Projects

Goals

- **Apply the knowledge of parallel algorithm design and GPU-oriented optimization toward real-life problems**
- **Propose the research projects according to three provided computational or algorithmic patterns**
  - Traverse hierarchical data structure (e.g. nearest neighbor search)
  - Neighborhood update/propagation (e.g. image segmentation, Eikonal equation)
  - Adaptive refinement (Adaptive mesh refinement, Supersampling)

Documents

- **Project proposal**
  - 4-Page ACM SIG Conference template (double column)
  - Included sections
    - Motivation (application)
    - Related work (literature reviews and direct motivation)
    - Research questions (in parallel computation)
    - Proposed solutions and contributions
- **Project Report**
  - 5 pages
  - Add research results/evaluation, modify proposal

Schedule

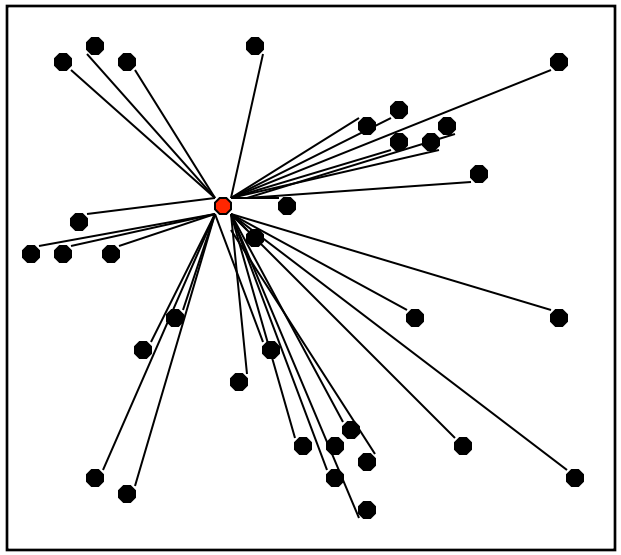
- Proposal due in one week (April 8th)
- Final presentation at last week of the class (May 8th)
- Final report due at May 13th (Monday)

Project Team

- **Two person team**
  - Start teaming now
  - Contact me if you can find a teammate
- **In final report, clearly define each team members work and contribution (one paragraph)**
- **Allow one person team for special cases**

Example 1: Nearest Neighbor Search

- Given a query point X.
- Scan through each point Y
- Takes O(N) time for each query!



33 Distance Computations

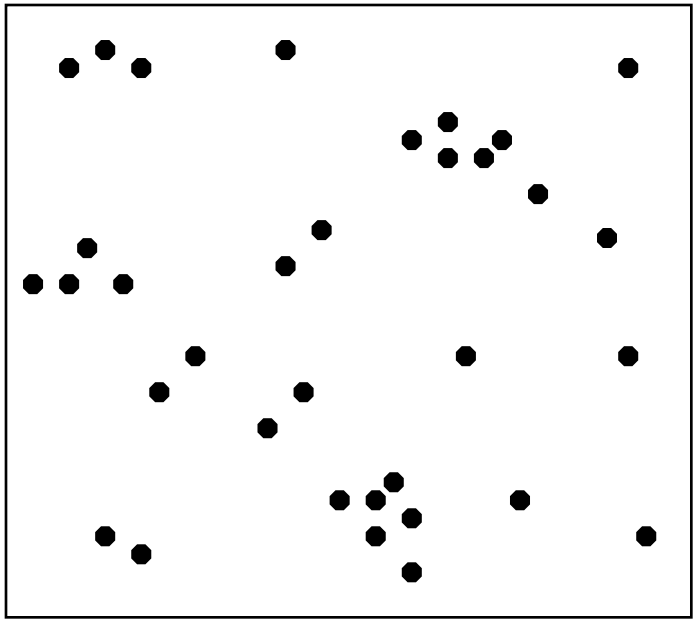
Acceleration: Spatial Index Structure

- **We can speed up the search for the nearest neighbor:**
  - Examine nearby points first.
  - Ignore any points that are further then the nearest point found so far.

K-D Tree

- k-d tree is a multidimensional binary search tree.
- Recursively partitions points into axis aligned boxes. One axis at a time.

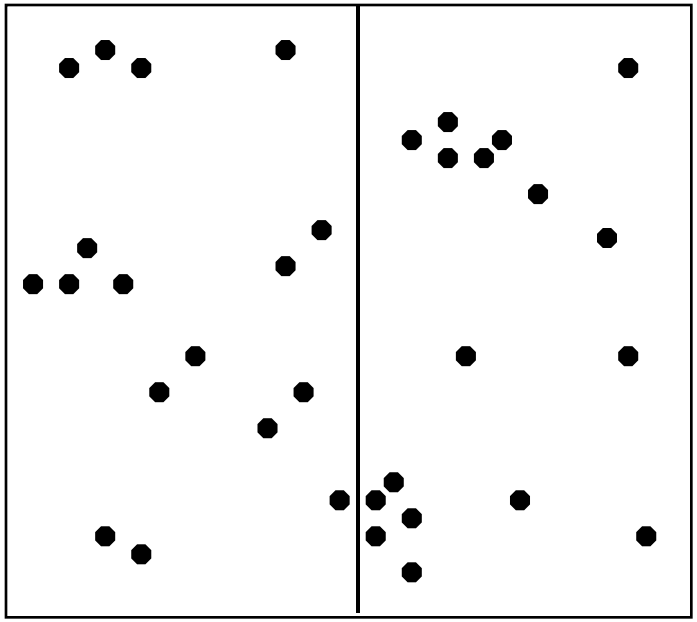
KD-Tree Construction

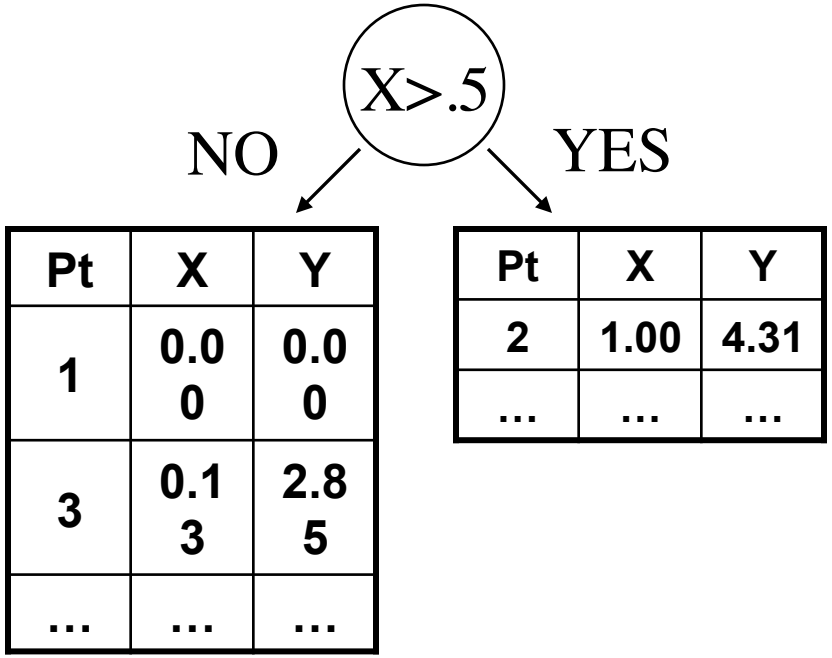


Pt	X	Y
1	0.00	0.00
2	1.00	4.31
3	0.13	2.85
…	…	…

We start with a list of n-dimensional points.

