New Lightweight N-body Algorithms

Alexander Gray
School of Computer Science
Carnegie Mellon University

N-body problems

- **Coulombic**

$$K(x, x_i) = \frac{m m_i}{\left\| x - x_i \right\|^a}$$

(high accuracy required)

N-body problems

- **Coulombic**

$$K(x, x_i) = \frac{mm_i}{\left\| x - x_i \right\|^a}$$

(high accuracy required)

- **Nonparametric statistics**

$$K(x, x_i) = e^{-\left\| x - x_i \right\|^2 / 2\sigma^2}$$

$$t = \left\| x - x_i \right\|^2 / \sigma^2$$

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

(only moderate accuracy required, often high-D)

N-body problems

- **Coulombic**

  (high accuracy required)

$$K(x,x_i)=\frac{mm_i}{\left\|x-x_i\right\|^a}$$

- **Nonparametric statistics**

$$K(x,x_i)=e^{-\left\|x-x_i\right\|^2/2\sigma^2}$$

$$t=\left\|x-x_i\right\|^2/\sigma^2$$

$$K(x,x_i)=\begin{cases}1-t^{2a} & 0\le t<1\\ 0 & t\ge 1\end{cases}$$

  (only moderate accuracy required, often high-D)

- **SPH** (smoothed particle hydrodynamics)

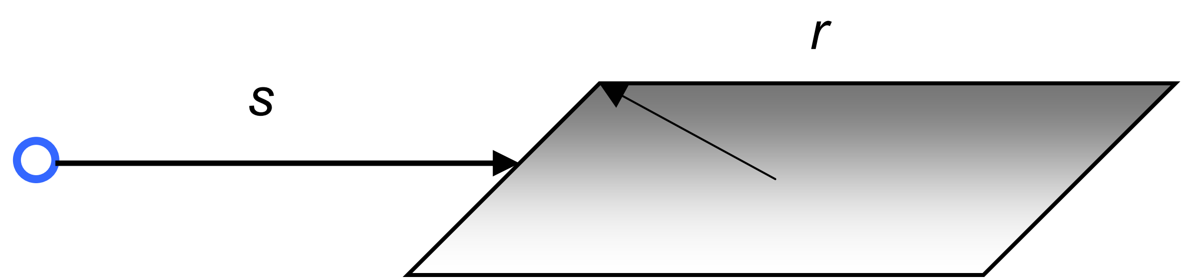
$$K(x,x_i)=\begin{cases}4-6t^2+3t^3 & 0\le t<1\\ (2-t)^3 & 1\le t<2\\ 0 & t\ge 2\end{cases}$$

  (only moderate accuracy required)

Also: different for every point, non-isotropic, edge-dependent, …

N-body methods: Approximation

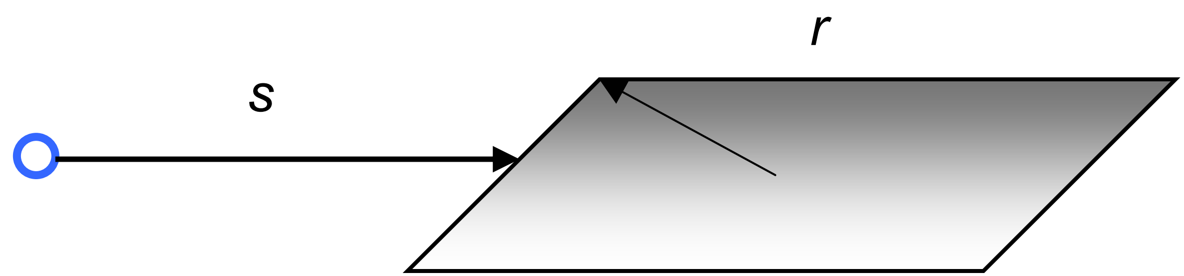
- **Barnes-Hut**



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

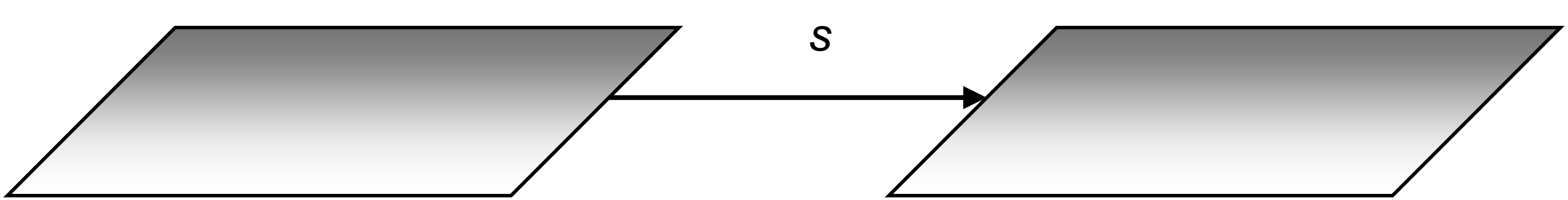
N-body methods: Approximation

- **Barnes-Hut**



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

- **FMM**



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

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

Expansions

- *Constants matter!* p^D factor is slowdown
- Large dimension infeasible
- Adds much complexity (software, human time)
- Non-trivial to do new kernels (assuming they're even analytic), heterogeneous kernels

Expansions

- *Constants matter!* p^D factor is slowdown
- Large dimension infeasible
- Adds much complexity (software, human time)
- Non-trivial to do new kernels (assuming they're even analytic), heterogeneous kernels
- BUT: Needed to achieve *O(N)*
  Needed to achieve high accuracy
  Needed to have hard error bounds

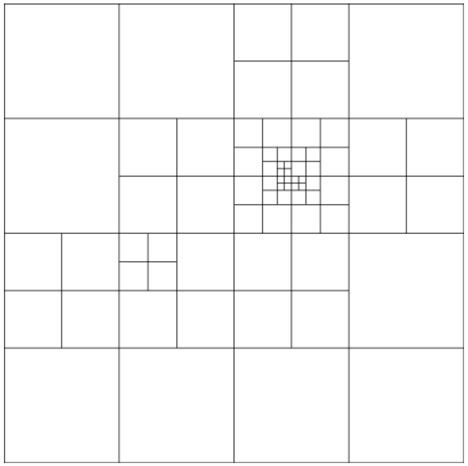
Expansions

- *Constants matter!* p^D factor is slowdown
- Large dimension infeasible
- Adds much complexity (software, human time)
- Non-trivial to do new kernels (assuming they're even analytic), heterogeneous kernels
- BUT: Needed to achieve *O(N)* (?)
  Needed to achieve high accuracy (?)
  Needed to have hard error bounds (?)

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

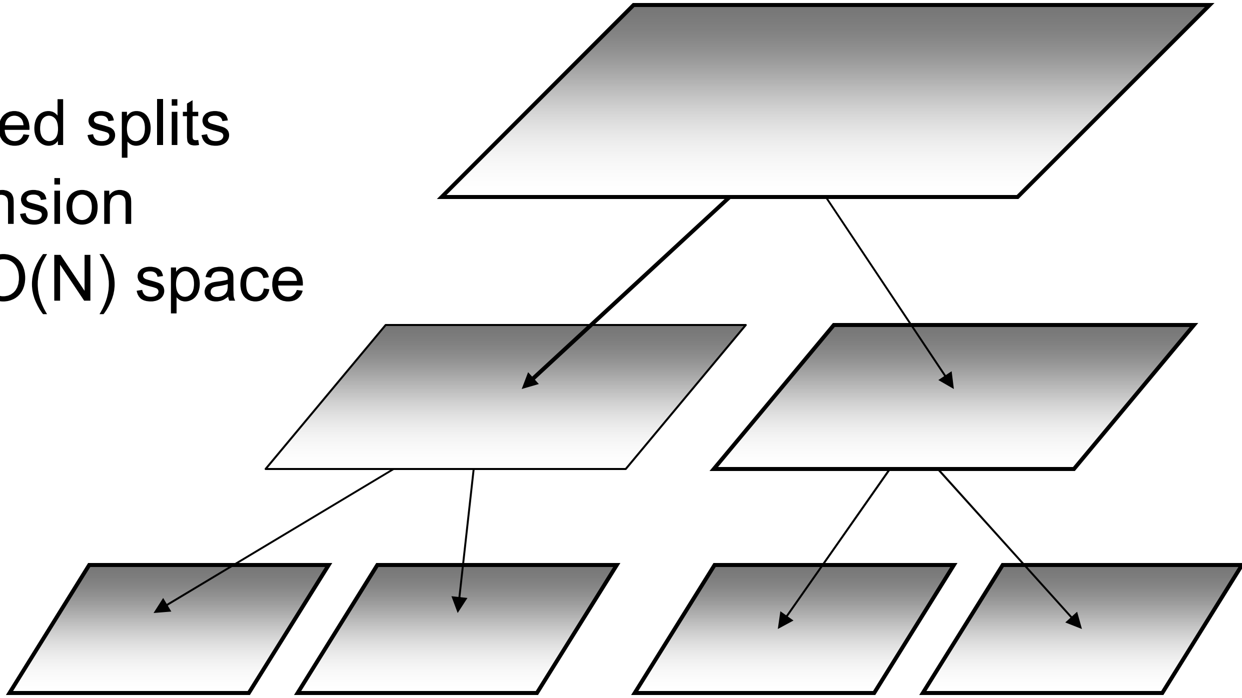


kd-trees:

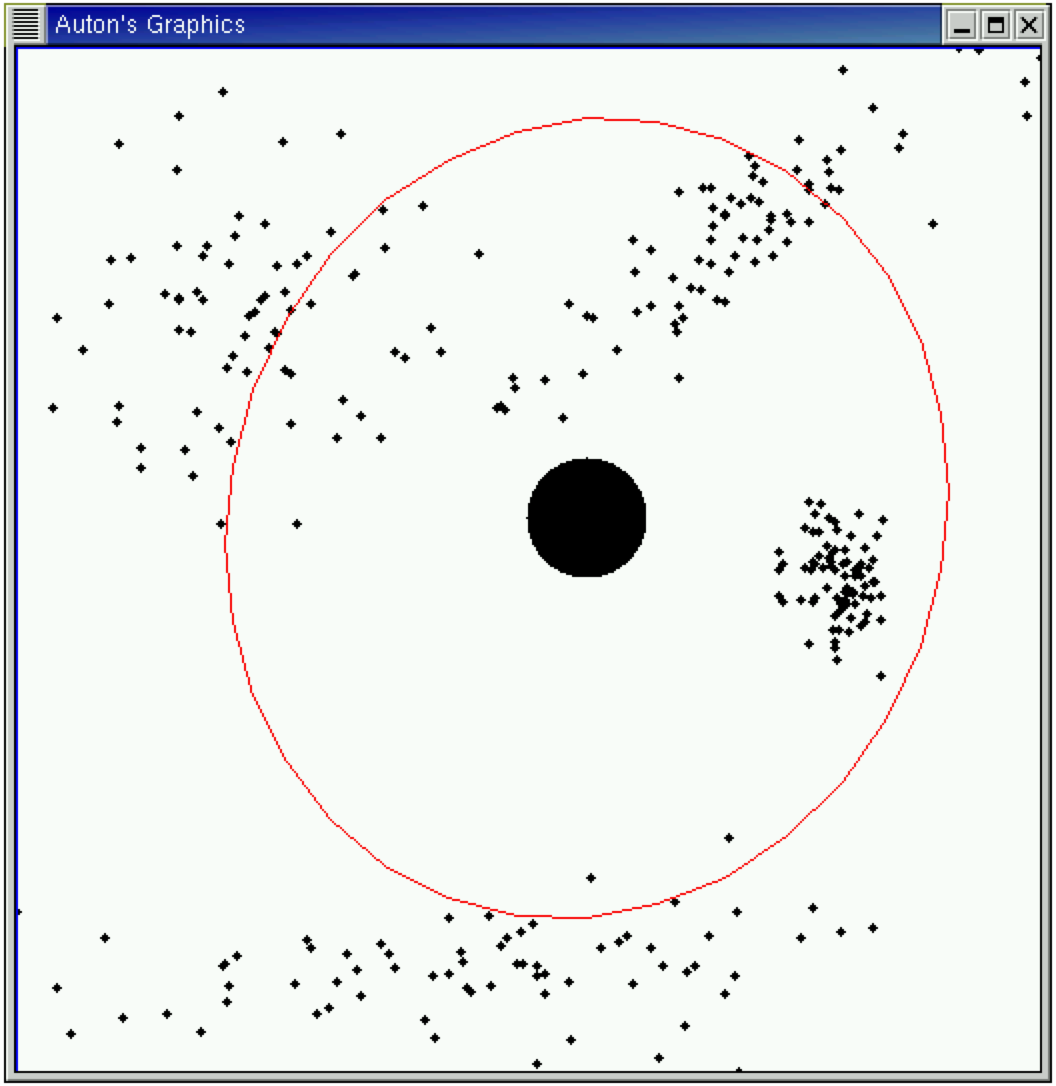
most widely-used space-partitioning tree

[Friedman, Bentley & Finkel 1977]

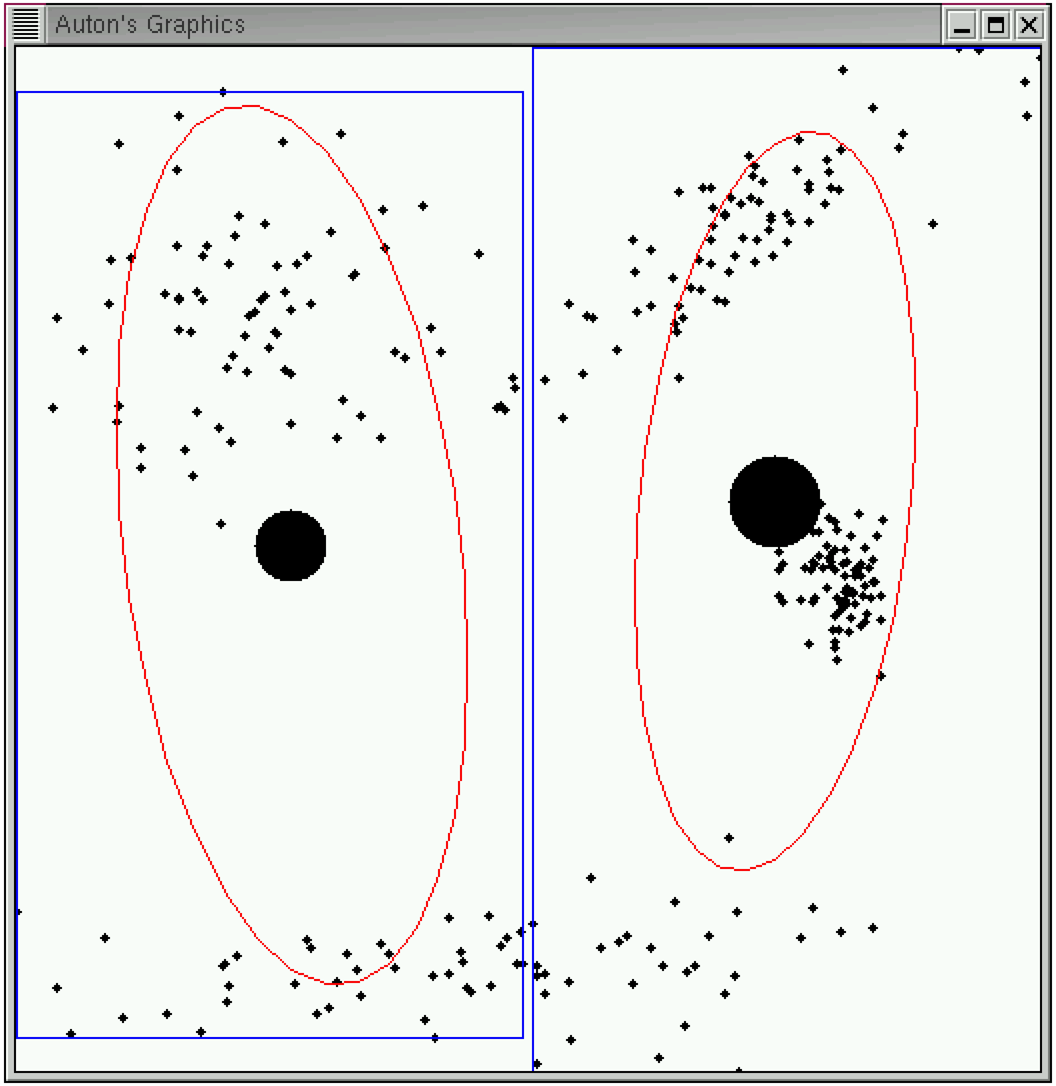
- Univariate axis-aligned splits
- Split on widest dimension
- O(N log N) to build, O(N) space



