Tutorial on **Statistical N-Body Problems** and **Proximity Data Structures**

Alexander Gray

School of Computer Science

Carnegie Mellon University

Outline:

1. **Physics problems and methods**
2. **Generalized N-body problems**
3. **Proximity data structures**
4. **Dual-tree algorithms**
5. **Comparison**

Outline:

1. **Physics problems and methods**

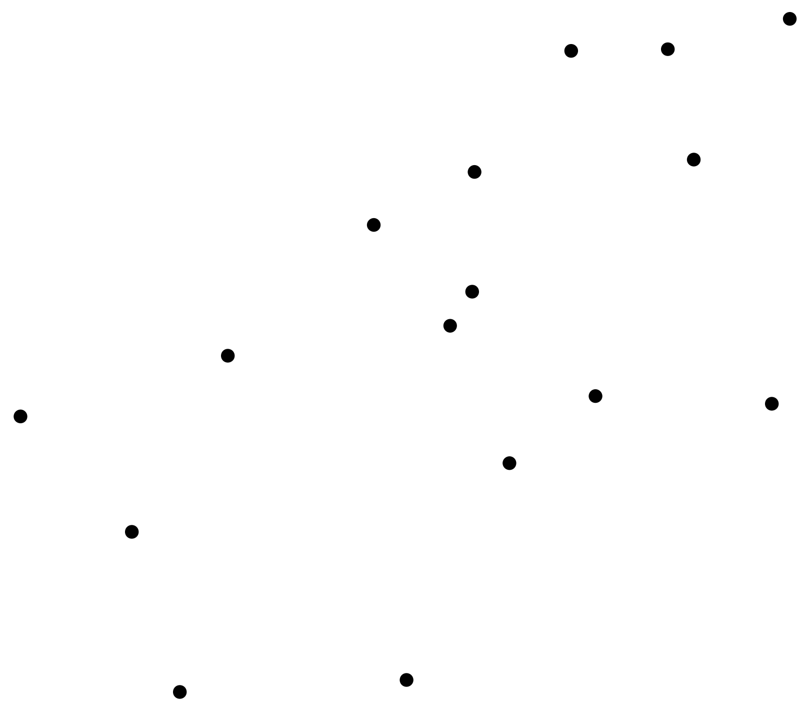
2. **Generalized N-body problems**

3. **Proximity data structures**

4. **Dual-tree algorithms**

5. **Comparison**

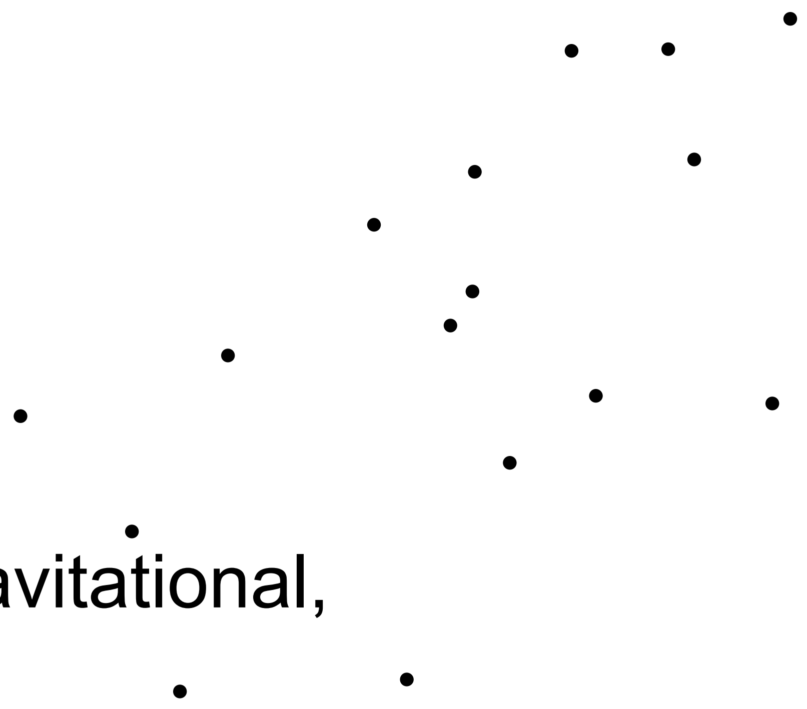
‘N-body problem’ of physics



‘N-body problem’ of physics

Compute:

$$\boxed{\forall i, \sum_{j \neq i}^{N} K(x_i, x_j)}$$



Simulation (electrostatic, gravitational, statistical mechanics):

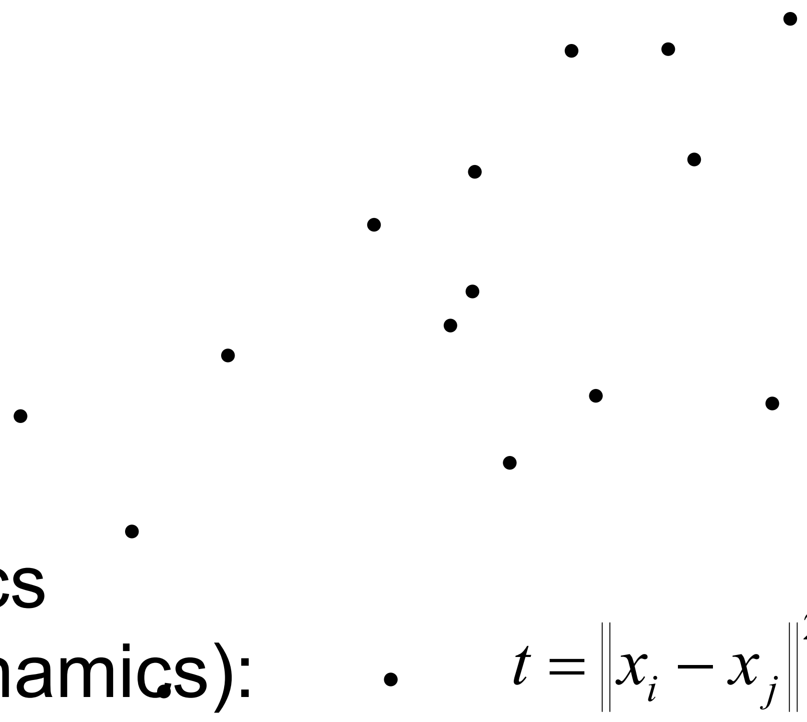
$$K(x_i, x_j) \propto \frac{m_i m_j}{\left\| x_i - x_j \right\|^a}$$

Some problems: Gaussian kernel

‘N-body problem’ of physics

Compute:

$$\boxed{\forall i, \sum_{j \neq i}^{N} K(x_i, x_j)}$$



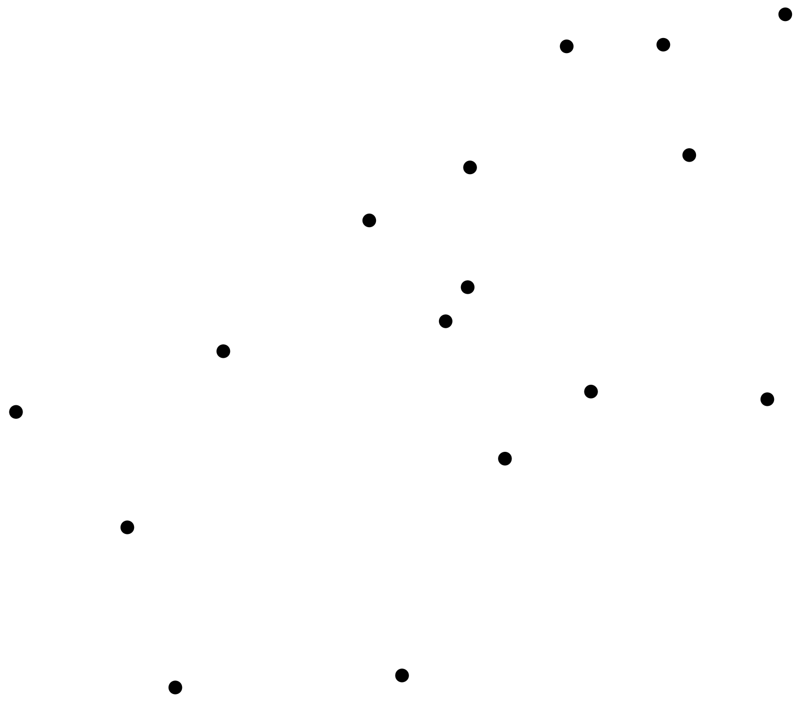
Computational fluid dynamics
(smoothed particle hydrodynamics):

$$t = \left\| x_i - x_j \right\|^2 / h^2$$

more complicated:
nonstationary,
anisotropic,
edge-dependent (Gray thesis 2003)

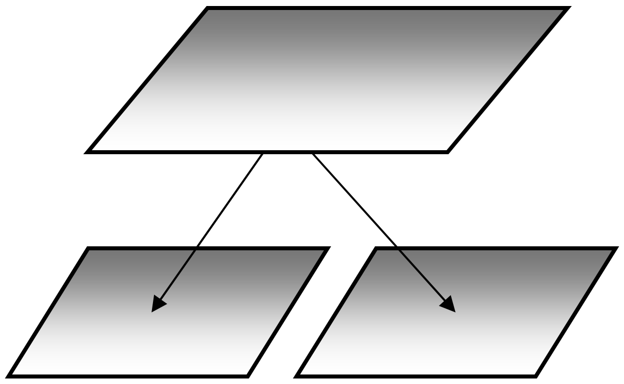
$$K(x, x_i) = \begin{matrix} 4 - 6t^2 + 3t^3 & 0 \le t < 1 \\ (2-t)^3 & 1 \le t < 2 \\ 0 & t \ge 2 \end{matrix}$$

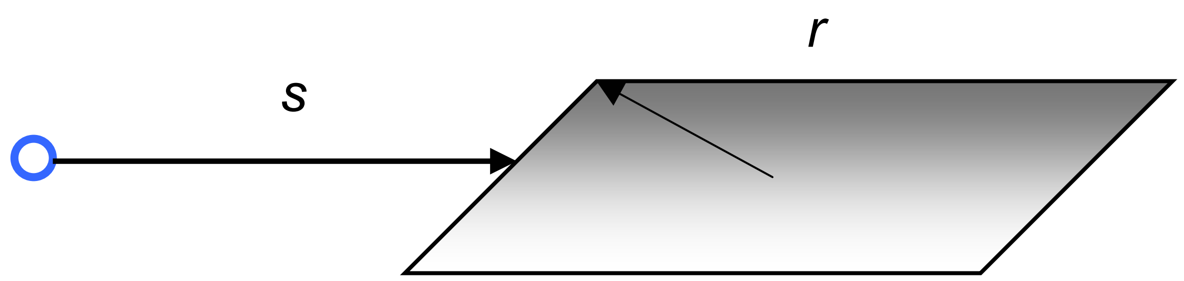
‘N-body problem’ of physics



➡ **Main obstacle:** $O(N^2)$

Barnes-Hut Algorithm
[Barnes and Hut, 87]

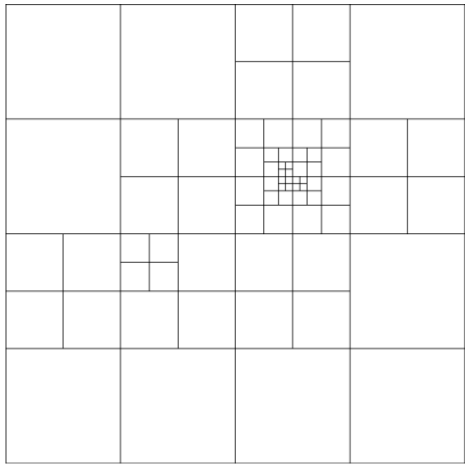




$$\sum_i K(x, x_i) \approx N_R K(x, \mu_R) \quad \text{if} \quad s > \frac{r}{\theta}$$

Fast Multipole Method [Greengard and Rokhlin 1987]

Quadtree/octree:



$\forall x, \sum_i K(x, x_i) \approx$ multipole/Taylor expansion of order *p* if $s > r$

For Gaussian kernel: “Fast Gauss Transform” [Greengard and Strain 91]

N-body methods: Runtime

- **Barnes-Hut** $\approx O(N \log N)$

  non-rigorous, $\approx$ uniform distribution

- **FMM** $\approx O(N)$

  non-rigorous, $\approx$ uniform distribution

N-body methods: Runtime

- **Barnes-Hut** $\approx O(N \log N)$

  non-rigorous, $\approx$ uniform distribution

- **FMM** $\approx O(N)$

  non-rigorous, $\approx$ uniform distribution

[Callahan-Kosaraju 95]: *O(N)* is impossible
for log-depth tree

In practice…

Both are used

Often Barnes-Hut is chosen for several reasons…

Expansions

- *Constants matter!* p^D factor is slowdown
- Adds much complexity (software, human time)
- Non-trivial to do new kernels (assuming they're even analytic), heterogeneous kernels
- Well-known papers in computational physics:
  - "Implementing the FMM in 3 Dimensions", J.Stat.Phys. 1991
  - "A New Error Estimate for the Fast Gauss Transform", J.Sci.Comput. 2002
  - "An Implementation of the FMM Without Multipoles", SIAM J.Sci.Stat.Comput. 1992

N-body methods: Adaptivity

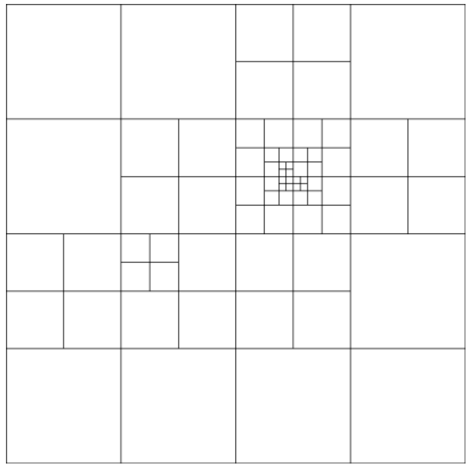
- **Barnes-Hut** **recursive**

  → can use any kind of tree

- **FMM** **hand-organized control flow**

  → requires grid structure

quad-tree/oct-tree	not very adaptive
kd-tree	adaptive
ball-tree/metric tree	very adaptive



N-body methods: Comparison

	Barnes-Hut	**FMM**
runtime	$O(N\log N)$	$O(N)$
expansions	optional	required
simple,recursive?	yes	no
adaptive trees?	yes	no
error bounds?	no	yes

Outline:

1. Physics problems and methods

2. **Generalized N-body problems**

3. Proximity data structures

4. Dual-tree algorithms

5. Comparison

N-body problems in statistical learning

[Gray and Moore, NIPS 2000]
[Gray PhD thesis 2003]

Obvious N-body problems:

- Kernel density estimation (Gray & Moore 2000, 2003abc)
- Kernel regression:
    - Locally-weighted regression
    - Nadaraya-Watson regression (Gray 2005, next talk)
- Gaussian process regression (Gray CMU-TR 2003)
- RBF networks
- Kernel machines
- Nonparametric Bayes classifiers (Gray et al. 2005)

N-body problems in statistical learning

Typical kernels: Gaussian, Epanechnikov (optimal):

$$K(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2h^2}$$

$$t = \|x_i - x_j\|^2 / h^2 \qquad K(x_i, x_j) = \begin{matrix} 1 - t^{2a} & 0 \le t < 1 \\ 0 & t \ge 1 \end{matrix}$$

N-body problems in statistical learning

[Gray and Moore, NIPS 2000]
[Gray PhD thesis 2003]

Less obvious N-body problems:

- n-point correlation (Gray & Moore 2000, 2004, Gray et al. 2005)
- Fractal/intrinsic dimension (Gray 2005)
- All-nearest-neighbors, bichromatic (Gray & Moore 2000, Gray, Lee, Rotella & Moore 2005)

Kernel density estimation

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

• The optimal smoothing parameter h is all-important
• Guaranteed to converge to the true underlying density (consistency)
• Nonparametric – distribution need only meet some weak smoothness conditions
• Optimal kernel doesn’t happen to be the Gaussian

Kernel density estimation

KDE: Compute $\boxed{\forall i, \sum_{j \neq i}^{N} K(x_i, x_j)}$

Abstract problem: $\{\forall, \Sigma, K_r(\delta), \cdot; \{r\}, 2\}$

Operator 1

Operator 2

Kernel function

Monadic function

Multiplicity

Chromatic number

All-nearest-neighbors (bichromatic, k)

All-NN: Compute $\boxed{\forall i, \arg\min^{k}{}_{j} \left\| x_i - x_j \right\|}$

Abstract problem: $\{\forall, \arg\min, \delta, \cdot; \cdot; 2\}$

Operator 1

Operator 2

Kernel function

Monadic function

Multiplicity

Chromatic number

These are examples of…

Generalized N-body problems

All-NN: $\{\forall, \arg\min, \delta, \cdot\}$

2-point: $\{\Sigma, \Sigma, I_r(\delta), w\}$

3-point: $\{\Sigma, \Sigma, \Sigma, I_R(\delta), w\}$

KDE: $\{\forall, \Sigma, K_r(\delta), \cdot; \{r\}\}$

SPH: $\{\forall, \Sigma, K_r(\delta), w; t\}$

etc.

[Gray PhD thesis 2003]

Physical simulation

- High accuracy required. (e.g. 6-12 digits)
- Dimension is 1-3.
- Most problems are covered by Coulombic kernel function.

Statistics/learning

- Accuracy on order of prediction accuracy required.
- Often high dimensionality.
- Test points can be different from training points.

FFT

- Approximate points by nearby grid locations
- Use M grid points in each dimension
- Multidimensional FFT: O($(M\log M)^D$)

Fast Gauss Transform
[Greengard and Strain 89, 91]

- Same series-expansion idea as FMM, but with Gaussian kernel
- However: no data structure
- Designed for low-D setting (borrowed from physics)

- "Improved FGT" [Yang, Duraiswami 03]:
  - appoximation is $O(D^p)$ instead of $O(p^D)$
  - also ignore Gaussian tails beyond a threshold
  - choose $K<\sqrt{N}$, find K clusters; compare each cluster to each other: $O(K^2)=O(N)$
  - not a tree, just a set of clusters

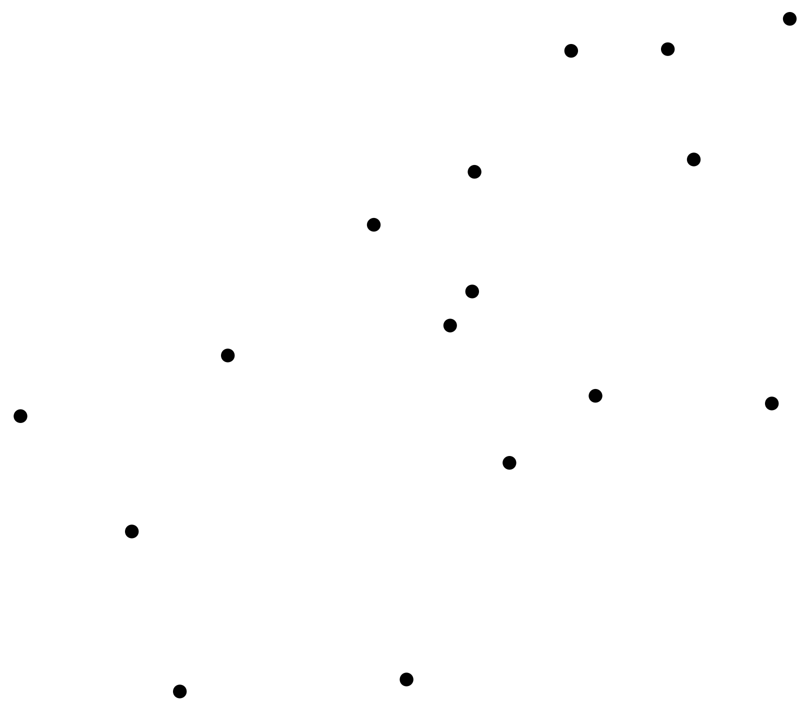
Observations

- FFT: Designed for 1-D signals (borrowed from signal processing). Considered state-of-the-art in statistics.

- FGT: Designed for low-D setting (borrowed from physics). Considered state-of-the-art in computer vision.

Runtime of both depends explicitly on D.

Observations



Data in high D basically always lie on manifold of (much) lower dimension, D'.

Degenerate N-body problems

Nearest neighbor: $\{\cdot, \arg\min, \delta, \cdot, \cdot, 1\}$

$$\boxed{\arg\min{}^{k}{}_{j} \left\| x - x_j \right\|}$$

Range-search (radial): $\{\cdot, \Sigma, I_r(\delta), \cdot, \cdot, 1\}$

$$\boxed{\sum_{j \neq i}^{N} I\left(\left\| x - x_j \right\| < r \right)}$$

How are these problems solved?

Outline:

1. Physics problems and methods

2. Generalized N-body problems

3. **Proximity data structures**

4. Dual-tree algorithms

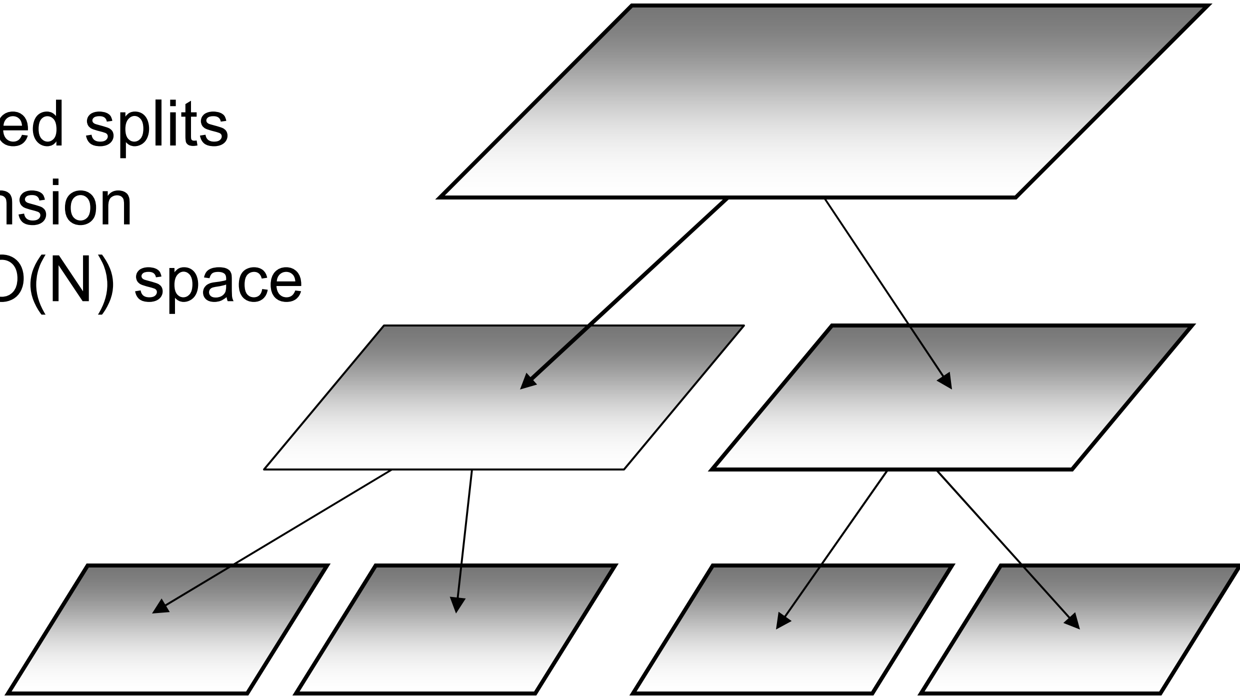
5. Comparison

kd-trees:

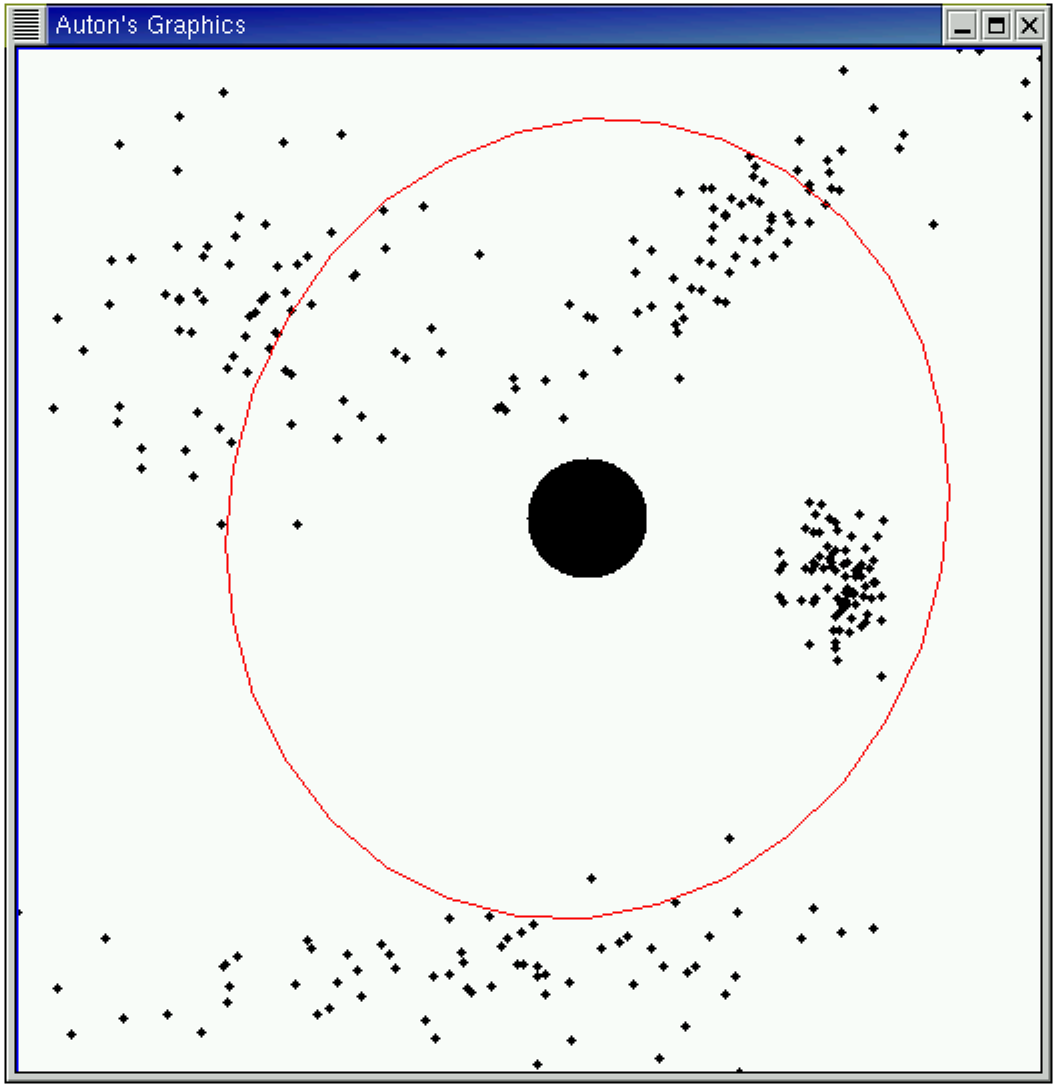
most widely-used space-partitioning tree

[Friedman, Bentley & Finkel 1977]

