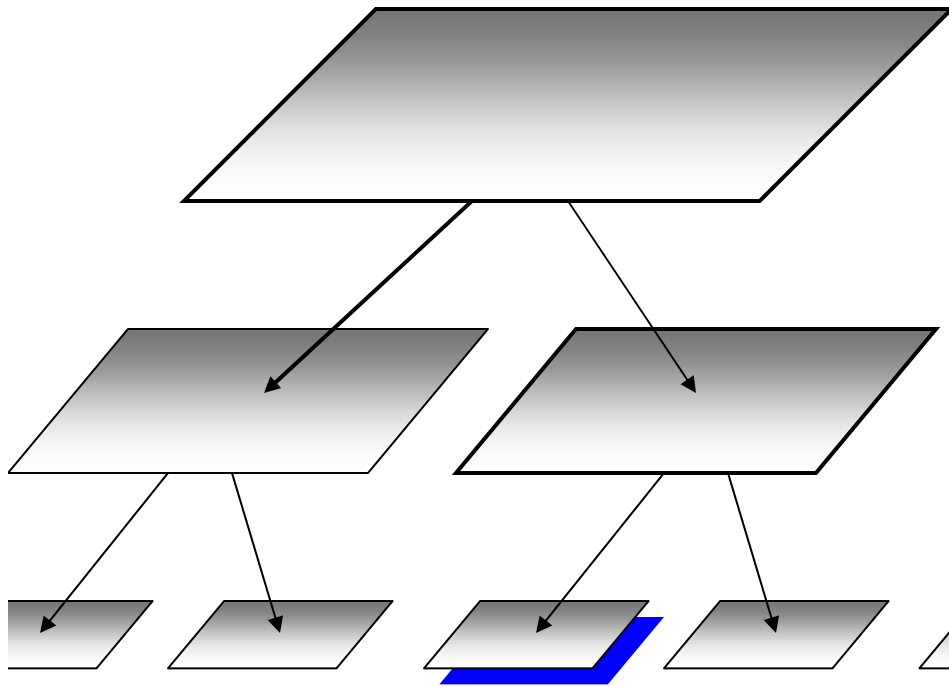


Reference points

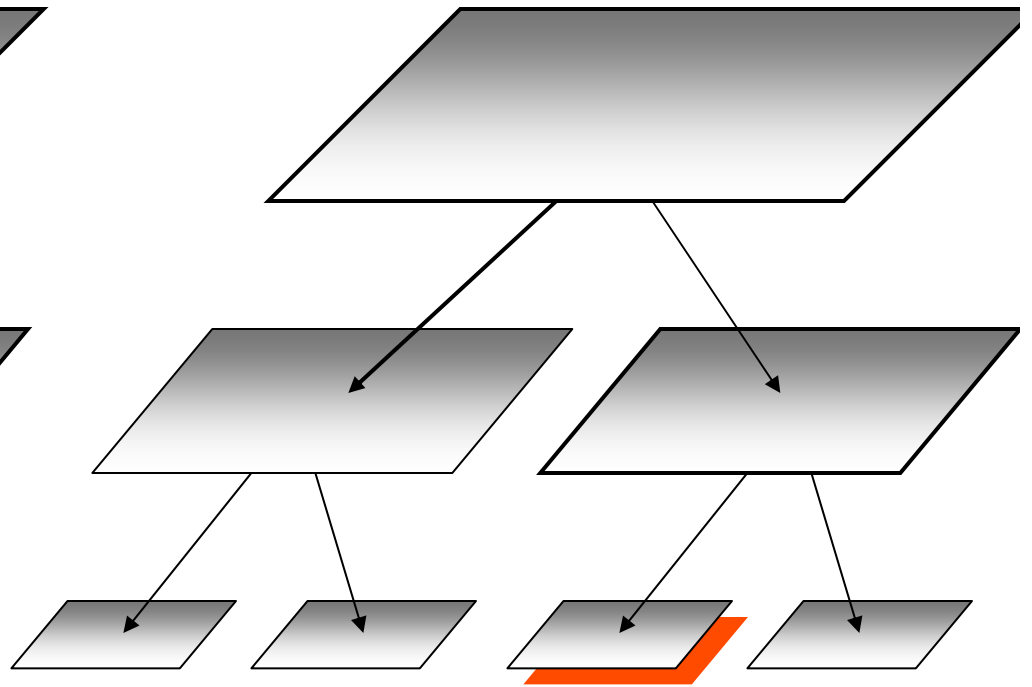


Dual-tree traversal

Query points

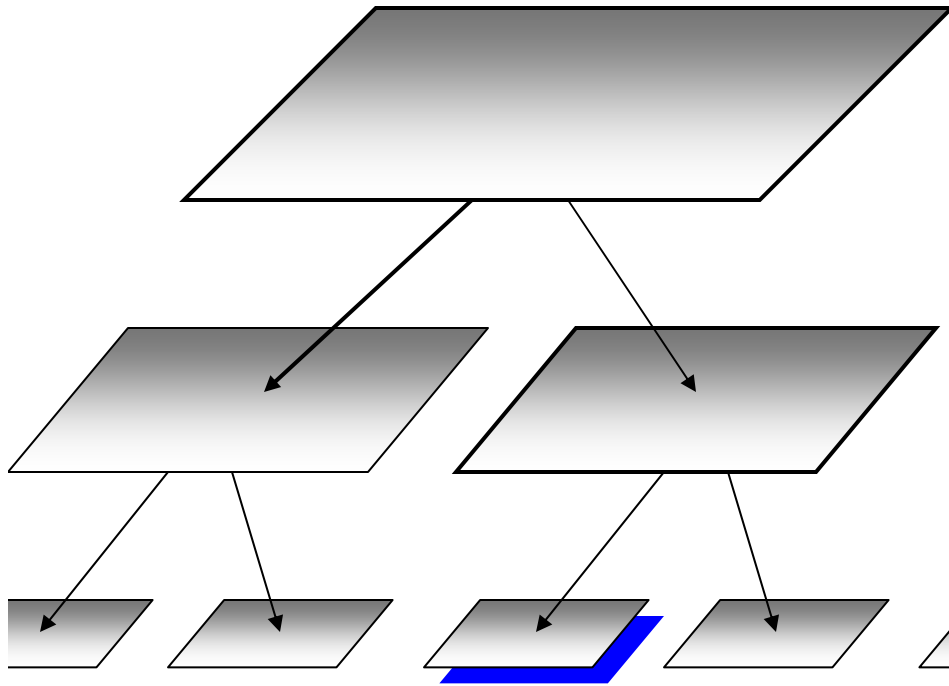