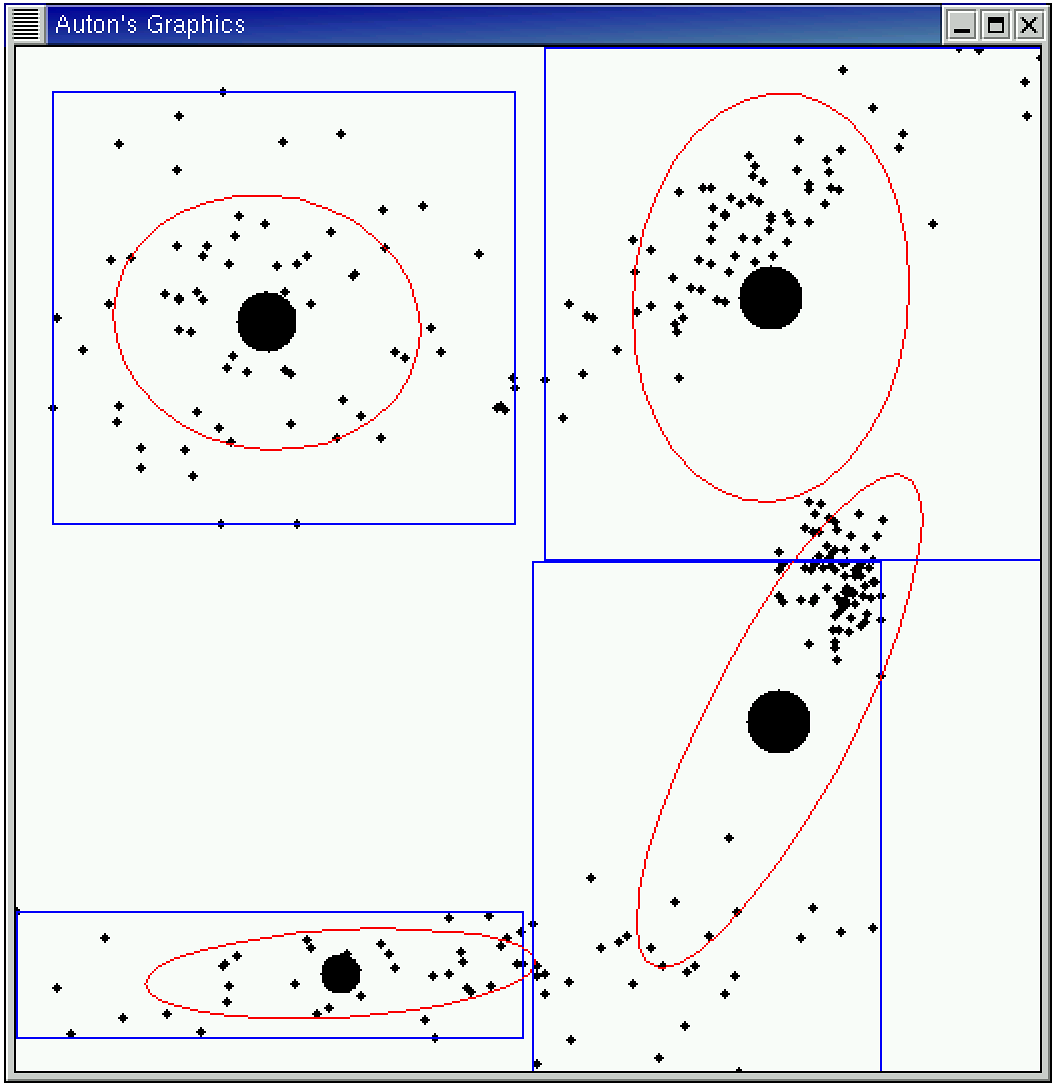
A *kd*-tree: level 1



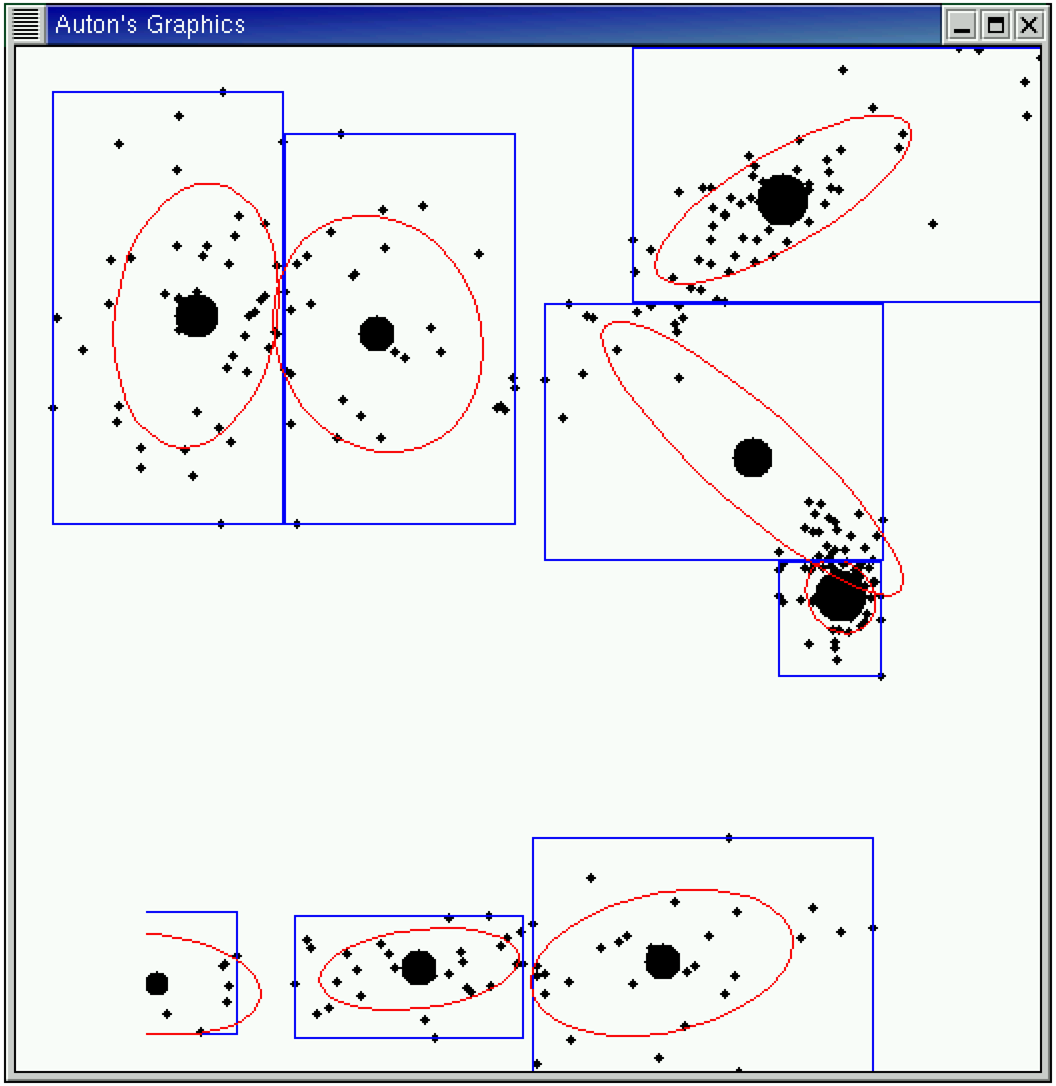
A *kd*-tree: level 2



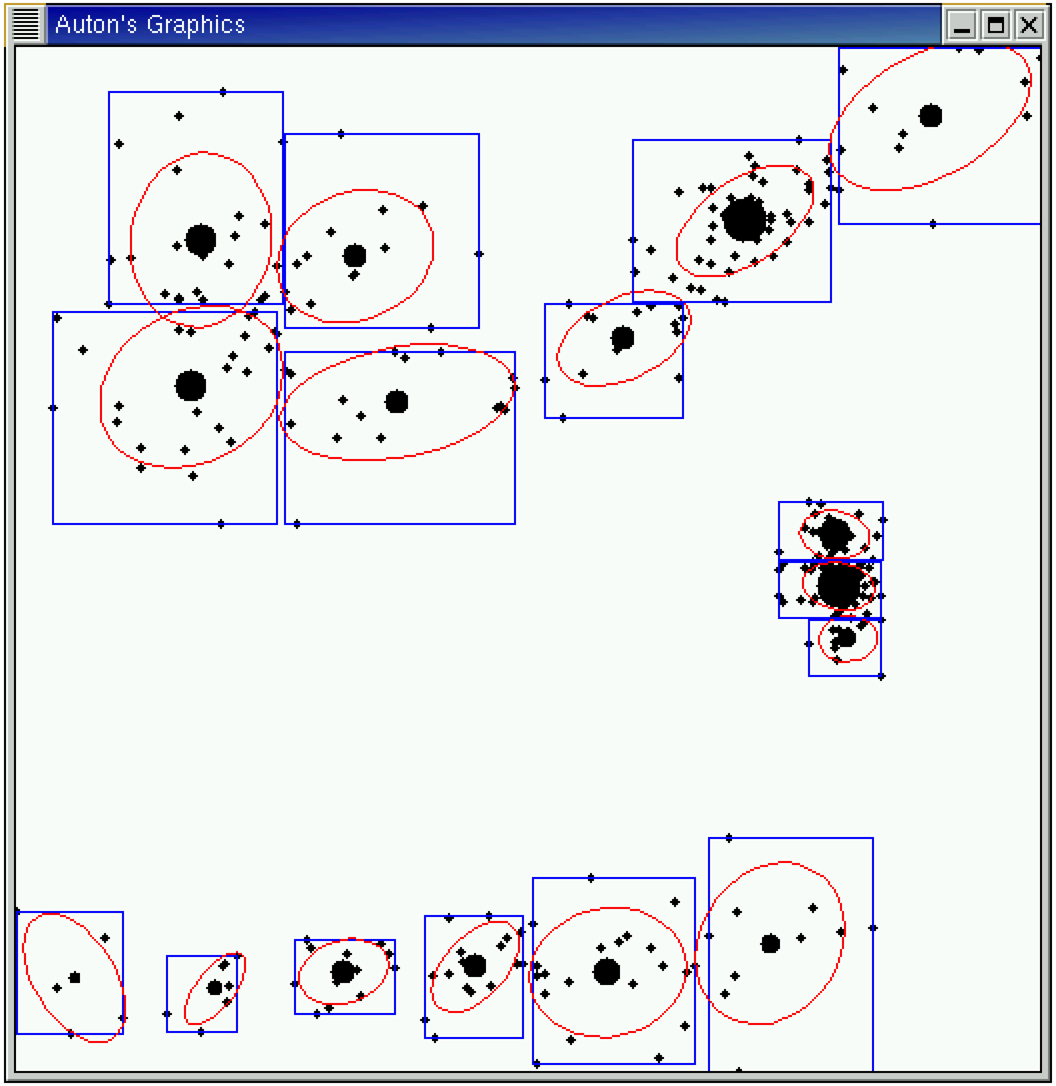
A *kd*-tree: level 3



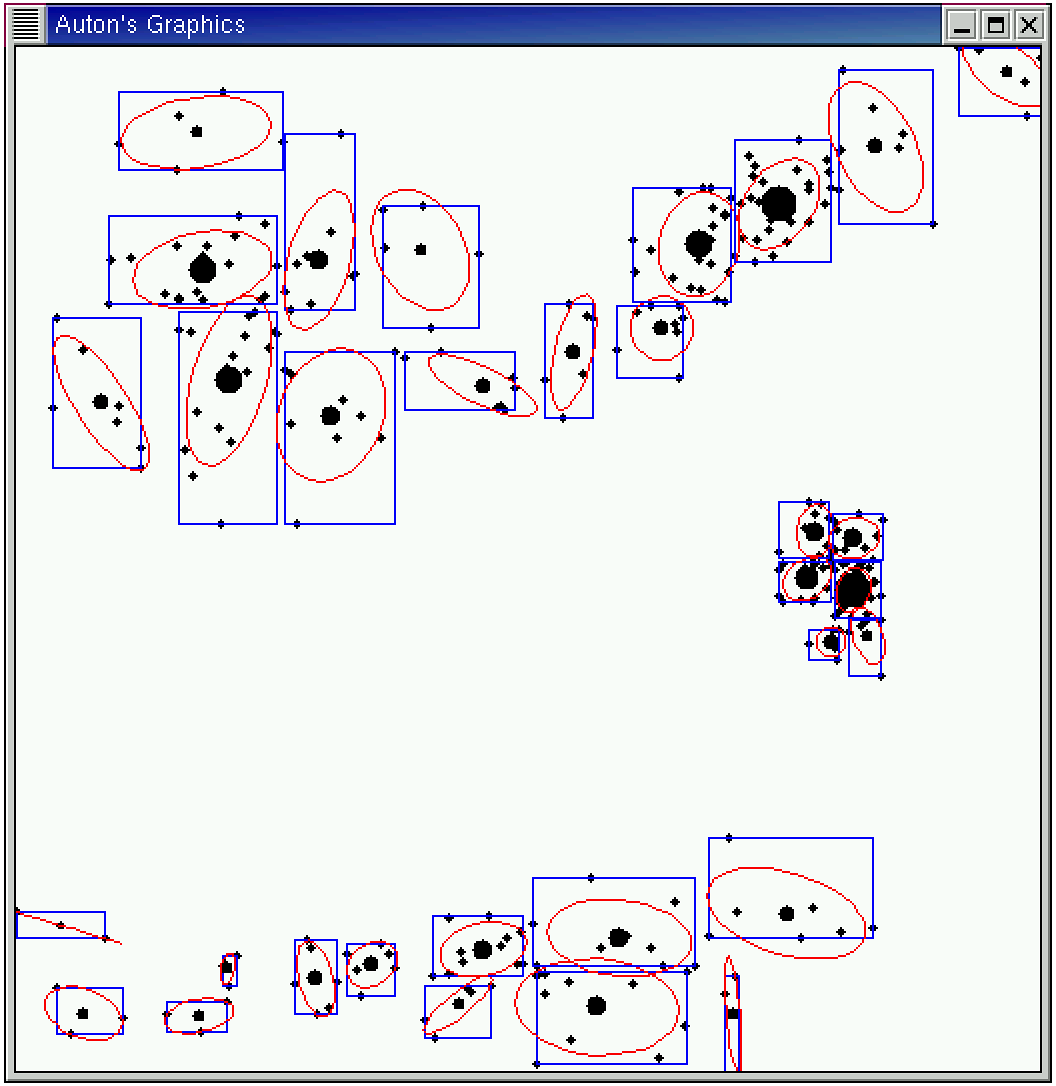
A *kd*-tree: level 4

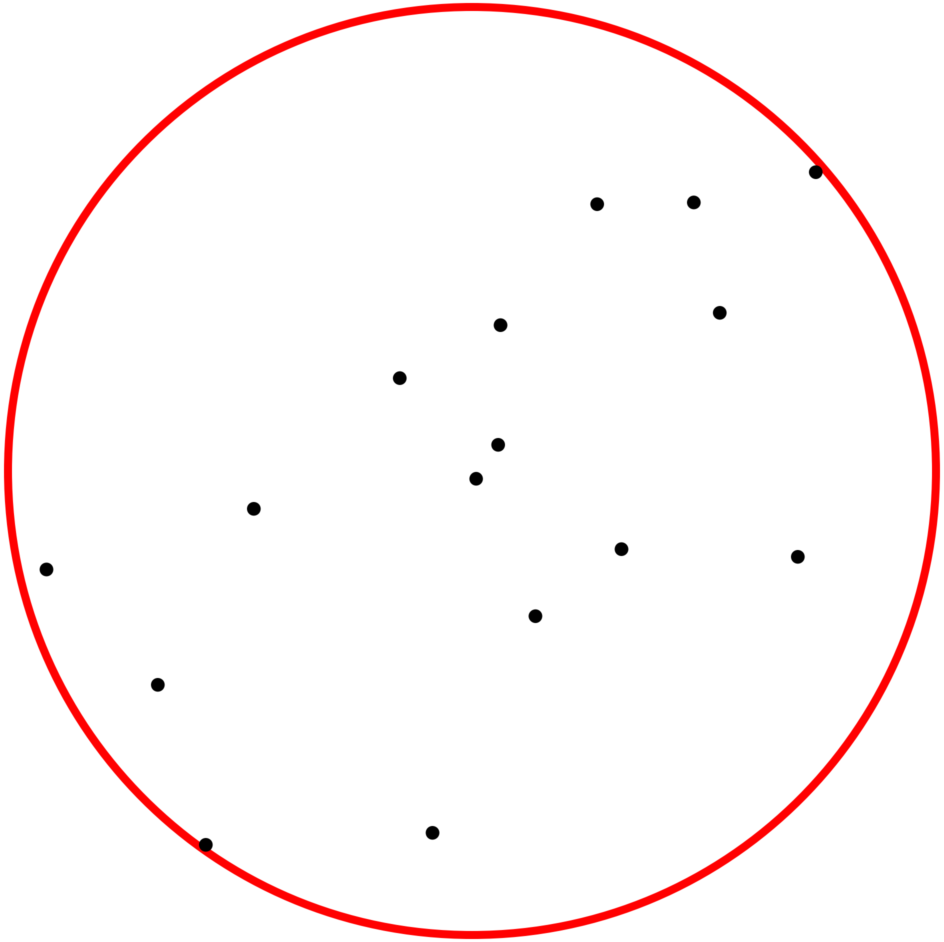


A *kd*-tree: level 5



A *kd*-tree: level 6

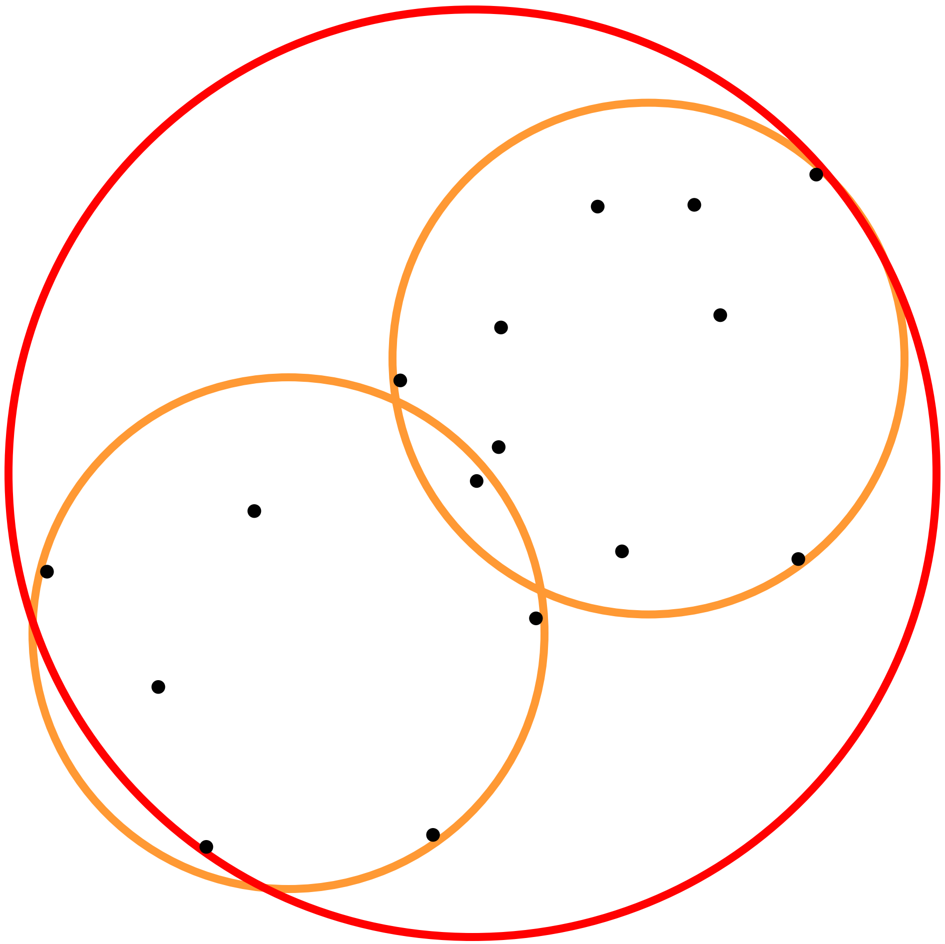


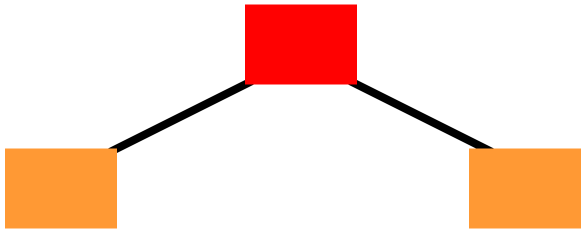


A ball-tree: level 1

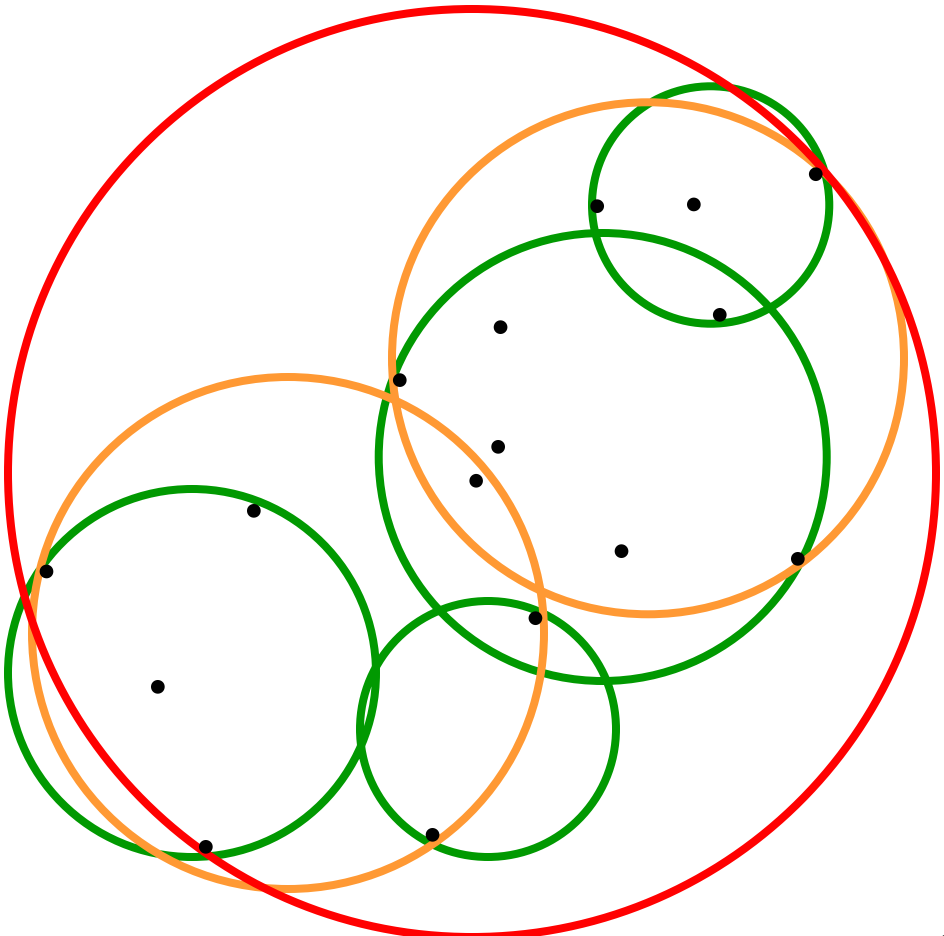