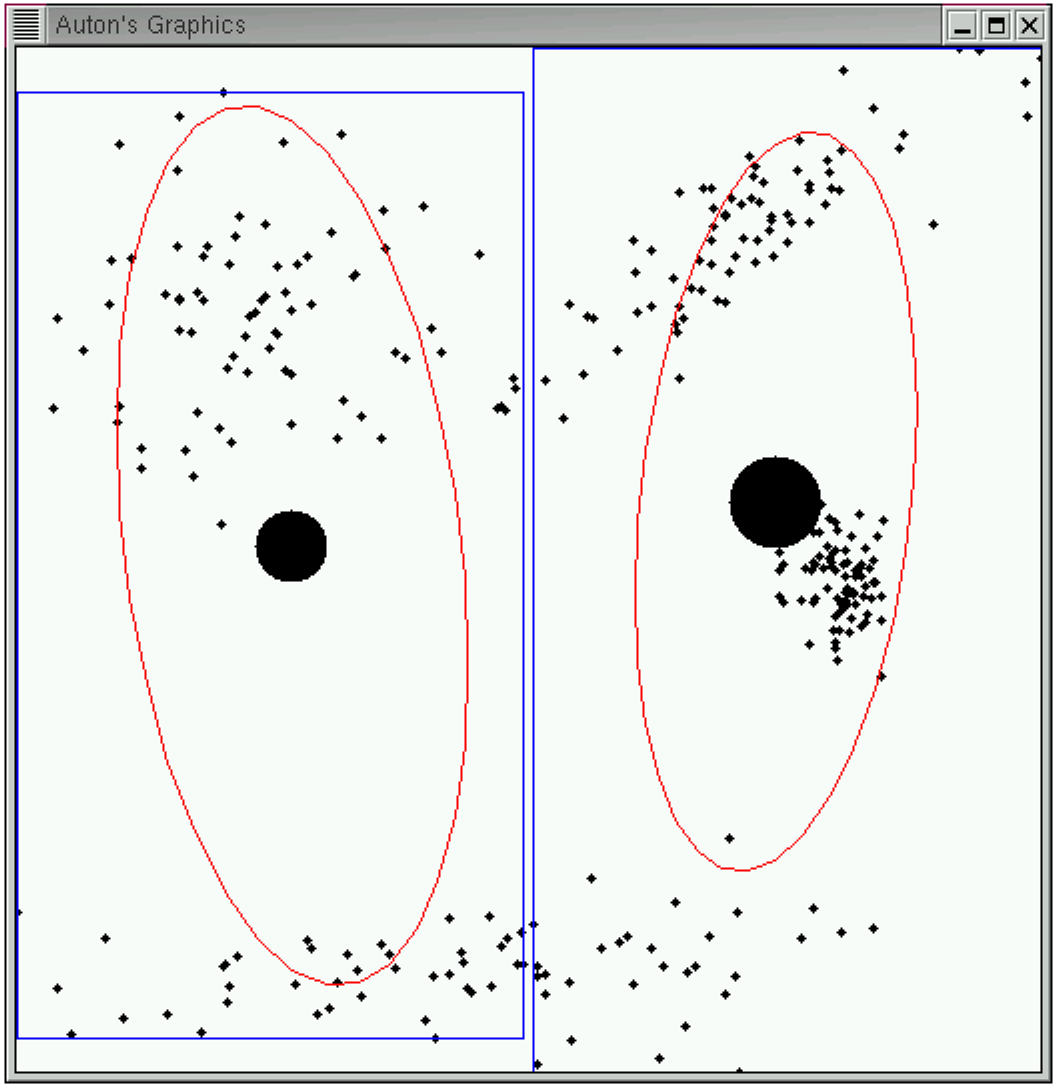
- Univariate axis-aligned splits
- Split on widest dimension
- O(N log N) to build, O(N) space



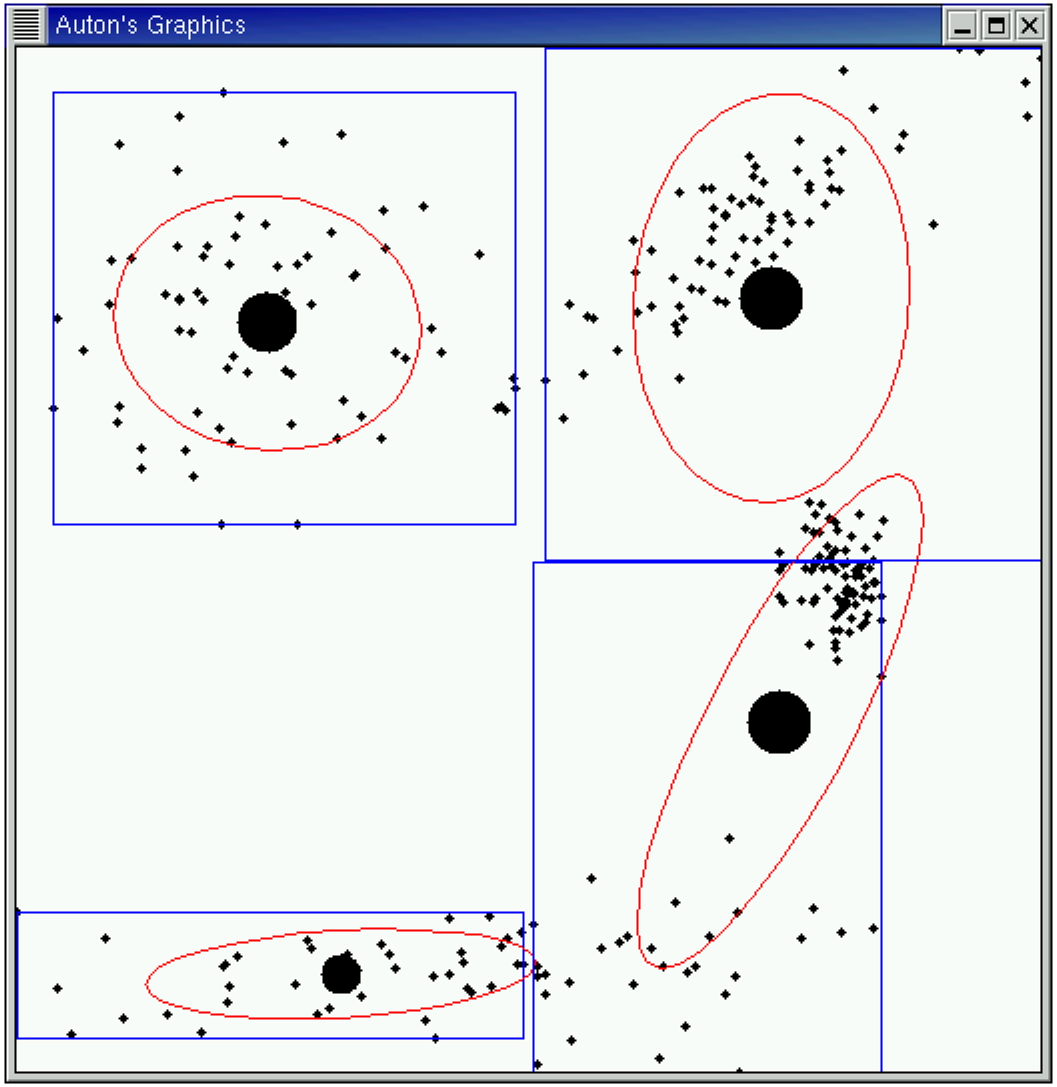
A *kd*-tree: level 1



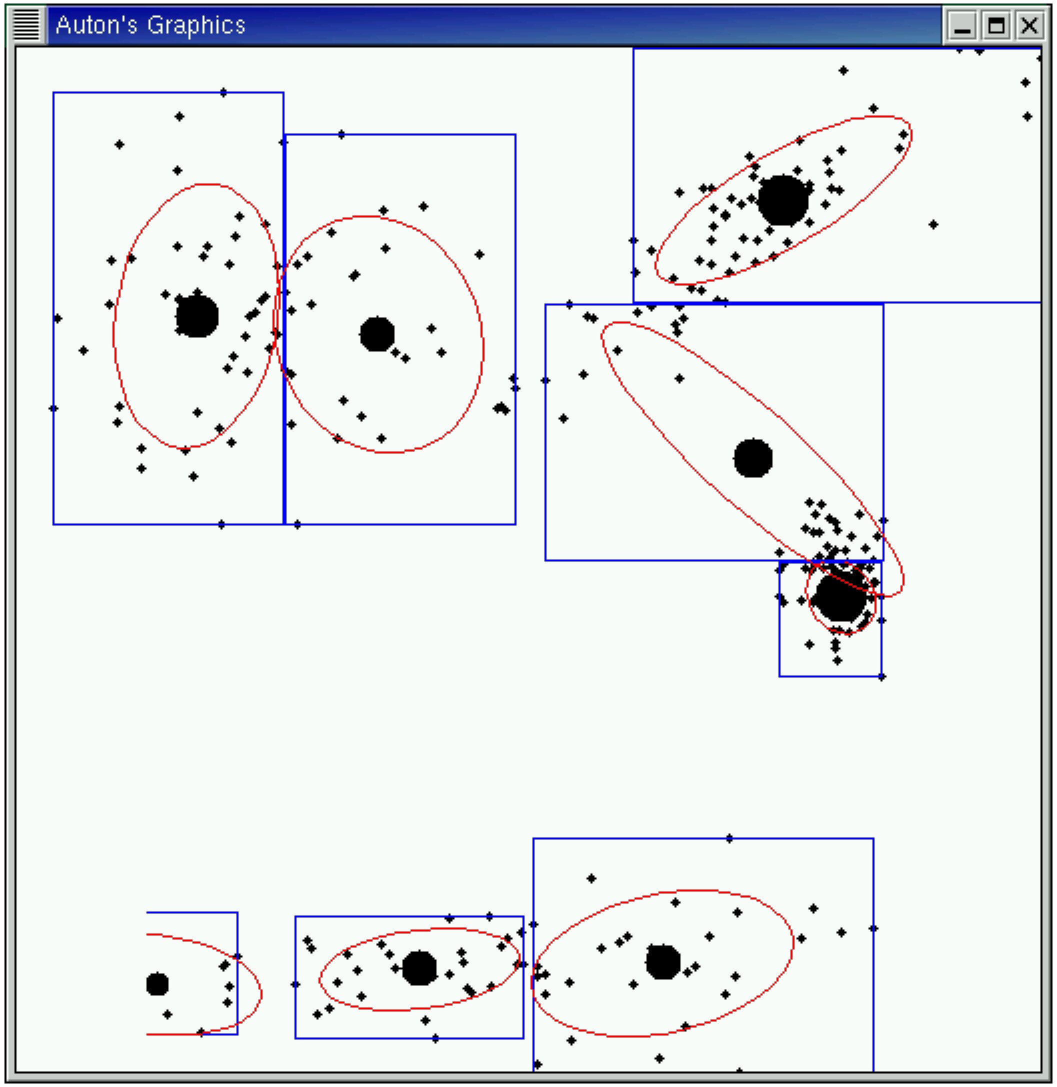
A *kd*-tree: level 2



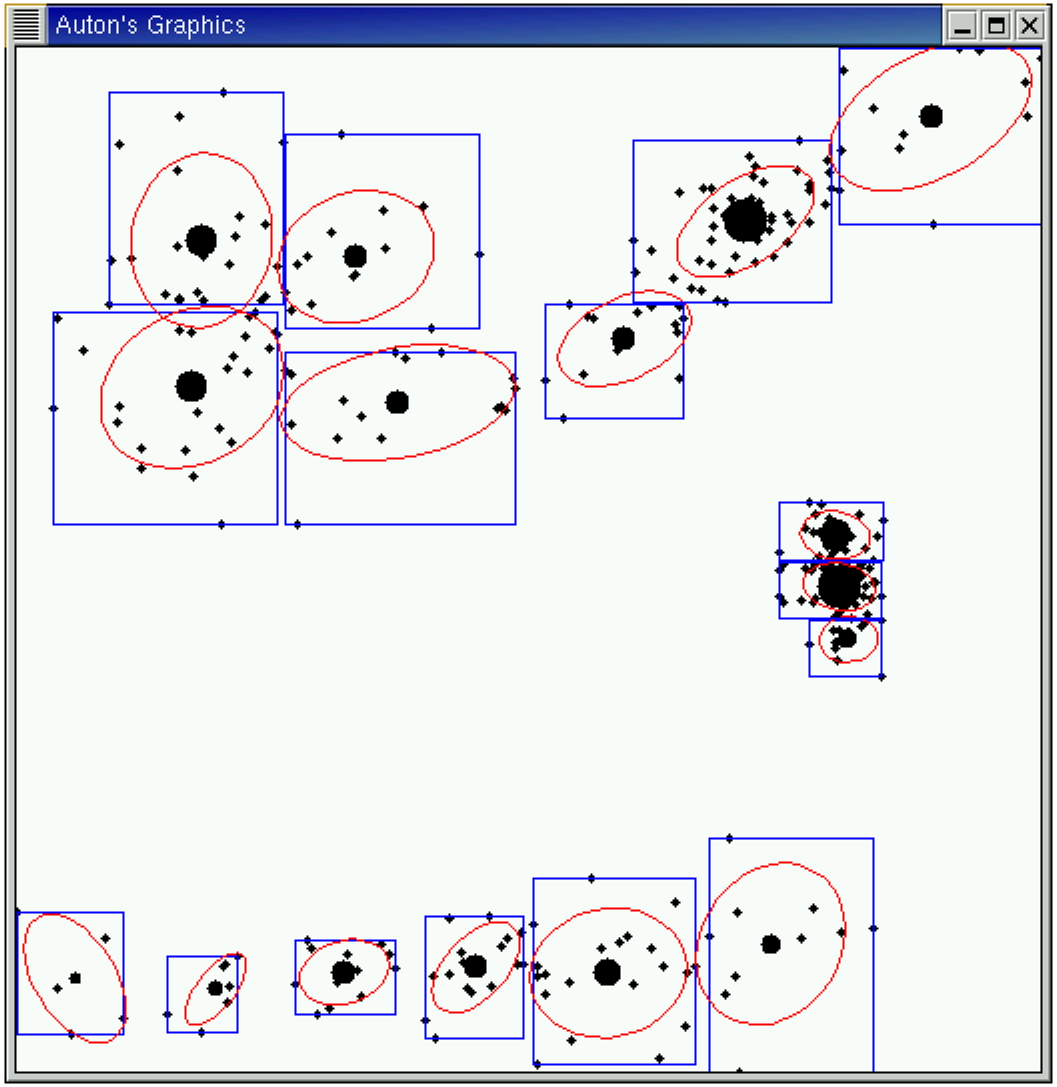
A *kd*-tree: level 3



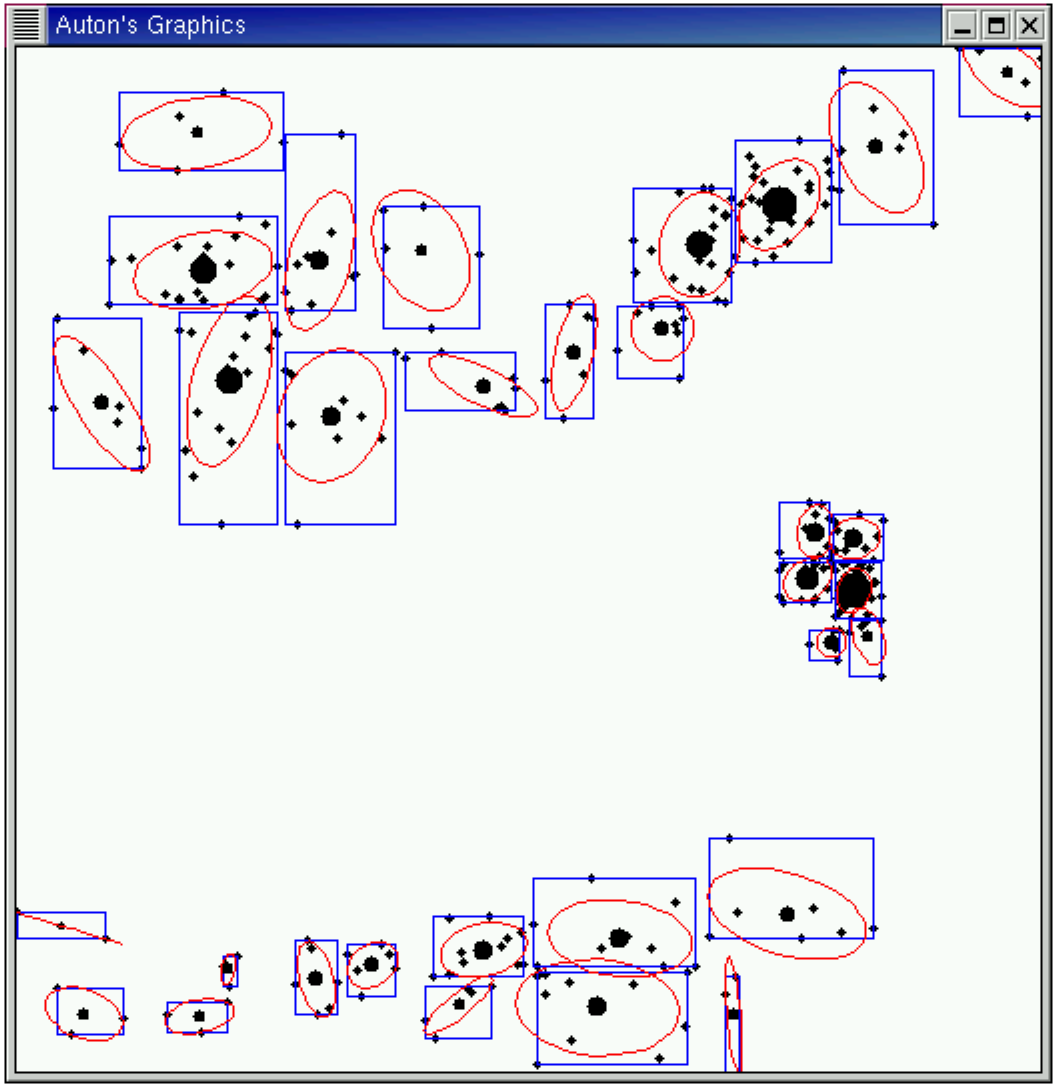
A *kd*-tree: level 4



A *kd*-tree: level 5



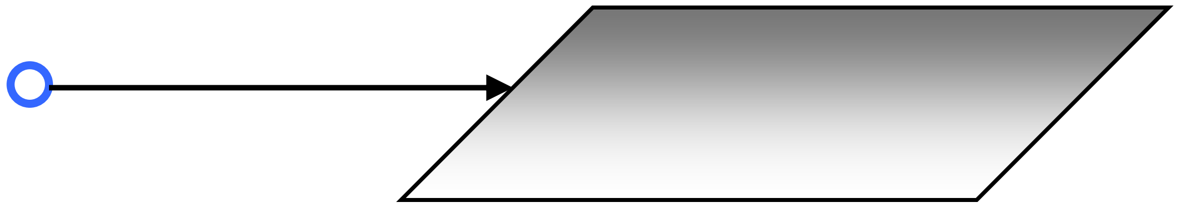
A *kd*-tree: level 6



Exclusion and inclusion,

using point-node *kd*-tree bounds.

O(D) bounds on distance minima/maxima:



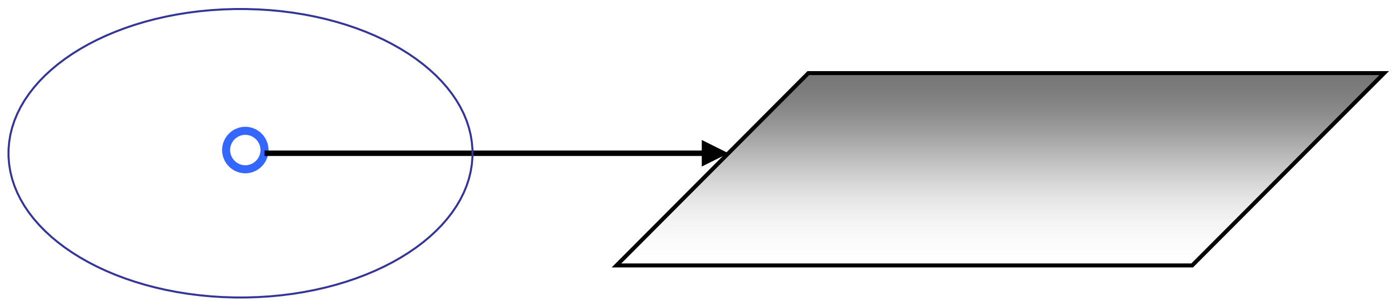
$$\min_i \|x - x_i\| \geq \sum_d^D \left[\max\left\{(l_d - x_d)^2, 0\right\} + \max\left\{(x_d - u_d)^2, 0\right\}\right]$$

$$\max_i \|x - x_i\| \leq \sum_d^D \max\left\{(u_d - x_d)^2, (x_d - l_d)^2\right\}$$

Exclusion and inclusion,

using point-node *kd*-tree bounds.

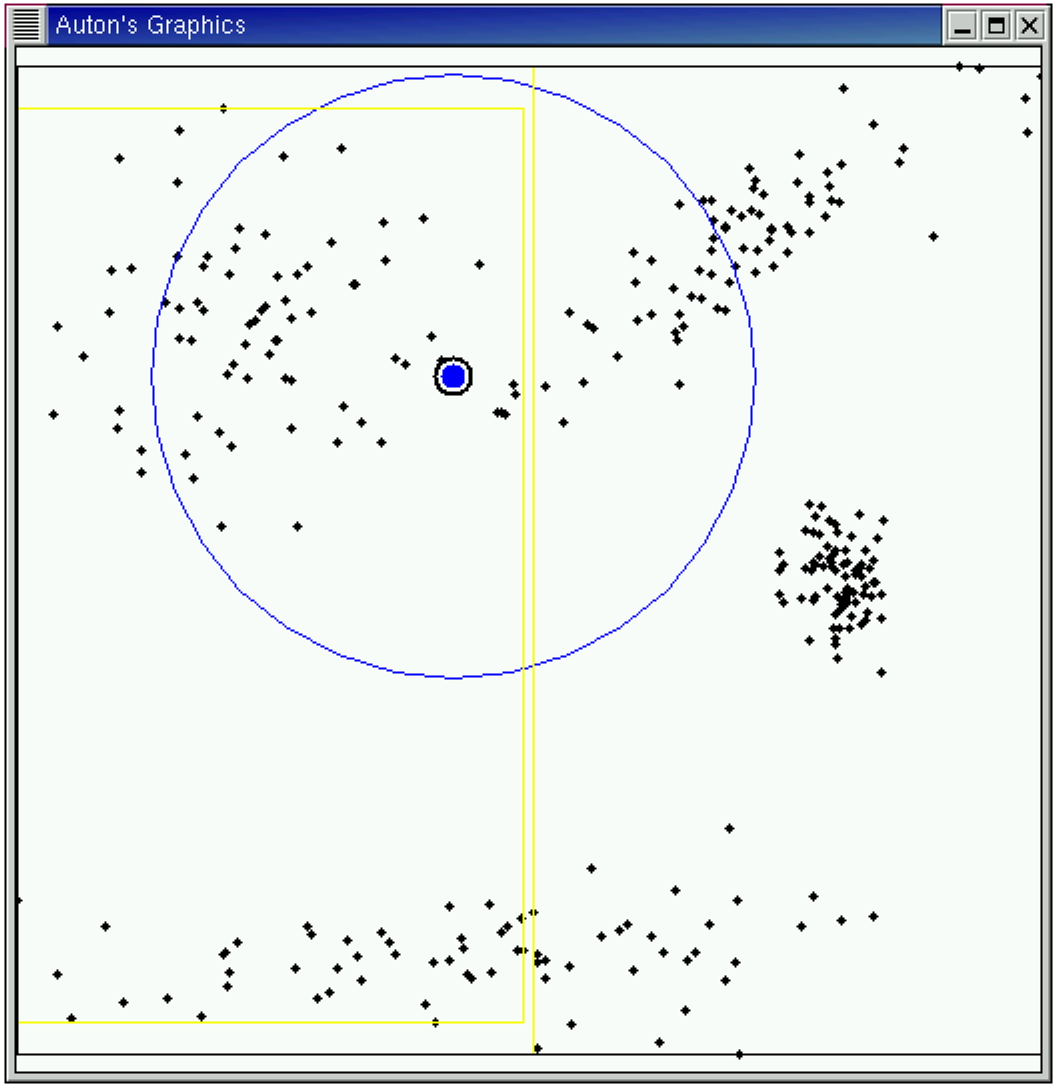
O(D) bounds on distance minima/maxima:



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

$$\max_i \left\| x - x_i \right\| \leq \sum_d^D \max\left\{ (u_d - x_d)^2, (x_d - l_d)^2 \right\}$$

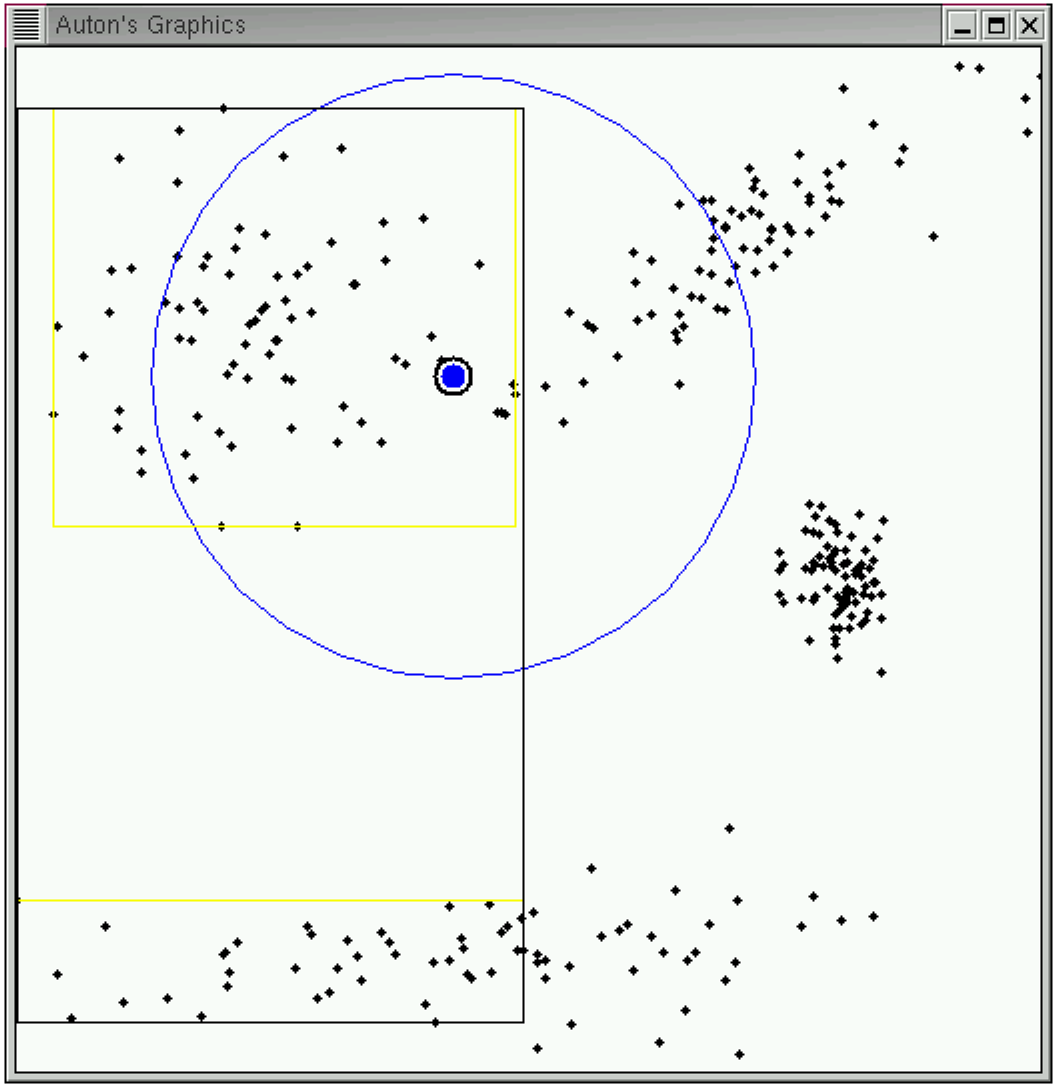
Range-count example



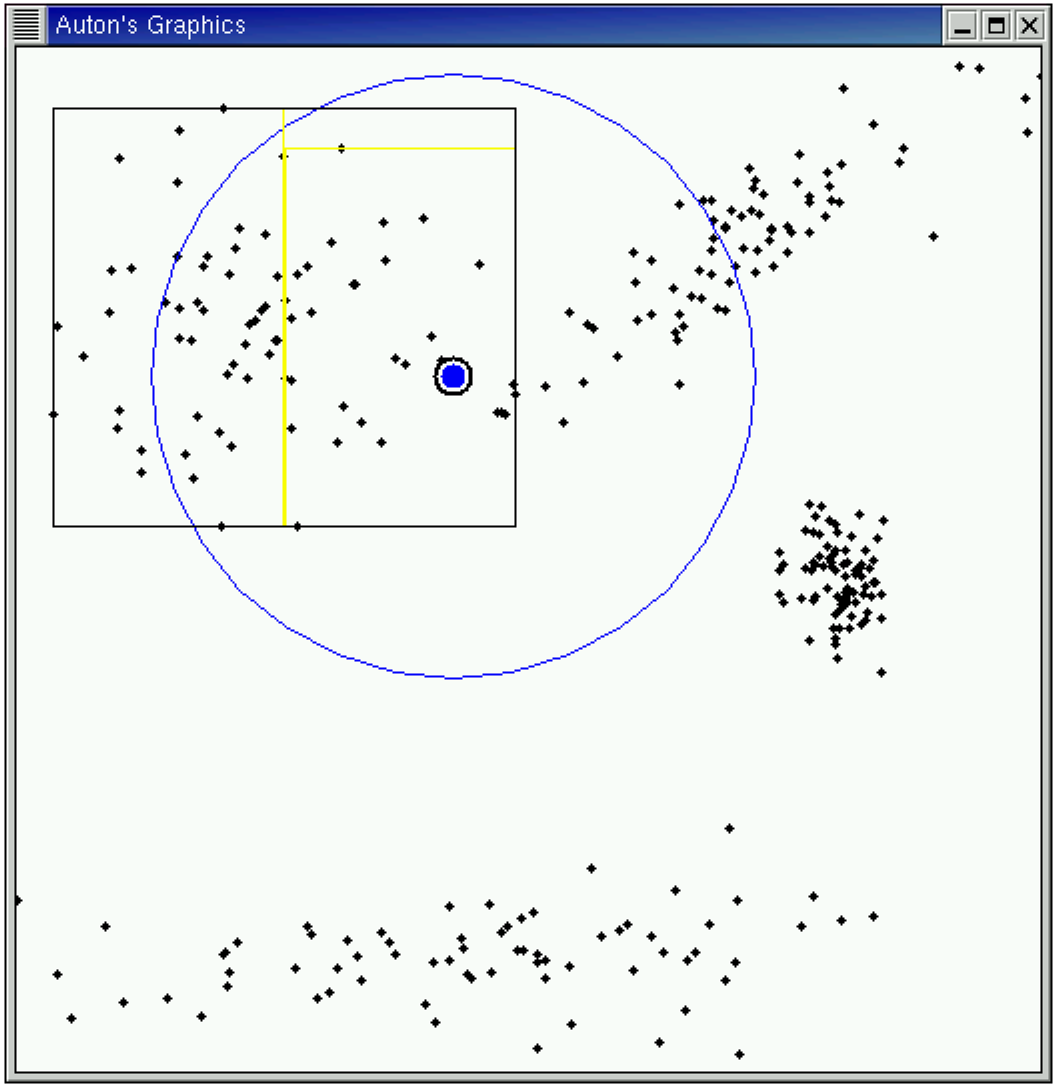
Idea #1.2: **Recursive range-count algorithm**