Reference points

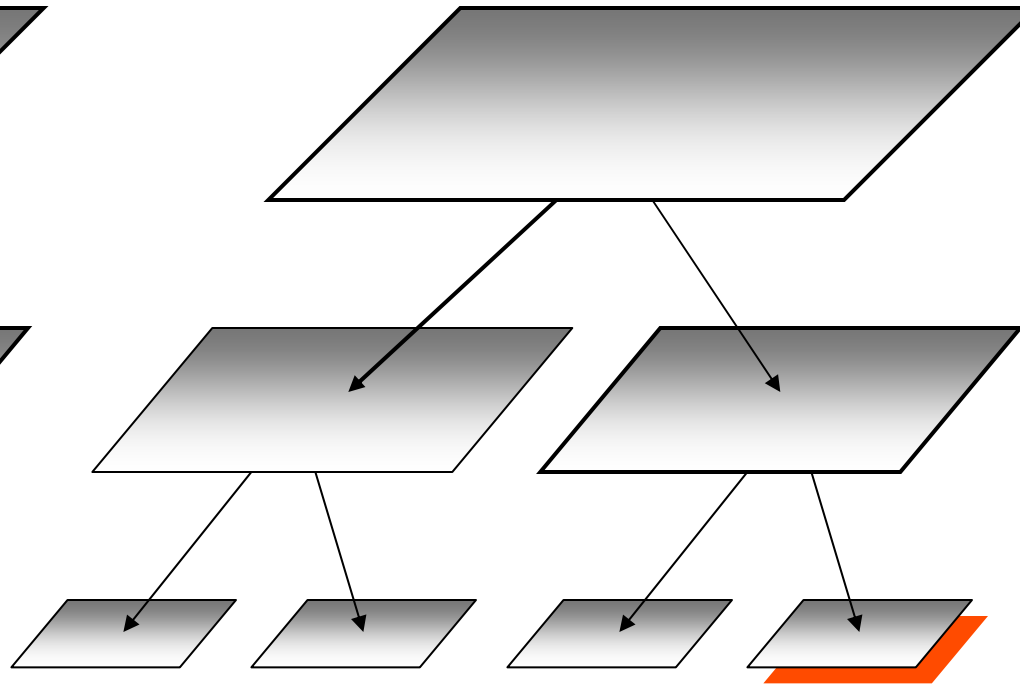


Dual-tree traversal

Query points

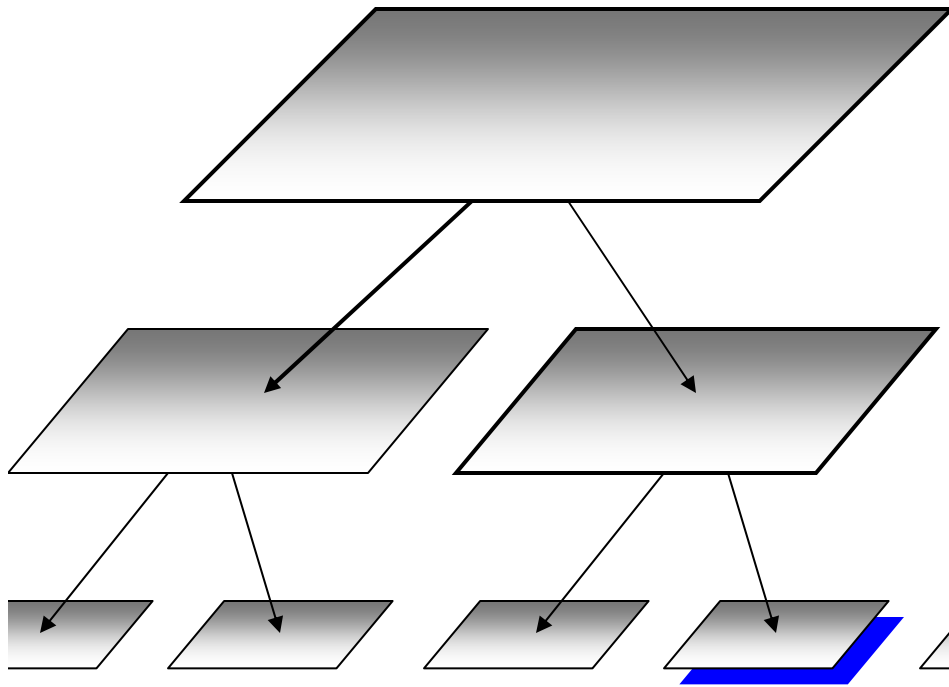


Reference points