[Uhlmann 1991], [Omohundro 1991]

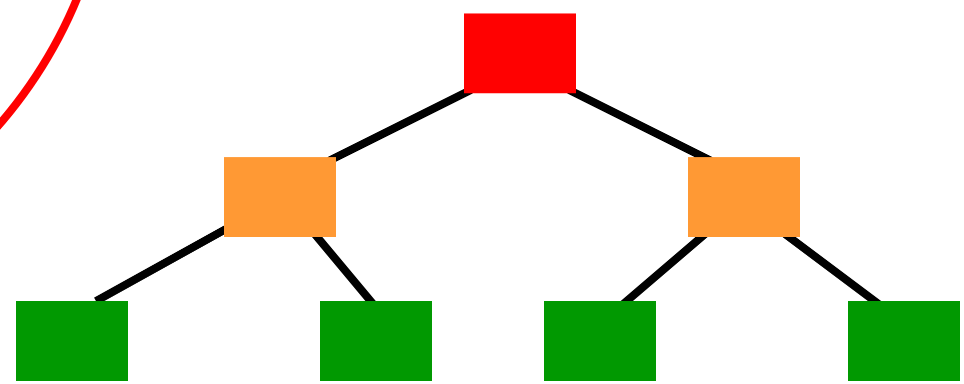
A ball-tree: level 2



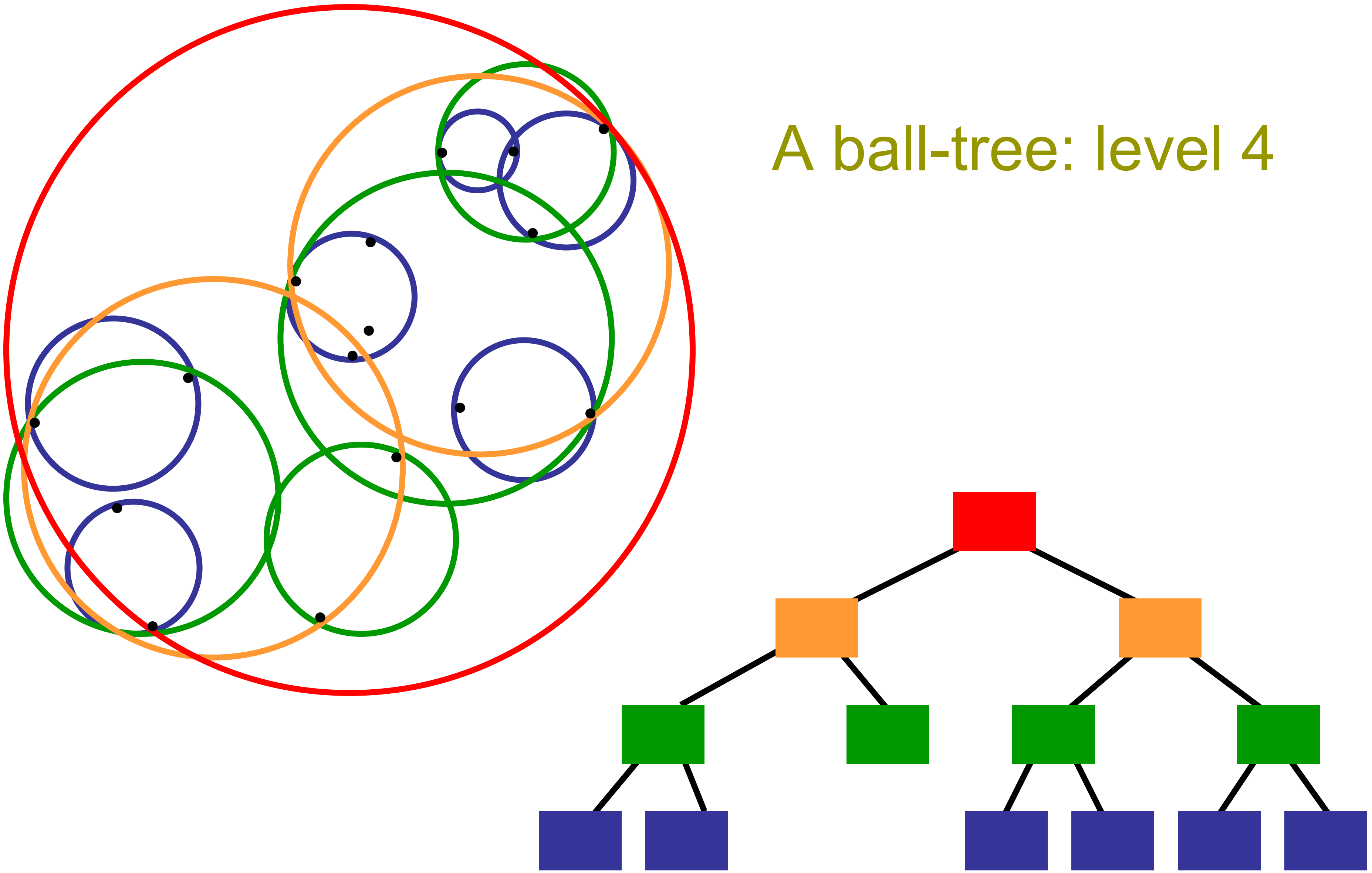


A ball-tree: level 3

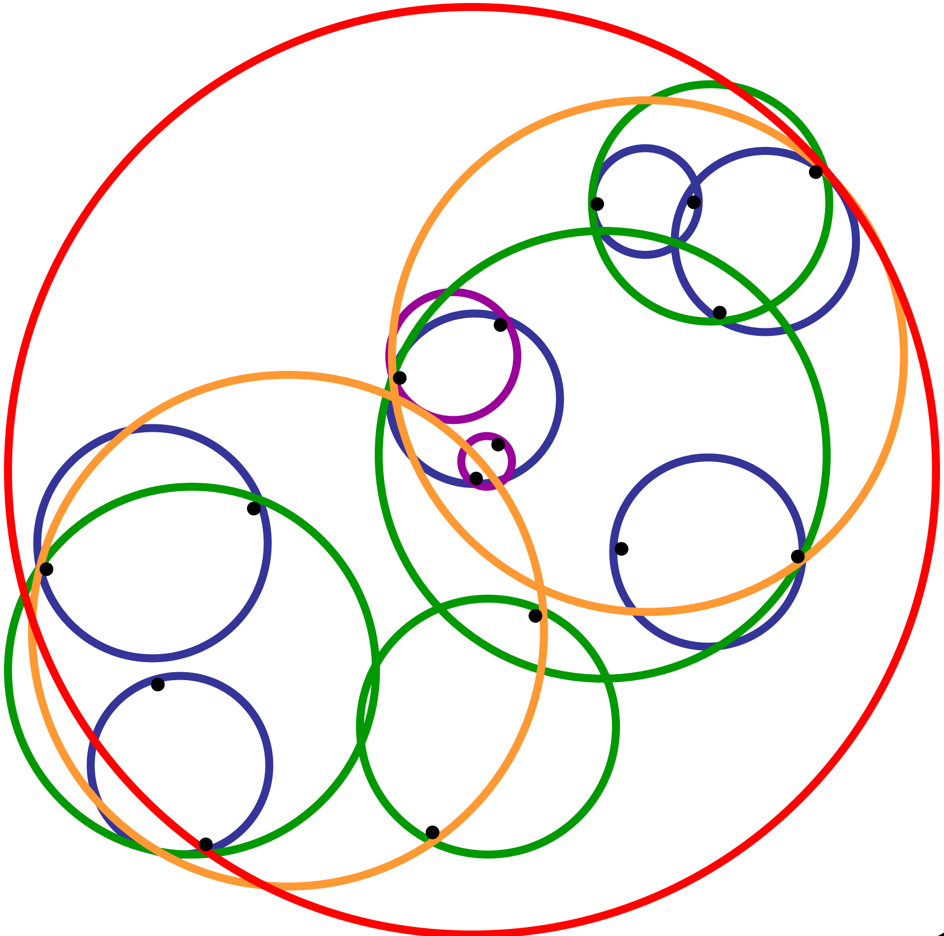


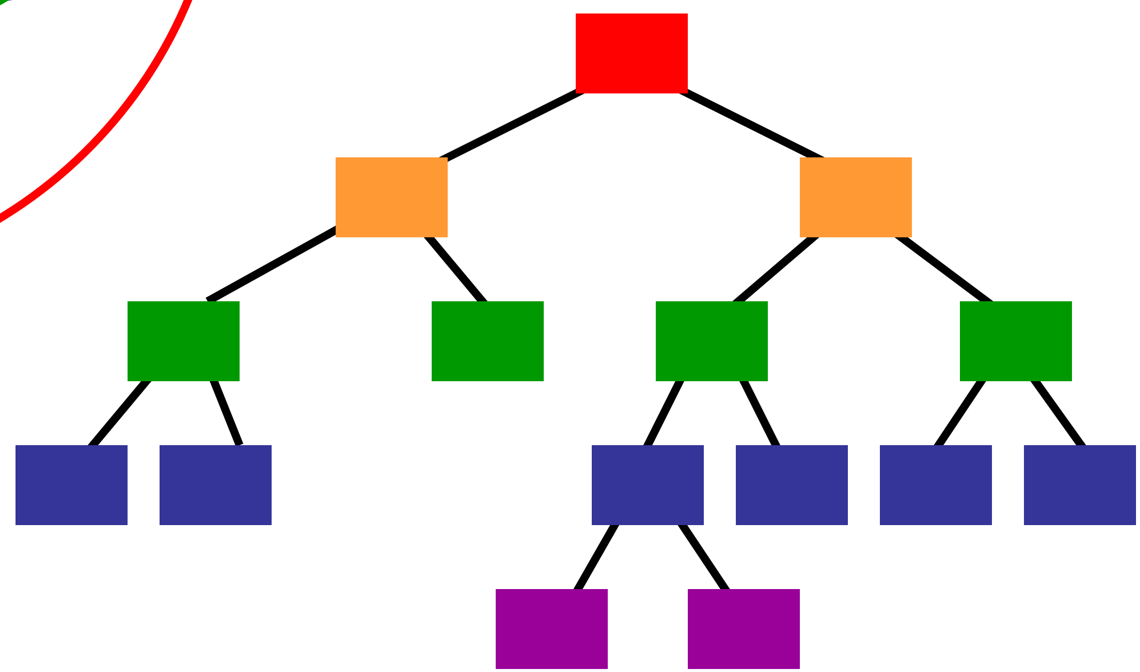


A ball-tree: level 4



A ball-tree: level 5





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

Questions

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

  Is it really because of the expansions?