KD-Tree Construction



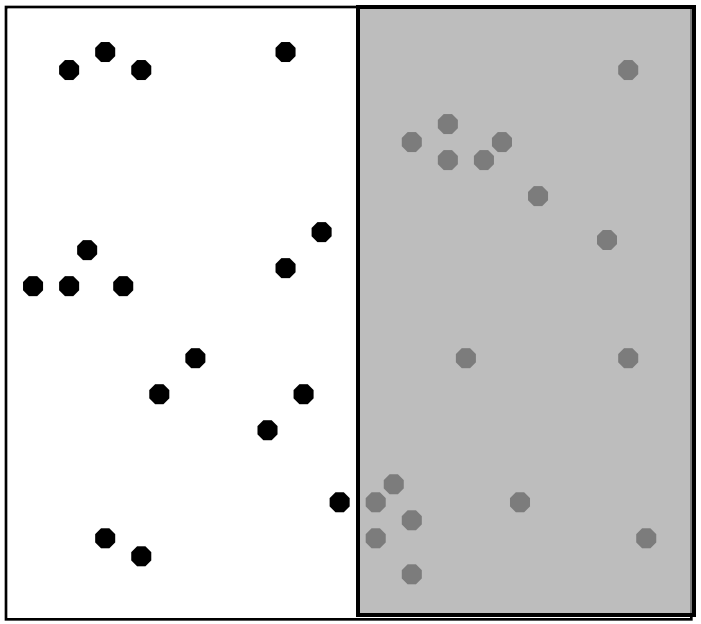


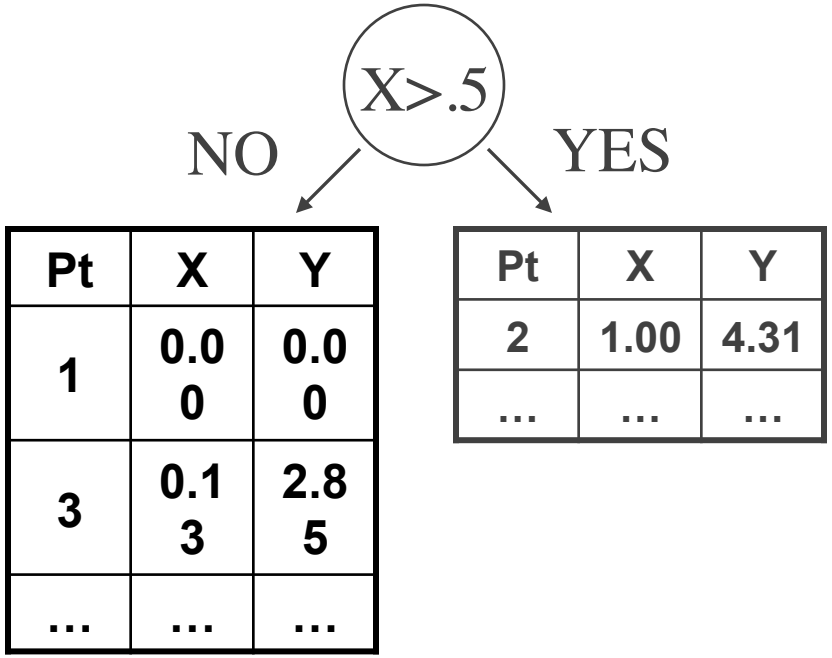
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	1.00	4.31
…	…	…

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 <= V.