[folk algorithm]

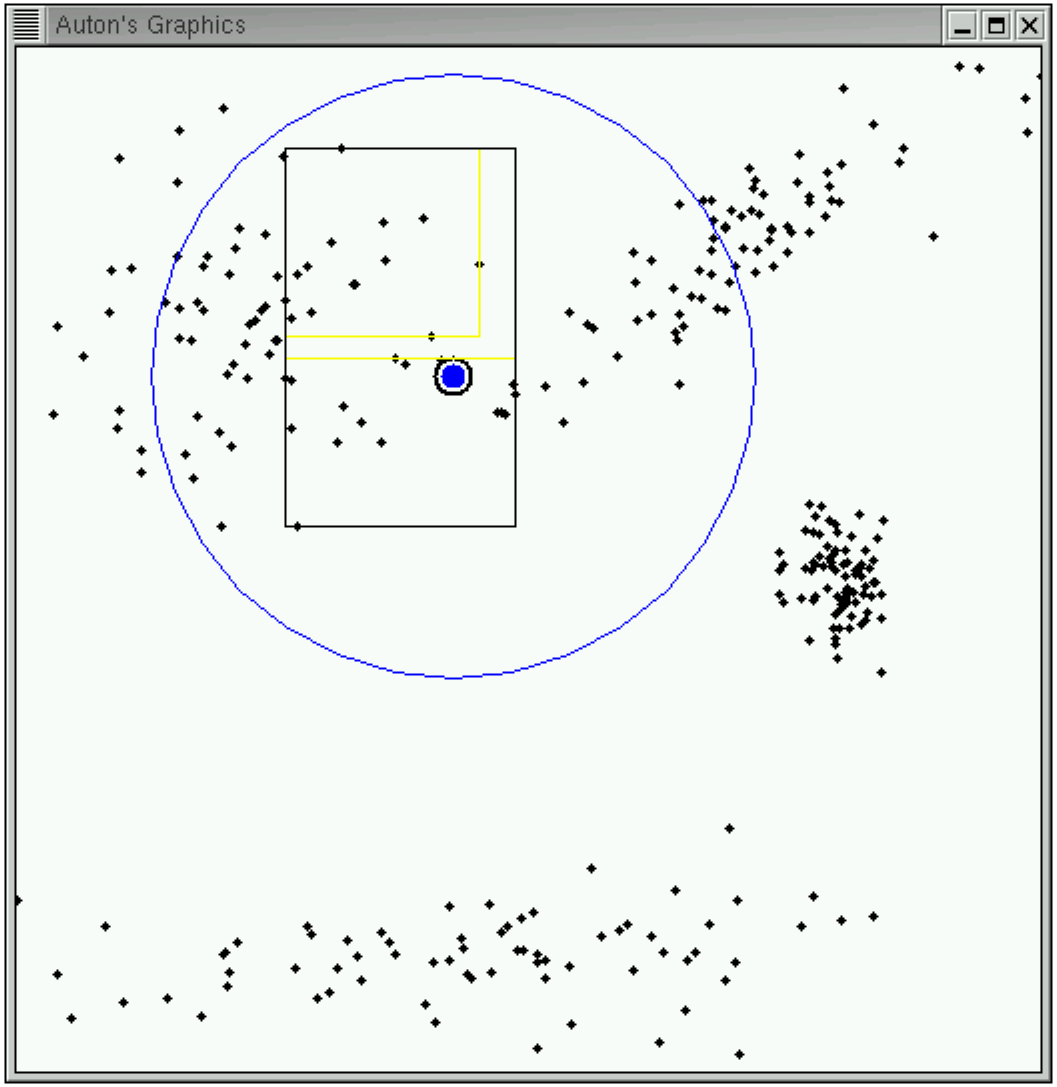
Range-count example



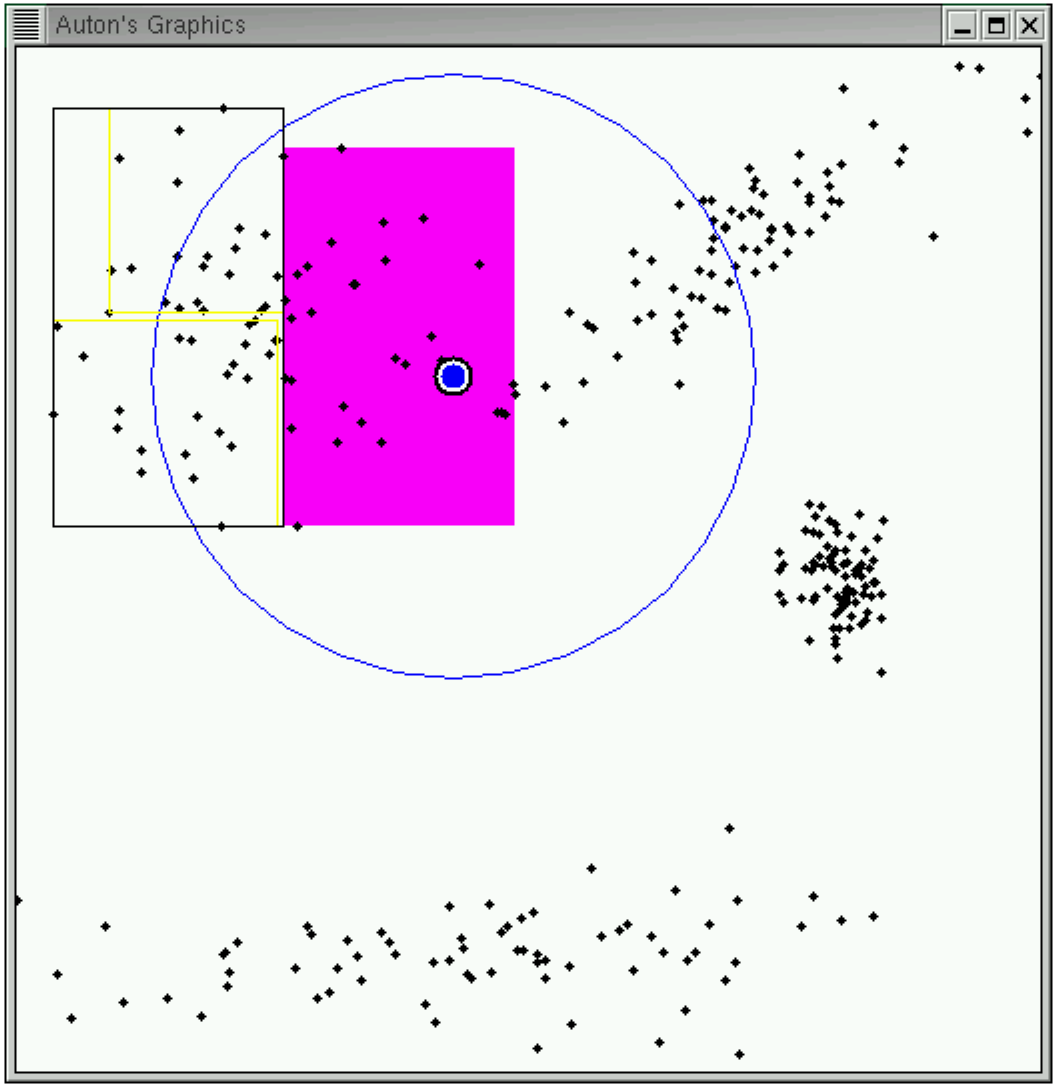
Range-count example



Range-count example



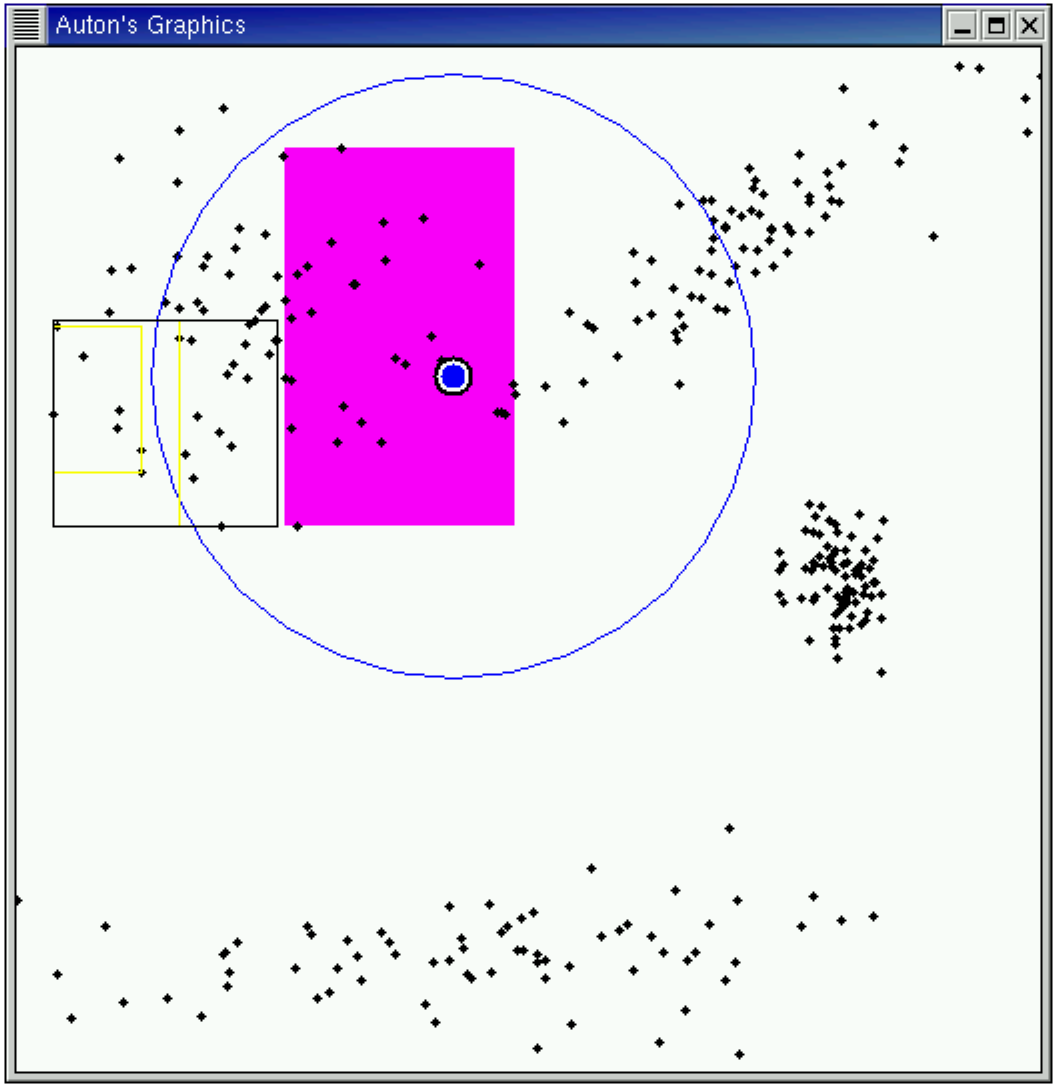
Range-count example



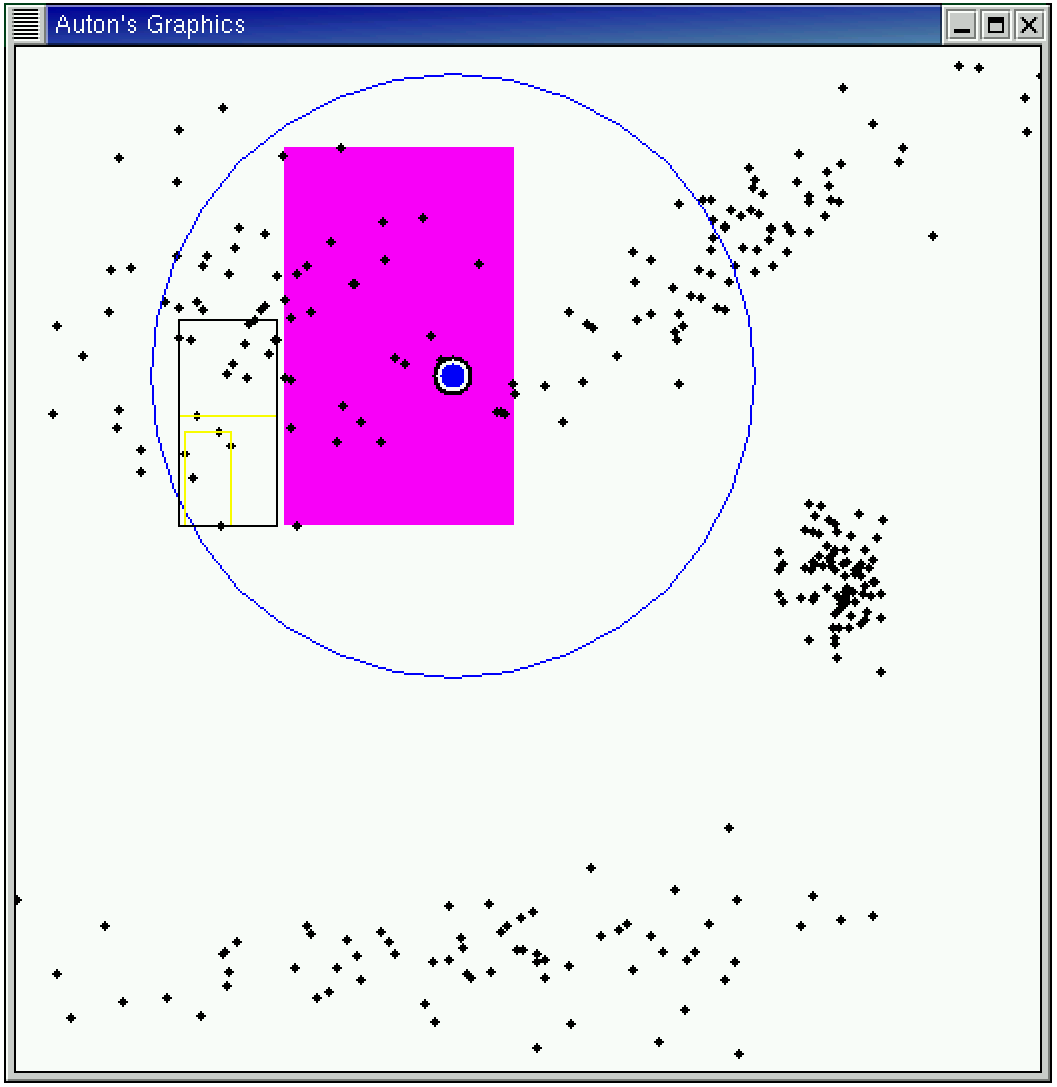
Pruned!

(inclusion)