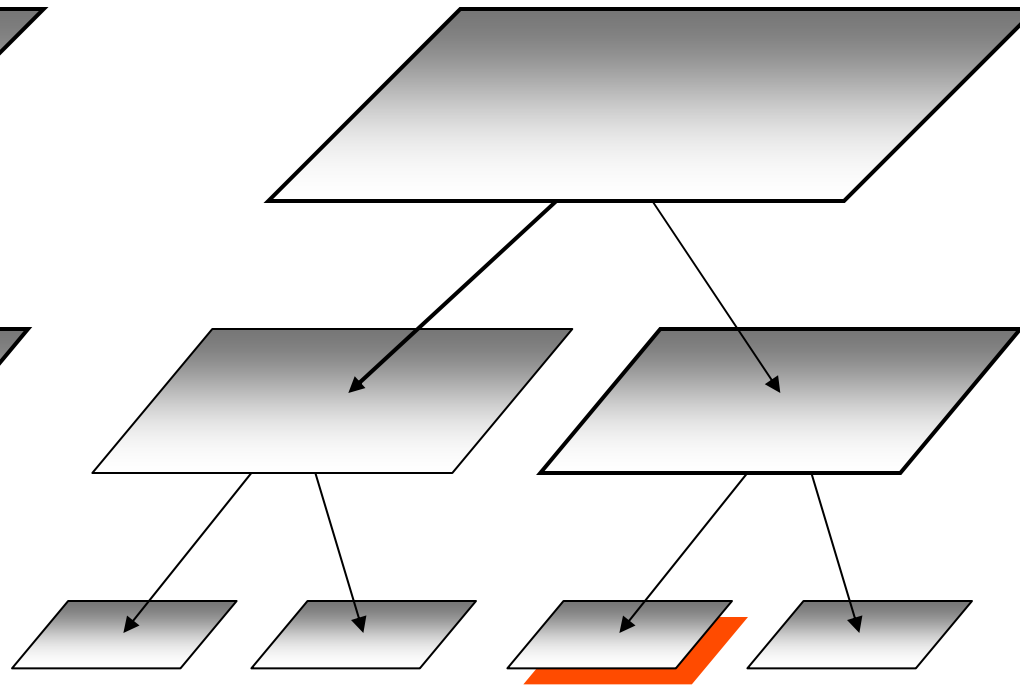


Dual-tree traversal

Query points

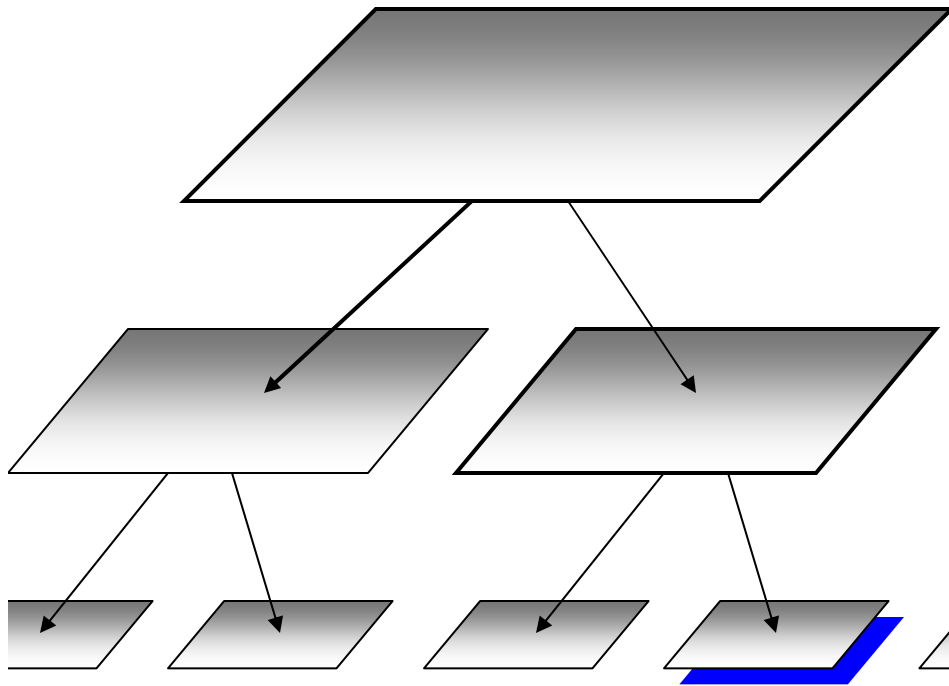


Reference points

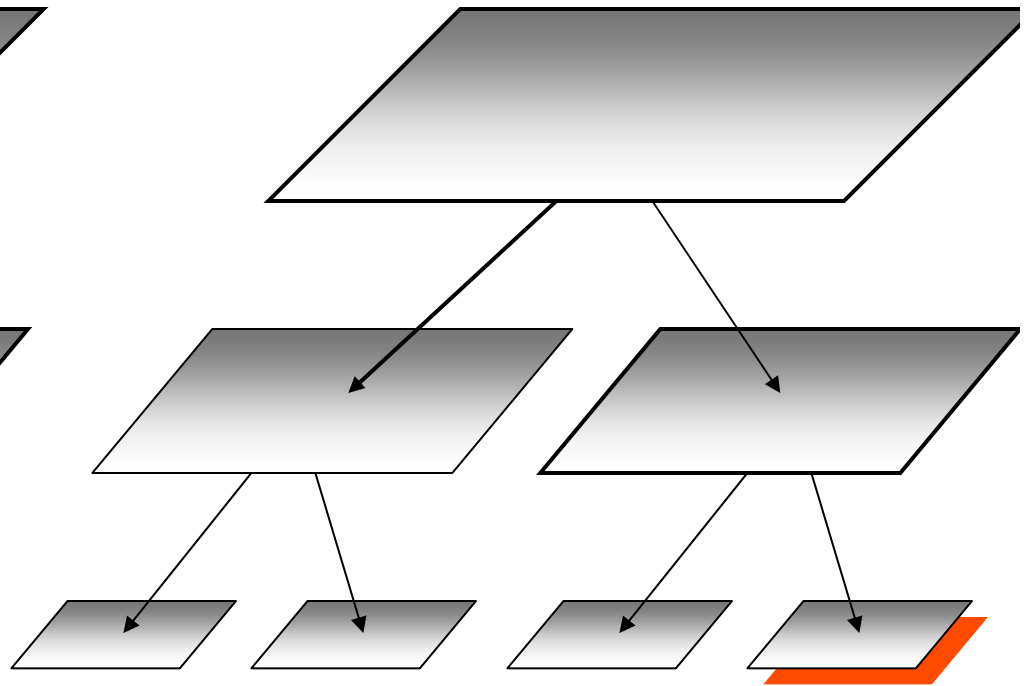


Dual-tree traversal

Query points



Reference points



Meta-idea: Higher-order Divide-and-conquer

Generalizes divide-and-conquer of a single set to divide-and-conquer of multiple sets.