KD-Tree Construction



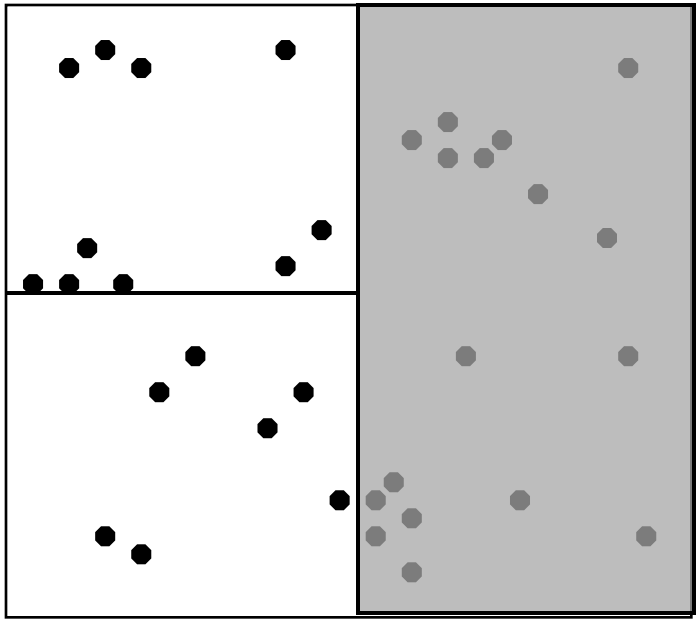


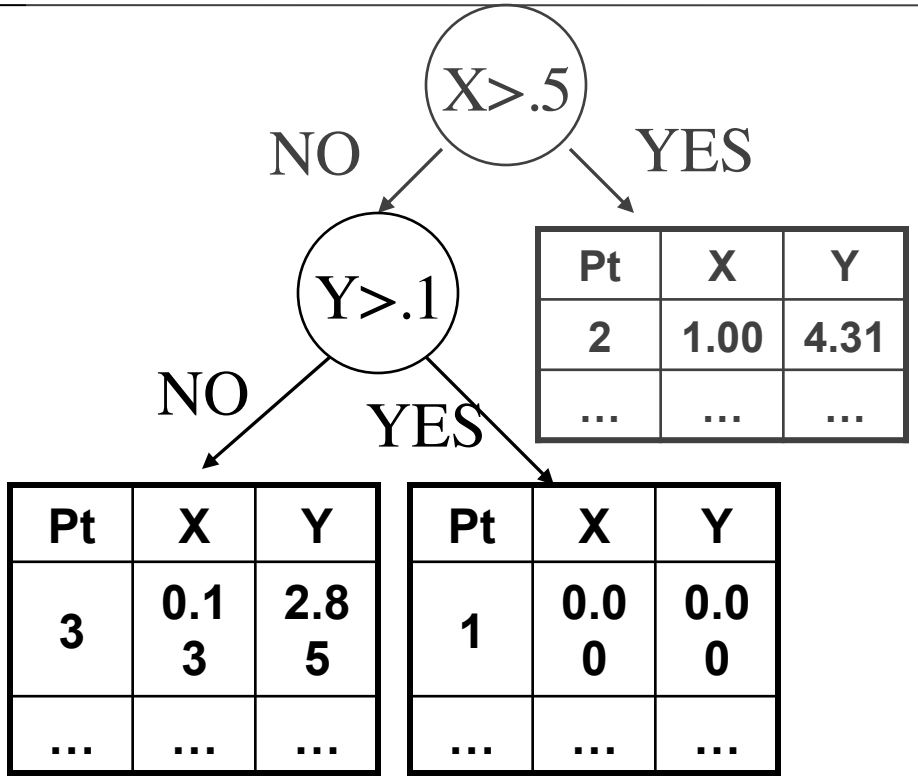
Pt	X	Y
1	0.00	0.00
3	0.13	2.85
…	…	…

Pt	X	Y
2	1.00	4.31
…	…	…

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

KD-Tree Construction





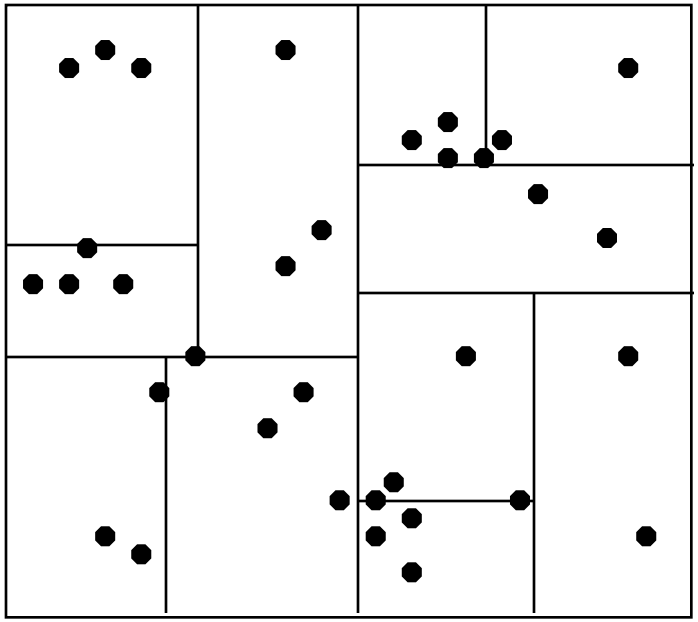
Pt	X	Y
2	1.00	4.31
…	…	…

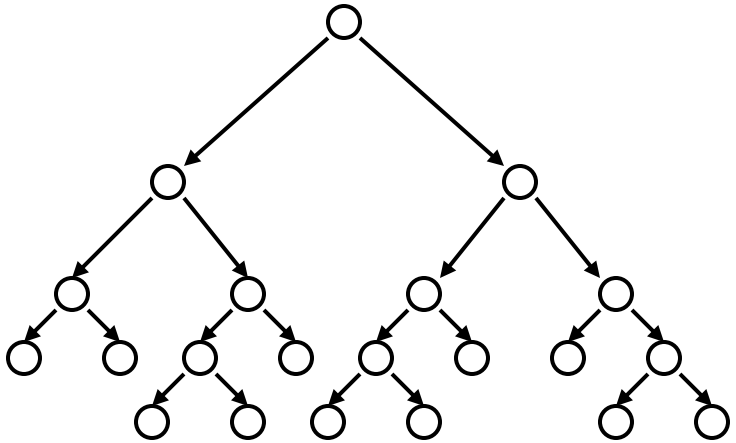
Pt	X	Y
3	0.13	2.85
…	…	…

Pt	X	Y
1	0.00	0.00
…	…	…

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

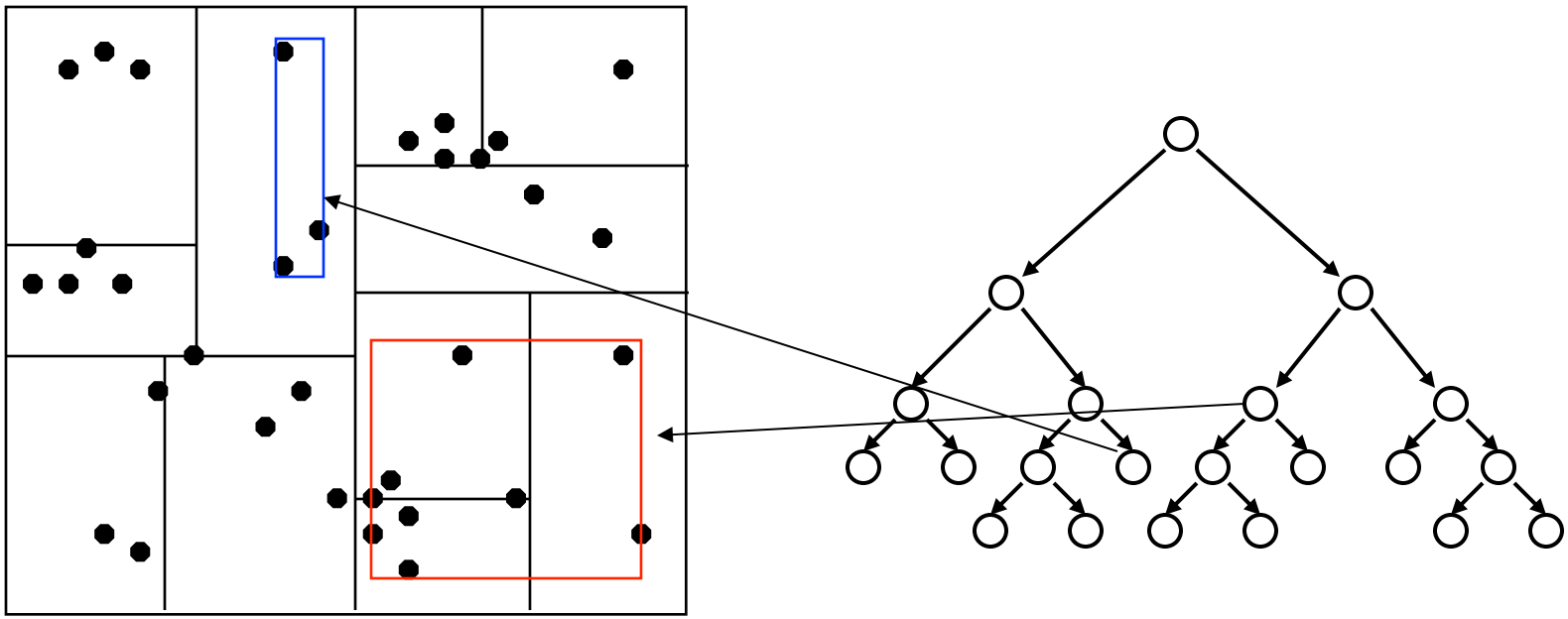
KD-Tree Construction





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

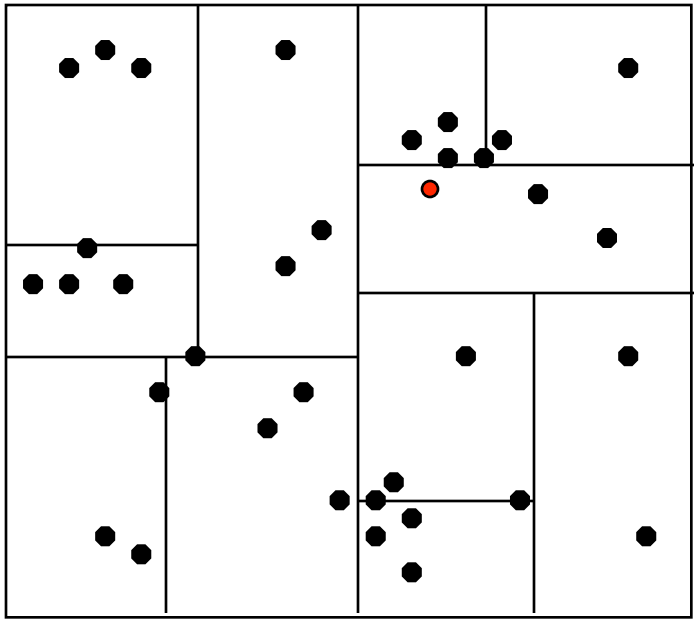


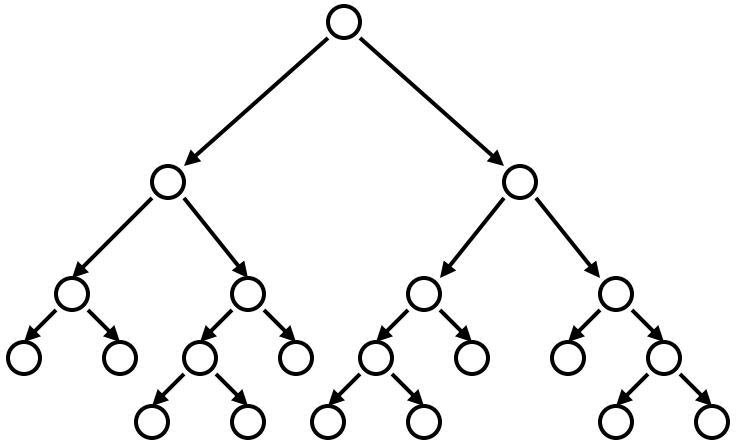
We will keep around one additional piece of information at each node. The (tight) bounds of the points at or below this node.

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 then m points left OR the box has hit some minimum width.