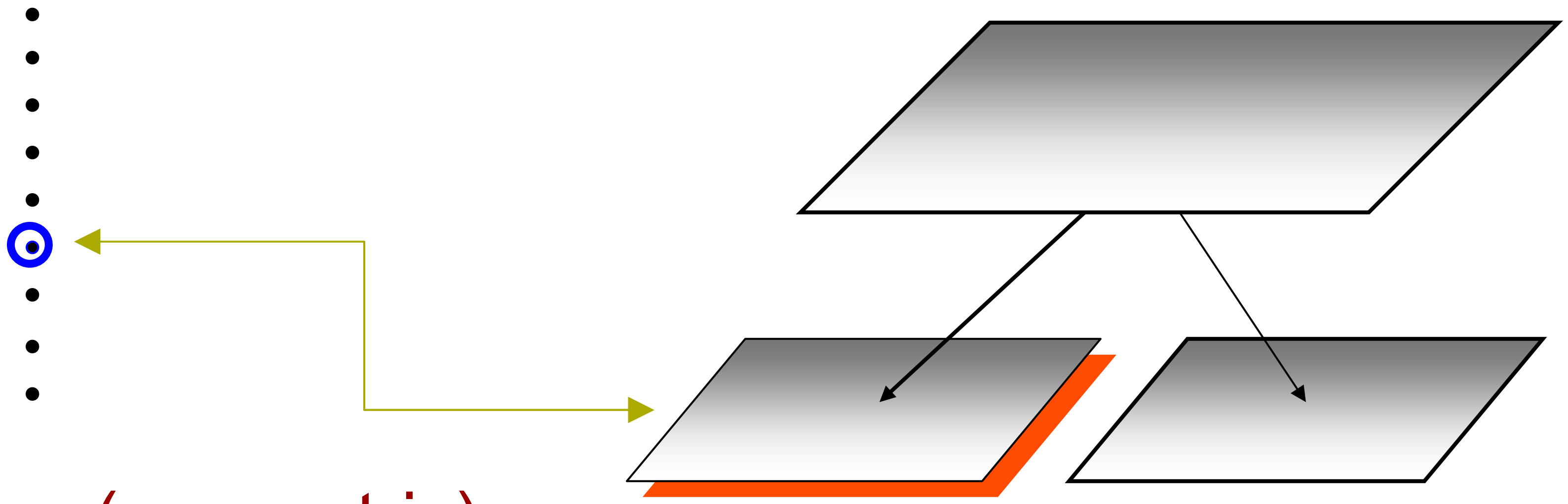
- Can we obtain an method that's:
  1. $O(N)$
  2. lightweight: works with or without expansions

     simple, recursive

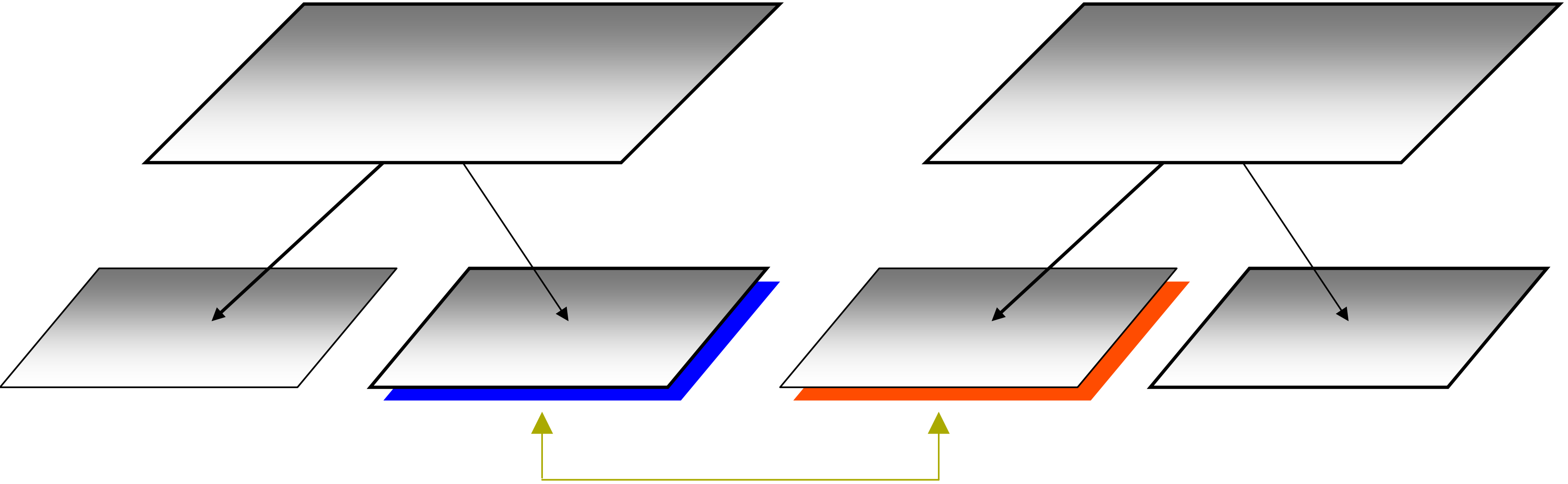
New algorithm

- 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)

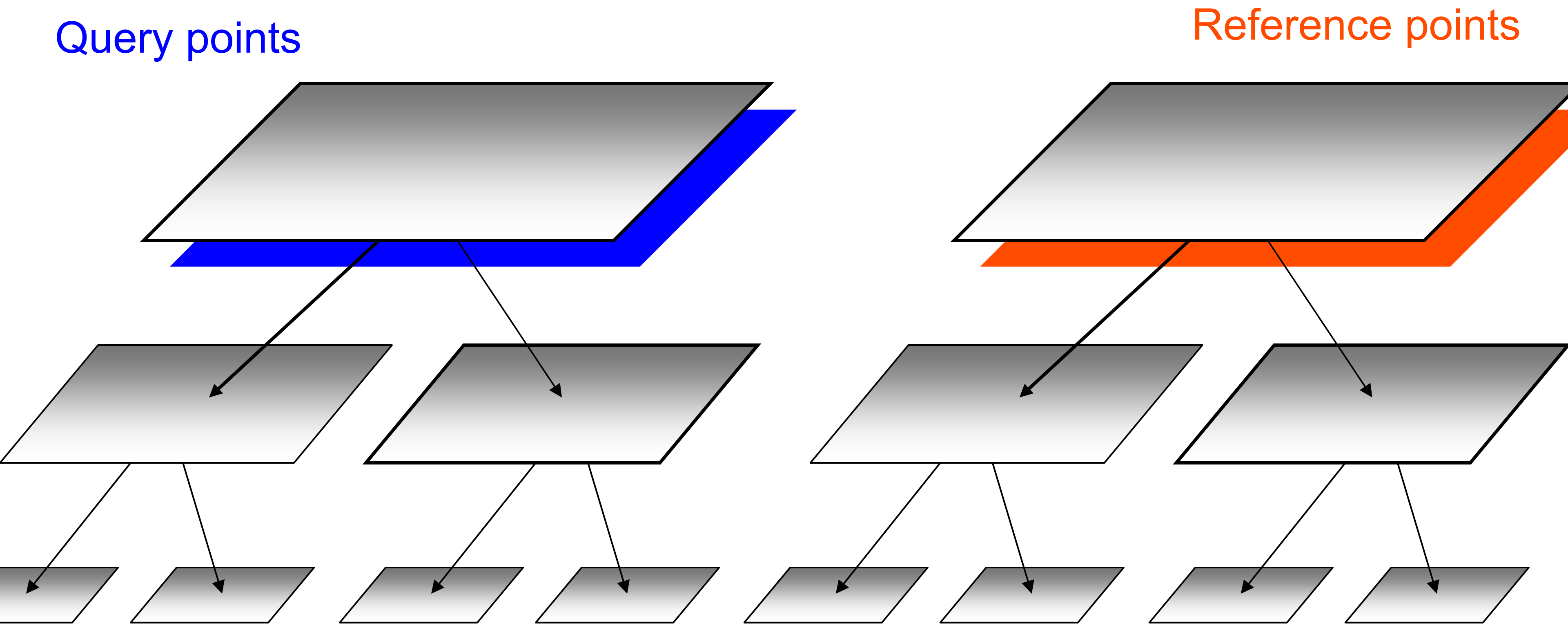
Simple recursive algorithm

```
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).
}
```

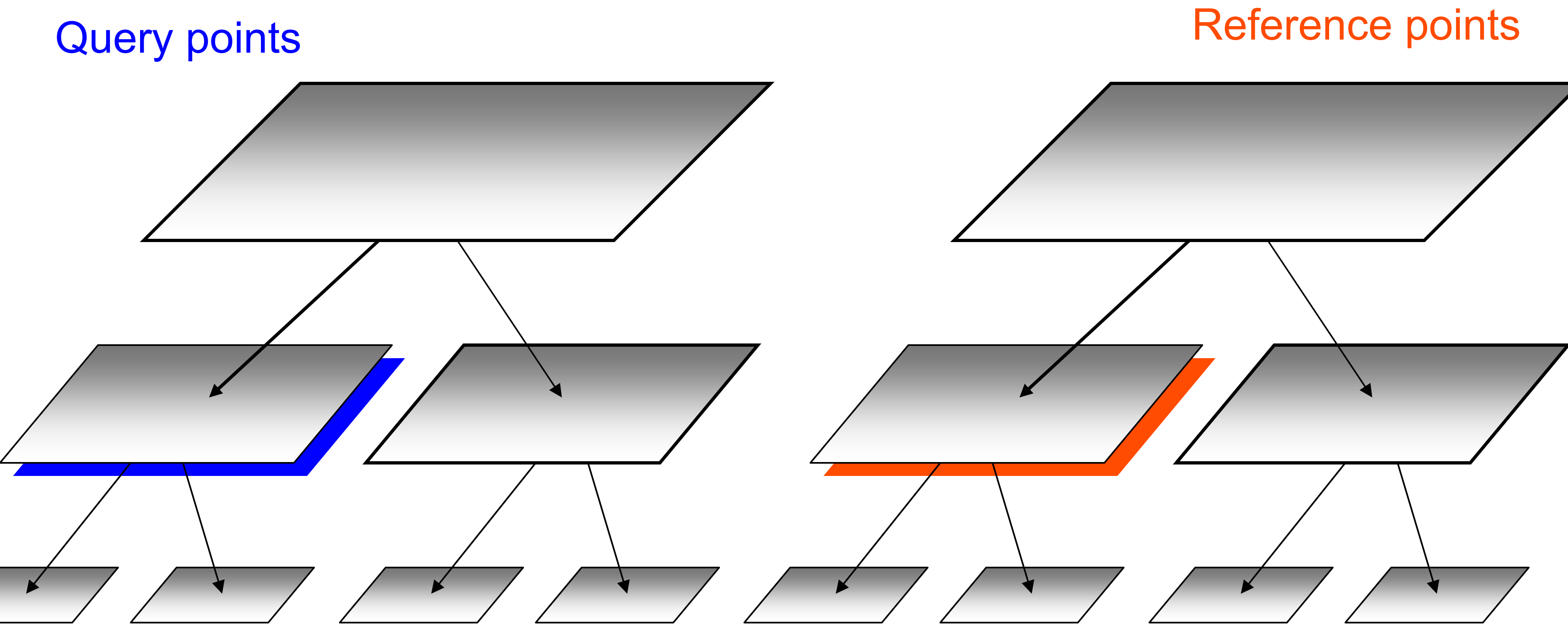
(NN or range-search: recurse on the closer node first)

Dual-tree traversal

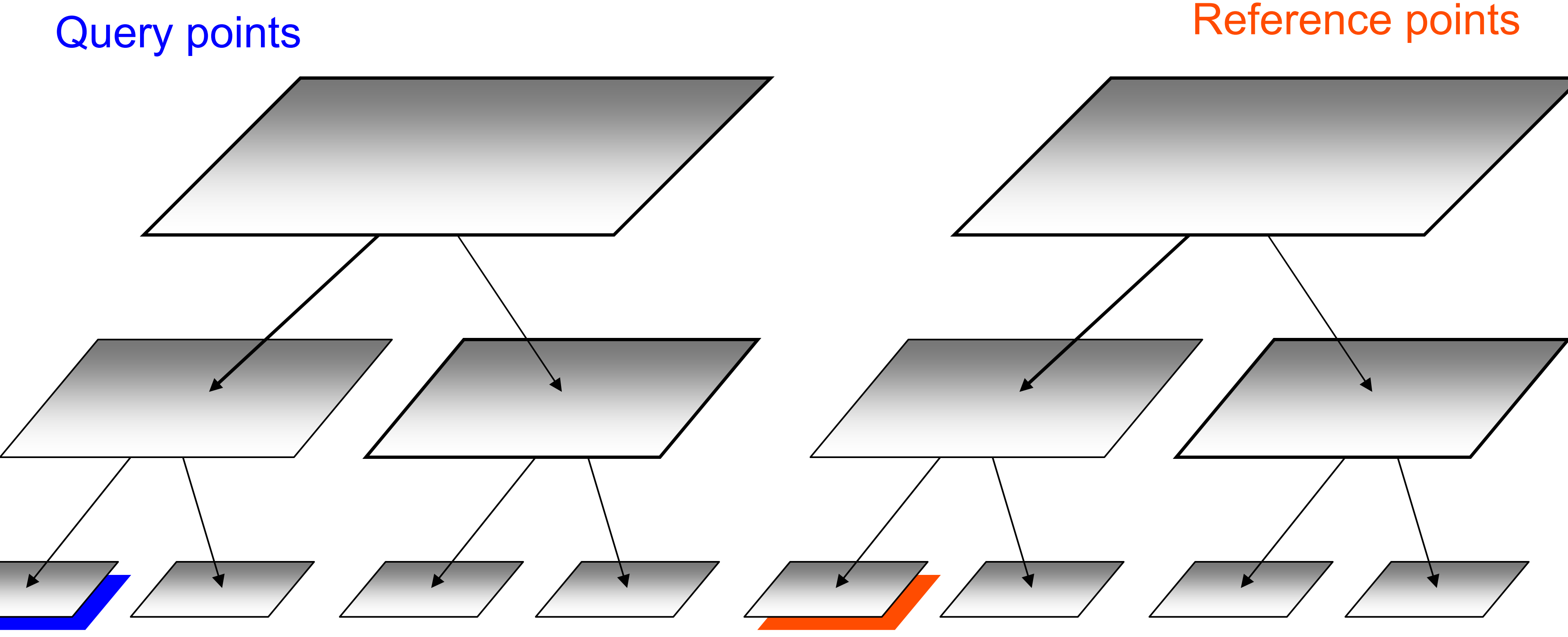
(depth-first)