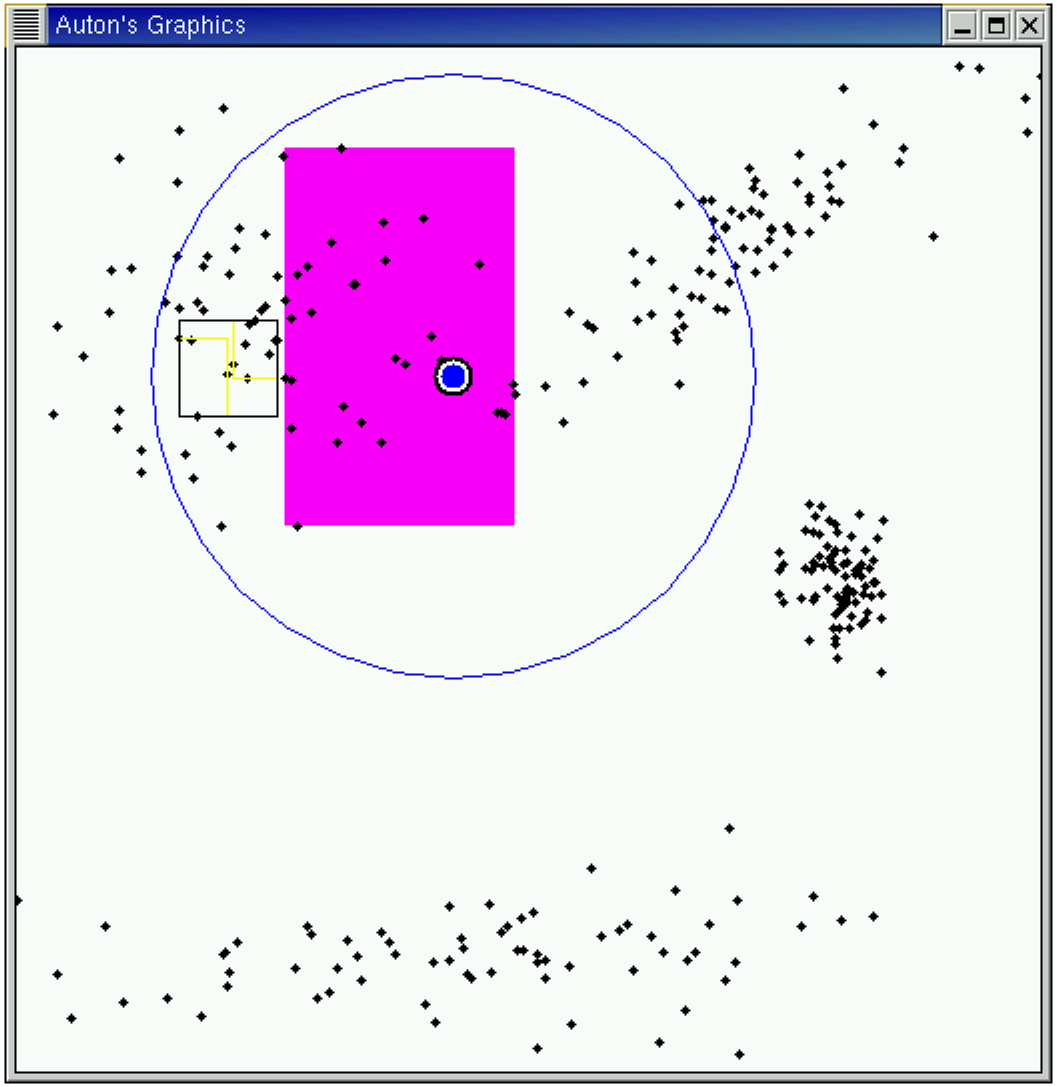
Range-count example



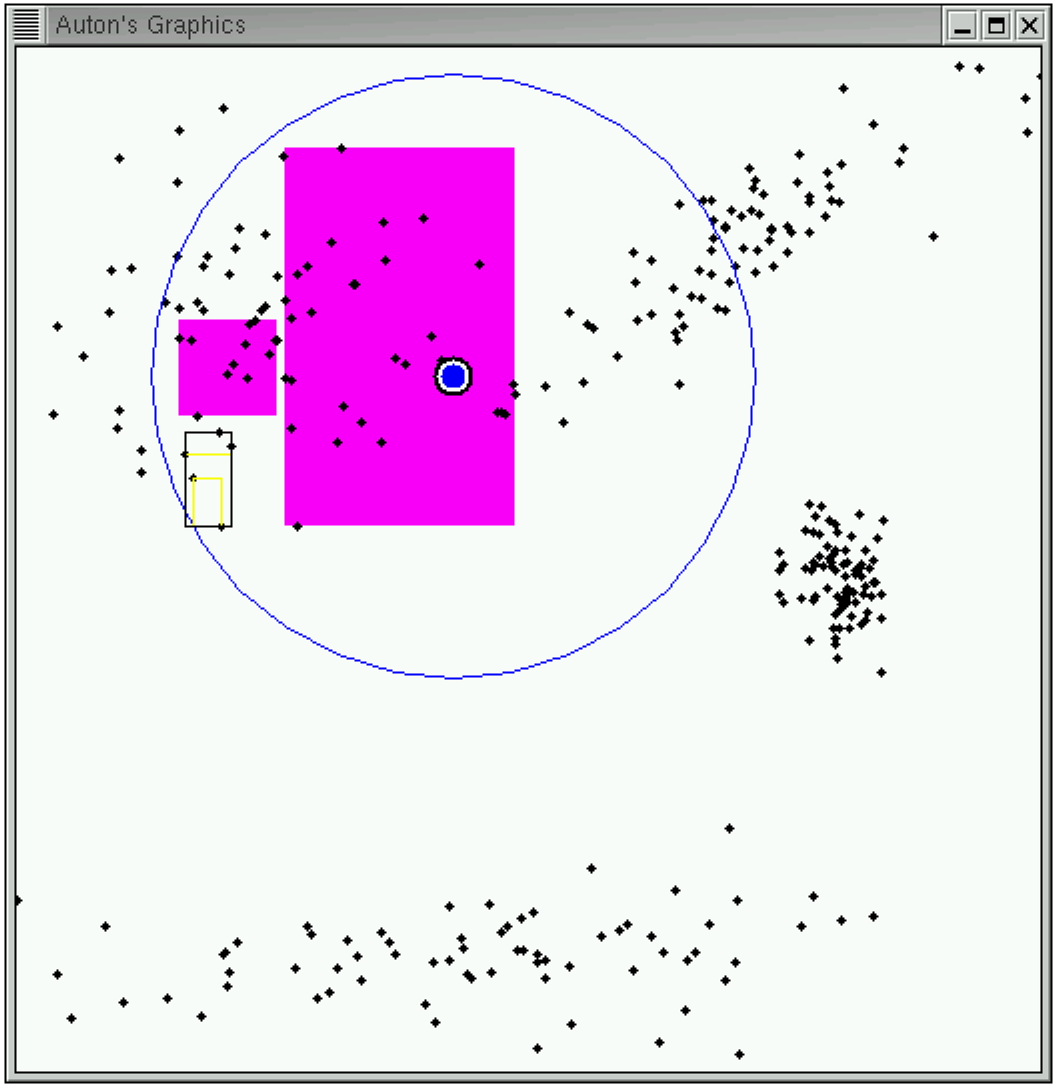
Range-count example



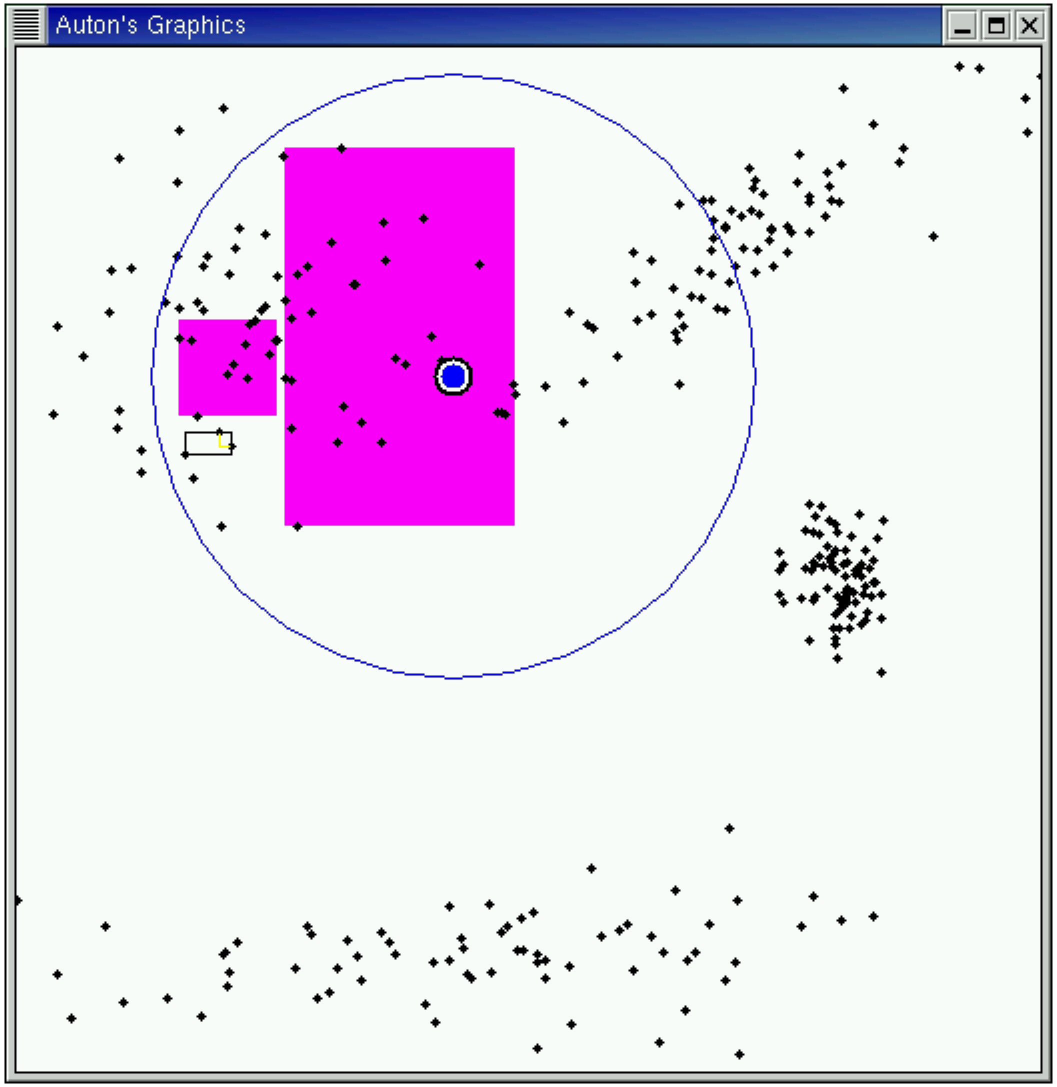
Range-count example



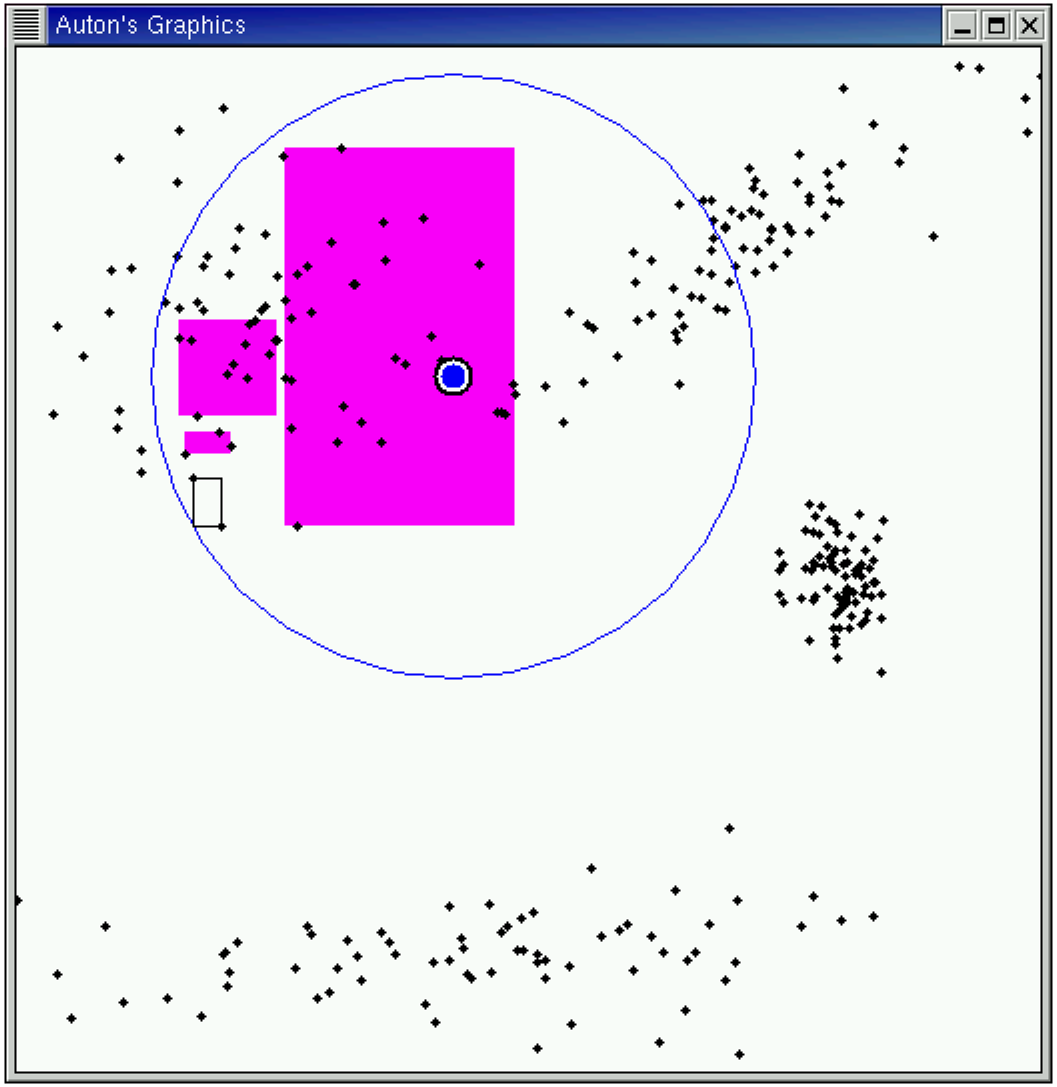
Range-count example



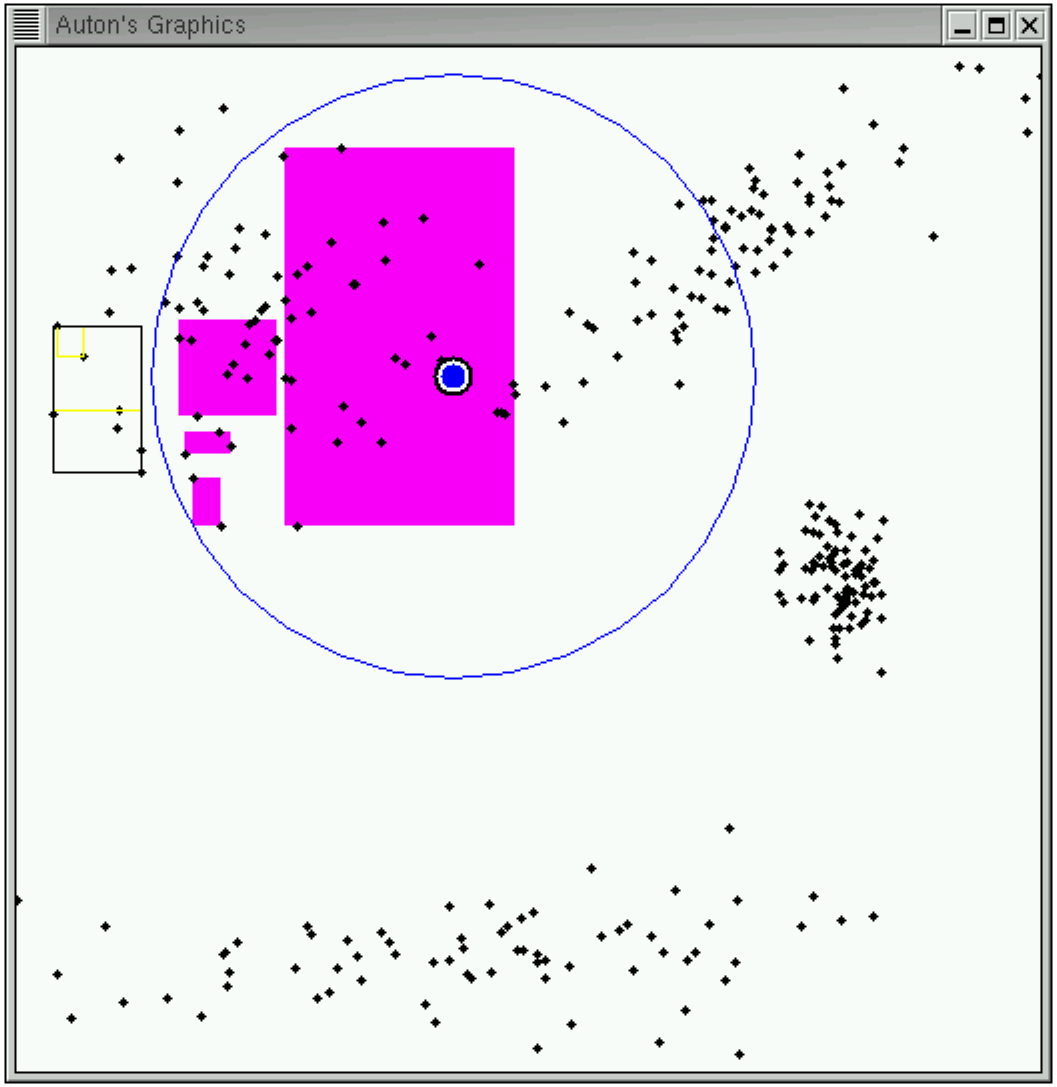
Range-count example



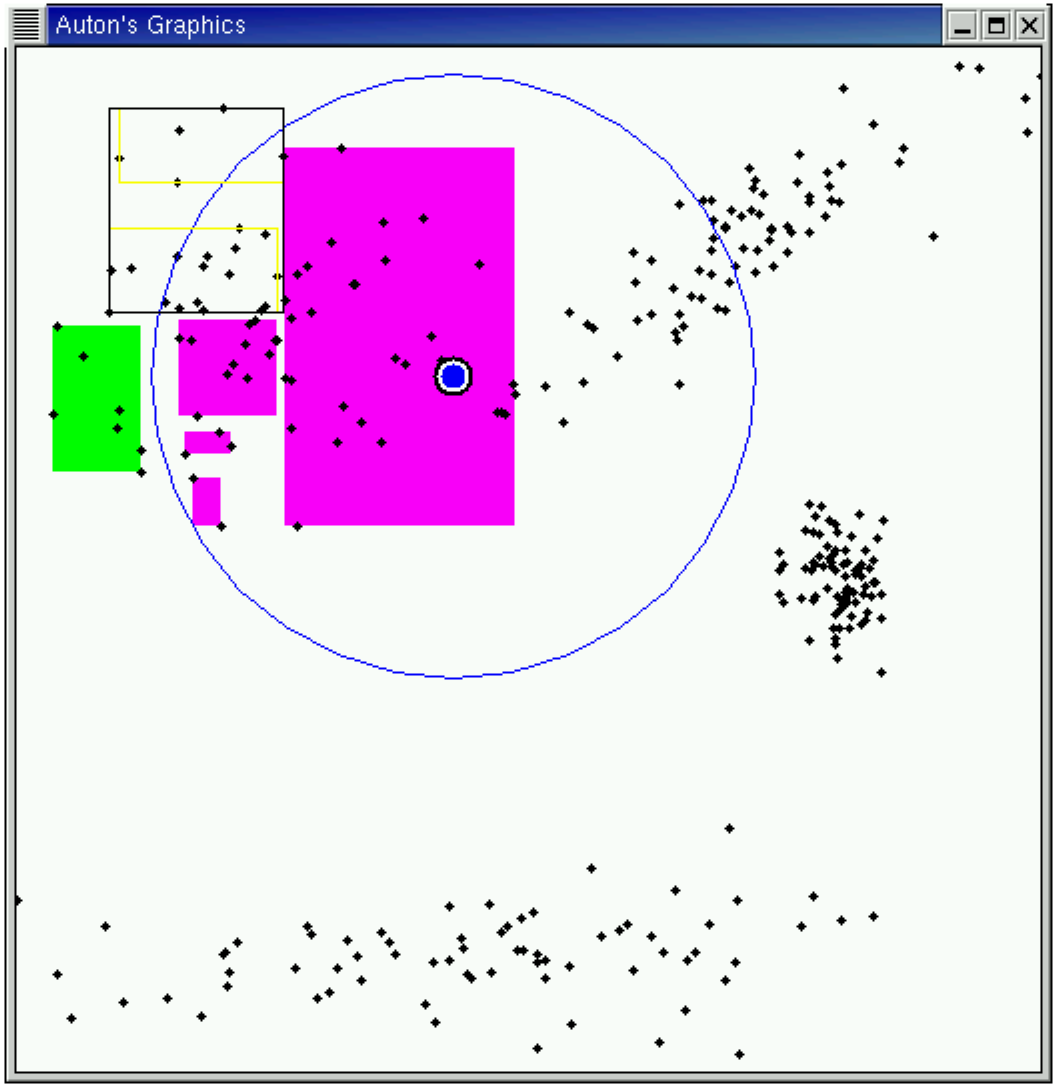
Range-count example



Range-count example

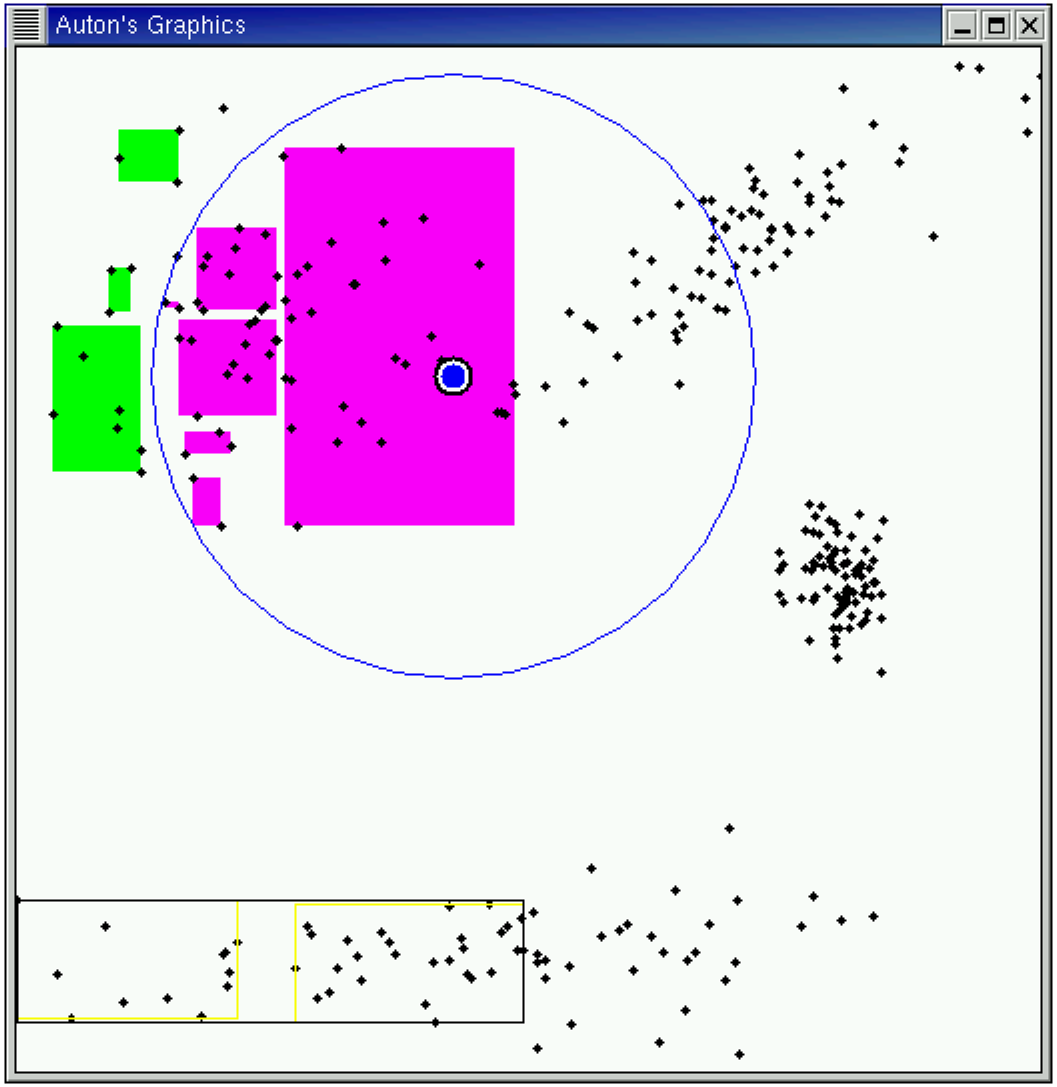


Range-count example

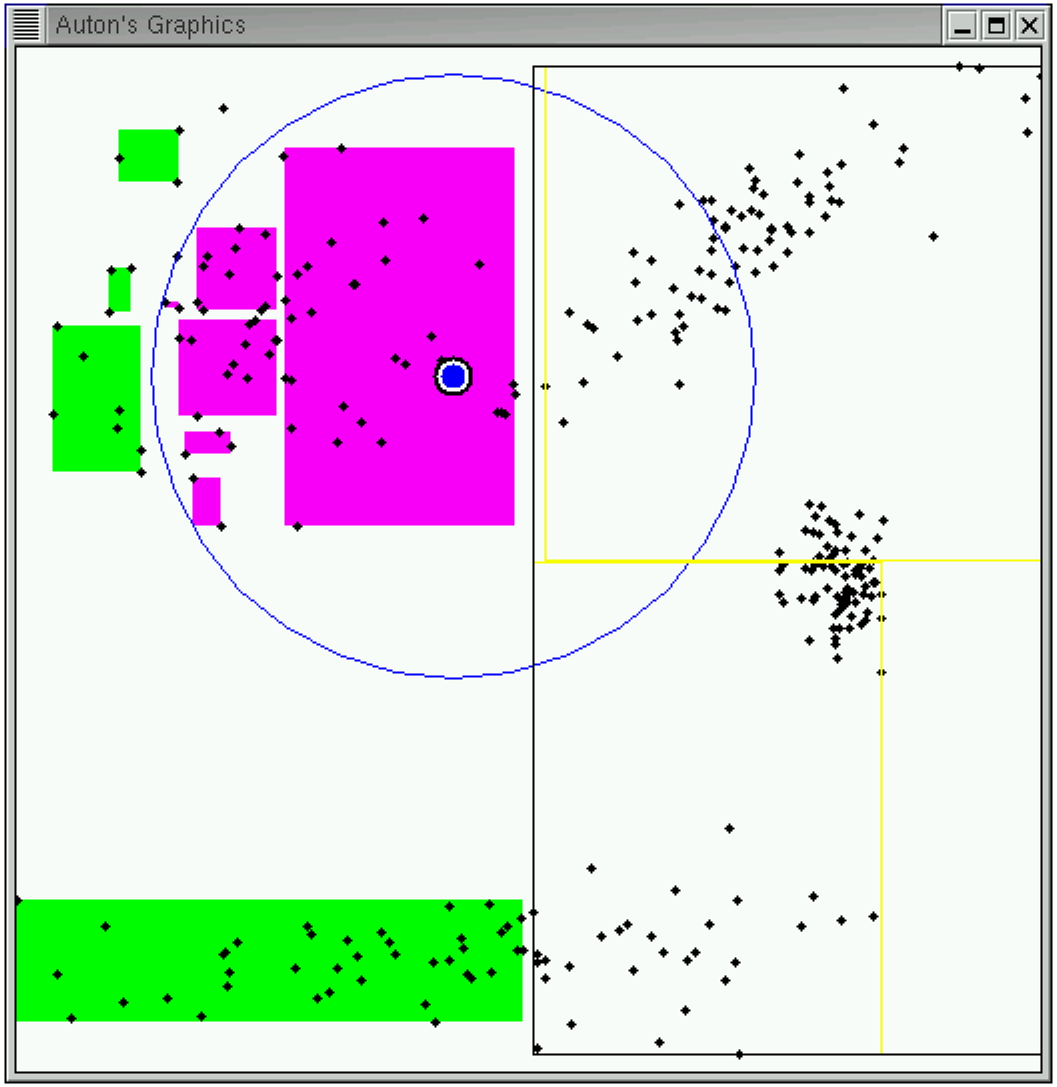


Pruned!
(exclusion)

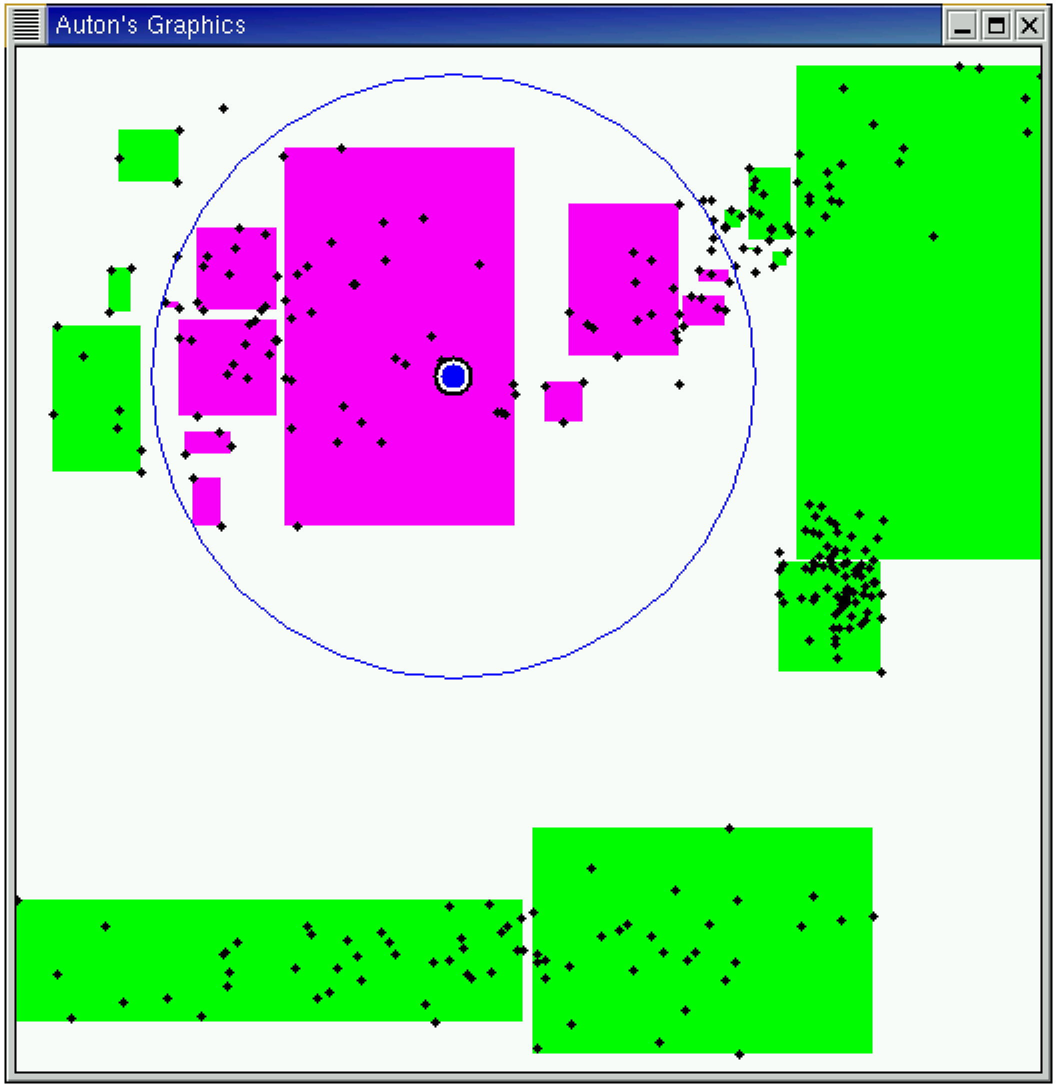
Range-count example



Range-count example



Range-count example



What’s the best data structure for proximity problems?

- There are hundreds of papers which have proposed nearest-neighbor data structures (maybe even thousands)
- Empirical comparisons are usually to one or two strawman methods

→ Nobody really knows how things compare

The Proximity Project

[Gray, Lee, Rotella, Moore 2005]

Careful agostic empirical comparison, open source
15 datasets, dimension 2-1M
The most well-known methods from 1972-2004

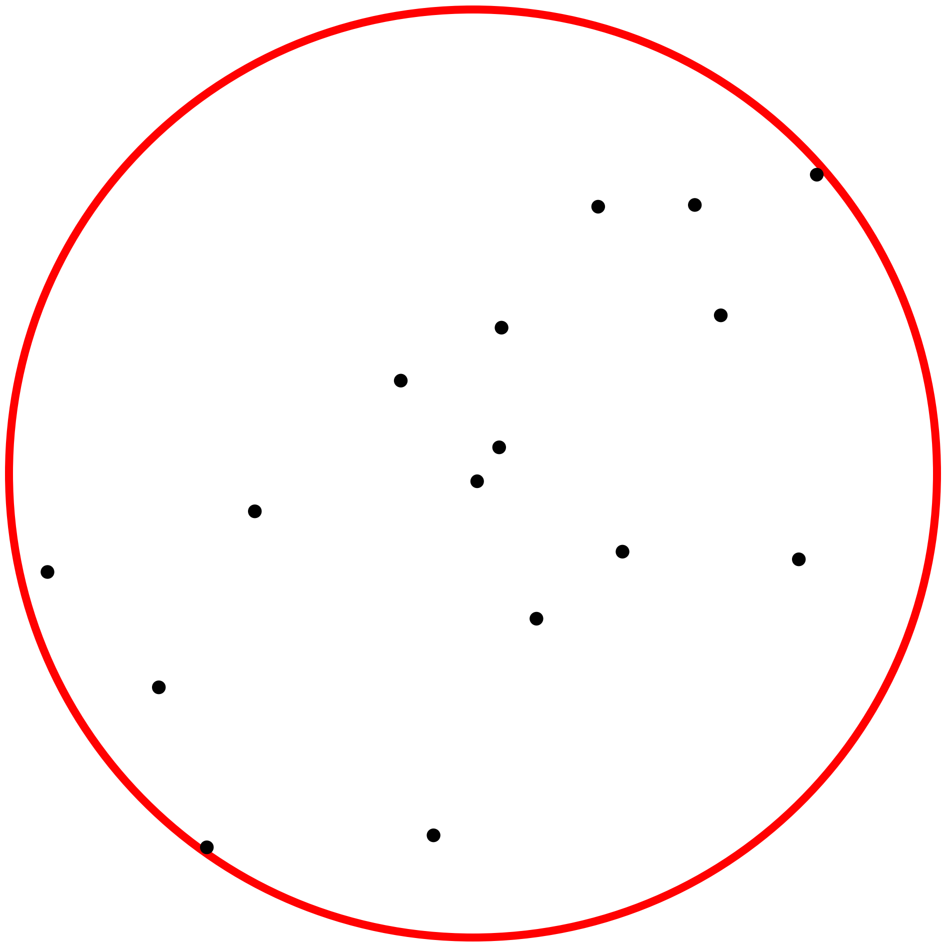
- Exact NN: 15 methods
- All-NN, mono & bichromatic: 3 methods
- Approximate NN: 10 methods
- Point location: 3 methods
- (NN classification: 3 methods)
- (Radial range search: 3 methods)

…and the overall winner is? (exact NN, high-D)

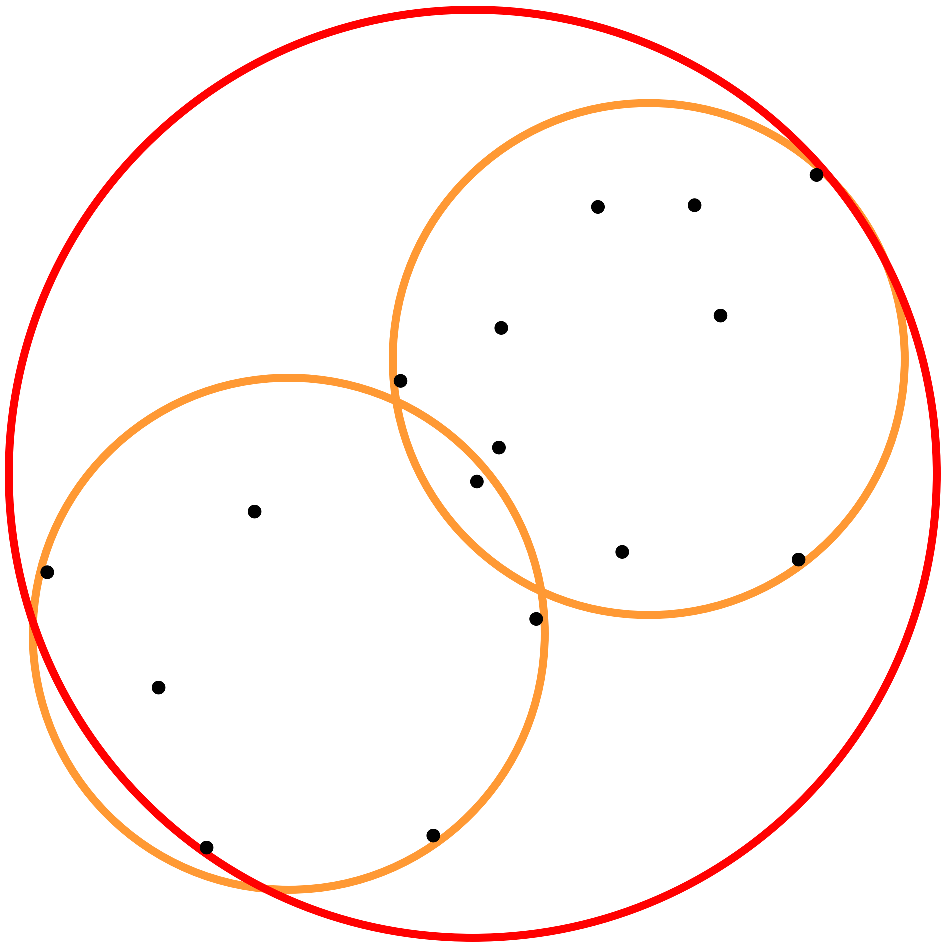
Ball-trees, basically – though there is high variance and dataset dependence

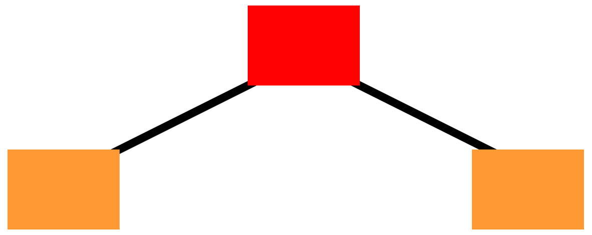
- Auton ball-trees III [Omohundro 91],[Uhlmann 91], [Moore 99]
- Cover-trees [Alina B.,Kakade,Langford 04]
- Crust-trees [Yianilos 95],[Gray,Lee,Rotella,Moore 2005]