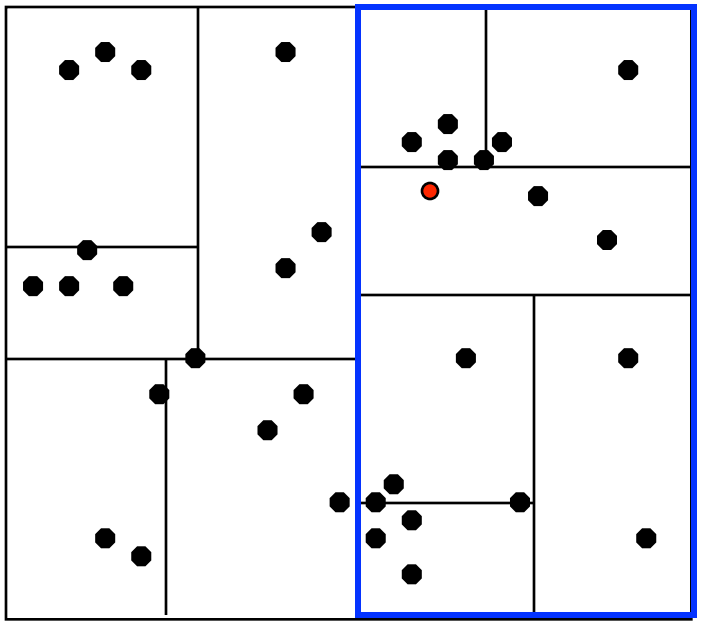
Nearest Neighbor with KD Trees

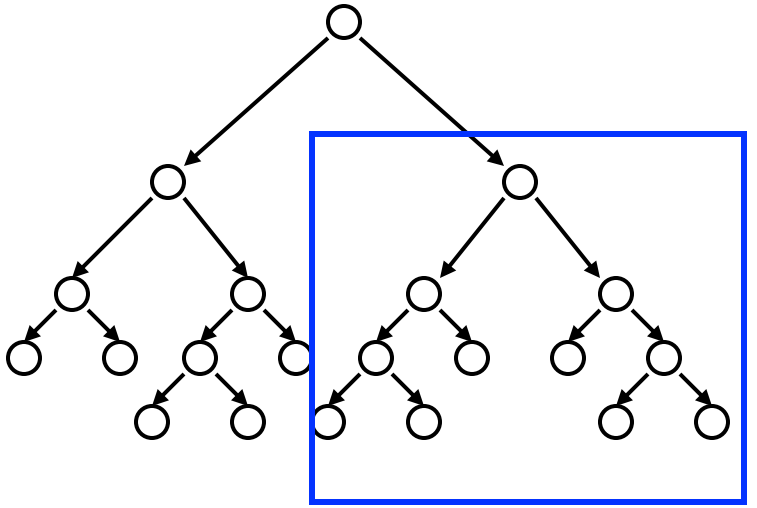




We traverse the tree looking for the nearest neighbor of the query point.

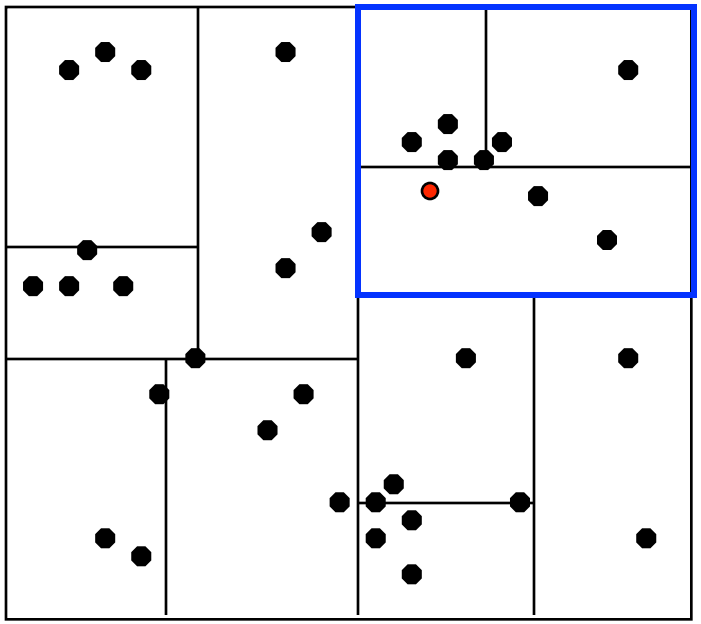
Nearest Neighbor with KD Trees

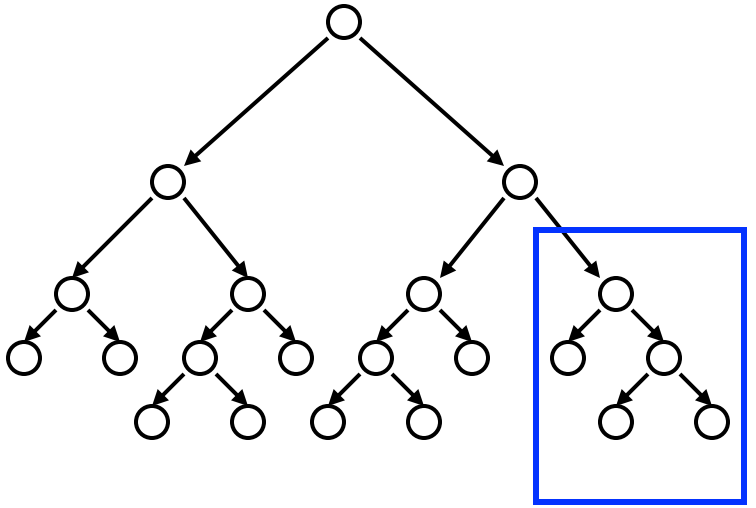




Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

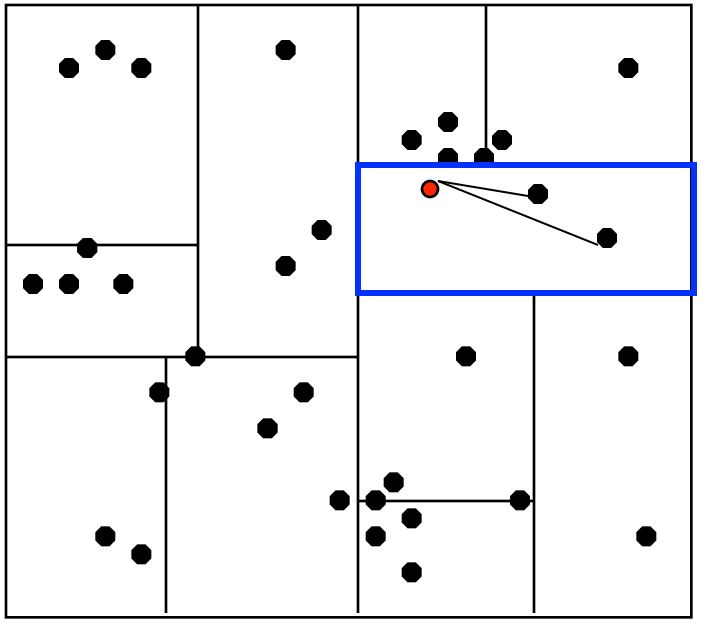
Nearest Neighbor with KD Trees

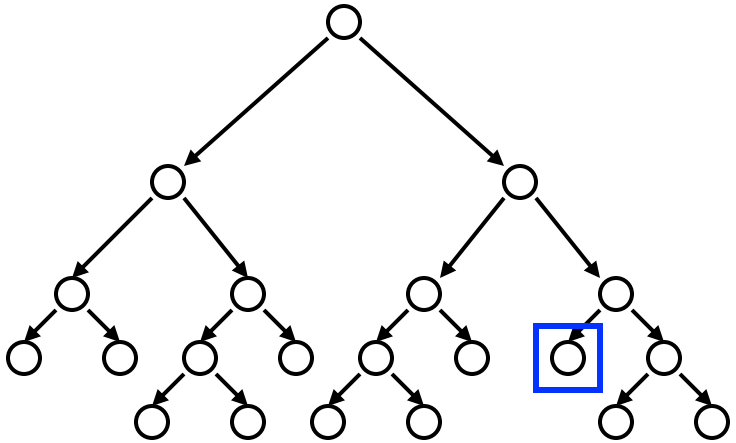




Examine nearby points first: Explore the branch of the tree that is closest to the query point first.