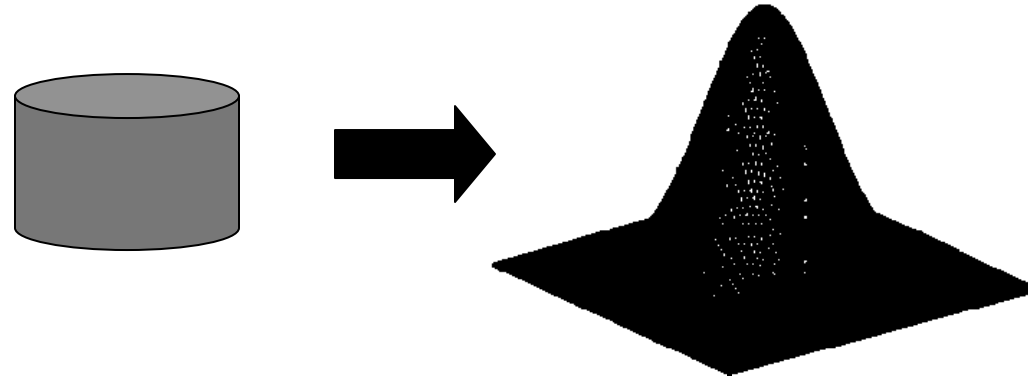
Break each set into pieces.

Solving the sub-parts of the problem and combining these sub-solutions appropriately might be easier than doing this over only one set.

Ideas

1. Data structures and how to use them
2. Monte Carlo
3. Multipole methods
4. Problem/solution abstractions

Kernel density estimation

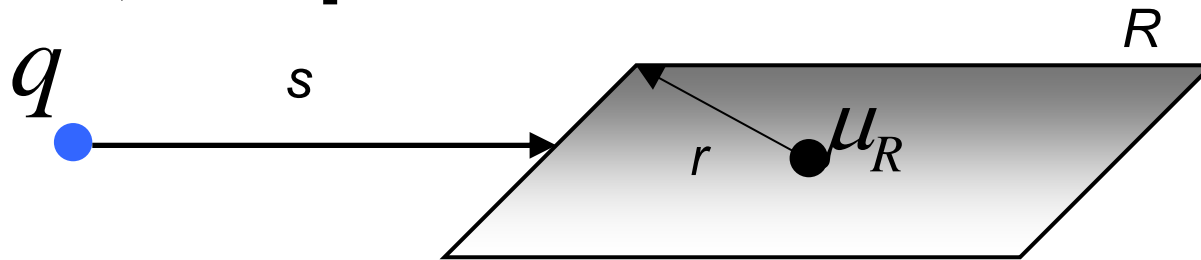


$$\forall x_q, \quad \hat{f}(x_q) = \frac{1}{N} \sum_{r \neq q}^N K_h(\|x_q - x_r\|)$$

How to use a tree...

1. How to approximate?
2. When to approximate?

[Barnes and Hut,
Science, 1987]



$$\sum_i K(q, x_i) \approx N_R K(q, \mu_R)$$

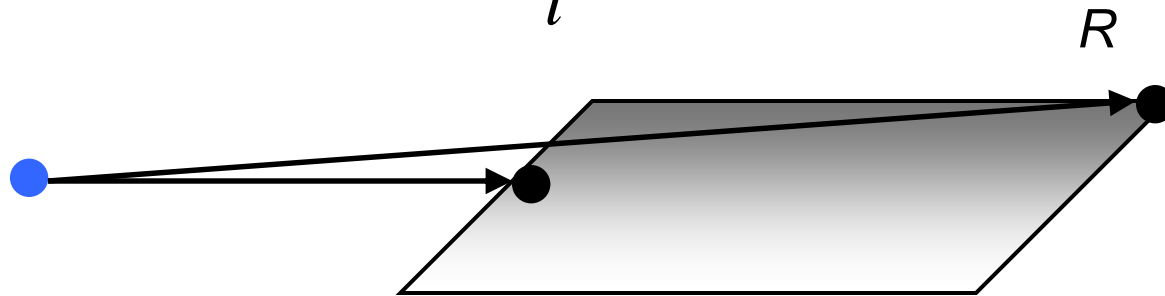
$$\text{if } s > \frac{r}{\theta}$$

How to use a tree...