A ball-tree: level 1

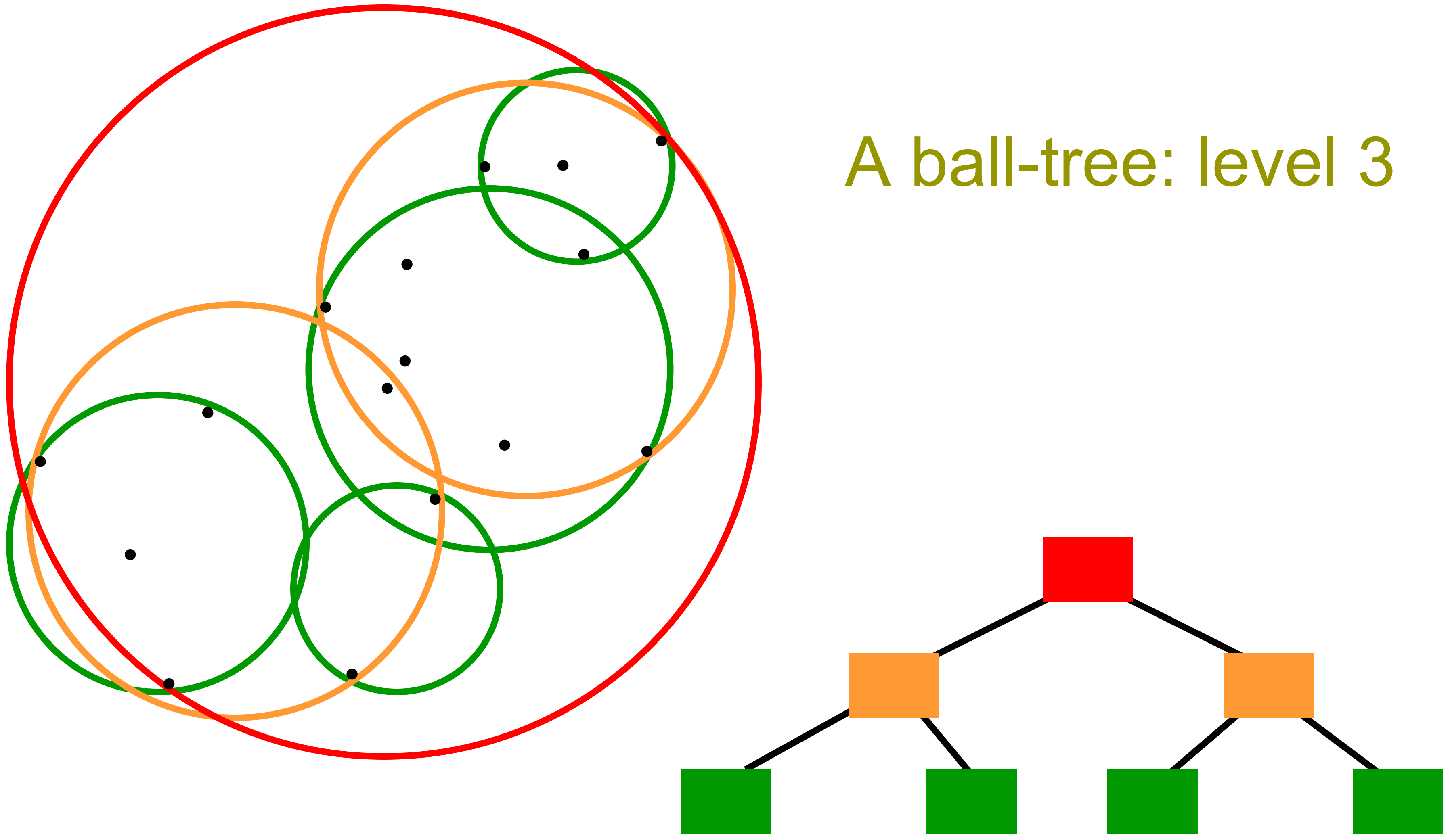


A ball-tree: level 2

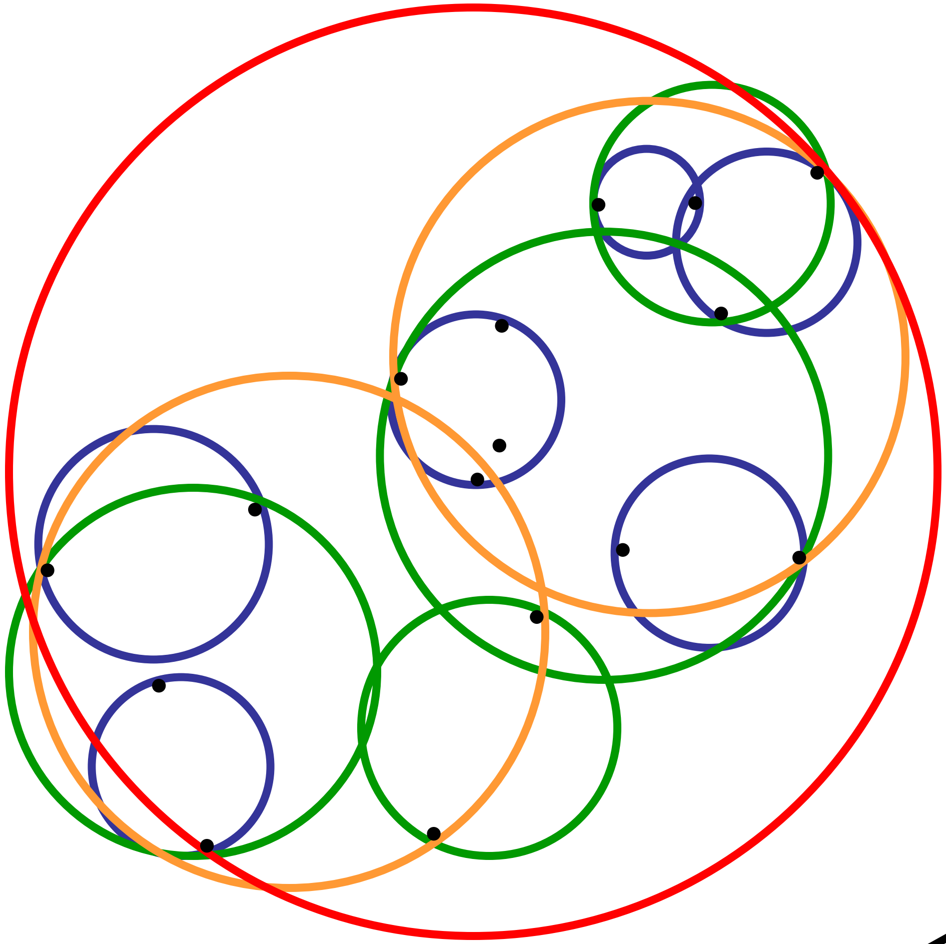


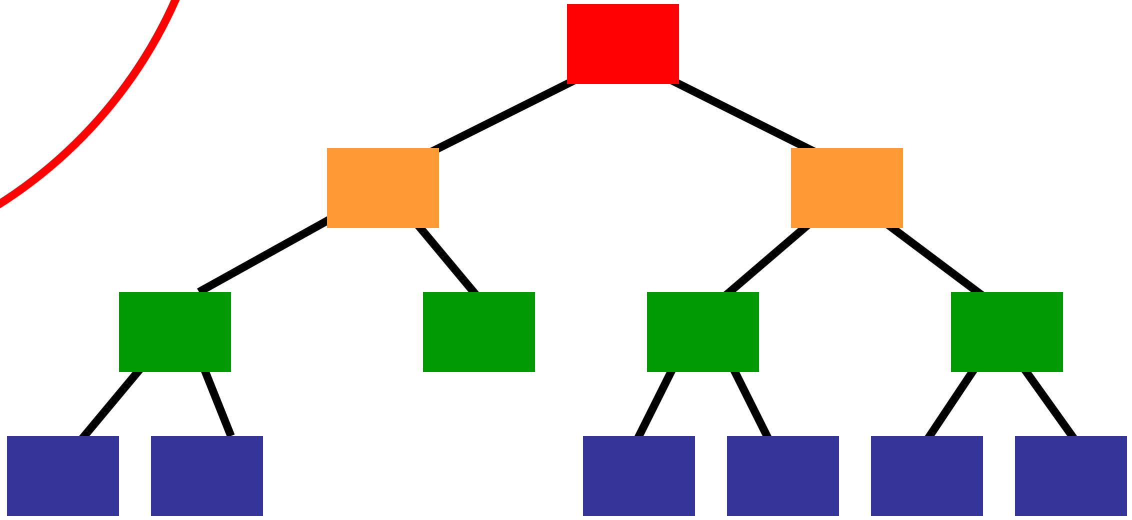


A ball-tree: level 3

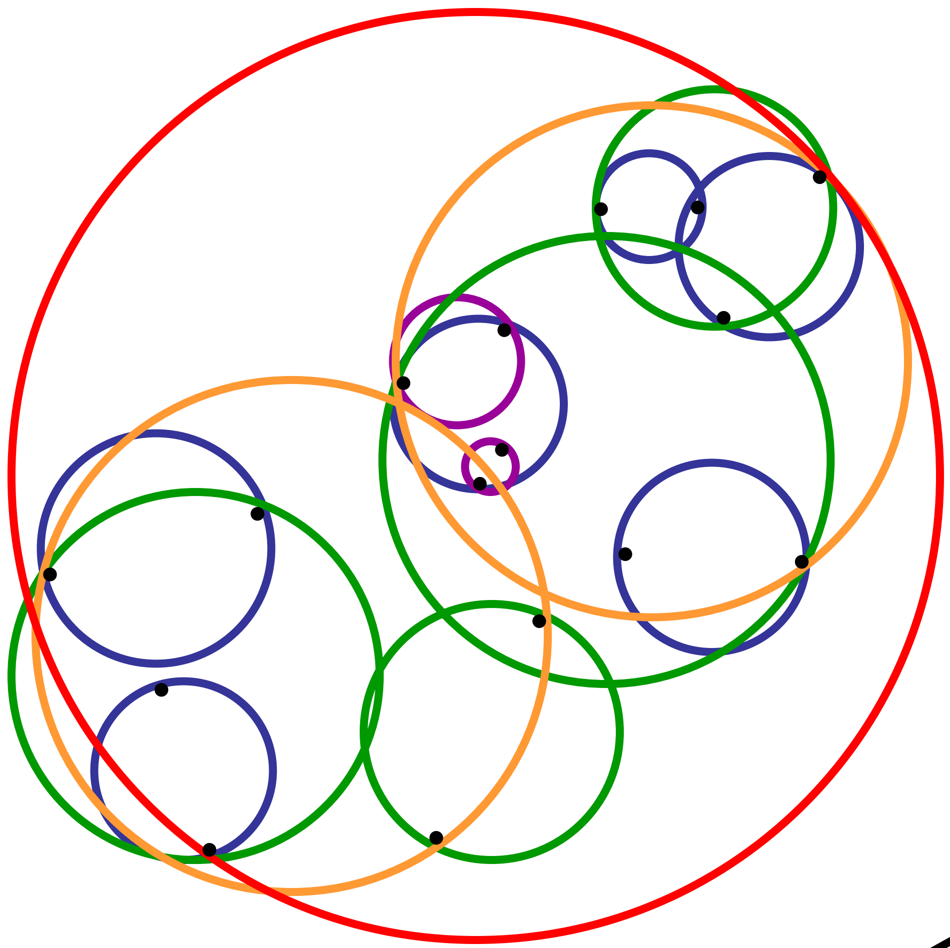


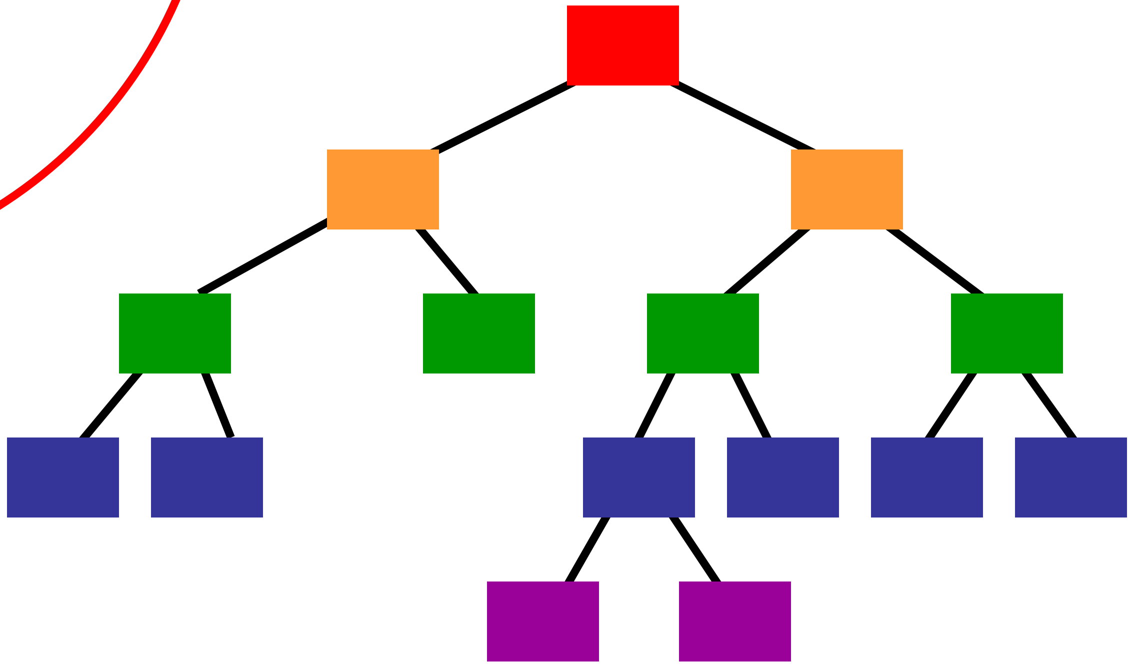
A ball-tree: level 4





A ball-tree: level 5





Anchors Hierarchy [Moore 99]

- ‘Middle-out’ construction
- Uses farthest-point method [Gonzalez 85] to find sqrt(N) clusters – this is the middle
- Bottom-up construction to get the top
- Top-down division to get the bottom
- Smart pruning throughout to make it fast
- (NlogN), very fast in practice

Outline:

1. Physics problems and methods

2. Generalized N-body problems

3. Proximity data structures

4. **Dual-tree algorithms**

5. Comparison

Questions

- What's the magic that allows *O(N)*?

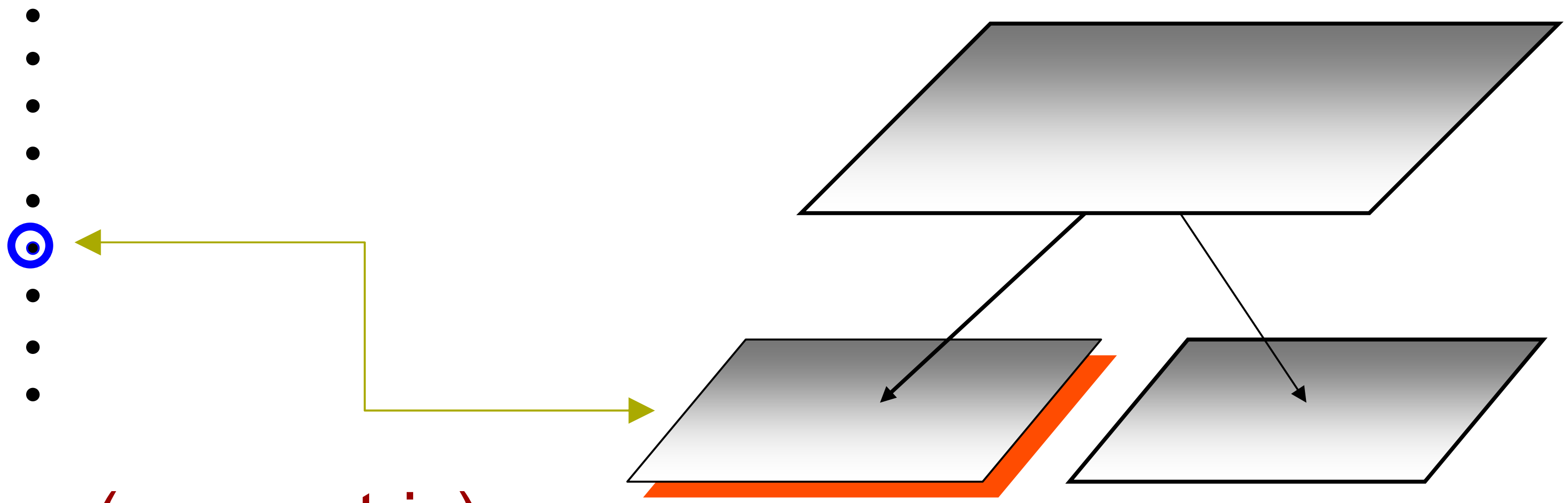
  Is it really because of the expansions?

- Can we obtain an method that's:
  1. *O(N)*
  2. Lightweight: - works with or without expansions
     - simple, recursive

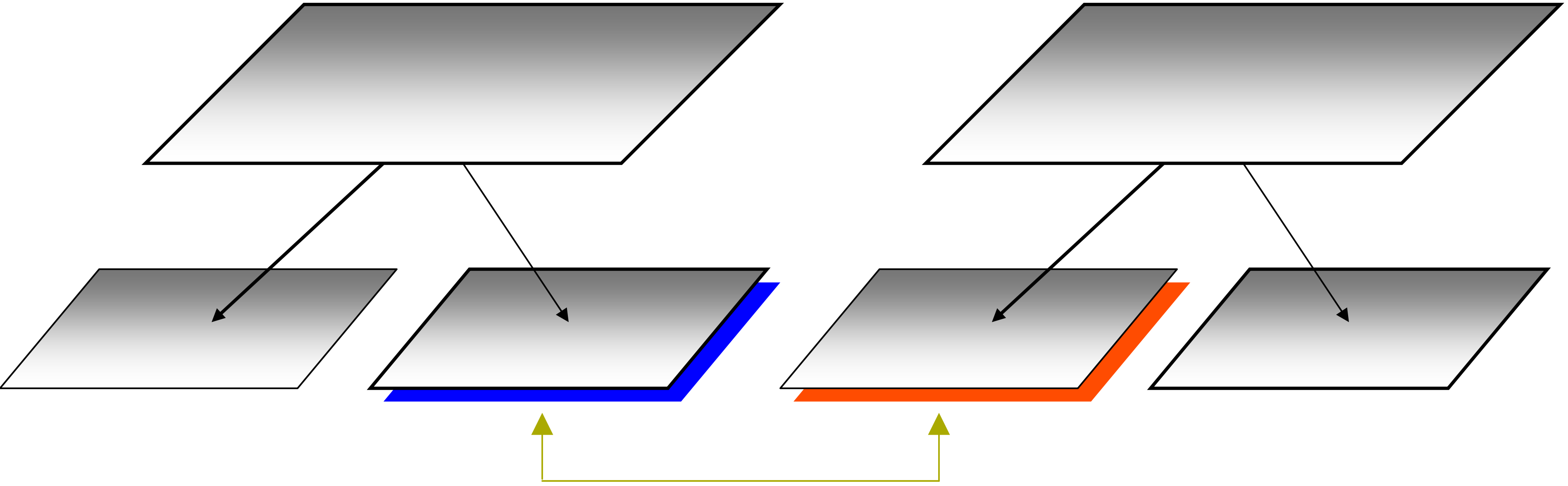
New algorithm

- Use an adaptive tree (*kd*-tree or ball-tree)
- Dual-tree recursion
- Finite-difference approximation

Single-tree:



Dual-tree (symmetric):



Simple recursive algorithm

```
SingleTree(q,R)
{
  if approximate(q,R), return.

  if leaf(R), SingleTreeBase(q,R).
  else,
    SingleTree(q,R.left).
    SingleTree(q,R.right).
}
```

(NN or range-search: recurse on the closer node first)

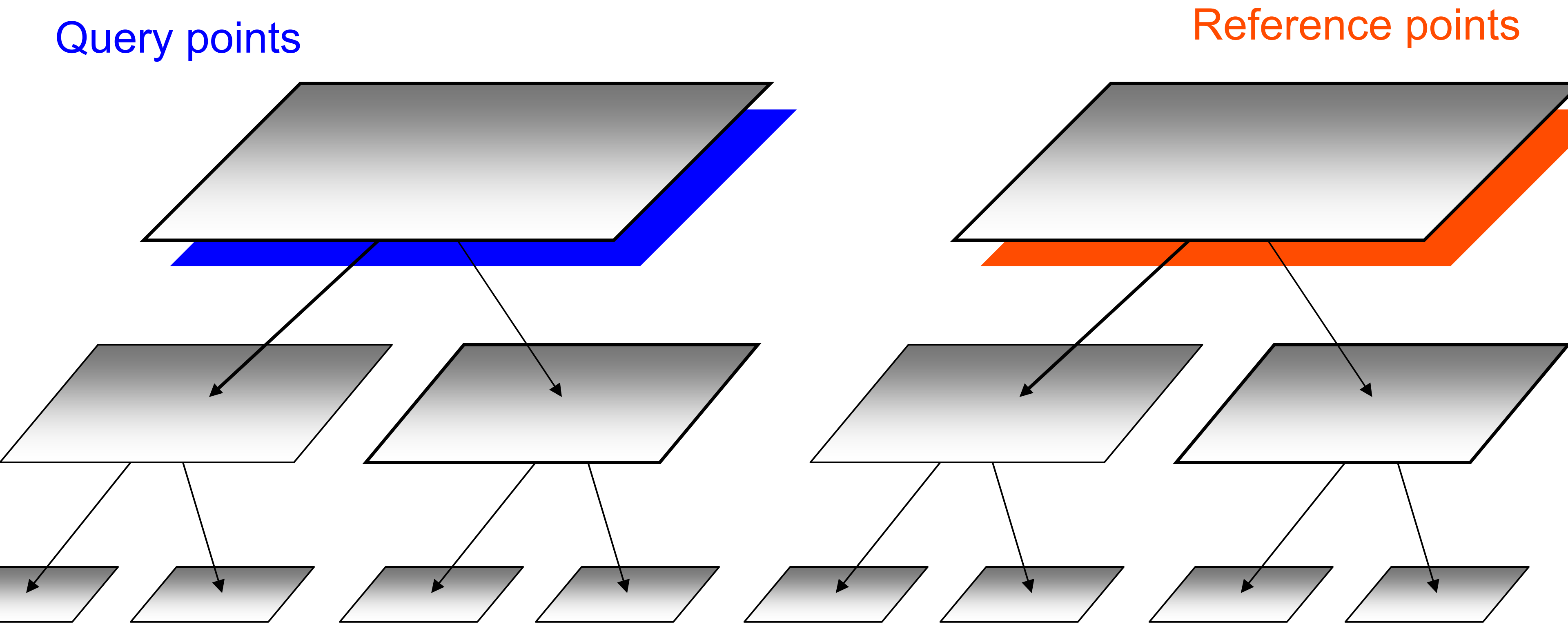
Simple recursive algorithm

```
DualTree(Q,R)
{
  if approximate(Q,R), return.

  if leaf(Q) and leaf(R), DualTreeBase(Q,R).
  else,
    DualTree(Q.left,R.left).
    DualTree(Q.left,R.right).
    DualTree(Q.right,R.left).
    DualTree(Q.right,R.right).
}
```

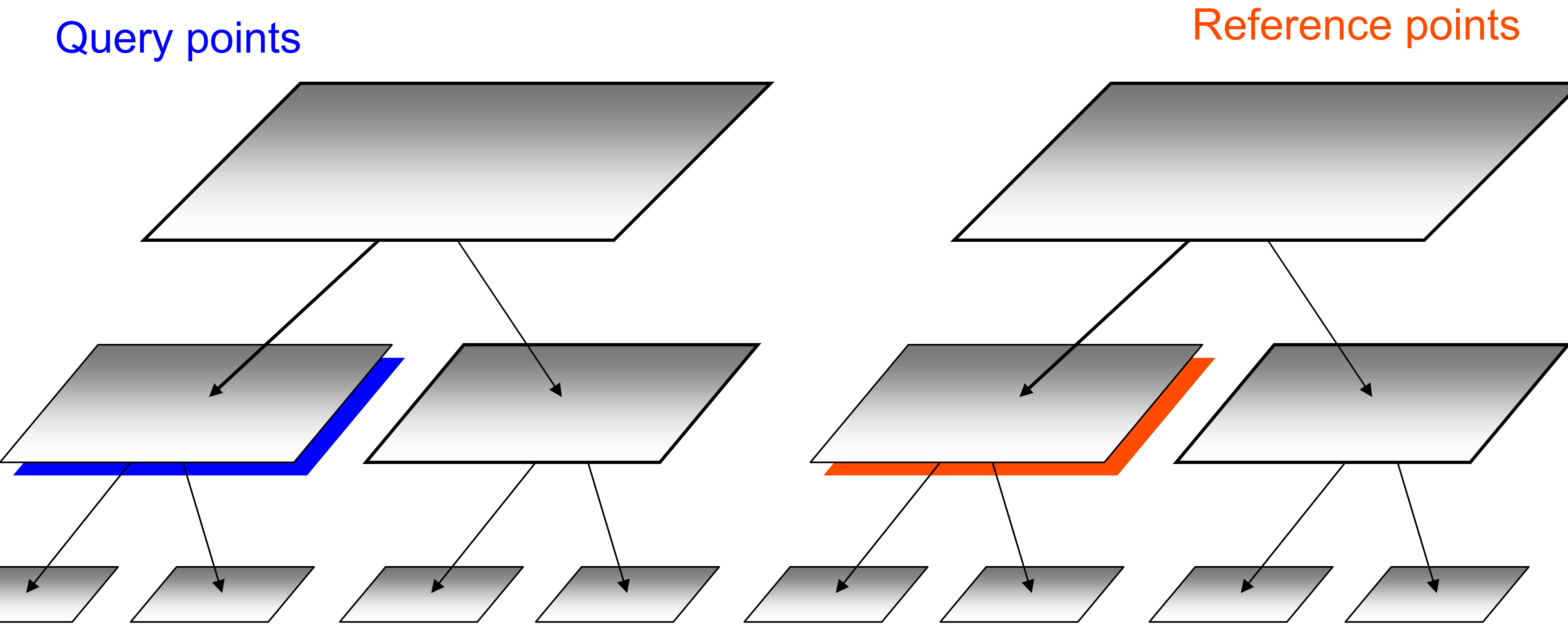
(NN or range-search: recurse on the closer node first)

Dual-tree traversal

(depth-first)