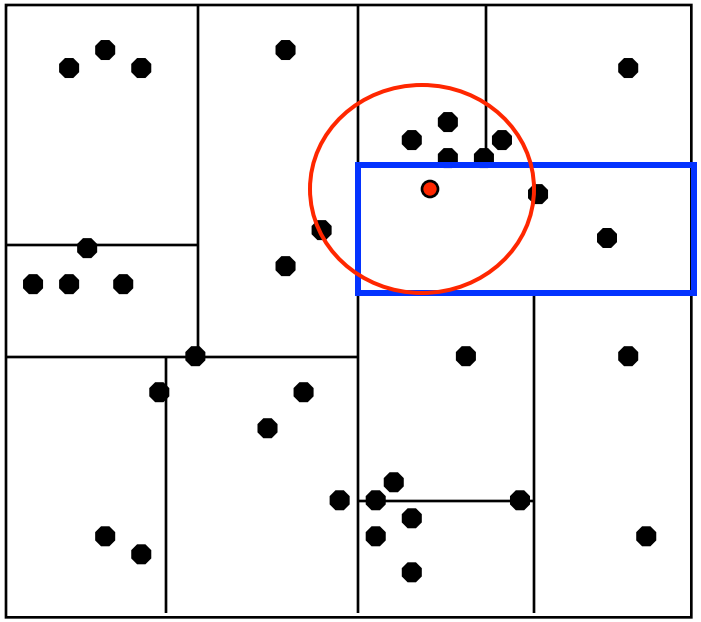
Nearest Neighbor with KD Trees

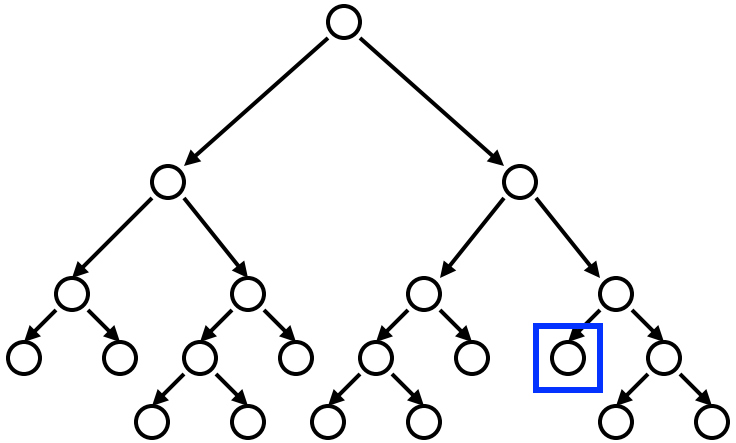




When we reach a leaf node: compute the distance to each point in the node.

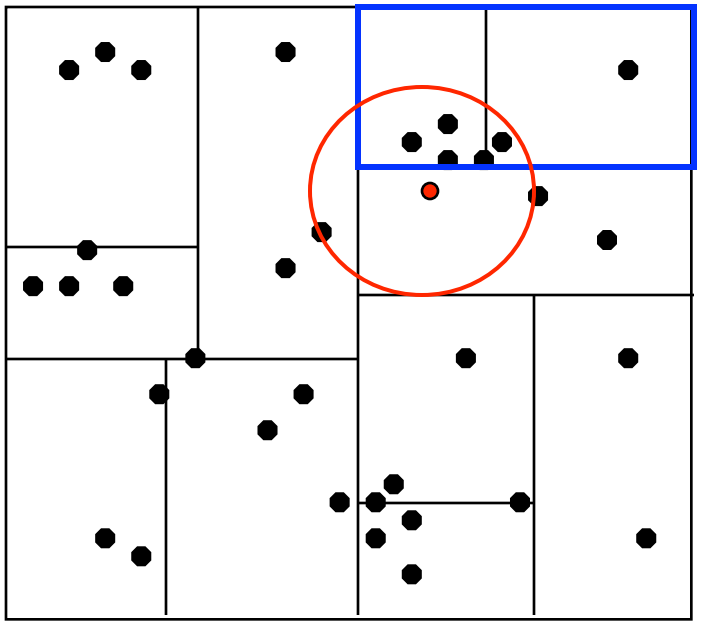
Nearest Neighbor with KD Trees

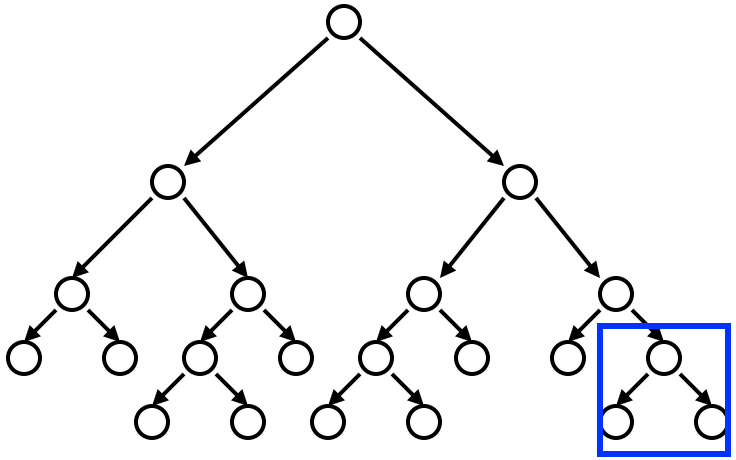




When we reach a leaf node: compute the distance to each point in the node.

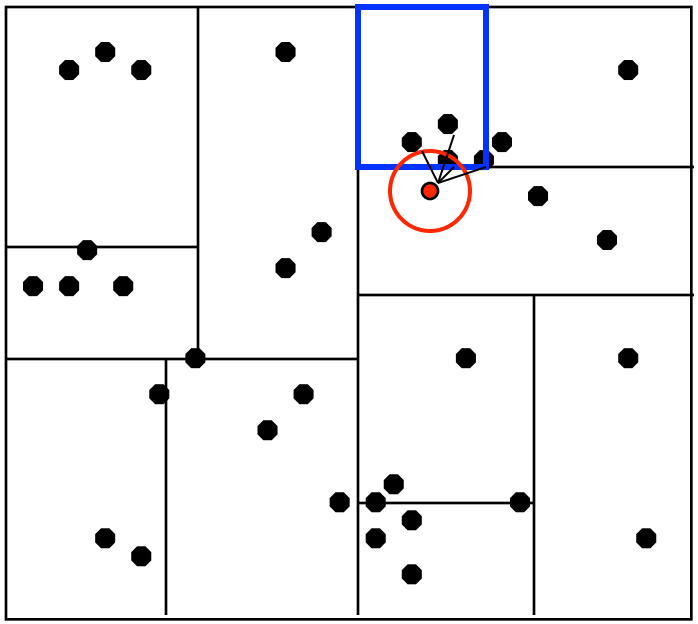
Nearest Neighbor with KD Trees

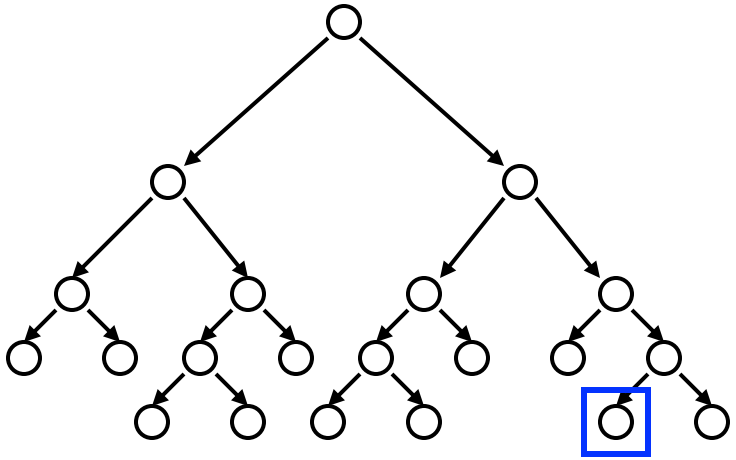




Then we can backtrack and try the other branch at each node visited.

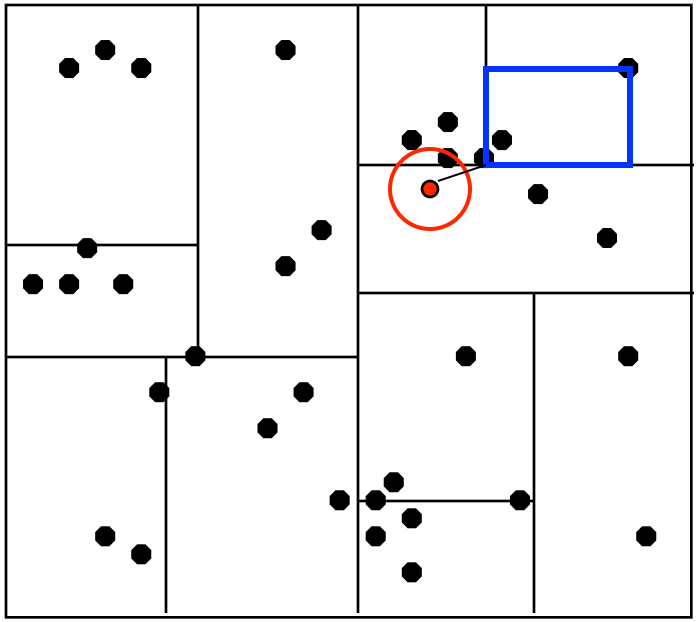
Nearest Neighbor with KD Trees

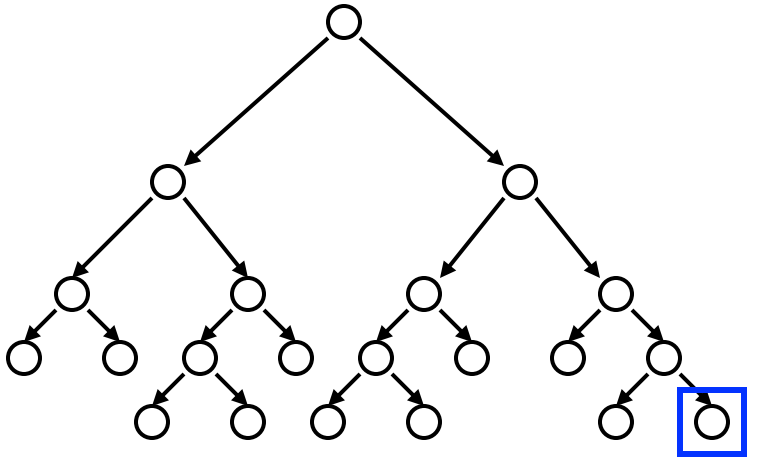




Each time a new closest node is found, we can update the distance bounds.