3. How to know potential error?

Let's maintain bounds on the true kernel sum

$$\Phi(q) \equiv \sum_i K(q, x_i)$$



At the beginning:

$$\Phi^{lo}(q) \leftarrow NK^{lo}$$

$$\Phi^{hi}(q) \leftarrow NK^{hi}$$

$$\Phi^{lo}(q) \leftarrow \Phi^{lo}(q) + N_R K(q, \delta_{qR}^{lo}) - N_R K^{lo}$$

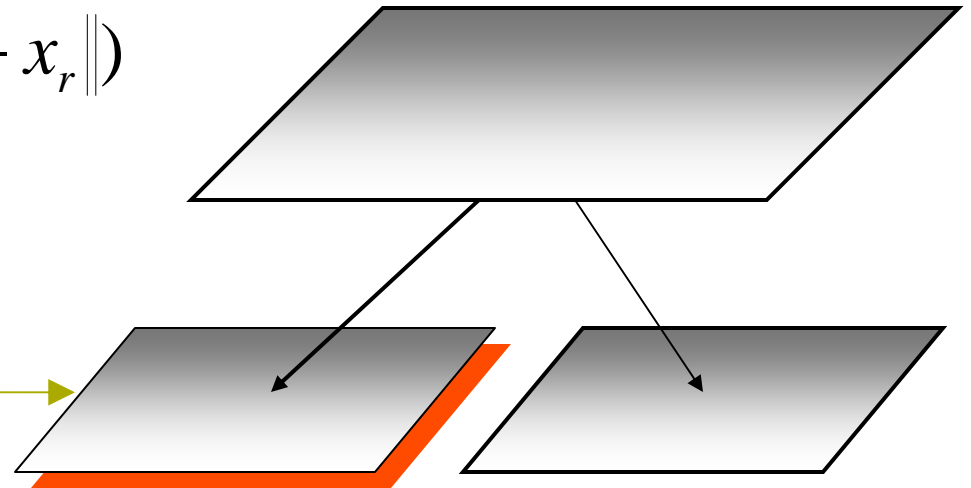
$$\Phi^{hi}(q) \leftarrow \Phi^{hi}(q) + N_R K(q, \delta_{qR}^{hi}) - N_R K^{hi}$$

How to use a tree...

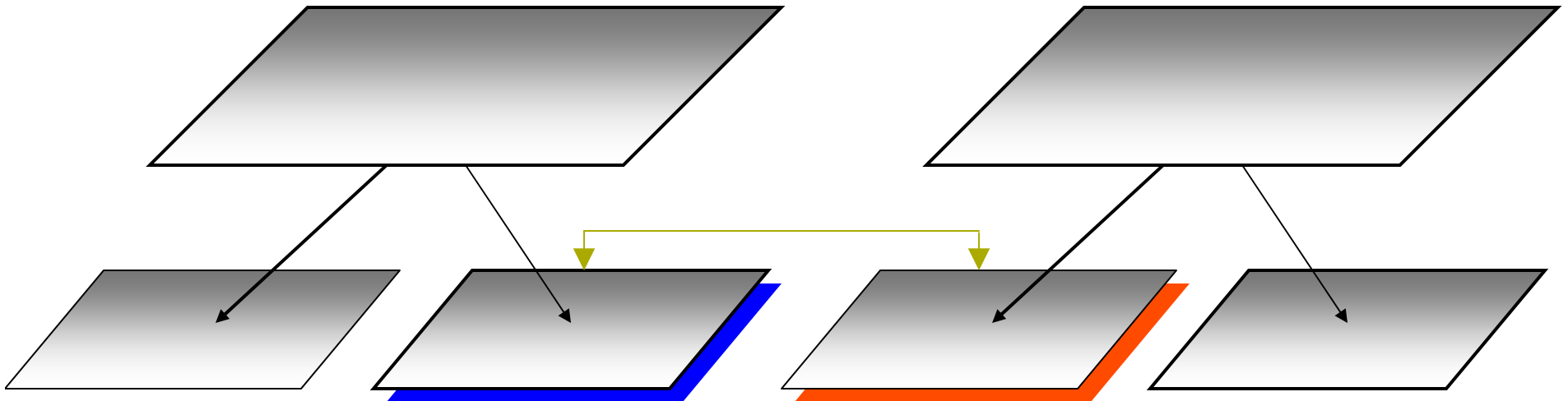
4. How to do 'all' problem?