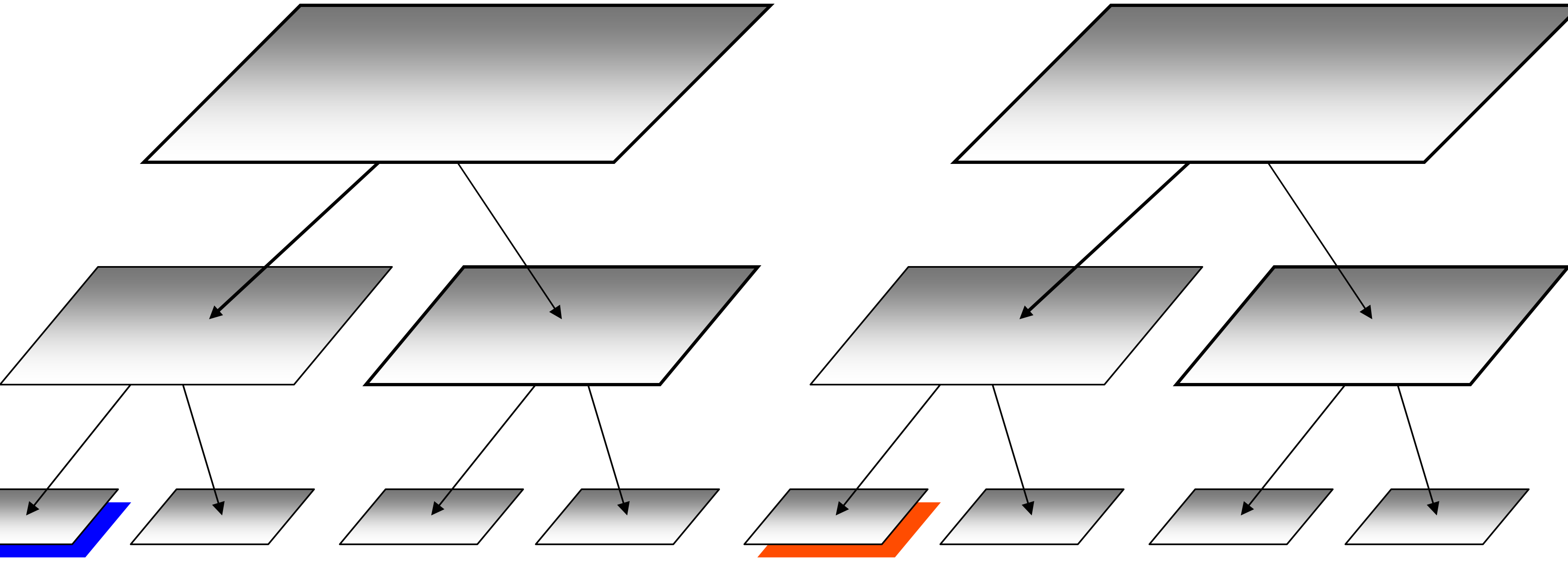
Dual-tree traversal



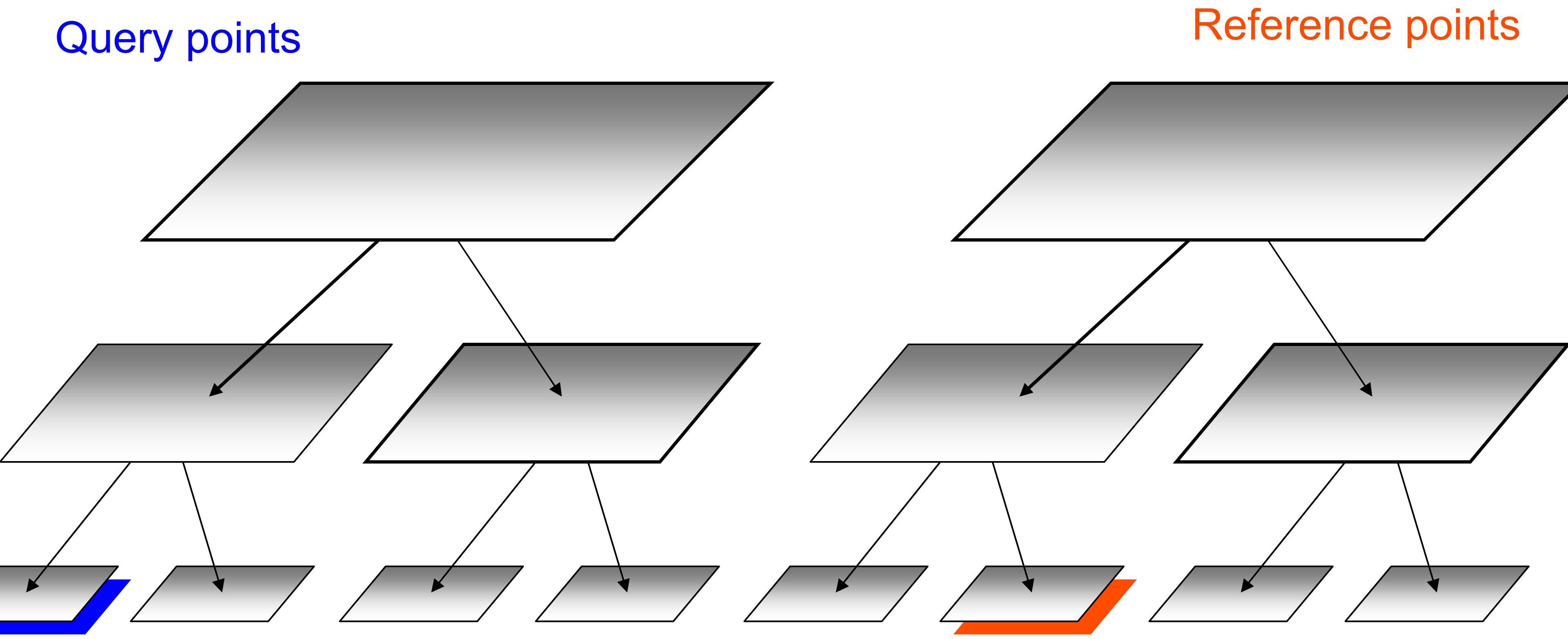
Dual-tree traversal

Query points

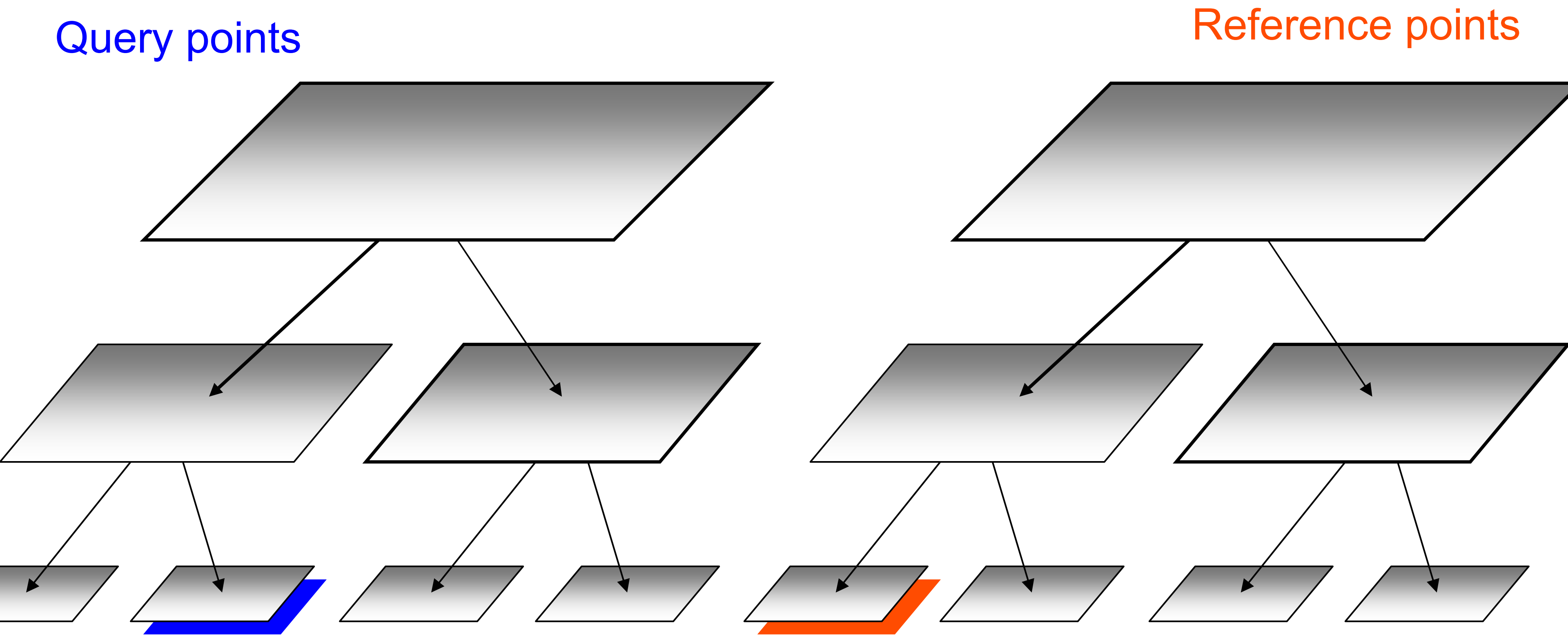
Reference points



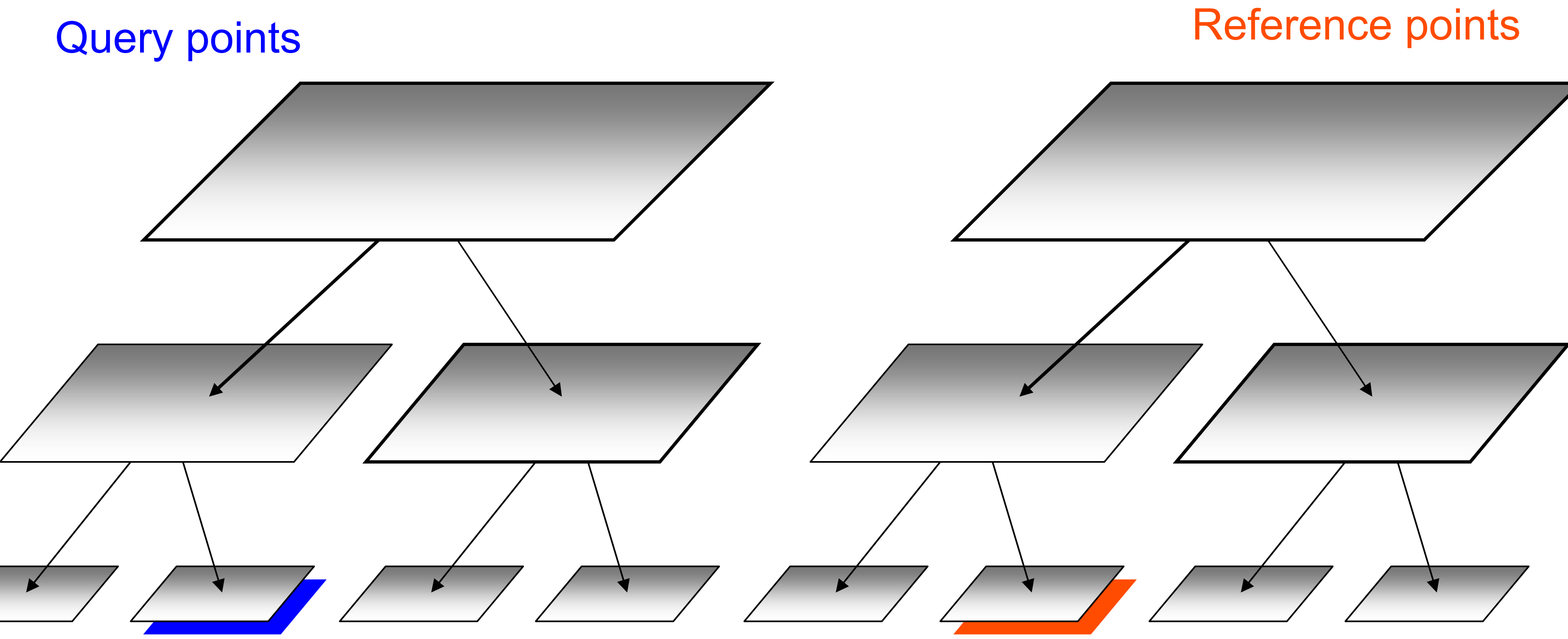
Dual-tree traversal



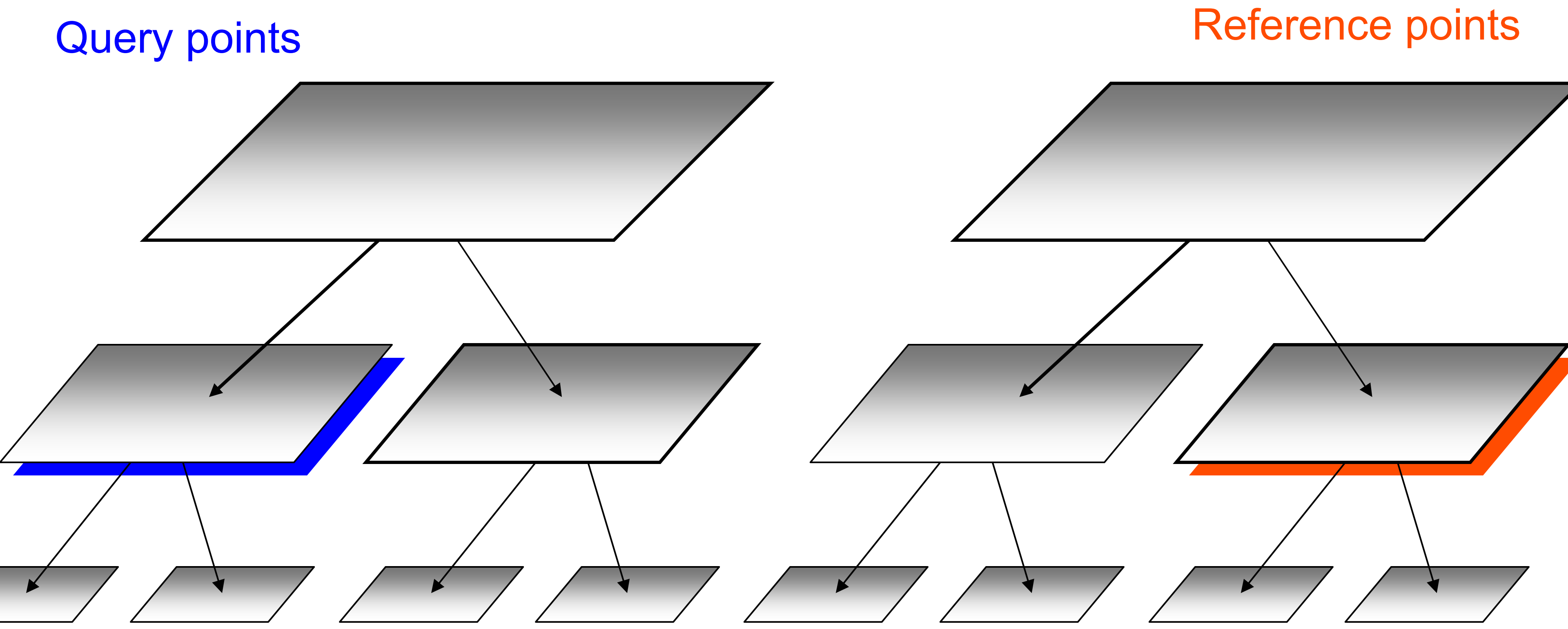
Dual-tree traversal



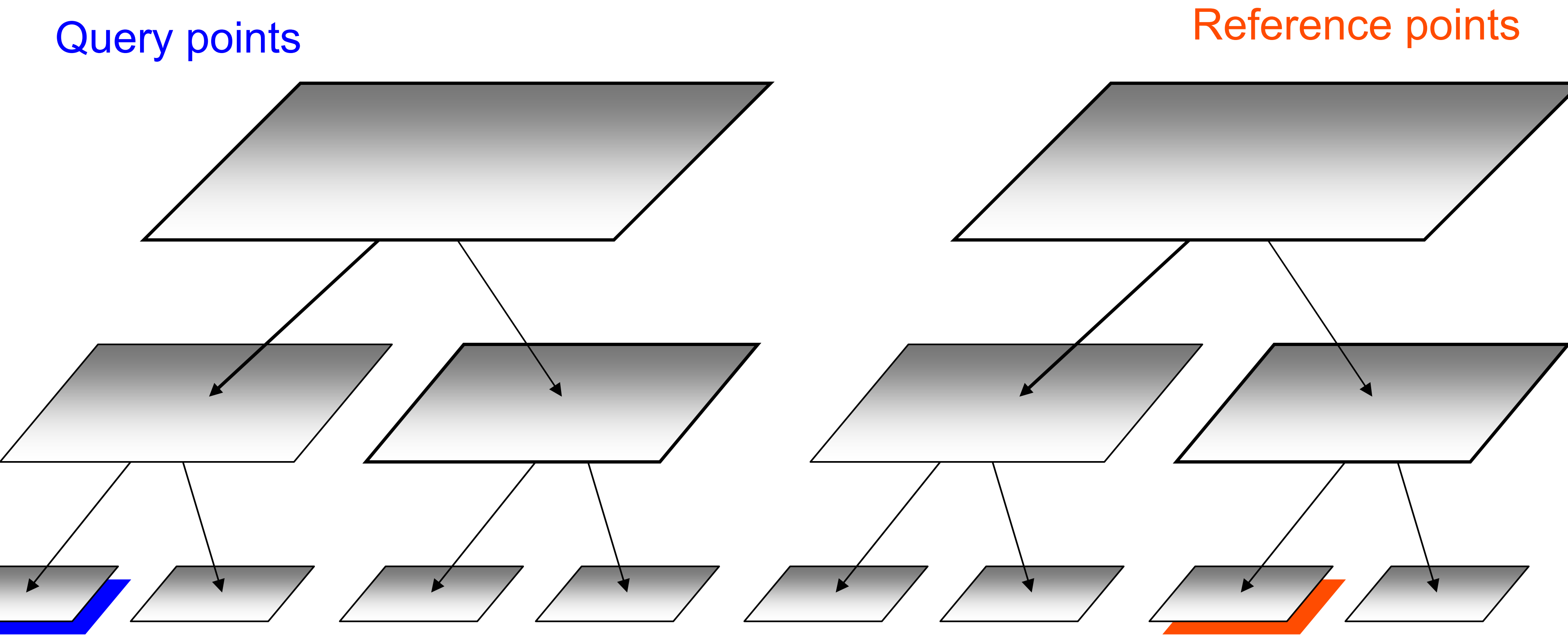
Dual-tree traversal



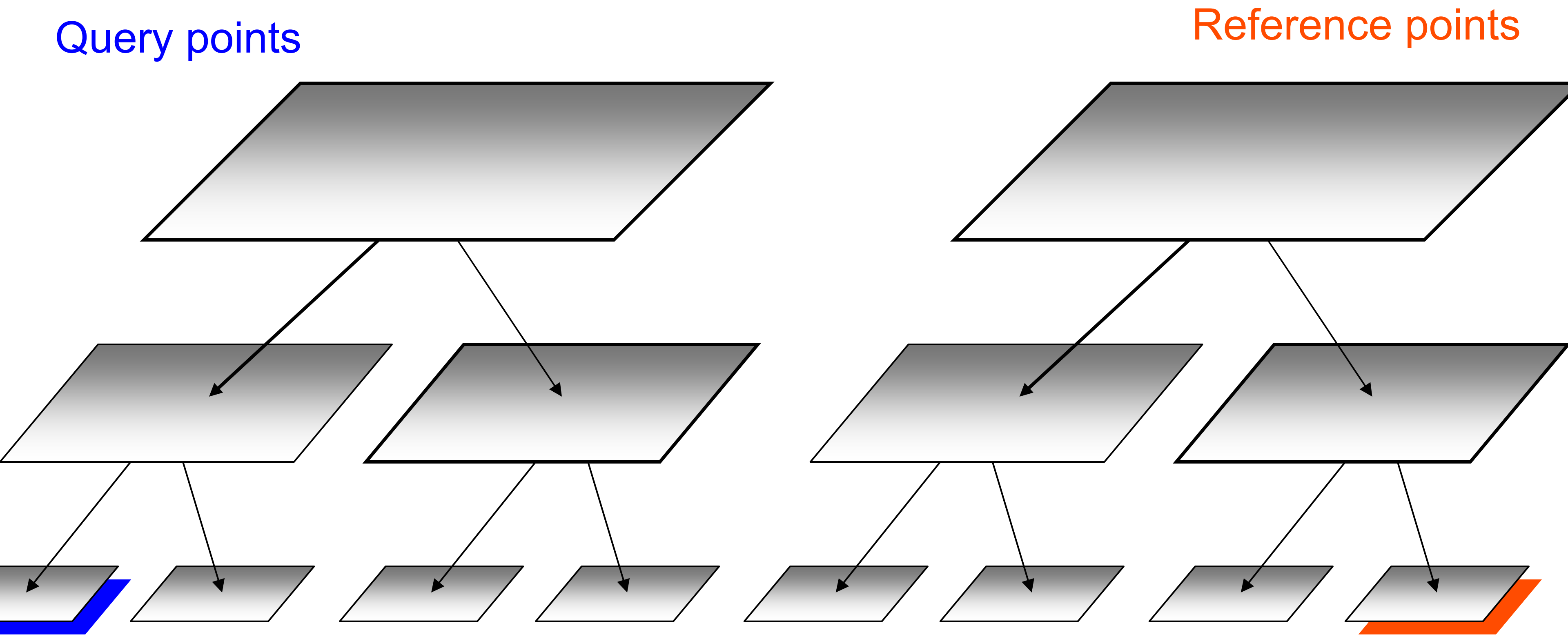
Dual-tree traversal



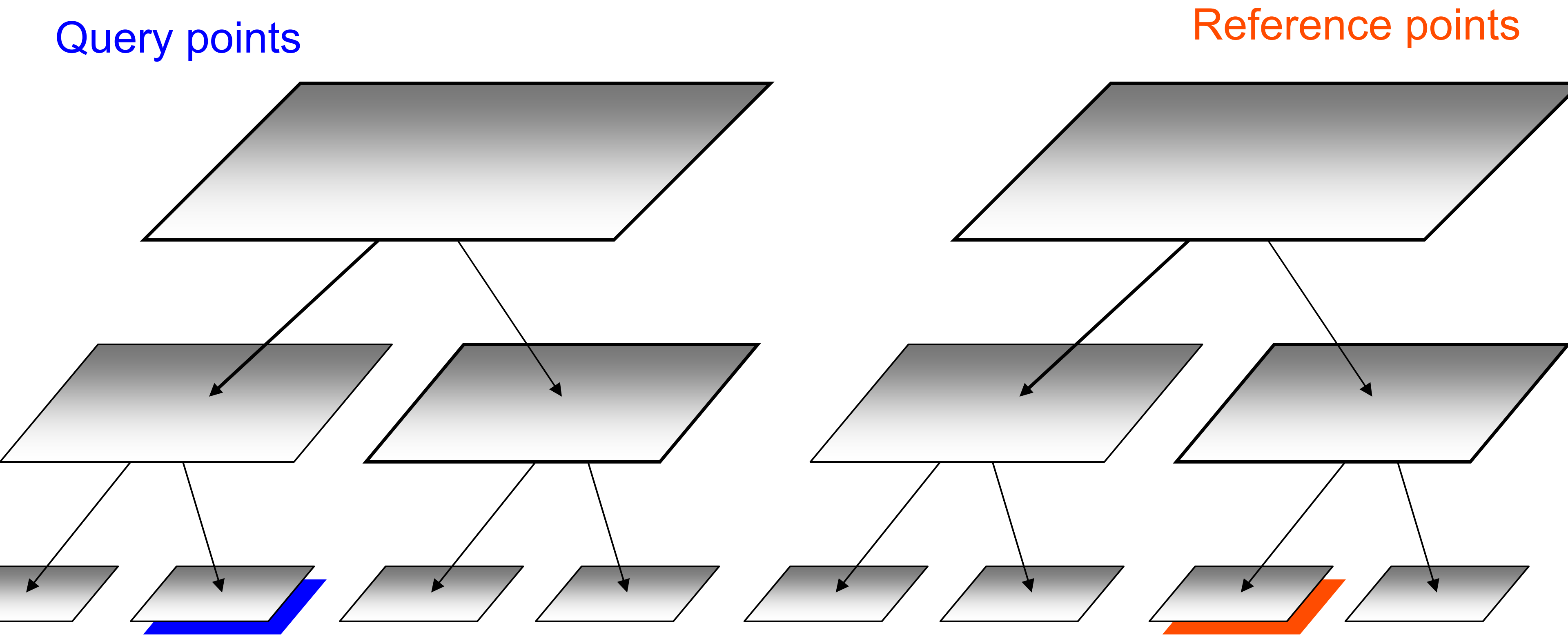
Dual-tree traversal



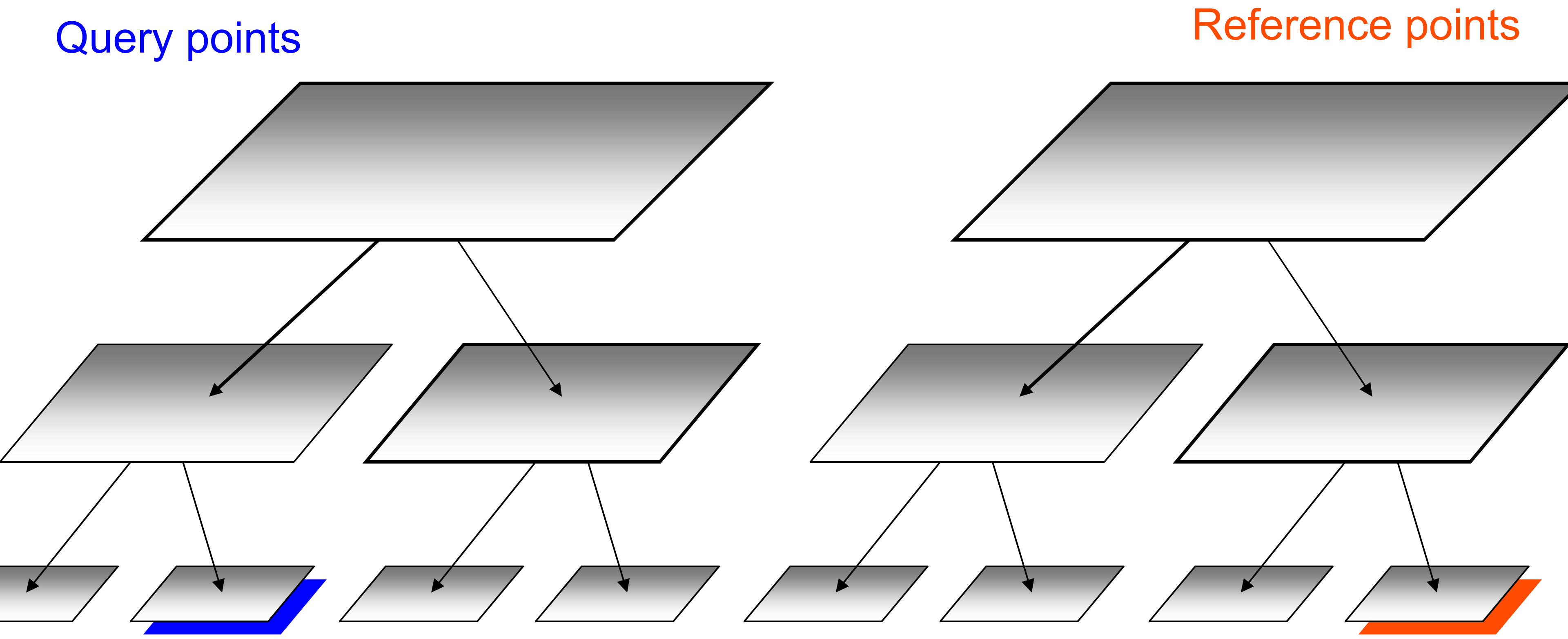
Dual-tree traversal



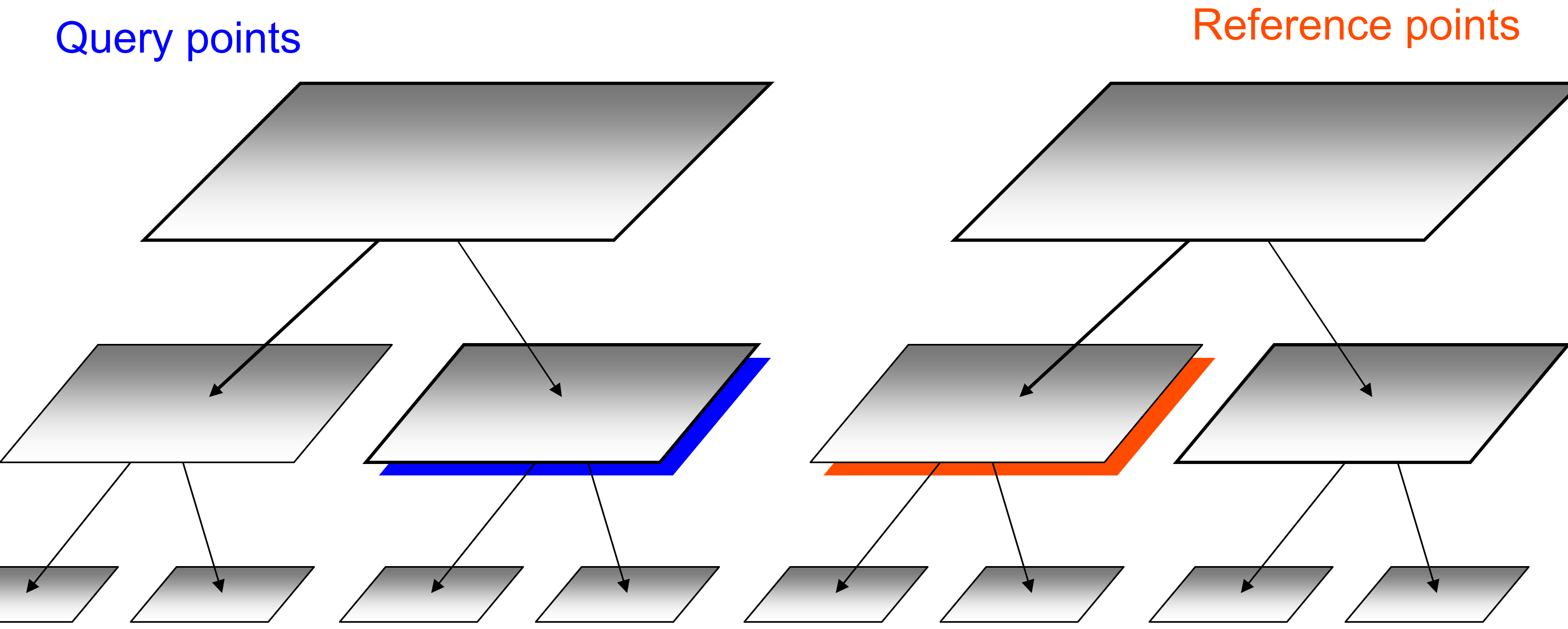
Dual-tree traversal



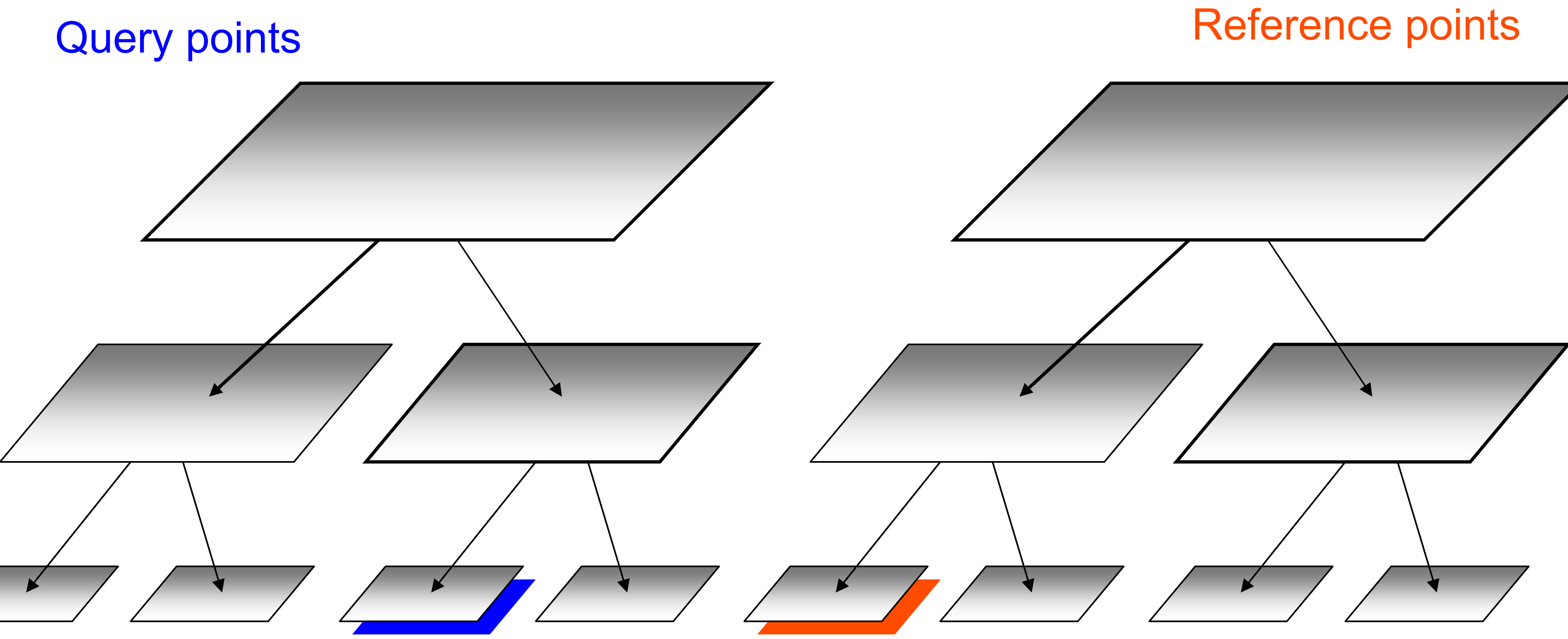