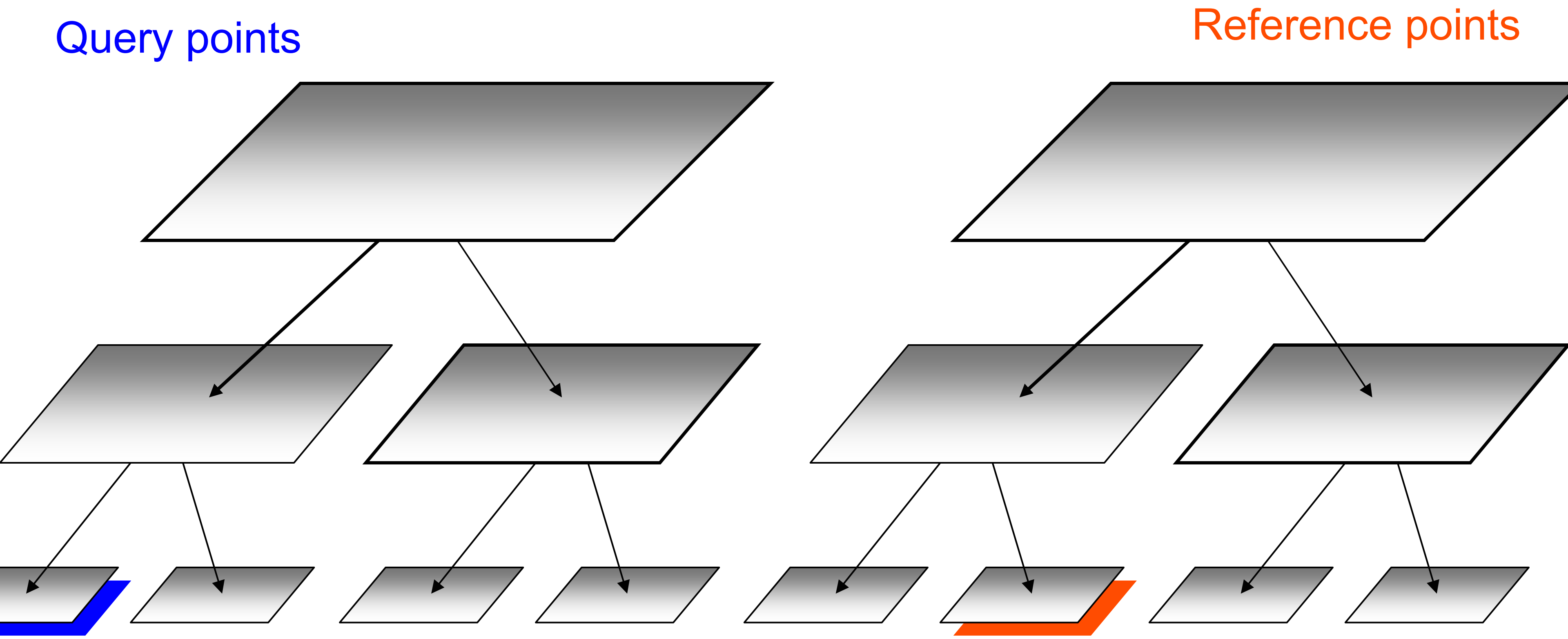
Dual-tree traversal



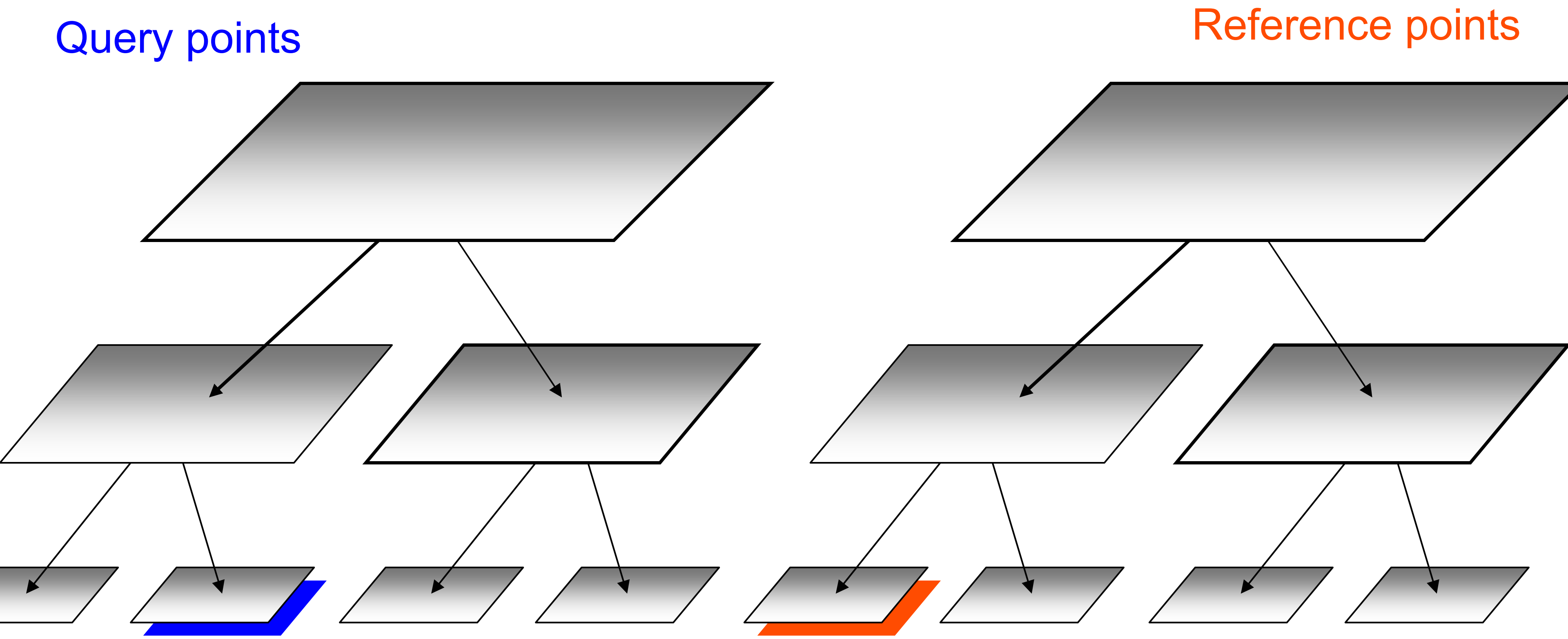
Dual-tree traversal



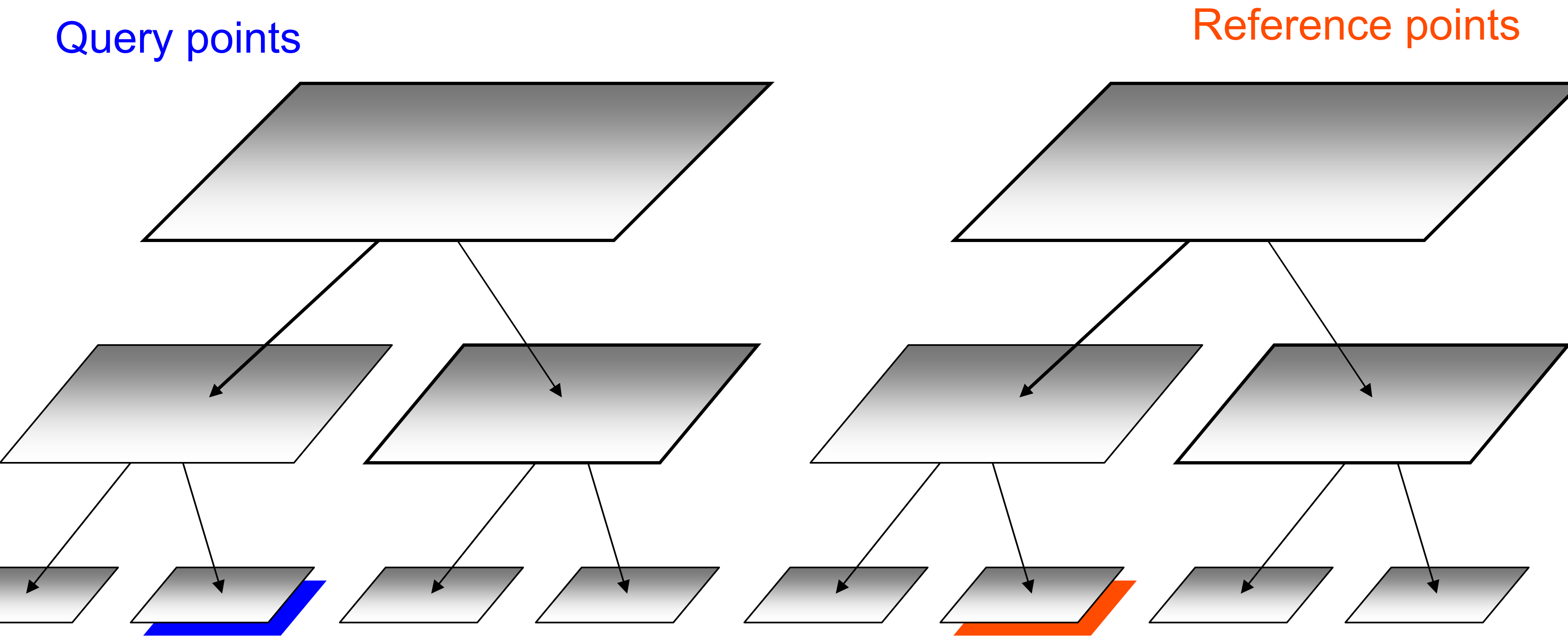
Dual-tree traversal



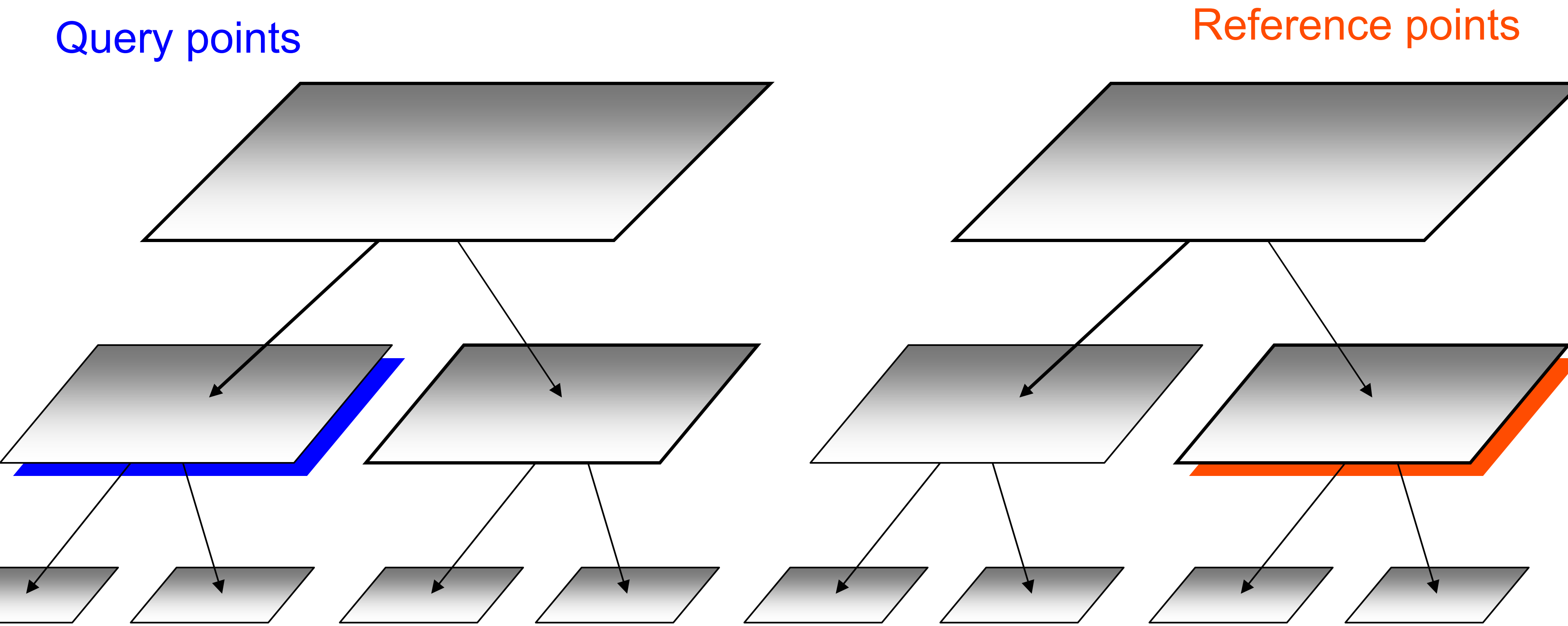
Dual-tree traversal



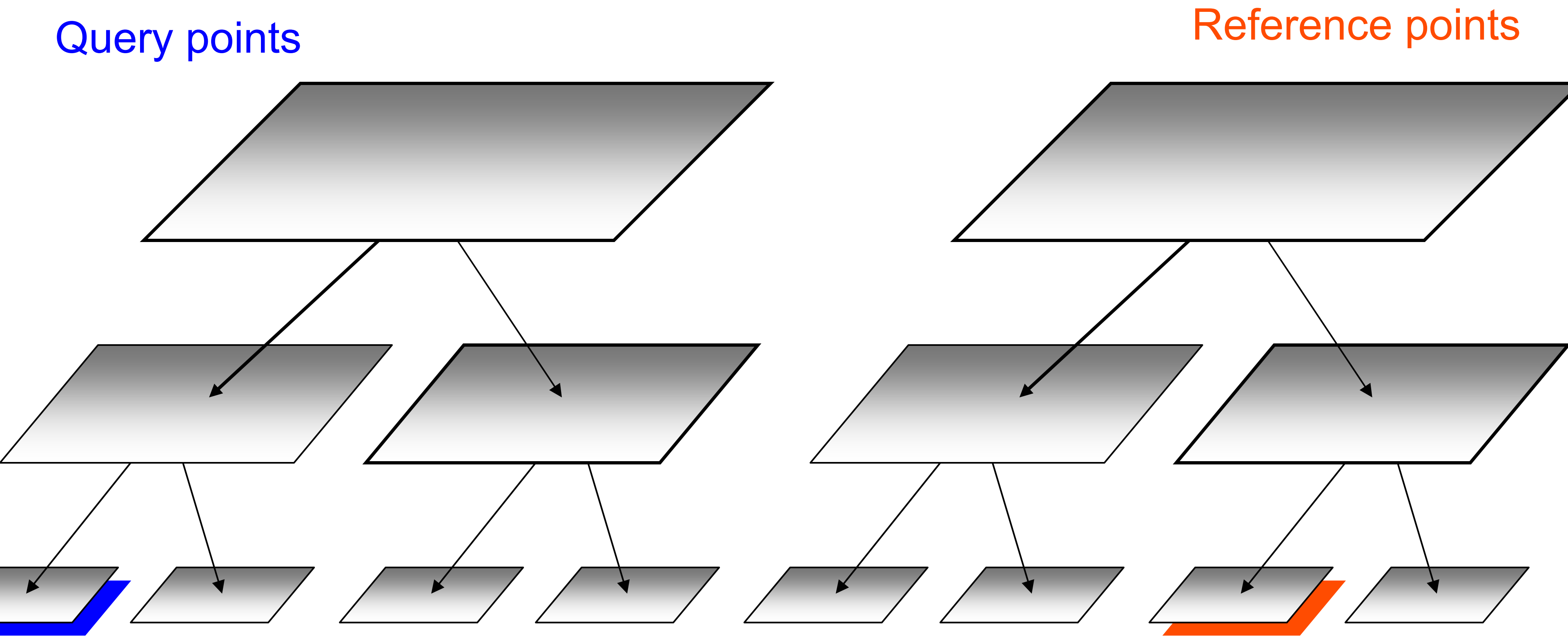
Dual-tree traversal



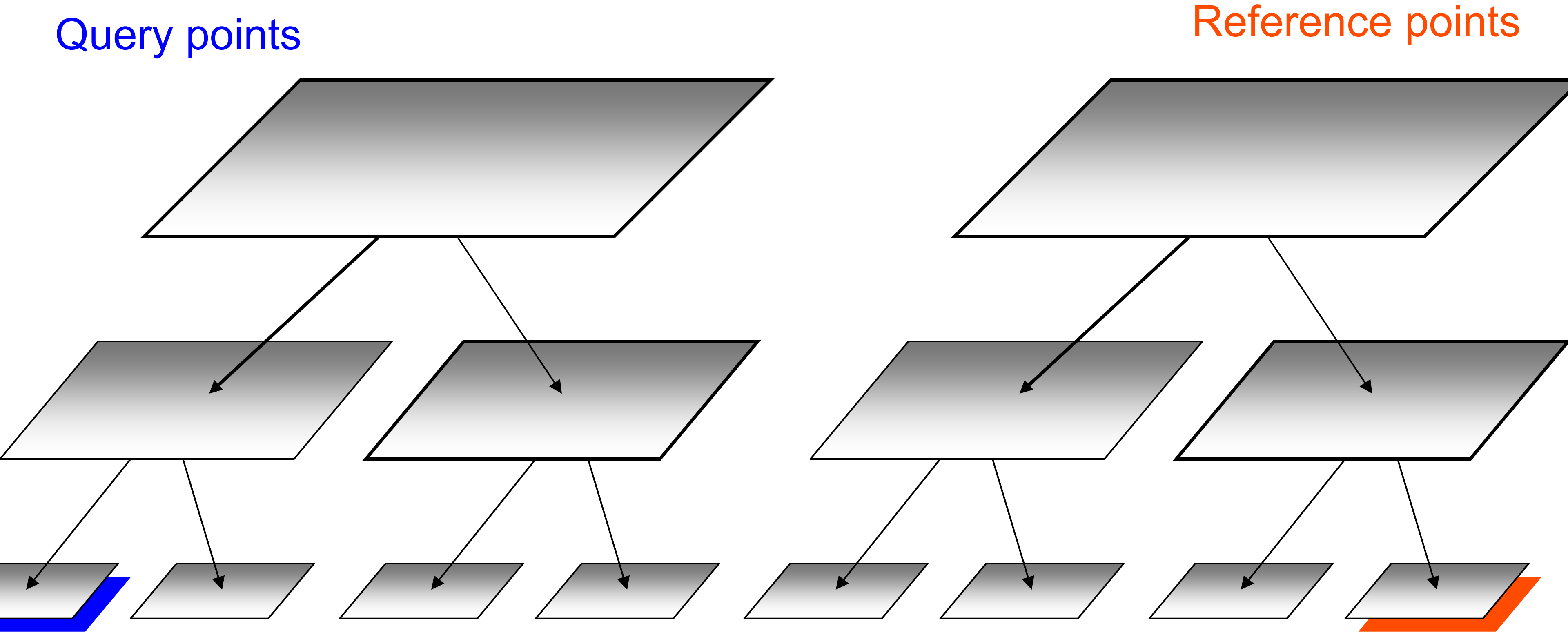
Dual-tree traversal



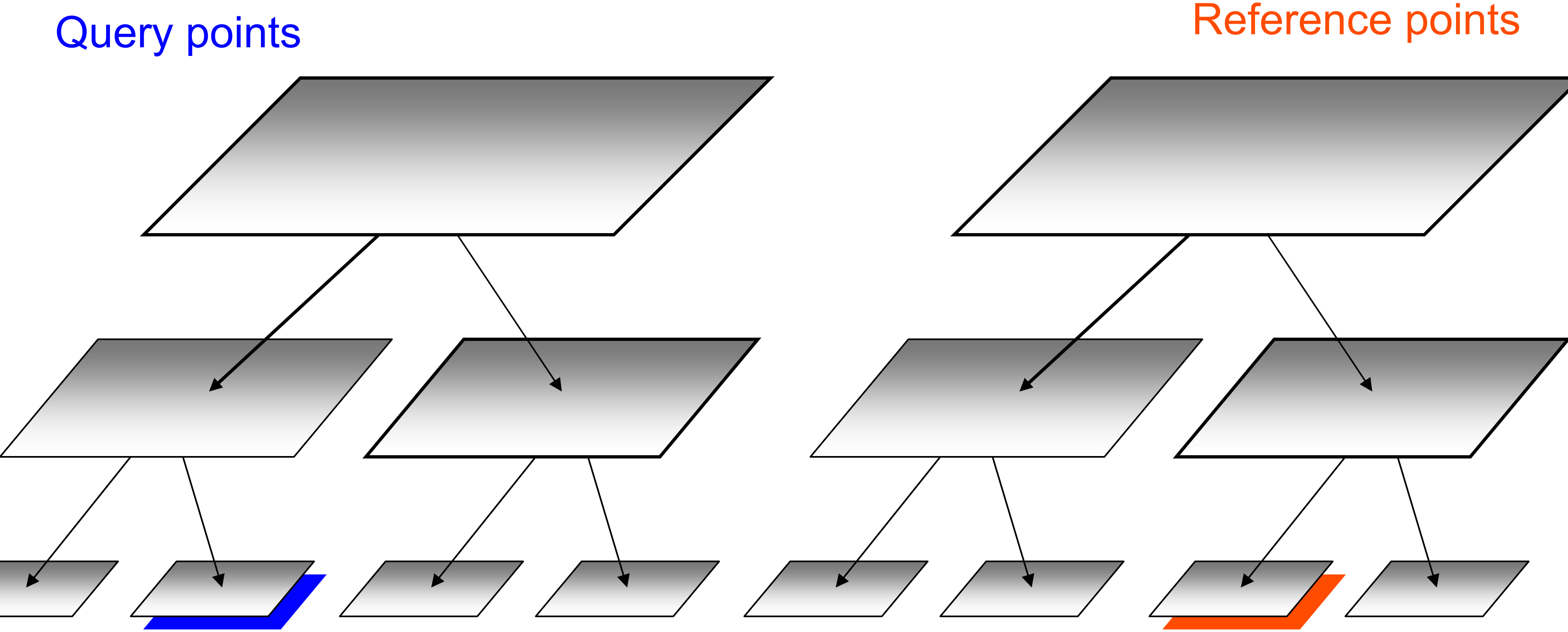