$$\forall x_q, \hat{f}(x_q) = \frac{1}{N} \sum_{r \neq q}^N K_h(\|x_q - x_r\|)$$

Single-tree:

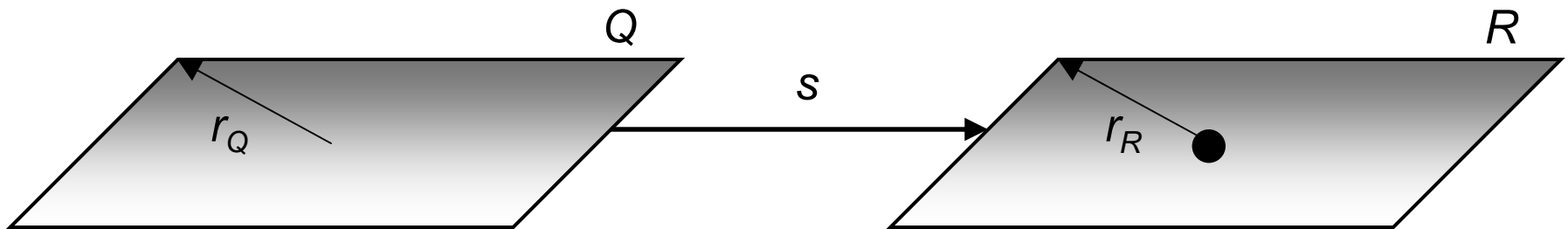


Dual-tree (symmetric): [Gray & Moore 2000]



How to use a tree...

4. How to do 'all' problem?



$$\forall q \in Q, \sum_i K(q, x_i) \approx N_R K(q, \mu_R)$$

$$\text{if } s > \frac{\max(r_Q, r_R)}{\theta}$$

Generalizes Barnes-Hut to dual-tree

BUT:

We have a tweak parameter: θ

Case 1 – alg. gives no error bounds

Case 2 – alg. gives error bounds, but must be rerun

Case 3 – alg. automatically achieves error tolerance

So far we have case 2;
let's try for case 3

Let's try to make an automatic stopping rule

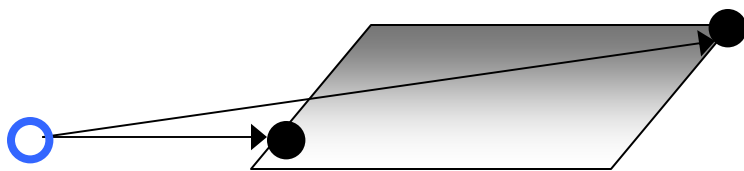
Finite-difference function approximation.

Taylor expansion:

$$f(x) \approx f(a) + f'(a)(x - a)$$

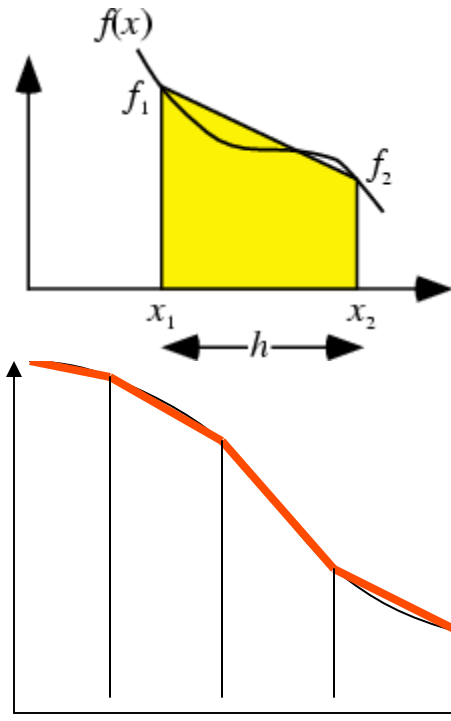
Gregory-Newton finite form:

$$f(x) \approx f(x_i) + \frac{1}{2} \left(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \right) (x - x_i)$$



$$K(\delta) \approx K(\delta^{lo}) + \frac{1}{2} \left(\frac{K(\delta^{hi}) - K(\delta^{lo})}{\delta^{hi} - \delta^{lo}} \right) (\delta - \delta^{lo})$$