Dual-tree traversal



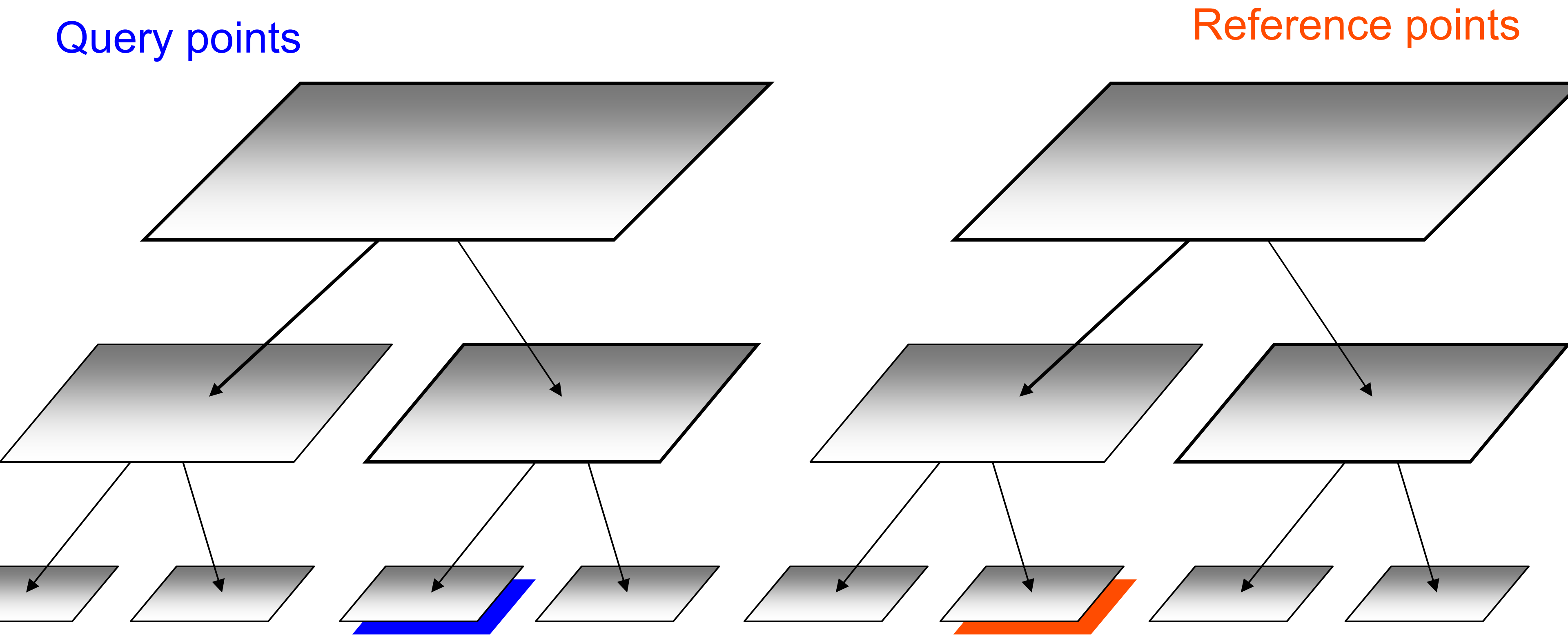
Dual-tree traversal



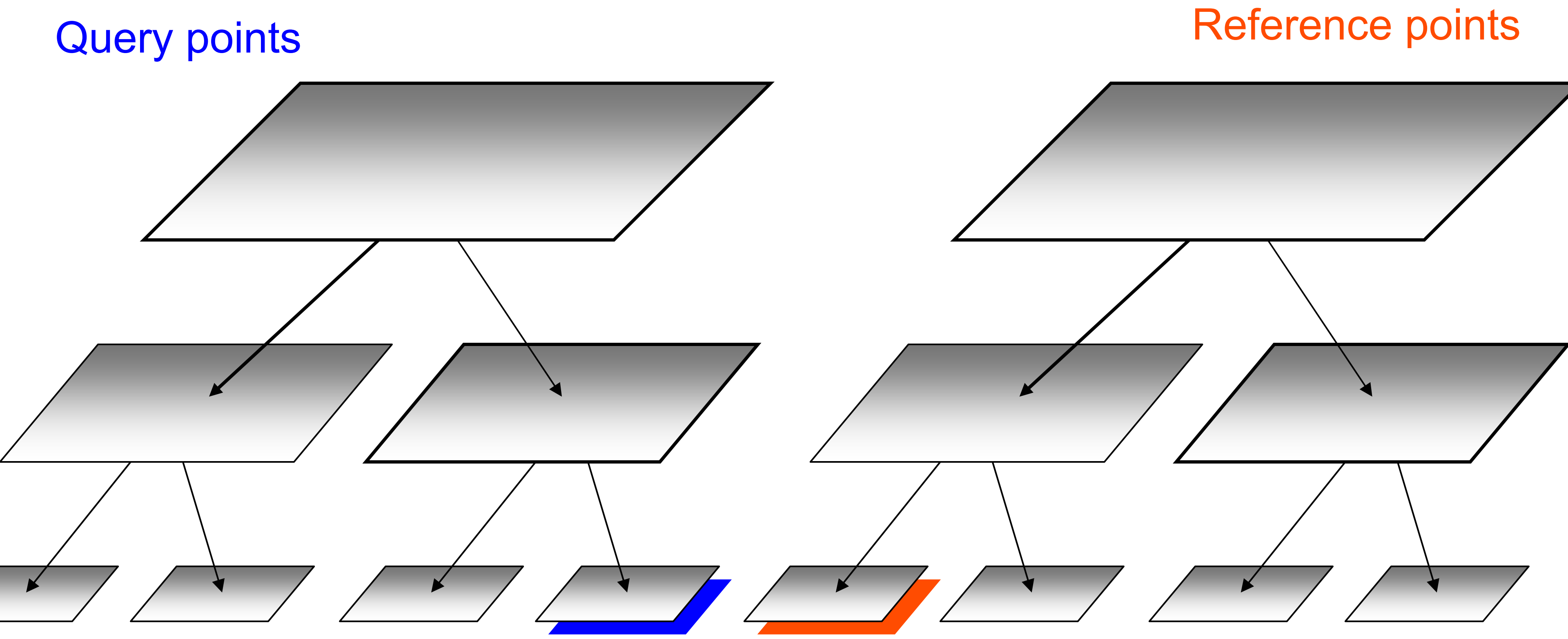
Dual-tree traversal



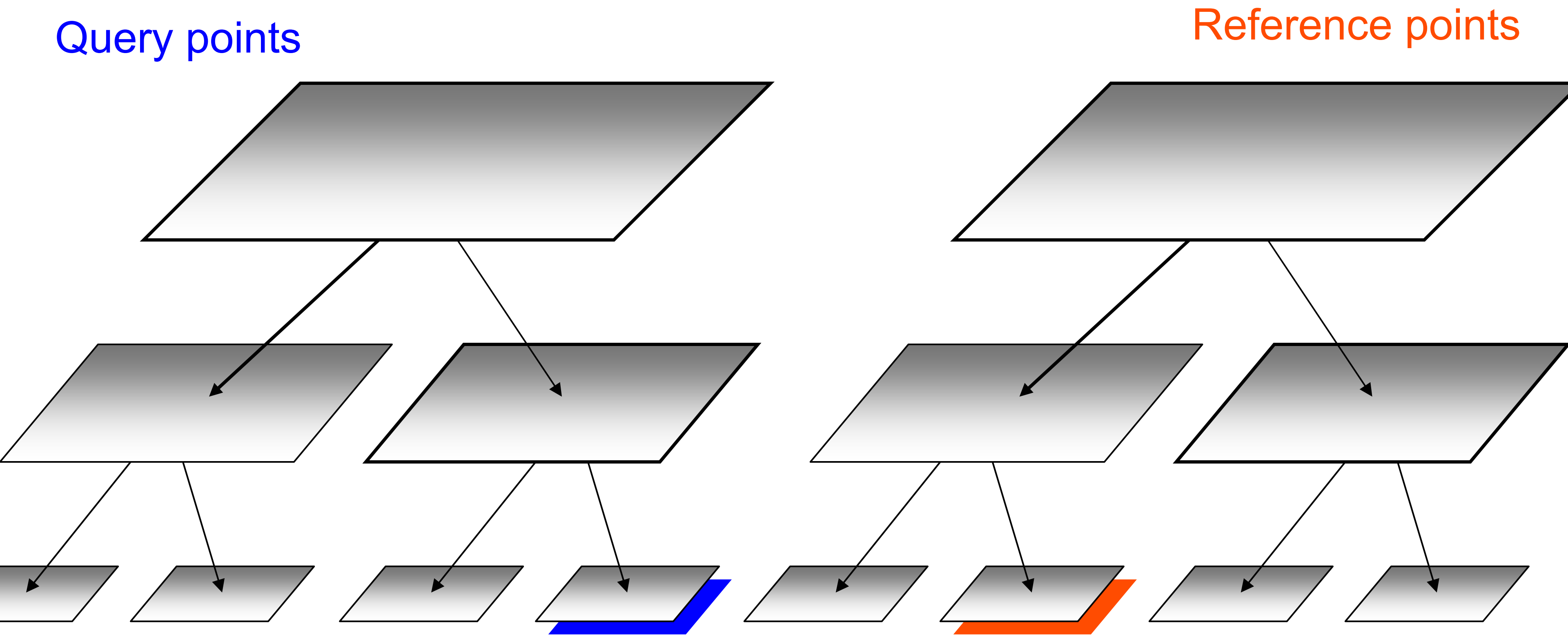
Dual-tree traversal



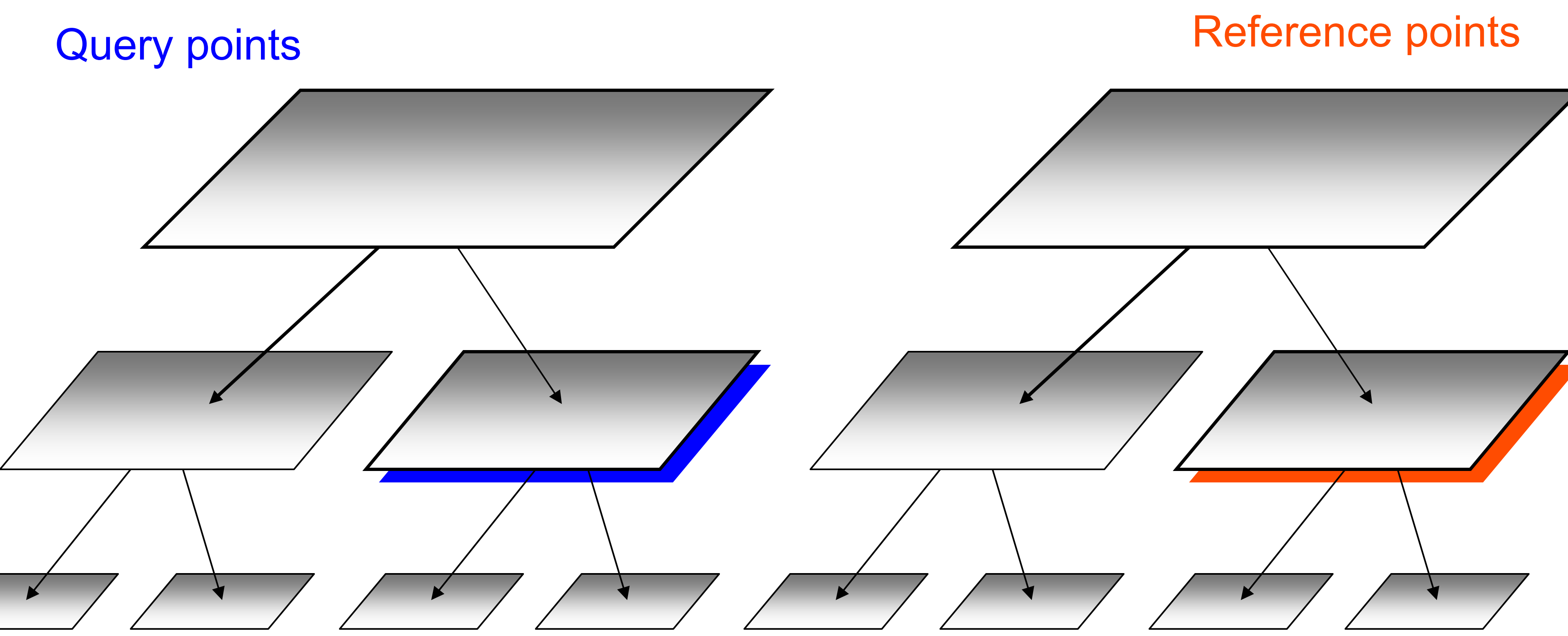
Dual-tree traversal



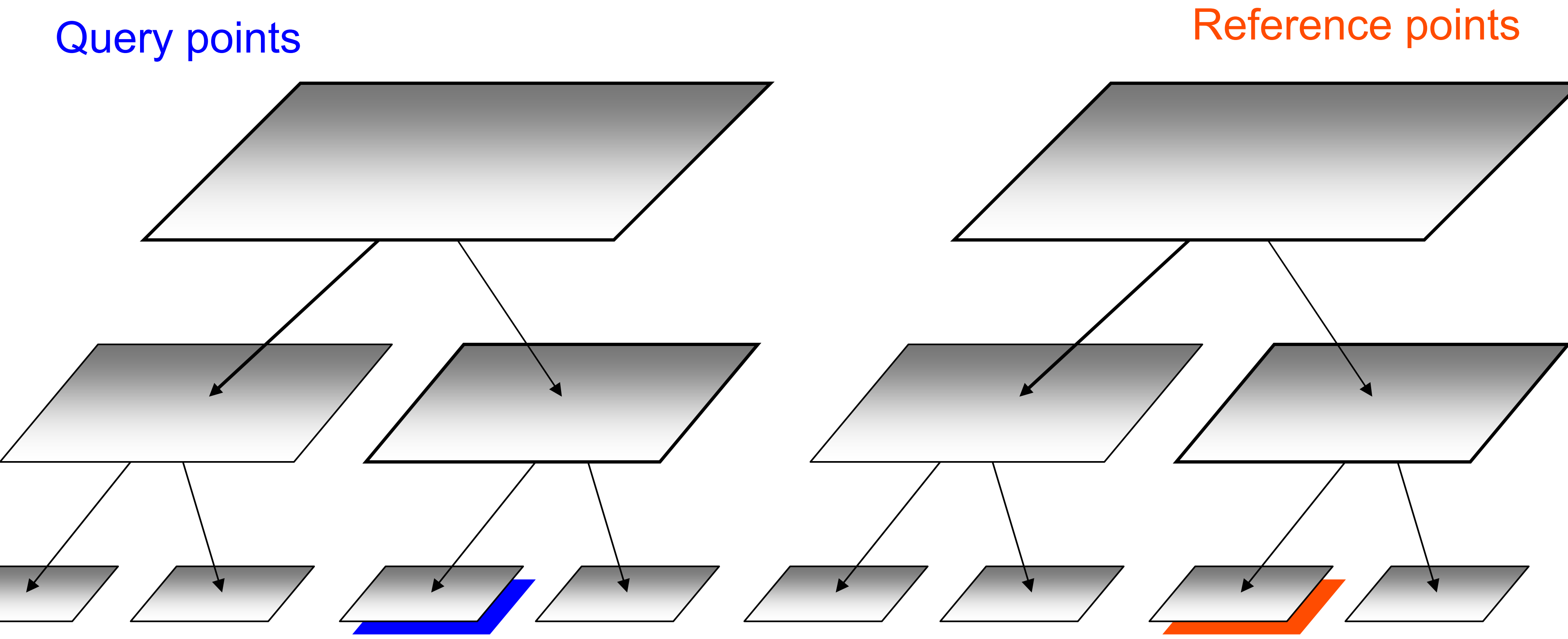
Dual-tree traversal



Dual-tree traversal



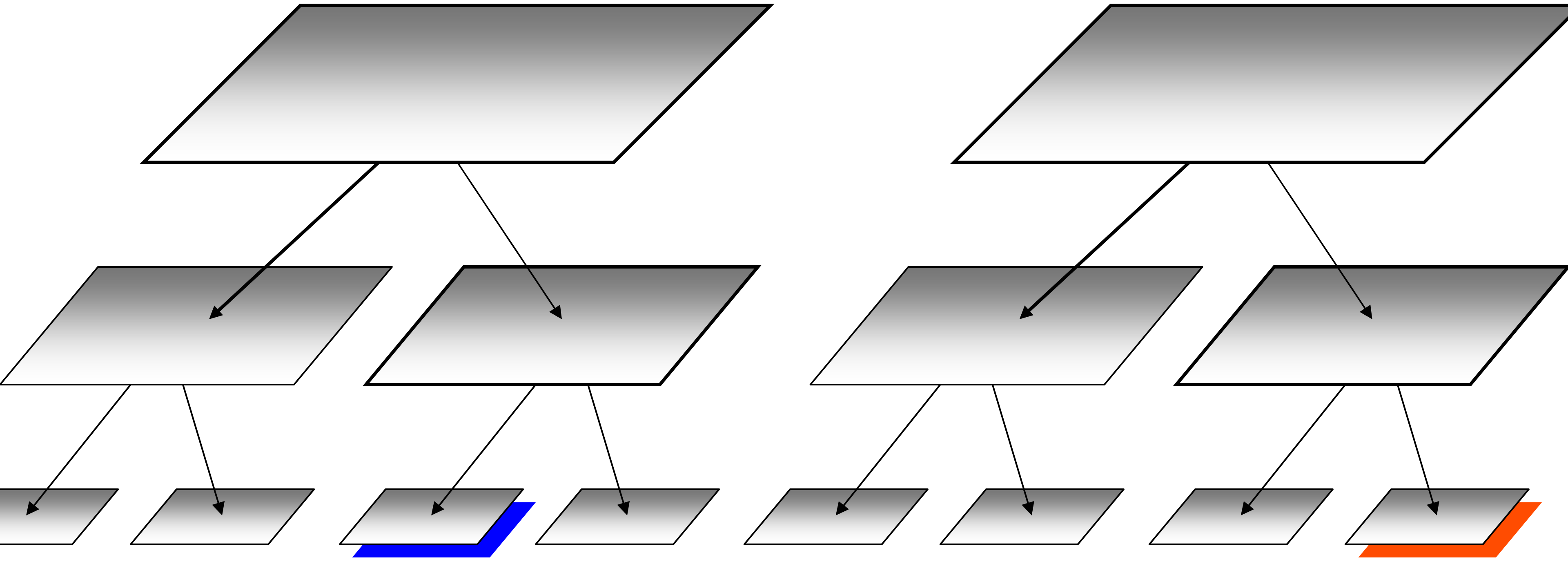
Dual-tree traversal



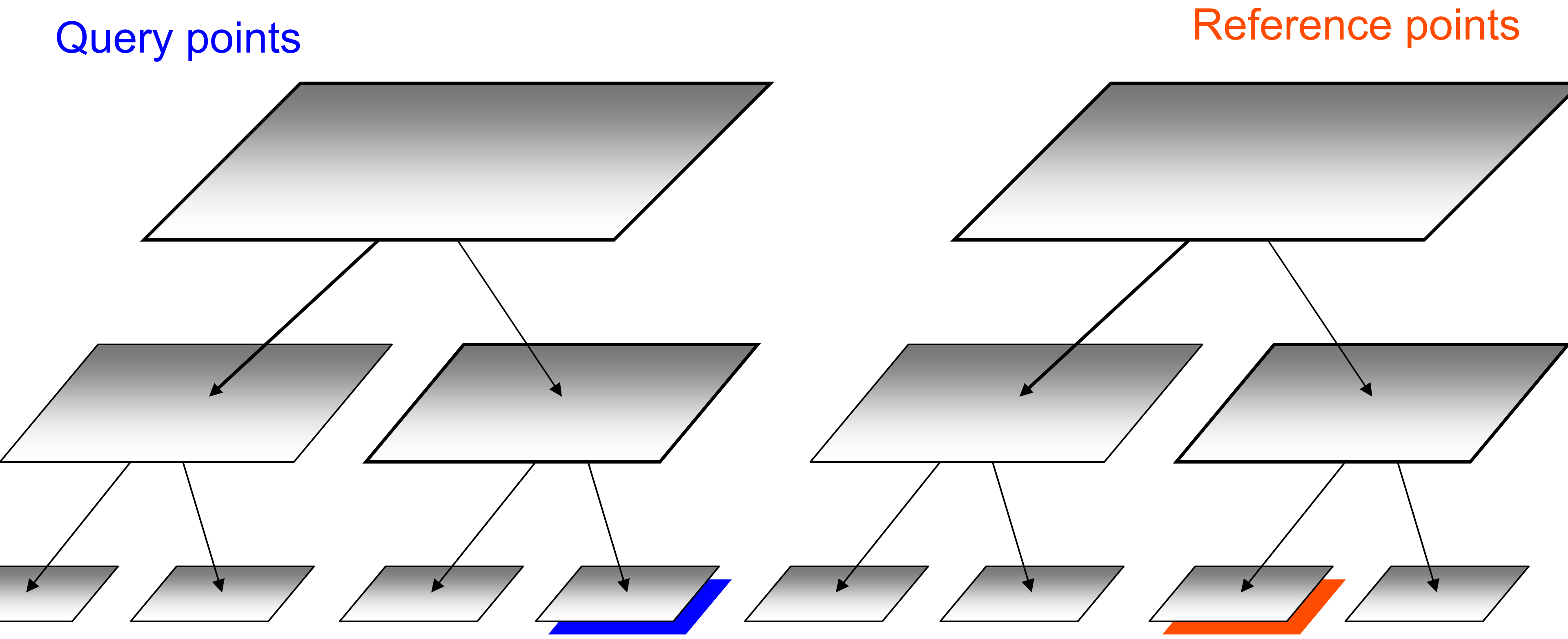
Dual-tree traversal

Query points

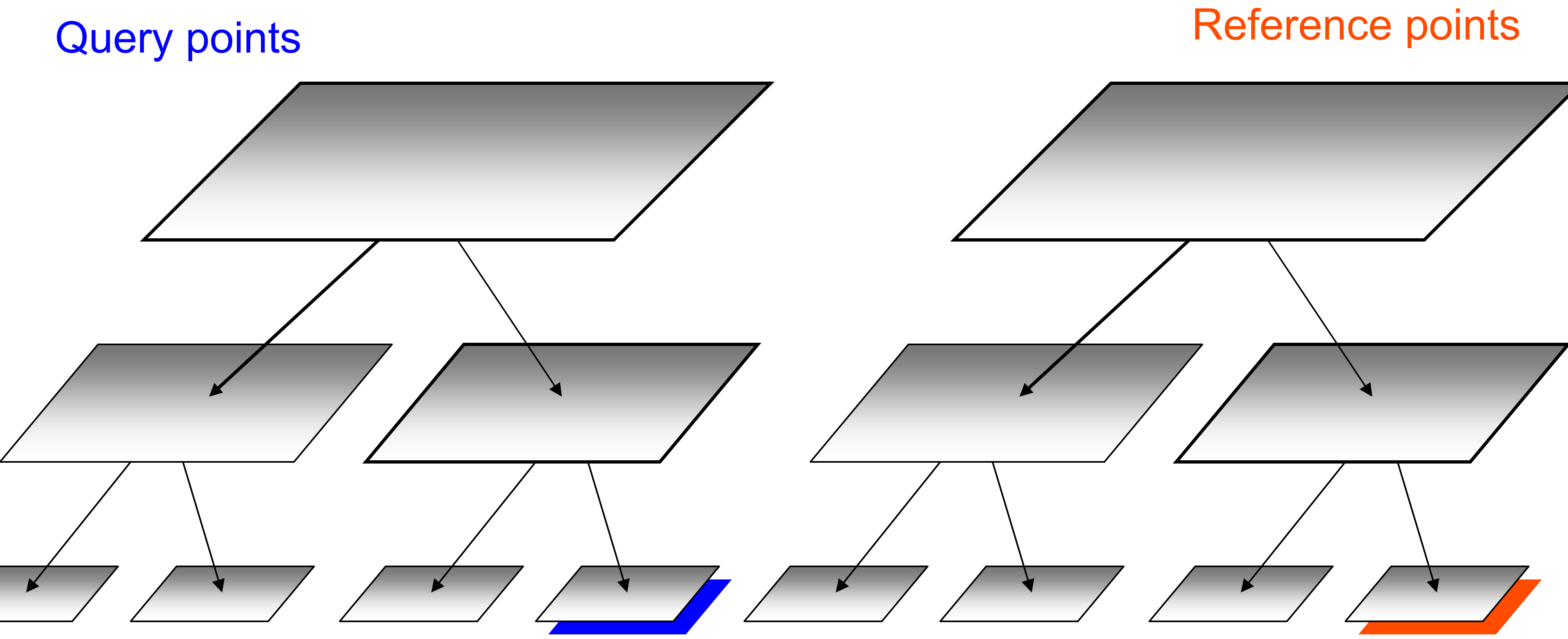
Reference points



Dual-tree traversal



Dual-tree traversal



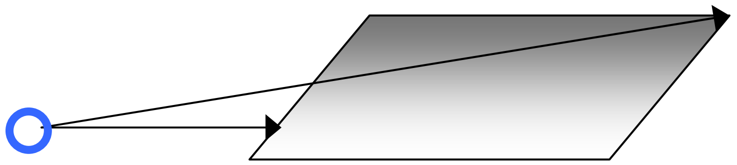
Finite-difference function approximation.