Dual-tree traversal



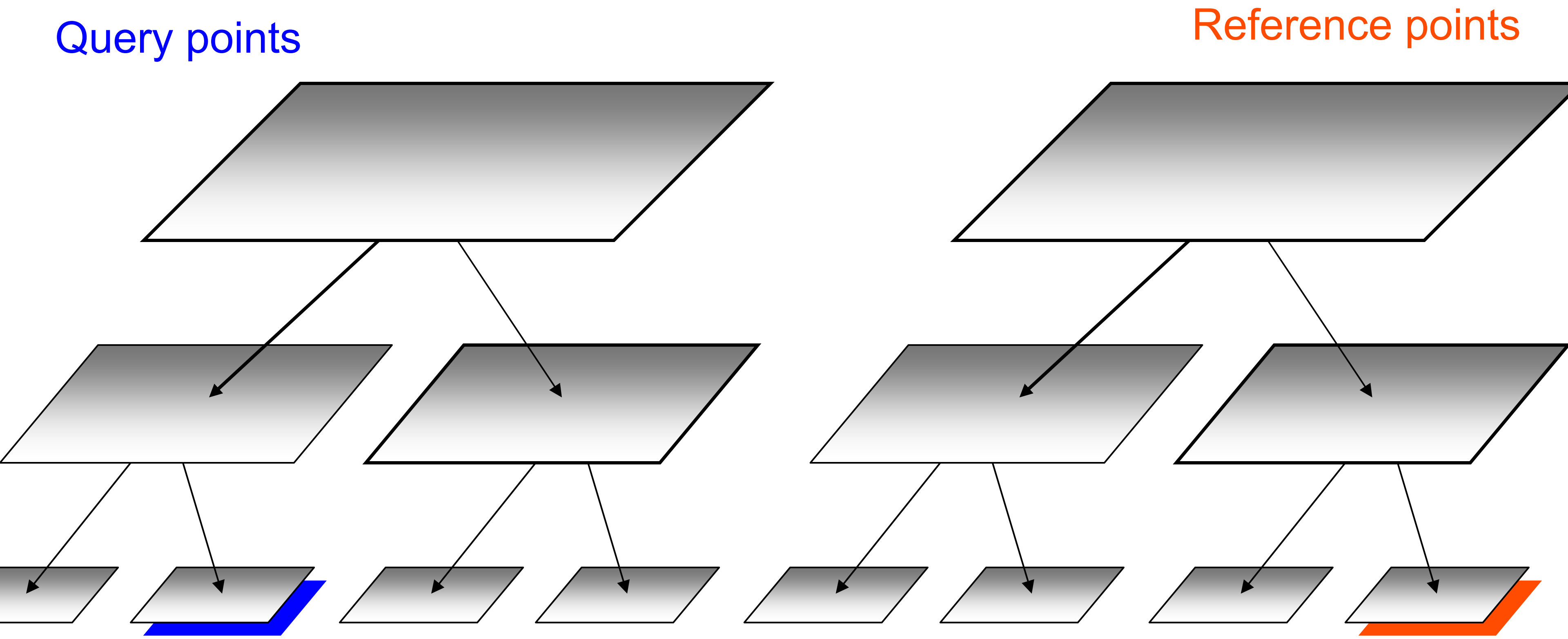
Dual-tree traversal



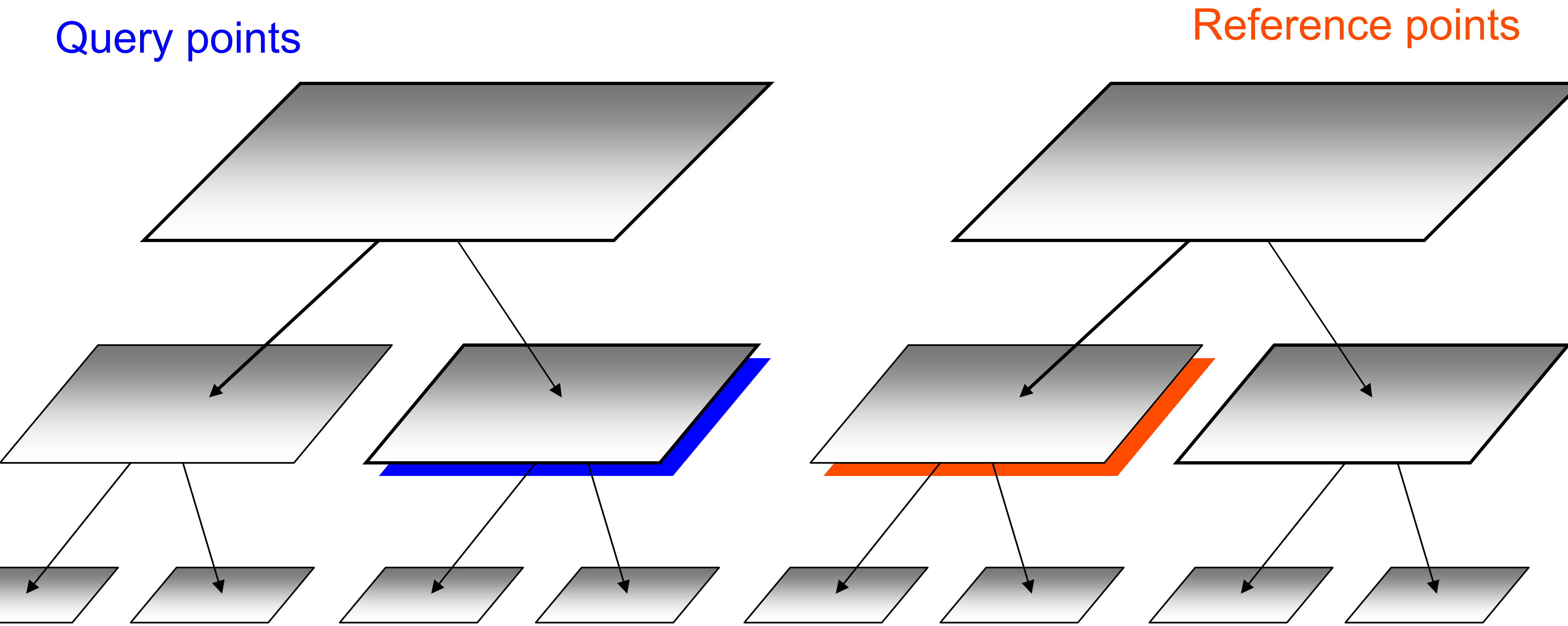
Dual-tree traversal



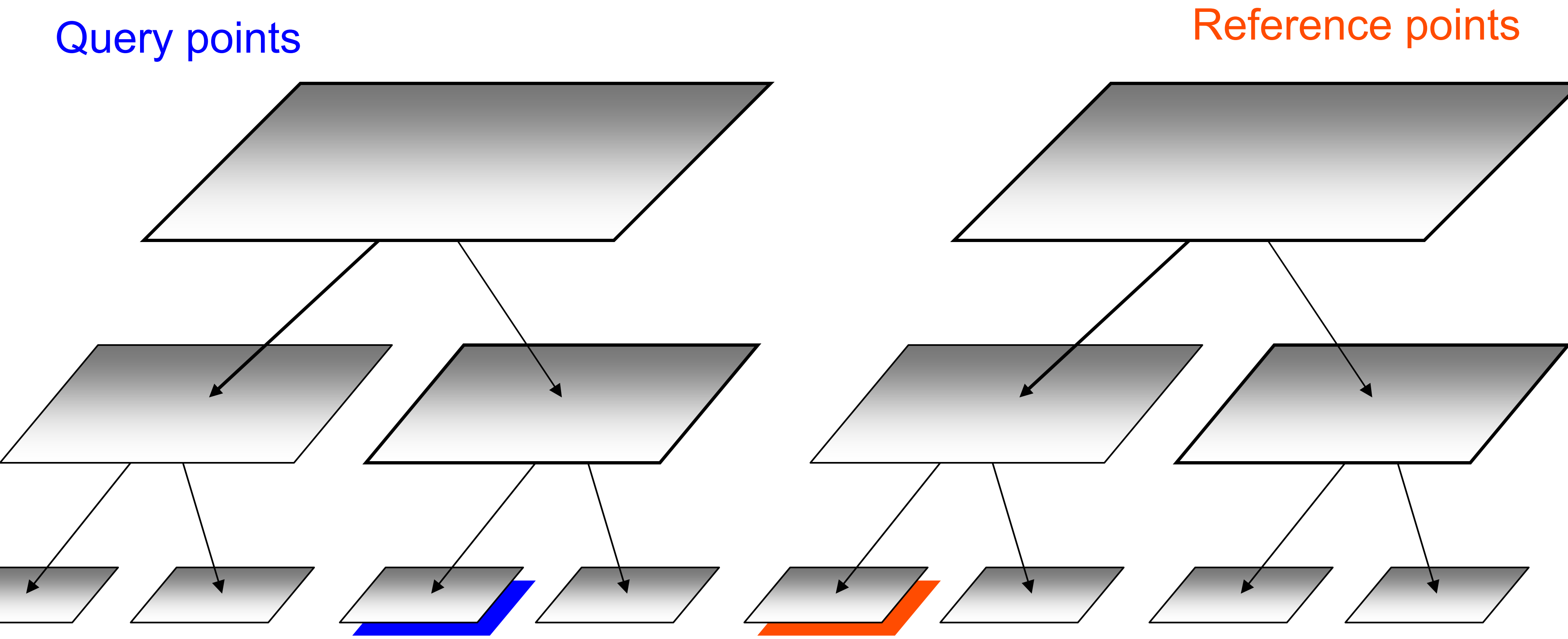
Dual-tree traversal



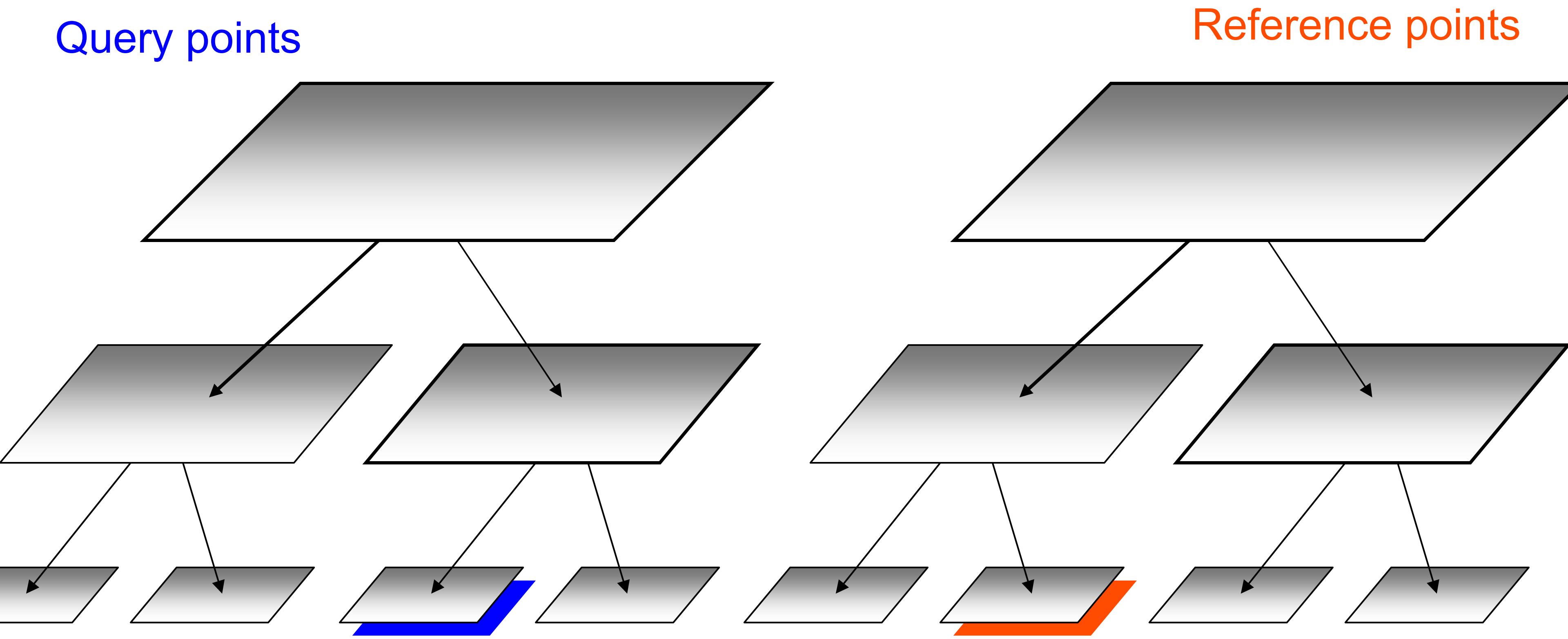
Dual-tree traversal



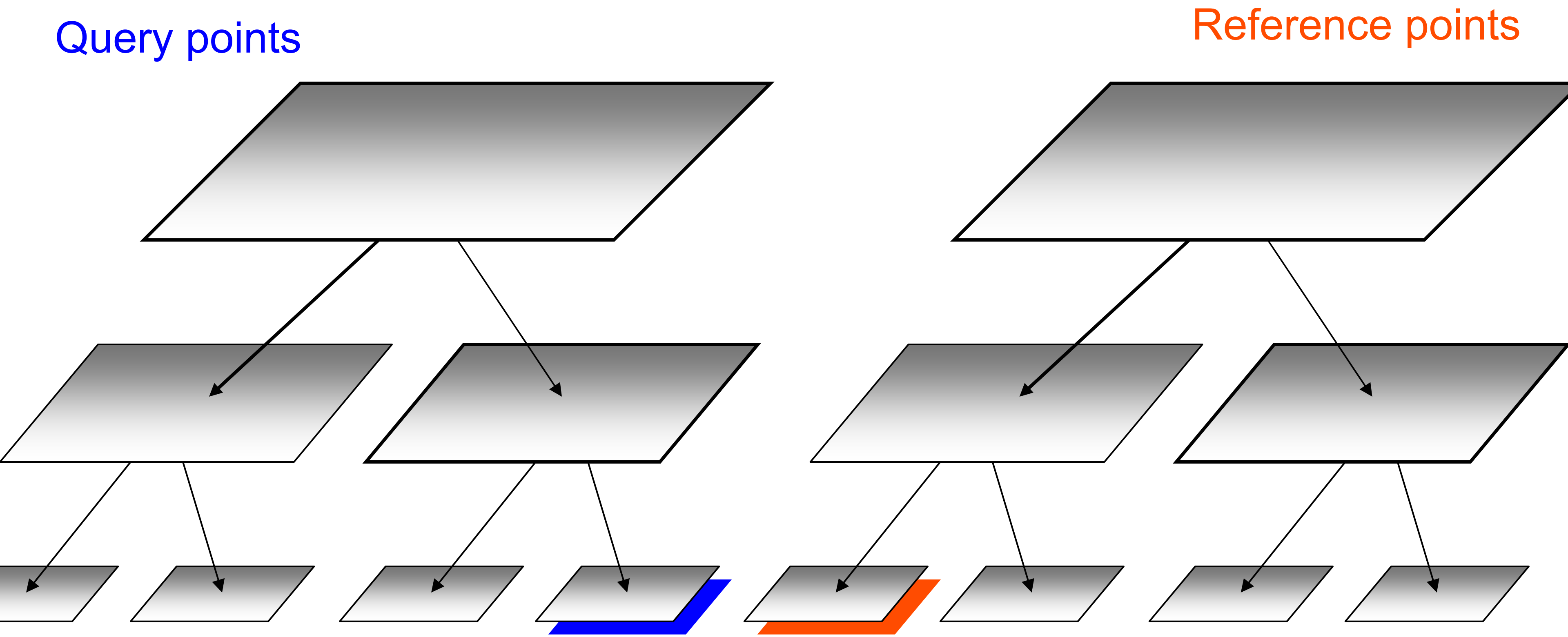
Dual-tree traversal



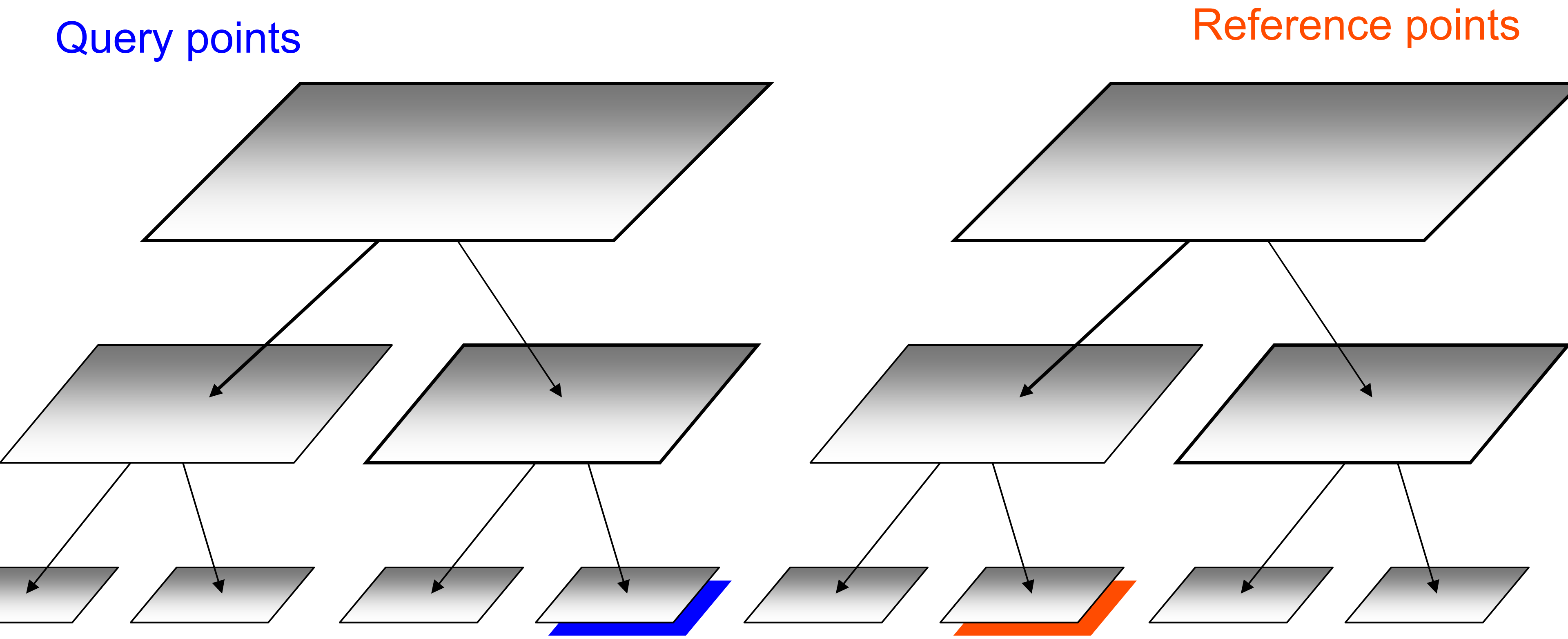
Dual-tree traversal



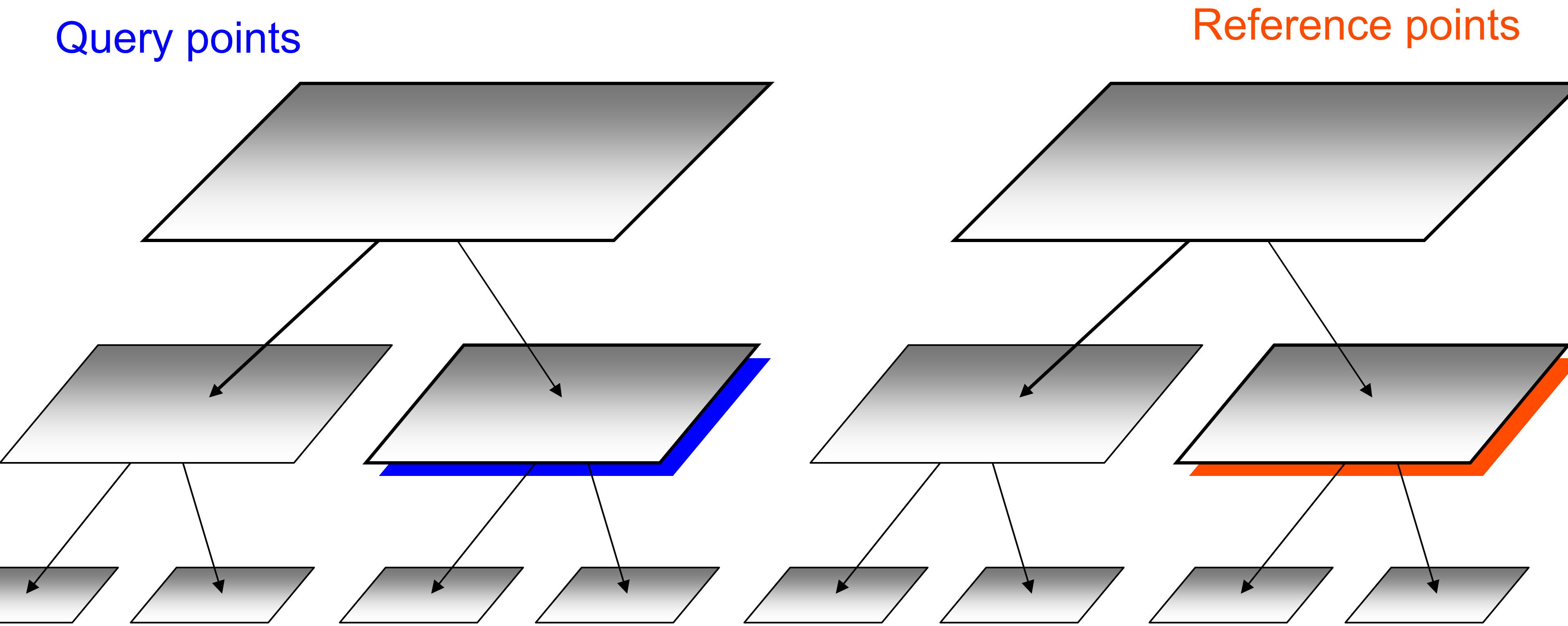