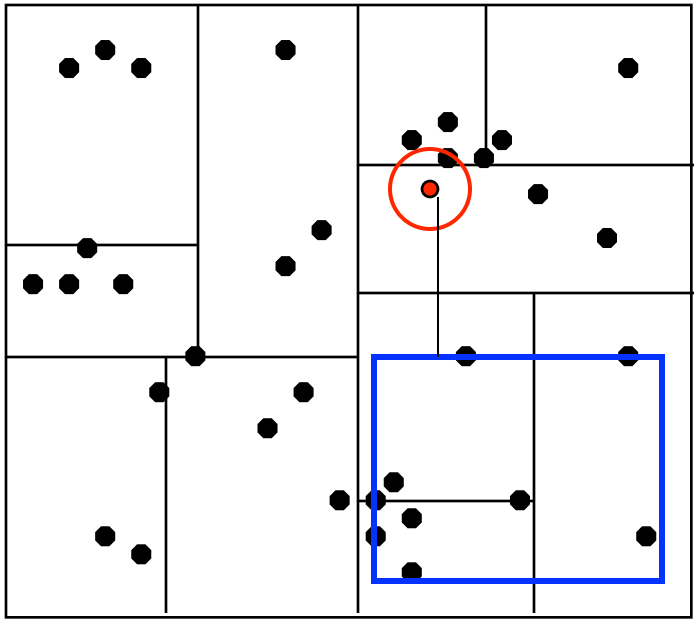
Nearest Neighbor with KD Trees

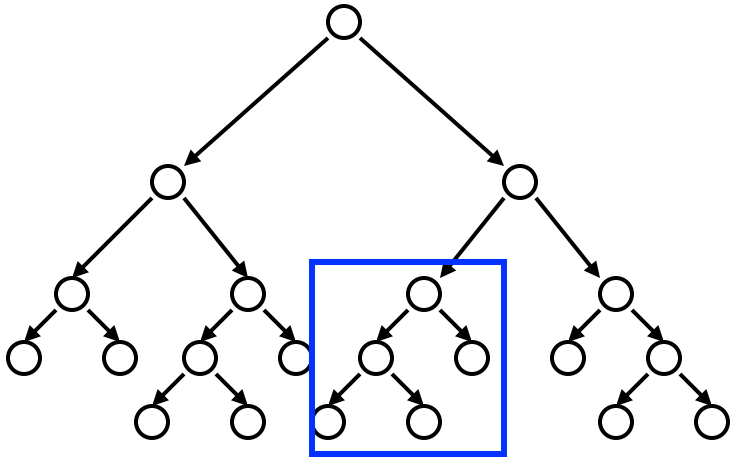




Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

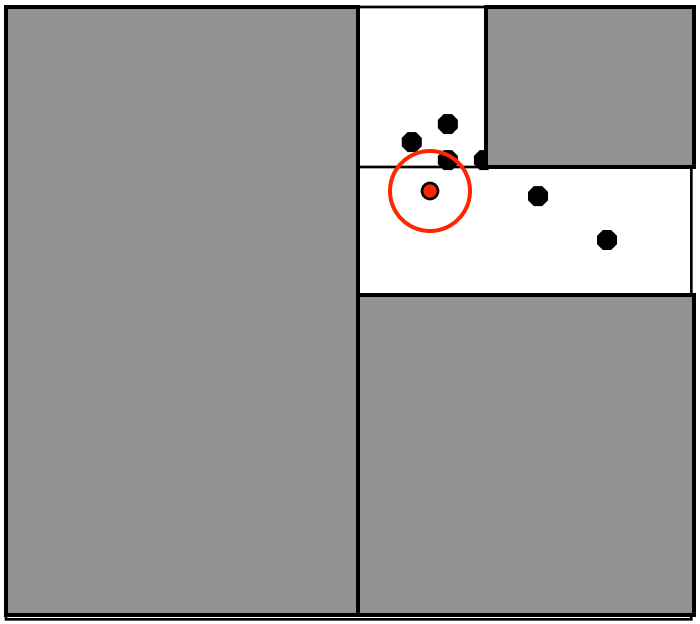
Nearest Neighbor with KD Trees

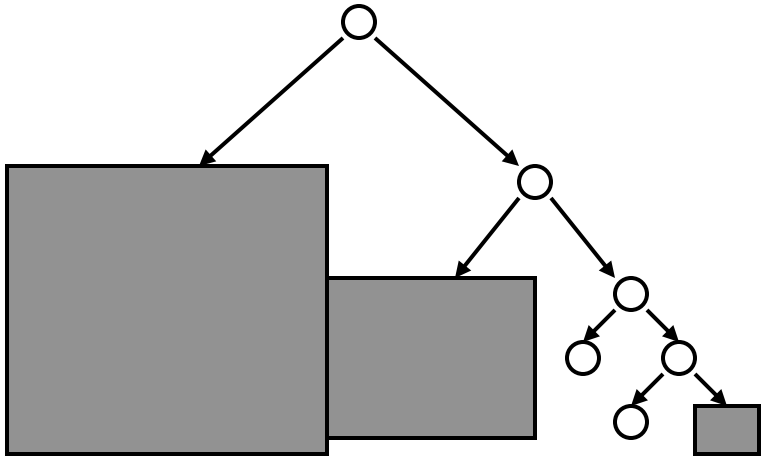




Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Nearest Neighbor with KD Trees

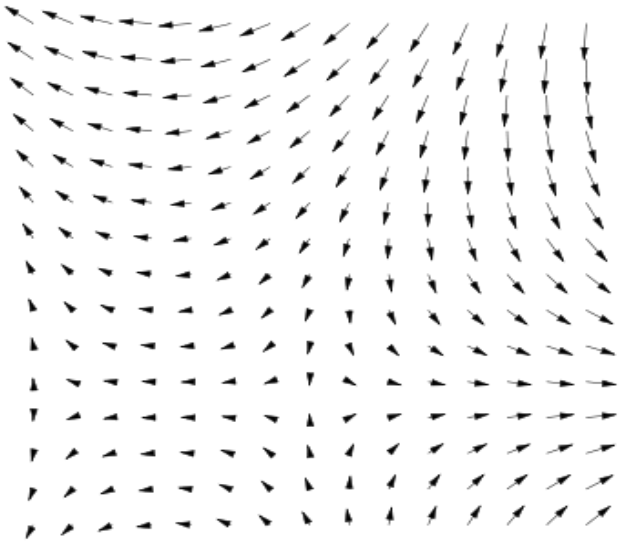




Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Crowd Simulation:

- Global path planning for massive agents in real-time
  - Can not solve for each agent (does not scale)
- Solution:
  - Models a crowd as a flow
  - Uses physics-based equations (PDEs)



- Divides the area into a discrete grid
- Creates a vector field for each grid cell
- Each agent simply follows the vectors of its nearest cells