Taylor expansion:

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

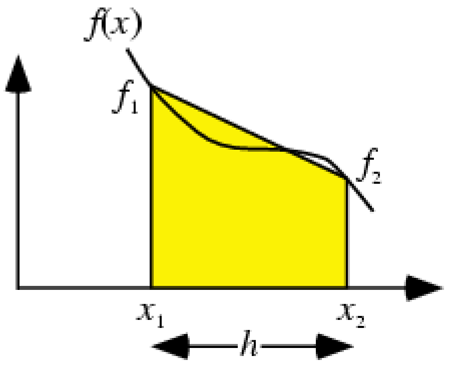
Gregory-Newton finite form:

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



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

Finite-difference function approximation.

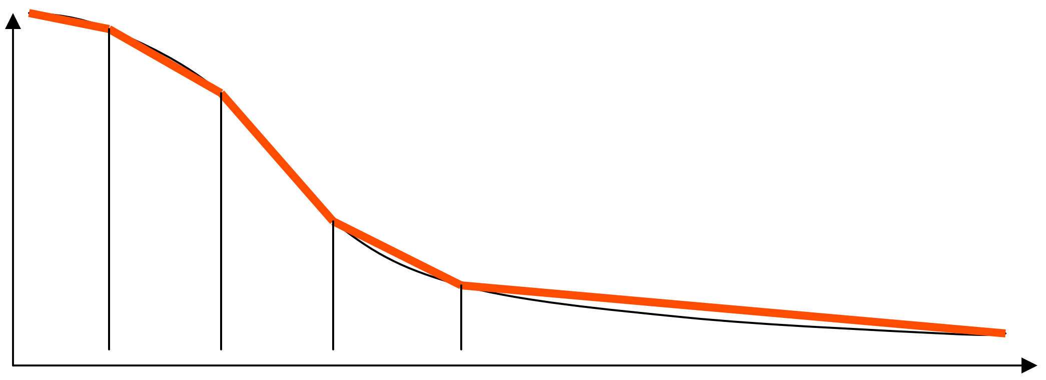


assumes monotonic decreasing kernel

$$\overline{K} = \tfrac{1}{2}\left[K(\delta_{QR}^{\min}) + K(\delta_{QR}^{\max})\right]$$

$$err_q = \sum_r^{N_R} \left| K(\delta_{qr}) - \overline{K} \right| \le \frac{N_R}{2}\left[K(\delta_{QR}^{\min}) - K(\delta_{QR}^{\max})\right]$$

could also use center of mass



Stopping rule?

Simple approximation method

```
**approximate**(Q,R)
{
    $dl = N_R K(\delta_{\max}), du = N_R K(\delta_{\min}).$

    if   $\delta_{\min} \geq \tau \cdot \max(diam(Q), diam(R))$

        incorporate(dl, du).
}
```

→trivial to change kernel
→hard error bounds

Big issue in practice…

Tweak parameters

Case 1 – algorithm gives no error bounds
Case 2 – algorithm gives hard error bounds: must run it many times
Case 3 – algorithm automatically achives your error tolerance

Automatic approximation method

```
**approximate**(Q,R)
{
    $dl = N_R K(\delta_{\max}), du = N_R K(\delta_{\min}).$

    if  $K(\delta_{\min}) - K(\delta_{\max}) \le \frac{2\varepsilon}{N} \phi_{\min}(Q)$

        incorporate(*dl*, *du*). return.
}
```

→just set error tolerance, no tweak parameters

→hard error bounds

Runtime analysis

THEOREM: Dual-tree algorithm is ***O(N)***

ASSUMPTION: *N* points from density *f*

$$0 < c \leq f \leq C$$

Recurrence for self-finding

single-tree (point-node)

$$T(N) = T(N/2) + O(1)$$
$$T(1) = O(1)$$
$$\Rightarrow N \cdot O(\log N)$$

dual-tree (node-node)

$$T(N) = 2T(N/2) + O(1)$$
$$T(1) = O(1)$$
$$\Rightarrow O(N)$$

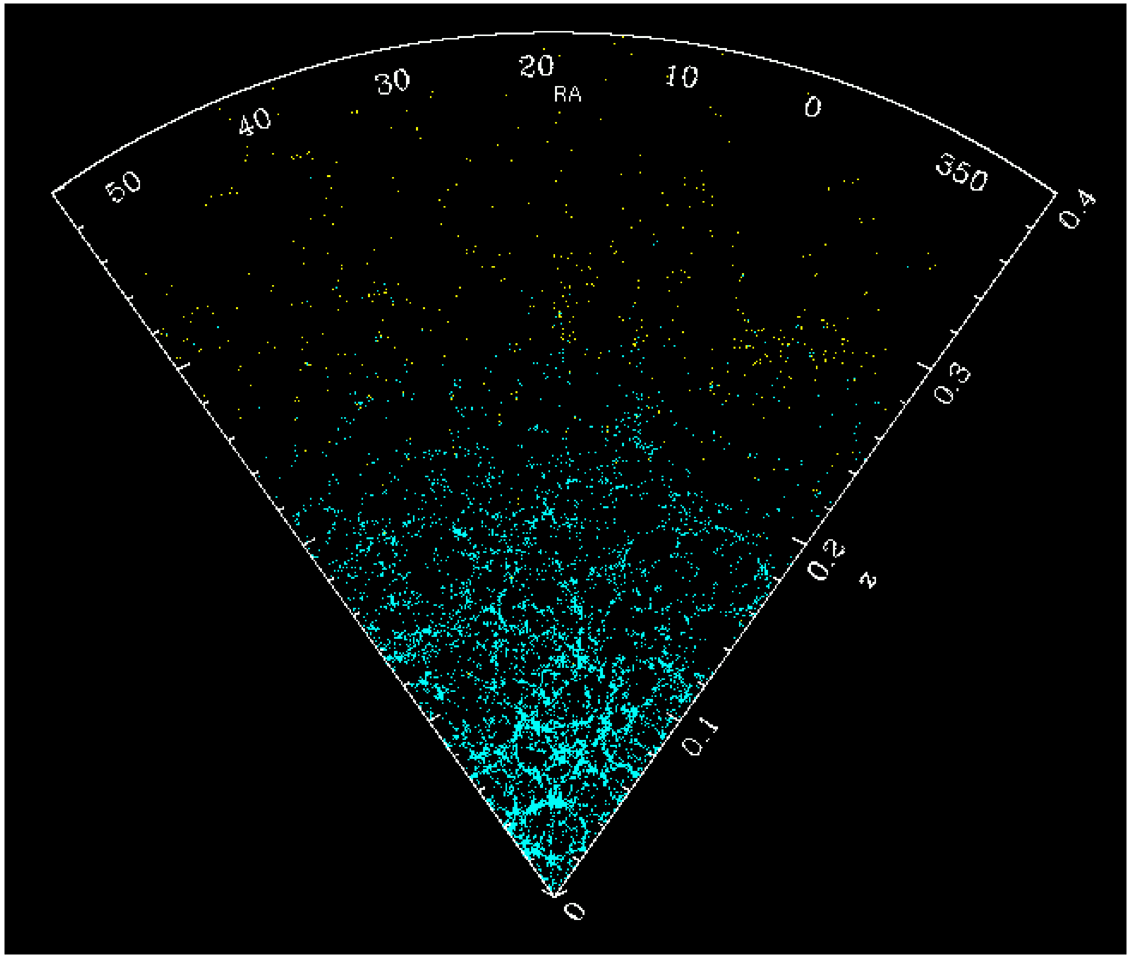
Packing bound

LEMMA: Number of nodes that are *well-separated* from a query node Q is bounded by a constant $\lceil 1 + g(s,c,C) \rceil^D$

Thus the recurrence yields the entire runtime. Done. (cf. [Callahan-Kosaraju 95])

On a manifold, use its dimension *D'* (the data's 'intrinsic dimension').

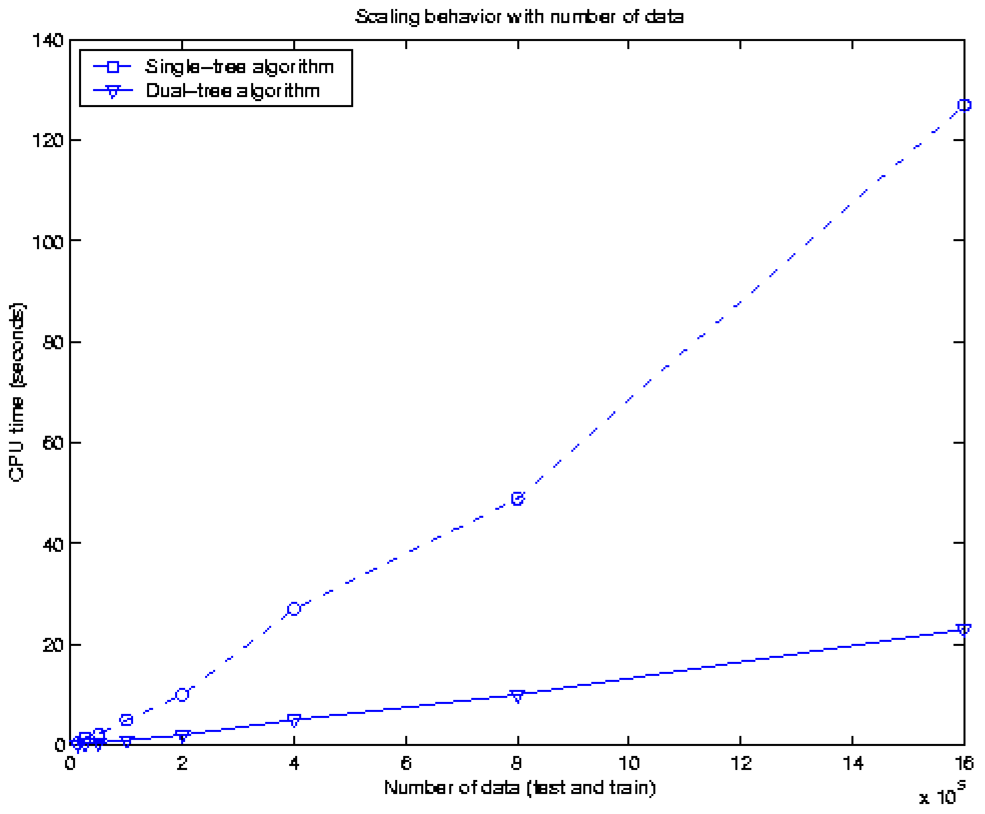
Real data: SDSS, 2-D



Speedup Results: Number of points

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

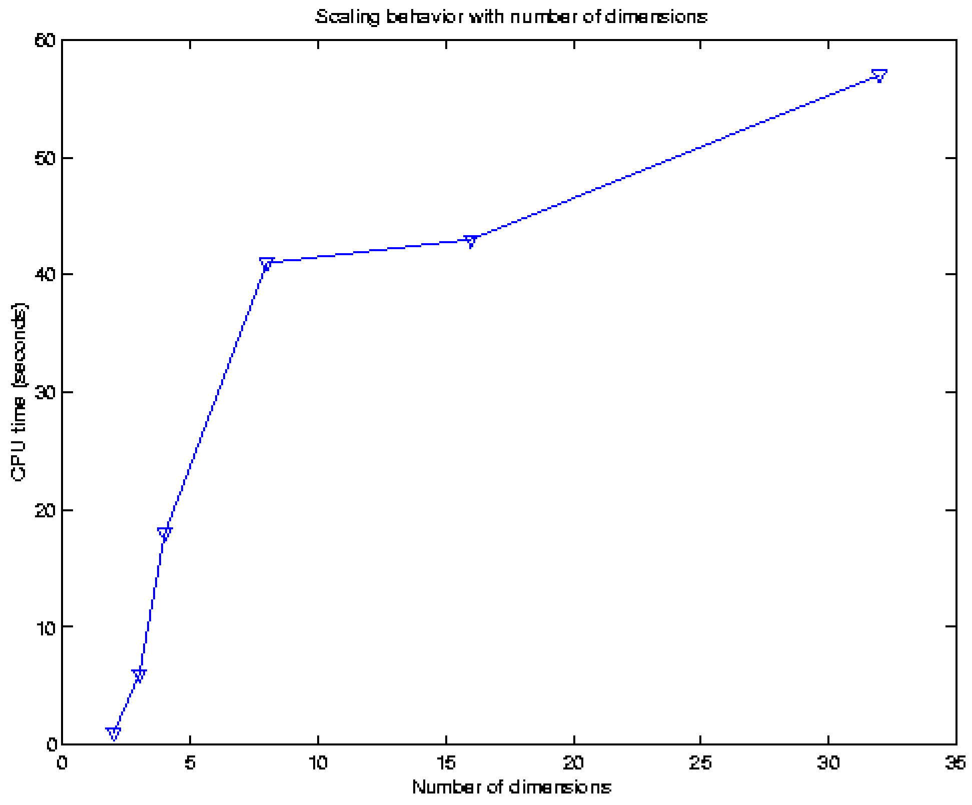
Speedup Results: Different kernels

N	Epan.	Gauss.
12.5K	.12	.32
25K	.31	.70
50K	.46	1.1
100K	1.0	2
200K	2	5
400K	5	11
800K	10	22
1.6M	23	51

Epanechnikov:
  10^{-6} relative error
Gaussian:
  10^{-3} relative error

Speedup Results: Dimensionality

N	Epan.	Gauss.
12.5K	.12	.32
25K	.31	.70
50K	.46	1.1
100K	1.0	2
200K	2	5
400K	5	11
800K	10	22
1.6M	23	51



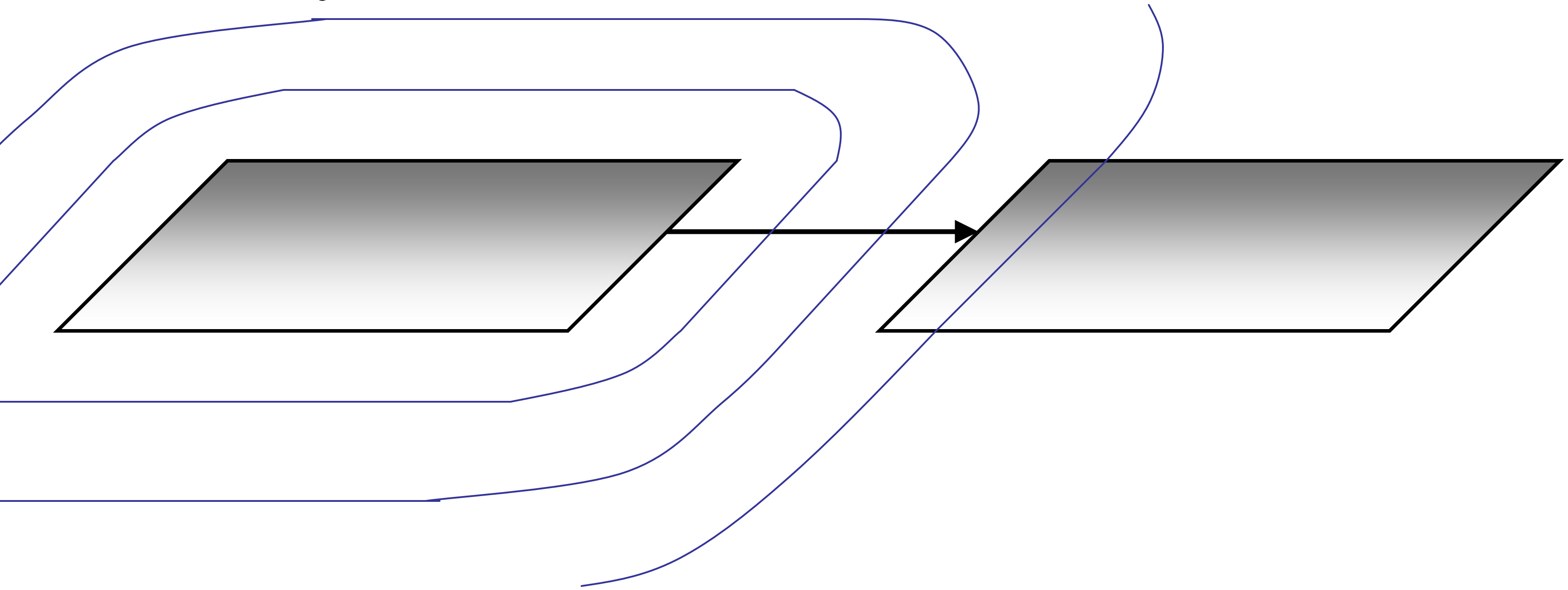
Speedup Results: Different datasets

Name	N	D	Time (sec)
Bio5	103K	5	10
CovType	136K	38	8
MNIST	10K	784	24
PSF2d	3M	2	9

Exclusion and inclusion, on multiple radii simultaneously.

Application of HODC principle

Use binary search to locate critical radius:

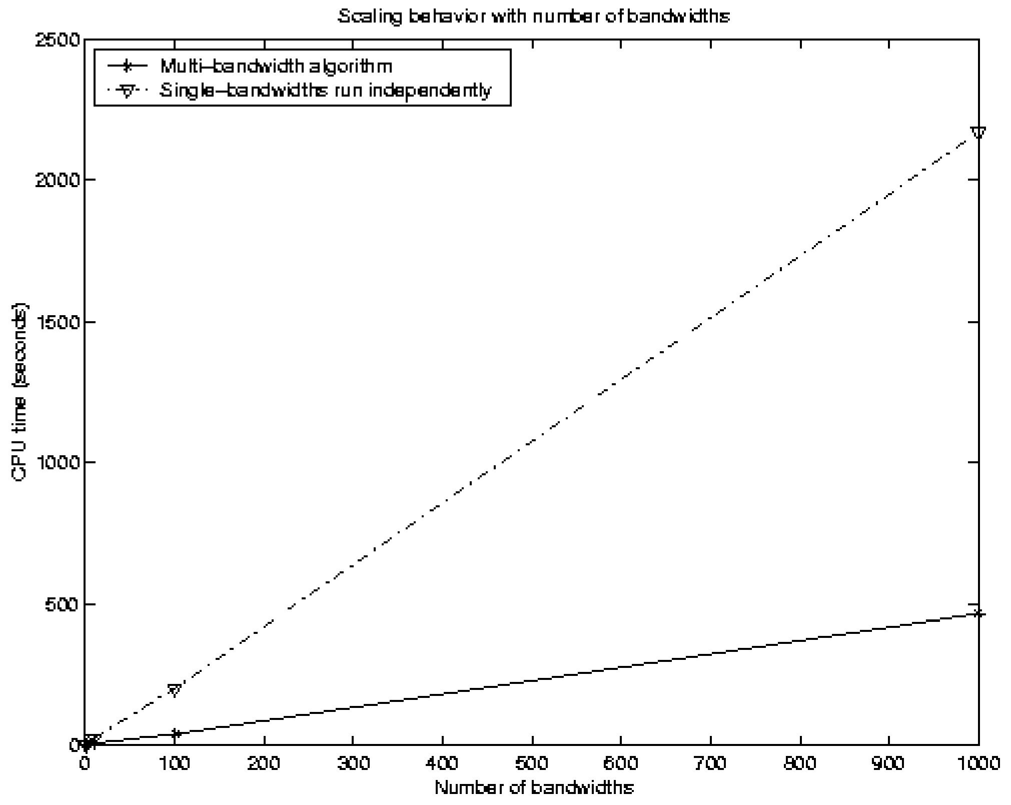


min||x-xi|| < h1 => min||x-xi|| < h2

Also needed:

b_lo,b_hi are arguments; store bounds for each b

Speedup Results



One order-of-magnitude speedup over single-radius at ~10,000 radii

Outline:

1. Physics problems and methods

2. Generalized N-body problems

3. Proximity data structures

4. Dual-tree algorithms

5. **Comparison**

Experiments

- Optimal bandwidth h* found by LSCV
- Error relative to truth: maxerr=max |est – true| / true
- Only require that 95% of points meet this tolerance
- Measure CPU time given this error level
- Note that these are small datasets for manageability
- Methods compared:
  - FFT
  - IFGT
  - Dual-tree (Gaussian)
  - Dual-tree (Epanechnikov)

Experiments: tweak parameters

- FFT tweak parameter M: M=16, double until error satisfied
- IFGT tweak parameters K, rx, ry, p: 1) K=√N, get rx, ry=rx; 2) K=10√N, get rx, ry=16 and doubled until error satisfied; hand-tune p for dataset: {8,8,5,3,2}
- Dualtree tweak parameter tau: tau=maxerr, double until error satisfied
- Dualtree auto: just give it maxerr

colors (N=50k, D=2)

	50%	10%	1%	Exact
Exhaustive	-	-	-	329.7 [111.0]
FFT	0.1	2.9	>Exhaust.	-
IFGT	1.7	>Exhaust.	>Exhaust.	-
Dualtree (Gaussian)	12.2 (65.1*)	18.7 (89.8*)	24.8 (117.2*)	-
Dualtree (Epanech.)	6.2 (6.7*)	6.5 (6.7*)	6.7 (6.7*)	58.2

sj2 (N=50k, D=2)

	50%	10%	1%	Exact
Exhaustive	-	-	-	301.7 [109.2]
FFT	3.1	>Exhaust.	>Exhaust.	-
IFGT	12.2	>Exhaust.	>Exhaust.	-
Dualtree (Gaussian)	2.7 (3.1*)	3.4 (4.8*)	3.8 (5.5*)	-
Dualtree (Epanech.)	0.8 (0.8*)	0.8 (0.8*)	0.8 (0.8*)	6.5

bio5 (N=100k, D=5)

	50%	10%	1%	Exact
Exhaustive	-	-	-	1966.3 [1074.9]
FFT	>Exhaust.	>Exhaust.	>Exhaust.	-
IFGT	>Exhaust.	>Exhaust.	>Exhaust.	-
Dualtree (Gaussian)	72.2 (98.8*)	79.6 (111.8*)	87.5 (128.7*)	-
Dualtree (Epanech.)	27.0 (28.2*)	28.4 (28.4*)	28.4 (28.4*)	408.9

corel (N=38k, D=32)

	50%	10%	1%	Exact
Exhaustive	-	-	-	710.2 [558.7]
FFT	>Exhaust.	>Exhaust.	>Exhaust.	-
IFGT	>Exhaust.	>Exhaust.	>Exhaust.	-
Dualtree (Gaussian)	155.9 (159.7*)	159.9 (163*)	162.2 (167.6*)	-
Dualtree (Epanech.)	10.0 (10.0*)	10.1 (10.1*)	10.1 (10.1*)	261.6

covtype (N=150k, D=38)

	50%	10%	1%	Exact
Exhaustive	-	-	-	13157.1 [11486.0]
FFT	>Exhaust.	>Exhaust.	>Exhaust.	-
IFGT	>Exhaust.	>Exhaust.	>Exhaust.	-
Dualtree (Gaussian)	139.9 (143.6*)	140.4 (145.7*)	142.7 (148.6*)	-
Dualtree (Epanech.)	54.3 (54.3*)	56.3 (56.3*)	56.4 (56.4*)	1572.0

Myths

Multipole expansions are needed to:

1. Achieve *O(N)*
2. Achieve high accuracy
3. Have hard error bounds

Generalized N-body solutions: **Multi-tree methods**

- Higher-order divide-and-conquer: generalizes divide-and-conquer to multiple sets
- Each set gets a space-partitioning tree
- Recursive with anytime bounds
- Generalized auto-approximation rule

[Gray PhD thesis 2003], [Gray 2005]

Tricks for different N-body problems

- All-k-NN, bichromatic (Gray & Moore 2000, Gray, Lee, Rotella, Moore 2005): vanilla
- Kernel density estimation (Gray & Moore 2000, 2003abc): multiple bandwidths
- Gaussian process regression (Gray CMU-TR 2003): error bound is crucial
- Nonparametric Bayes classifiers (Gray et al. 2005): possible to get exact predictions
- n-point correlation (Gray & Moore 2000, 2004): n-tuples > pairs are possible; Monte Carlo for large radii

Discussion

- Related ideas: WSPD, spatial join, Appel’s algorithm
- FGT with a tree: coming soon
- Auto-approx FGT with a tree: unclear how to do this

Summary

- Statistics problems have their own properties, and benefit from a **fundamentally rethought methodology**
- *O(N)* can be achieved **without multipole expansions;** via **geometry**
- **Hard anytime error bounds** are given to the user
- **Tweak parameters** should and can be eliminated
- Very **general** methodology
- Future work: tons (even in physics)

→ Looking for comments and collaborators!
agray@cs.cmu.edu

THE END

Simple recursive algorithm

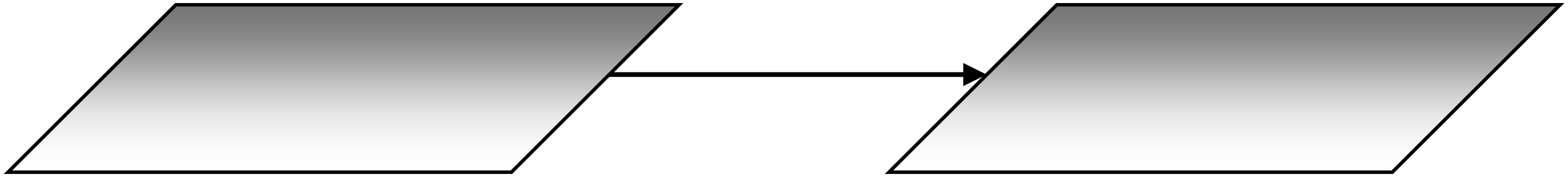
```
DualTree(Q,R)
{
  if approximate(Q,R), return.

  if leaf(Q) and leaf(R), DualTreeBase(Q,R).
  else,
    DualTree(Q.left,closer-of(R.left,R.right)).
    DualTree(Q.left,farther-of(R.left,R.right)).
    DualTree(Q.right,closer-of(R.left,R.right)).
    DualTree(Q.right,farther-of(R.left,R.right)).
}
```

(Actually, recurse on the closer node first)

Exclusion and inclusion,

using *kd*-tree node-node bounds.

O(D) bounds on distance minima/maxima:



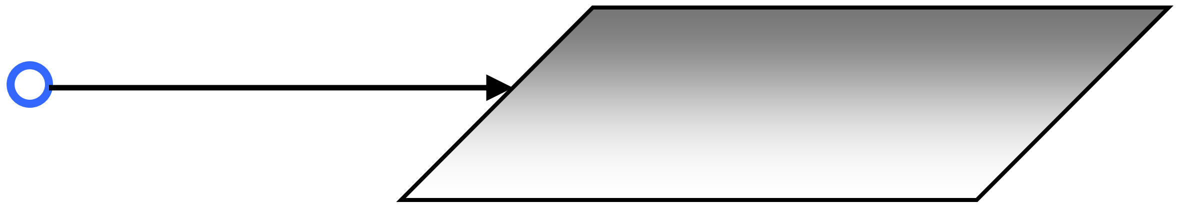
(Analogous to point-node bounds.)

Also needed:

Nodewise bounds.

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

O(D) bounds on distance minima/maxima:



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

$$\max_i \|x - x_i\| \leq \sum_d^D \max\left\{(u_d - x_d)^2, (x_d - l_d)^2\right\}$$