Dual-tree traversal



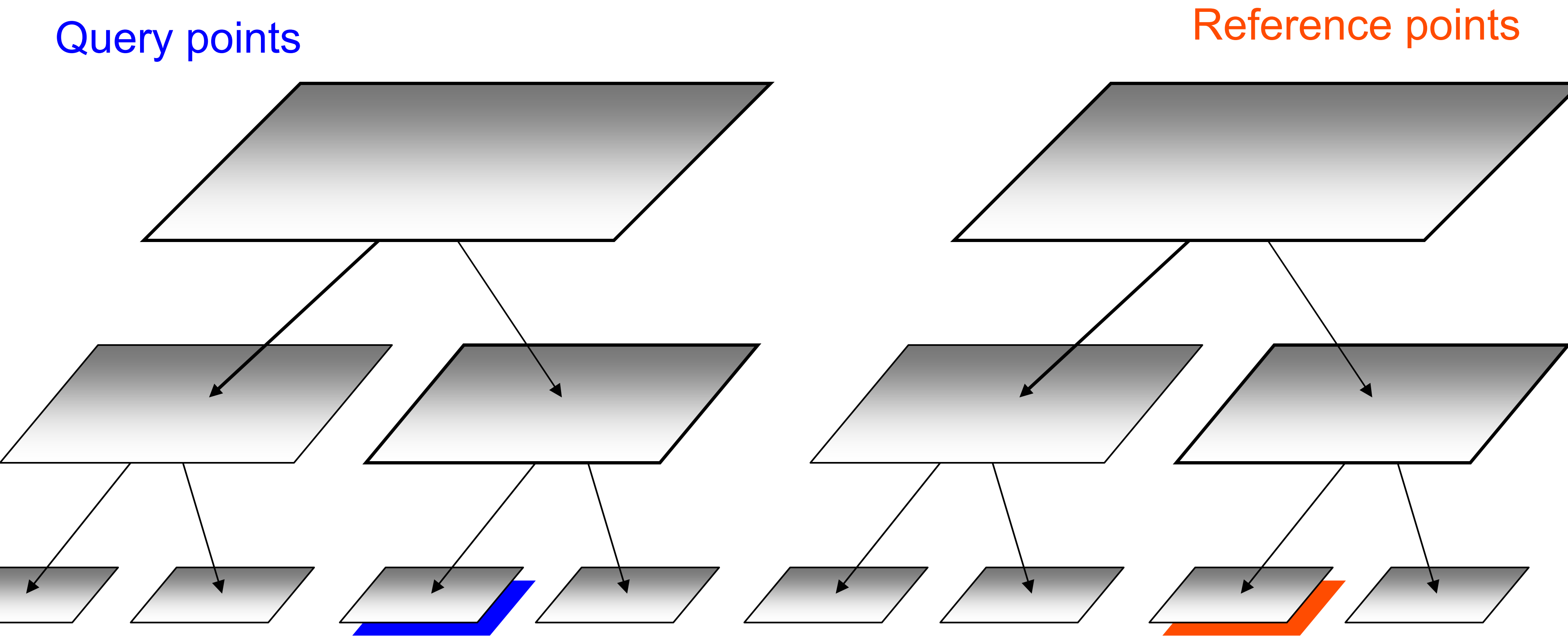
Dual-tree traversal



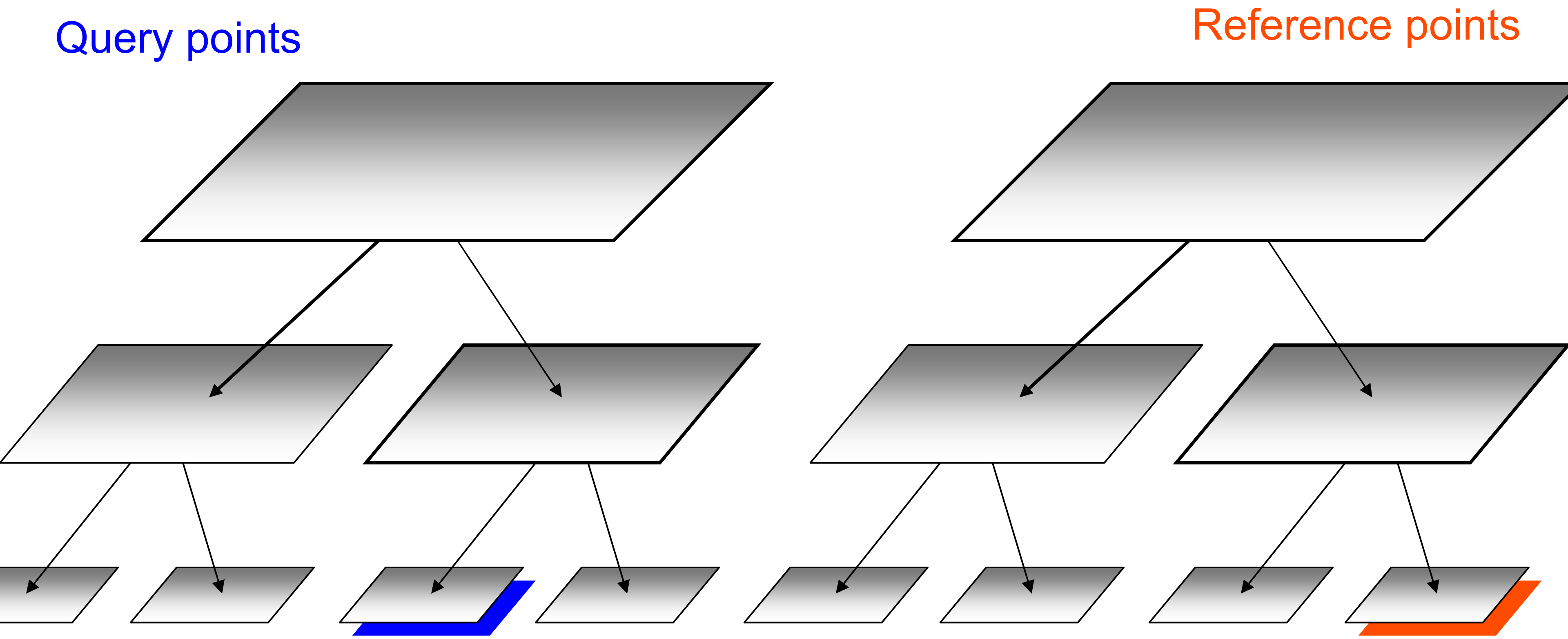
Dual-tree traversal



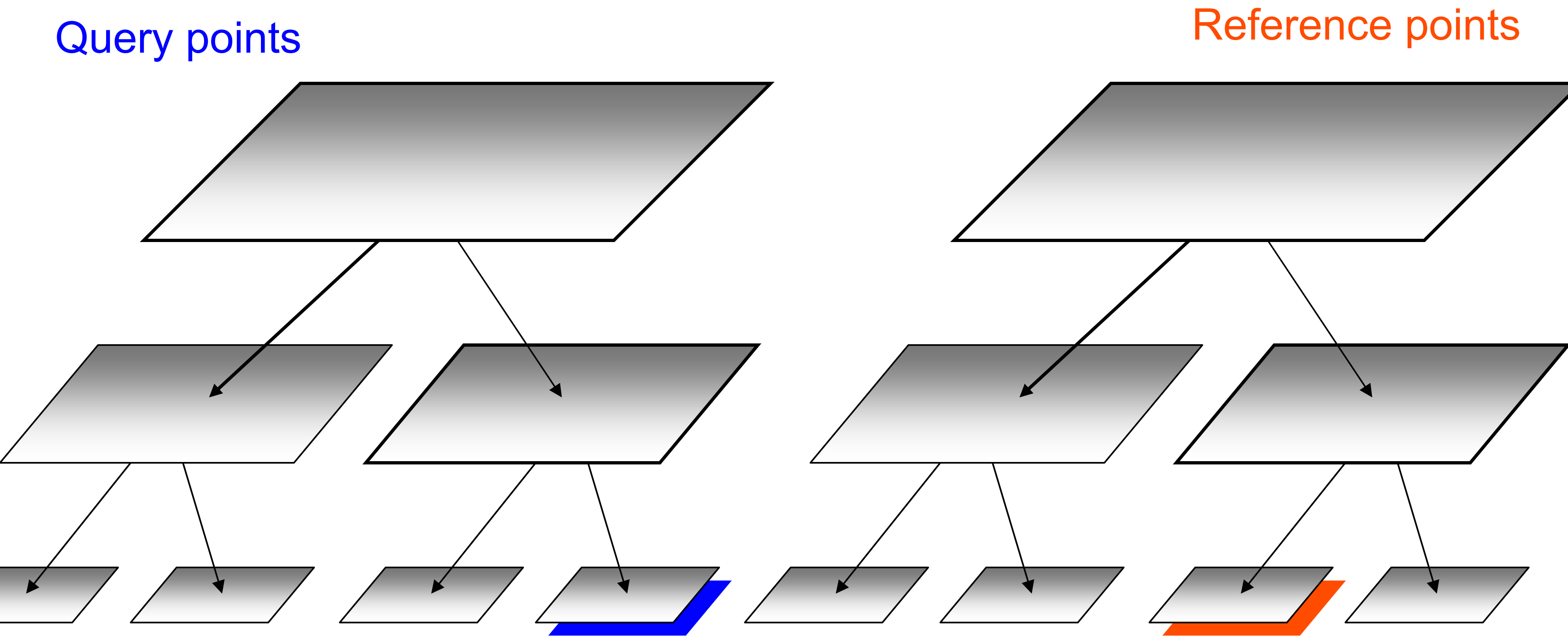
Dual-tree traversal



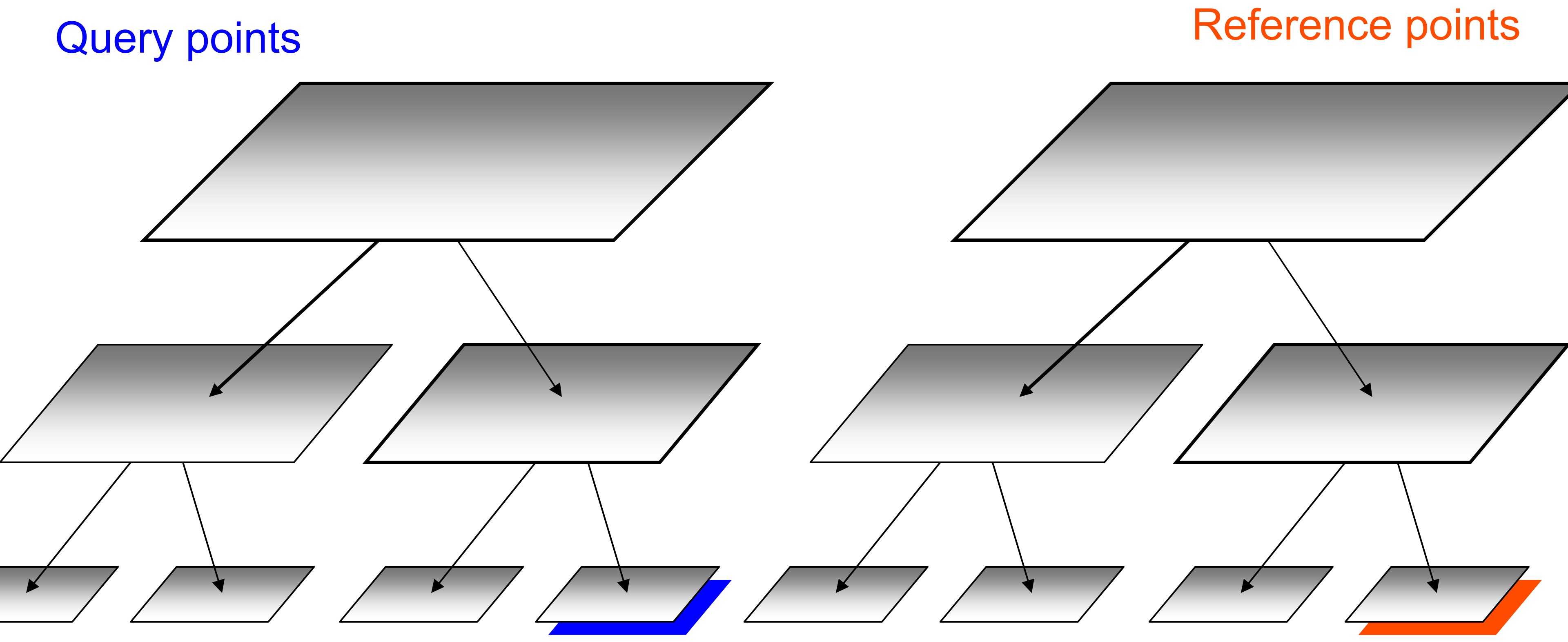
Dual-tree traversal



Dual-tree traversal



Dual-tree traversal



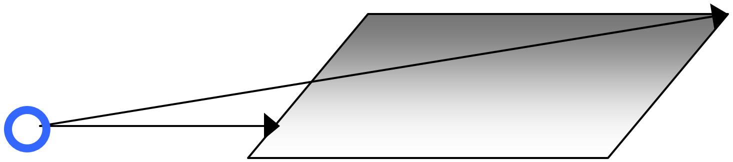
Finite-difference function approximation.