Finite-difference function approximation.



assumes monotonic decreasing kernel

$$\bar{K} = \frac{1}{2} [K(\delta_{QR}^{lo}) + K(\delta_{QR}^{hi})]$$

$$err_q = \sum_r^{N_R} |K(\delta_{qr}) - \bar{K}| \leq \frac{N_R}{2} [K(\delta_{QR}^{lo}) - K(\delta_{QR}^{hi})]$$

$$\forall q, R: \frac{err_{qR}}{\phi(x_q)} \leq \frac{N_R}{N} \varepsilon \Rightarrow \forall q: \frac{err_q}{\phi(x_q)} \leq \varepsilon$$

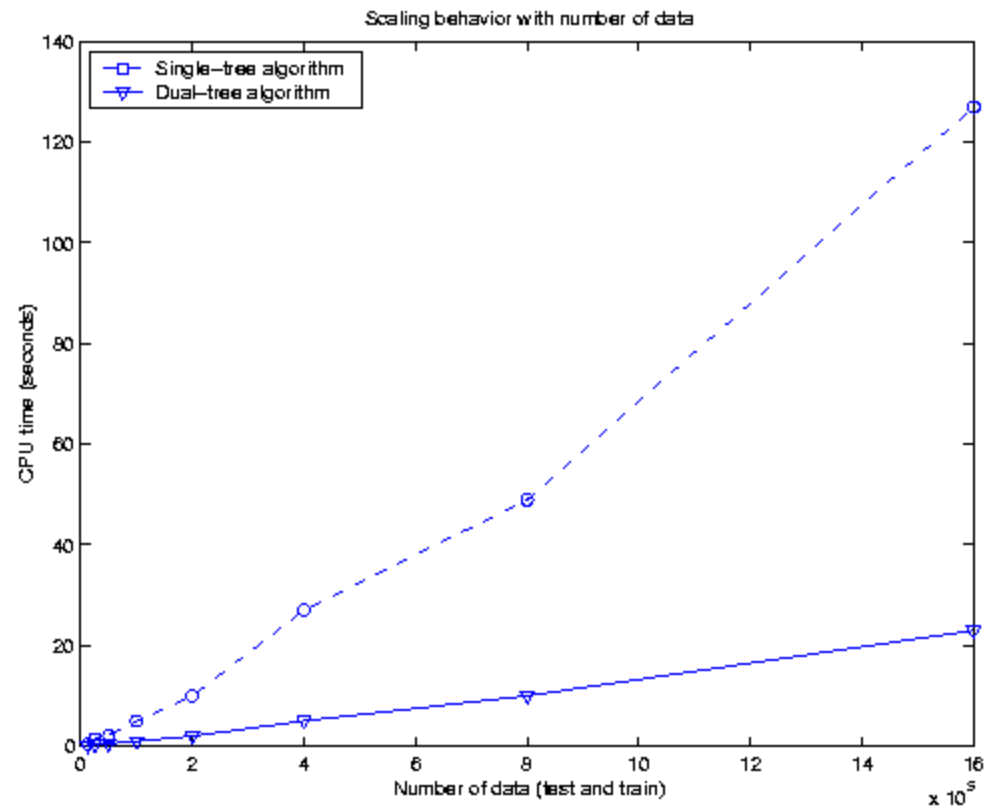
approximate $\{Q, R\}$ if

$$K(\delta_{lo}) - K(\delta_{hi}) \leq \frac{2\varepsilon}{N} \Phi_{lo}(Q)$$

Speedup Results (KDE)

| N | naïve | dual-tree |
|-------|--------|-----------|
| 12.5K | 7 | .12 |
| 25K | 31 | .31 |
| 50K | 123 | .46 |
| 100K | 494 | 1.0 |
| 200K | 1976* | 2 |
| 400K | 7904* | 5 |
| 800K | 31616* | 10 |
| 1.6M | 35 hrs | 23 |

5500x



One order-of-magnitude speedup over single-tree at ~2M points

N-Body Problems

- **Ubiquitous N-Body Problems**

- Astrophysics
- Molecular Dynamics
- Particle discretizations for PDEs
- Data Mining
- Irregular Sampling in Graphics
- etc.

- **Simple Observations**

- Points have no intrinsic topology
- Metric/Kernel relations matter $K(x,y)$
- N^2 interactions for all to all

- **Typical Problems**

- Nearest neighbors
- Weighted interpolation
- Partition of unity
- Kernel density