Taylor expansion:

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

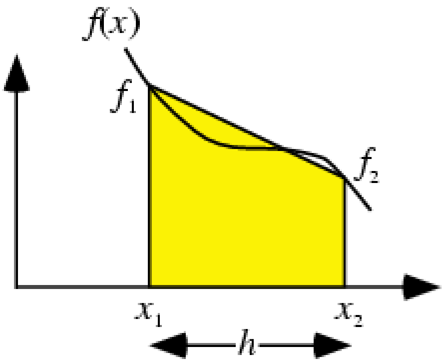
Gregory-Newton finite form:

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



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

Finite-difference function approximation.

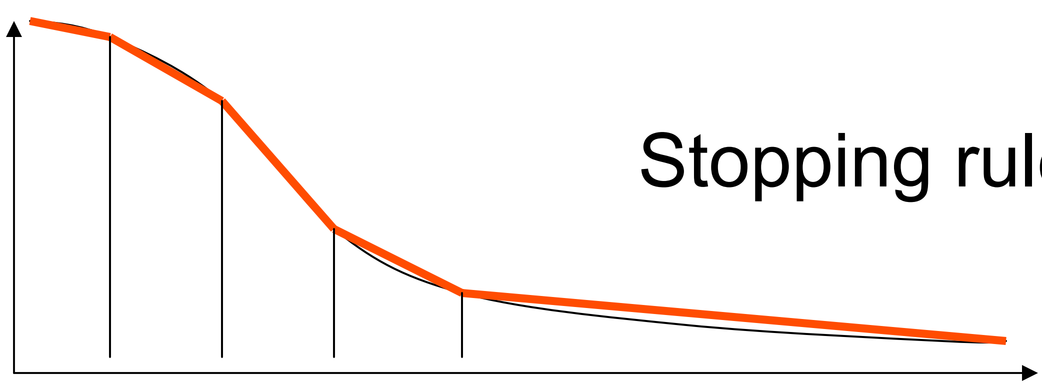


assumes monotonic decreasing kernel

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

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

could also use center of mass



Stopping rule: approximate if *s* > *r*

Simple approximation method

approximate(Q,R)
{

$$dl = N_R K(\delta_{\max}), du = N_R K(\delta_{\min}).$$

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

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

→trivial to change kernel
→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)

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

$$T(1) = O(1)$$

$$\Rightarrow O(N)$$

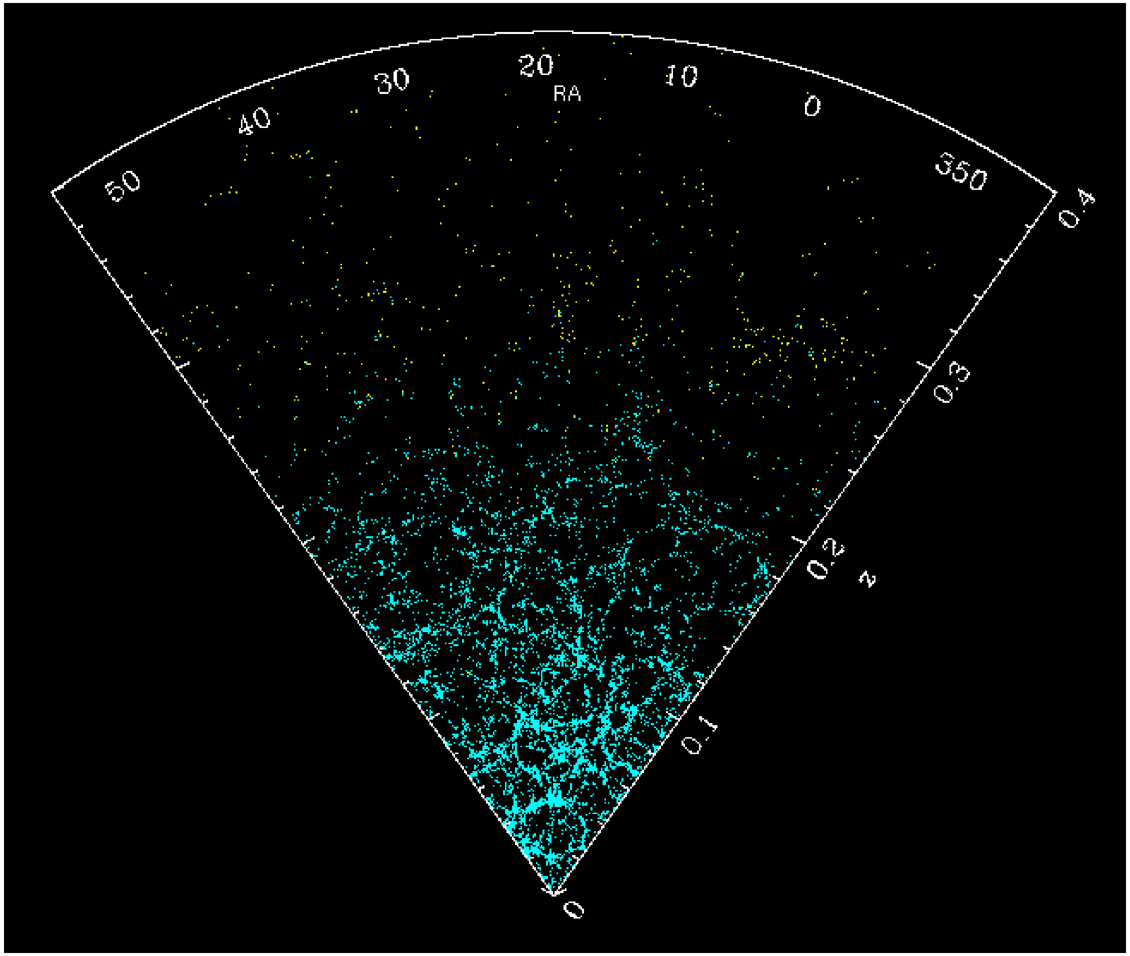
Packing bound

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.

CONJECTURE: should actually be *D'* (the intrinsic dimension).

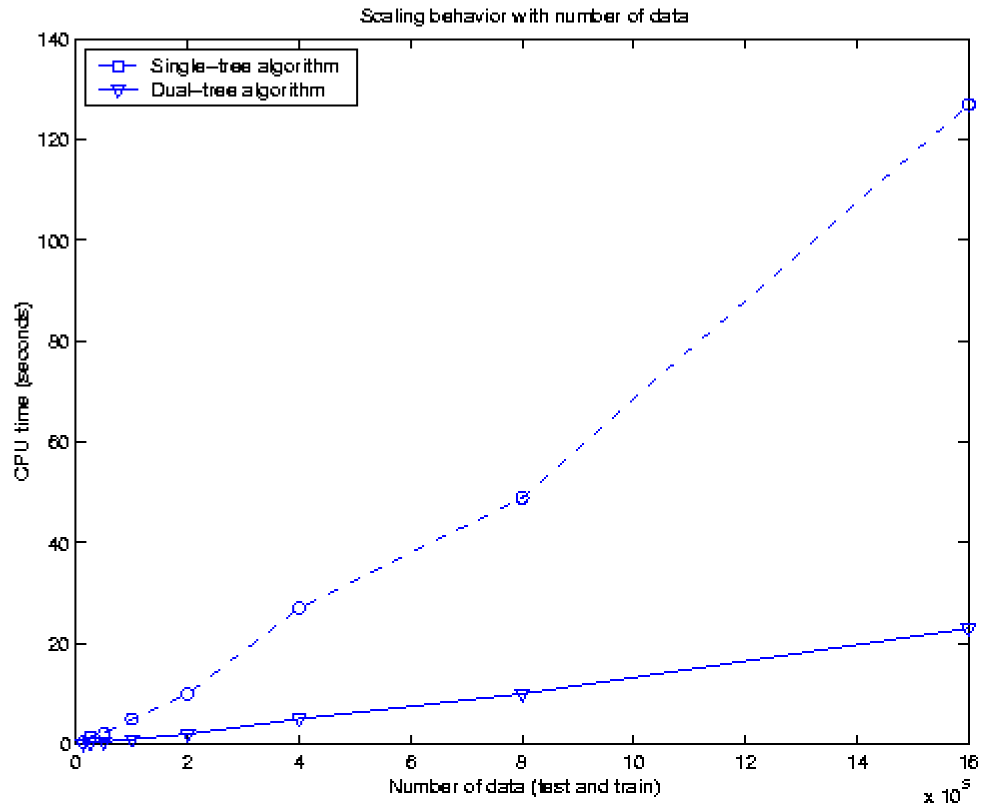
Real data: SDSS, 2-D



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

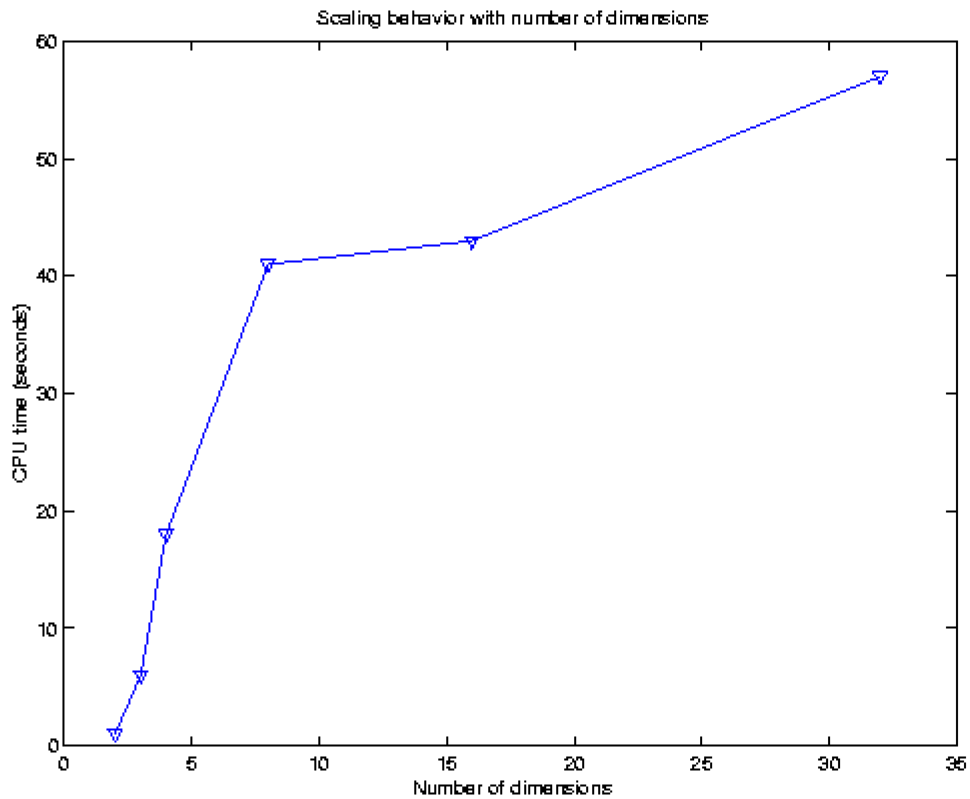
Speedup Results: Different kernels

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

Meets desiderata? Nonparametric statistics

- Accuracy good enough? yes
- Separate query and reference datasets? yes
- Variable-scale kernels? yes
- Multiple scales simultaneously? yes
- Nonisotropic kernels? yes
- Arbitrary dimensionality? yes, but not ultra-high
- Allows all desired kernels? mostly
- Extends to regression, locally-weighted, etc.? yes
- Field-tested, compared to existing methods? yes

→ [Gray and Moore, 2003]

Meets desiderata?
Smoothed particle hydrodynamics

- Accuracy good enough?   yes
- Variable-scale kernels?   yes
- Nonisotropic kernels?   yes
- Allows all desired kernels?   yes
- Edge-effect corrections (mixed kernels)?   yes
- Highly non-uniform data?   yes
- Fast tree-rebuilding?   yes, soon perhaps faster
- Time stepping integrated? no
- Field-tested, compared to existing methods?   no

Meets desiderata?
Coulombic simulation

- Accuracy good enough?   open question
- Allows multipole expansions?   yes
- Allows all desired kernels?   yes
- Fast tree-rebuilding?   yes, soon perhaps faster
- Time stepping integrated?   no
- Field-tested, compared to existing methods?   no
- Parallelized?   no

Summary

- *O(N)* can be achieved **independent of multipole expansions;** provable rather than arguable
- New lightweight dual-tree algorithm: explores **tradeoff between geometry and approximation**
- Well-suited to statistics problems; plausibly useful in physics problems

→ Looking for comments and collaborators!

agray@cs.cmu.edu

THE END

Simple recursive algorithm

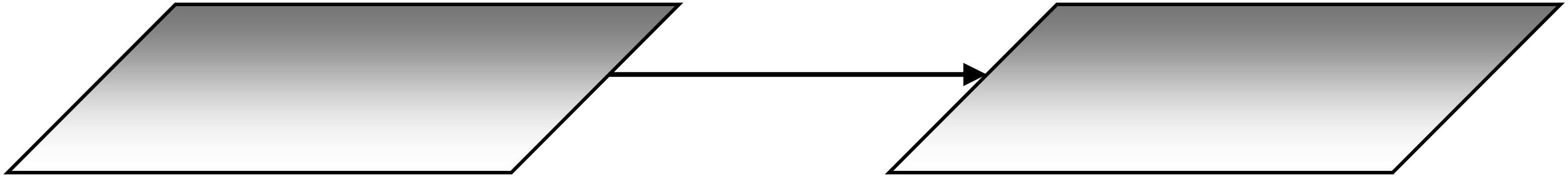
```
DualTree(Q,R)
{
  if approximate(Q,R), return.

  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 <u>node-node</u> bounds.

O(D) bounds on distance minima/maxima:



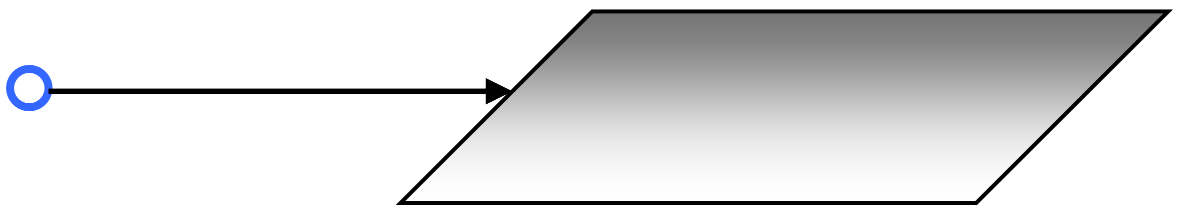
(Analogous to point-node bounds.)

Also needed:

Nodewise bounds.

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

O(D) bounds on distance minima/maxima:



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

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

old stopping criterion

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

old 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