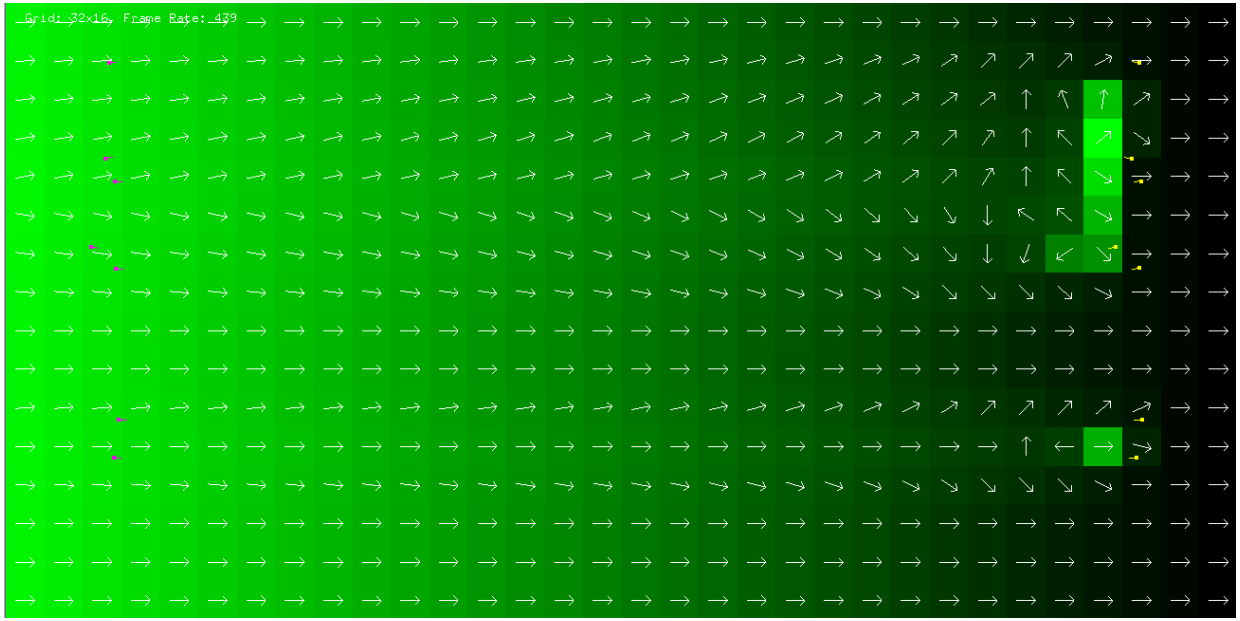
Example 2: Eikonal Equation

- Physics-based wave propagation equation
- Guarantees minimum cost path with no local minima

$$\left\|\nabla\phi(x)\right\| = C(x)$$

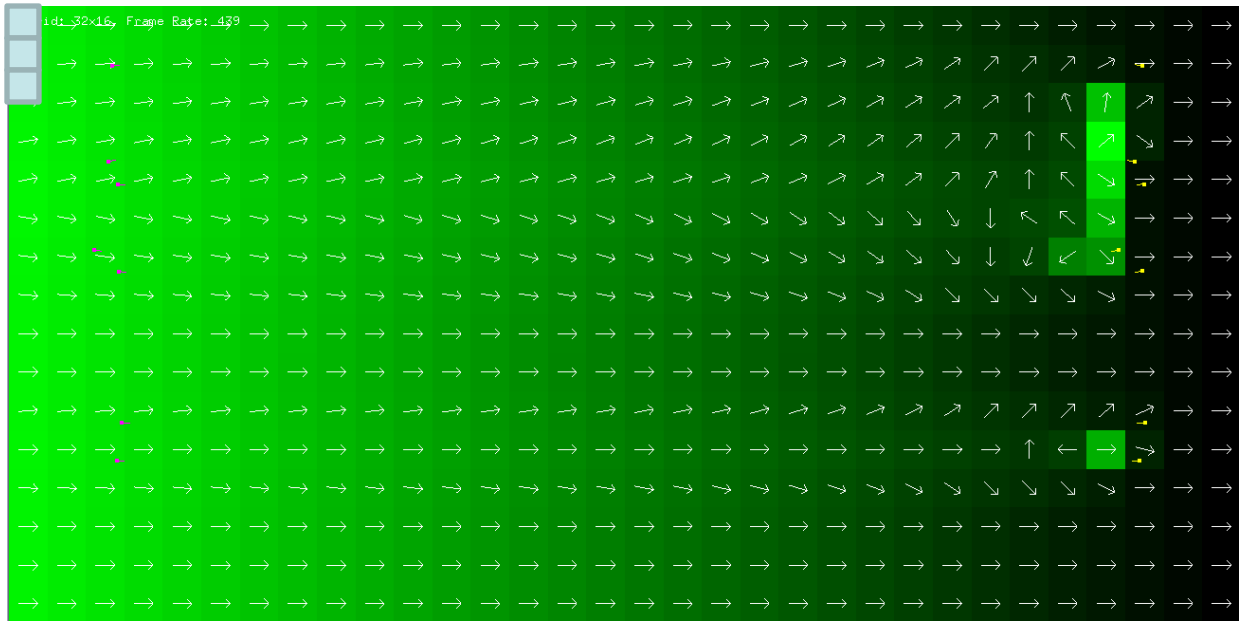
Cost function

Potential field



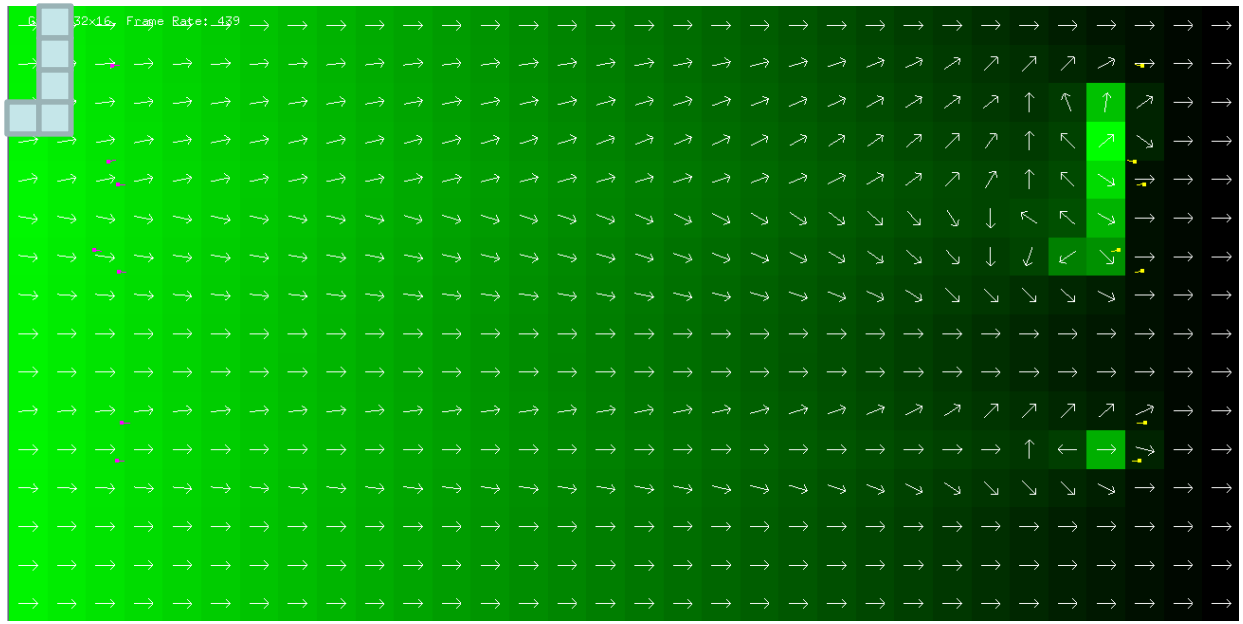
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima



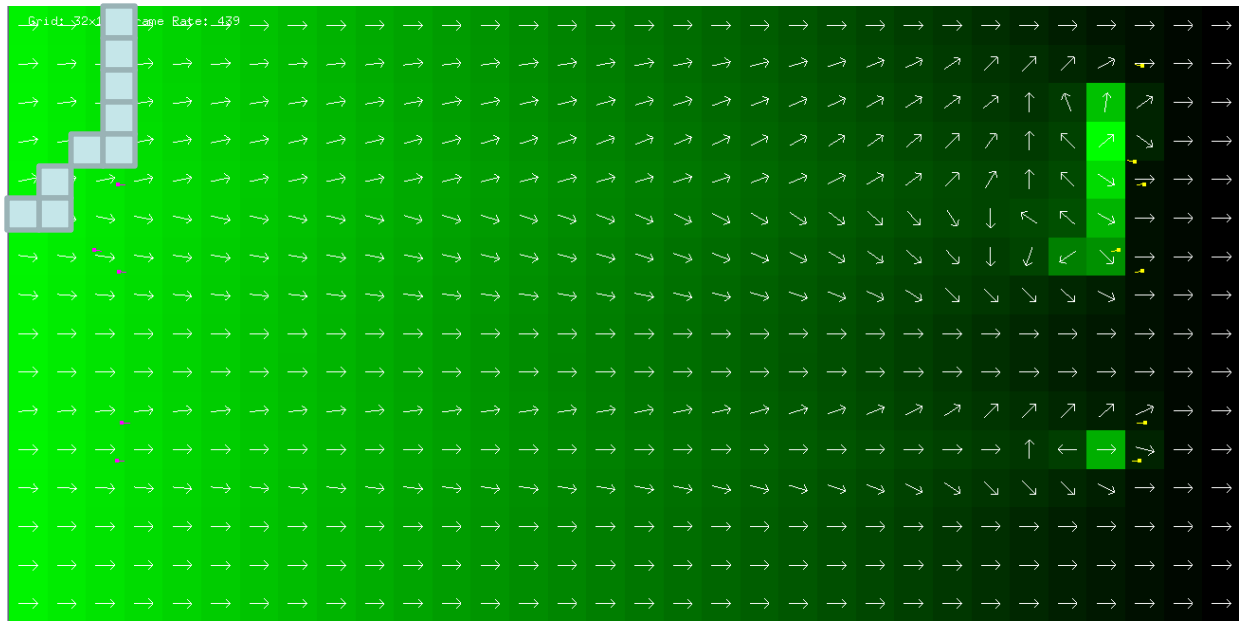
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima



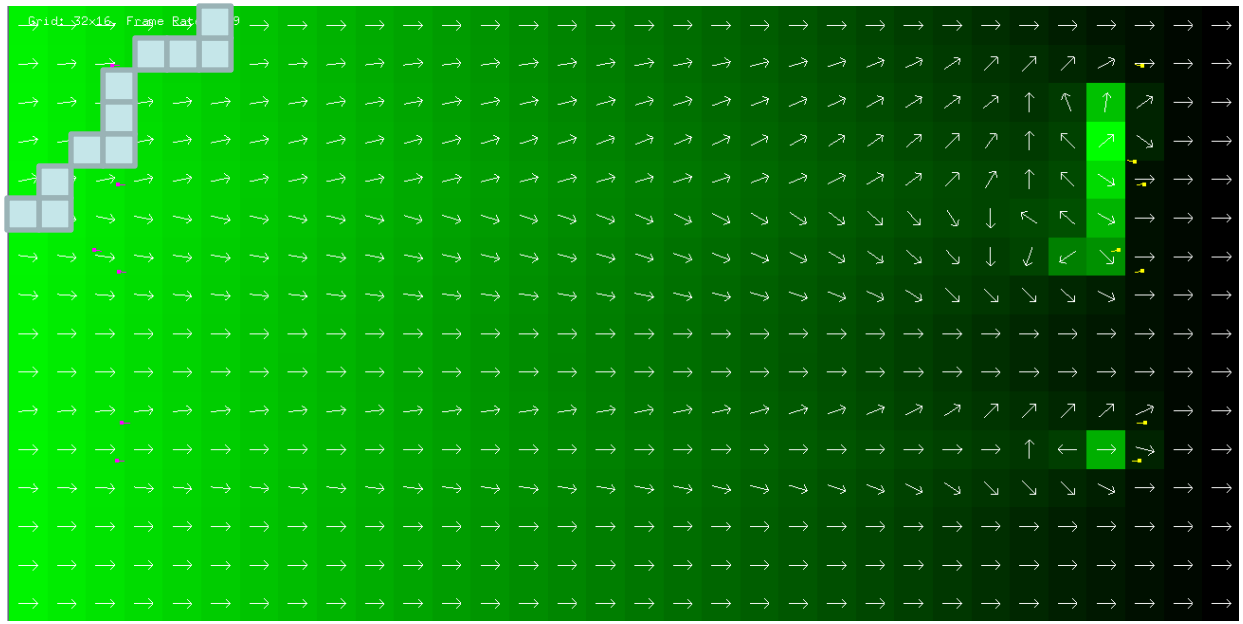
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima



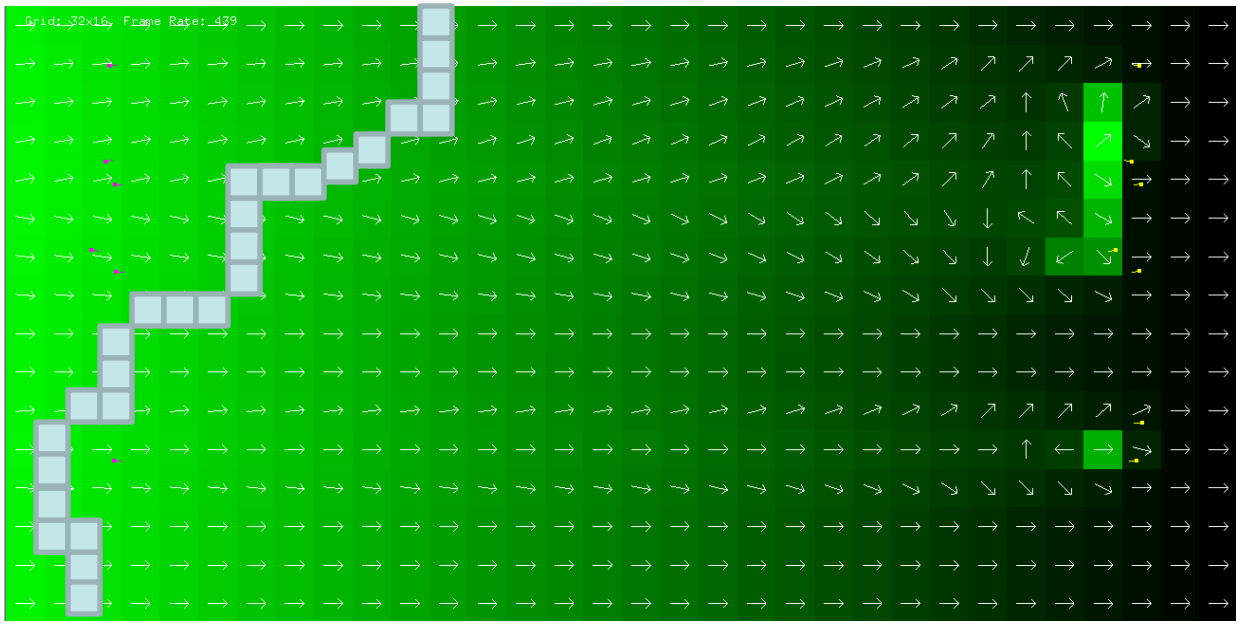
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima

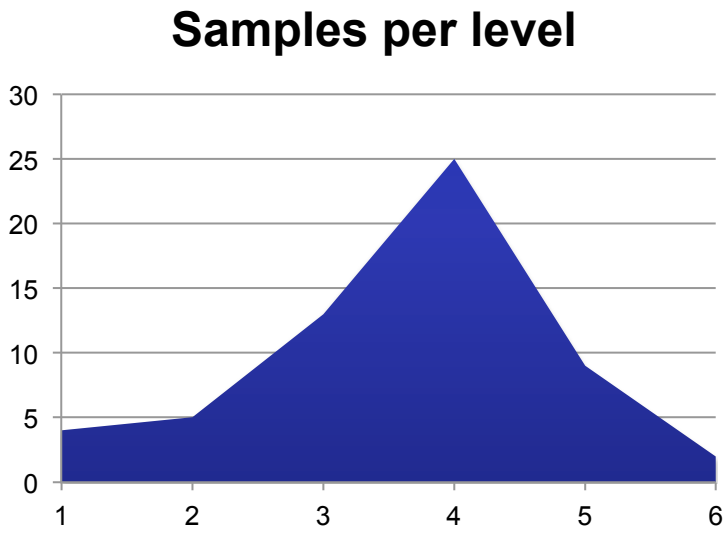


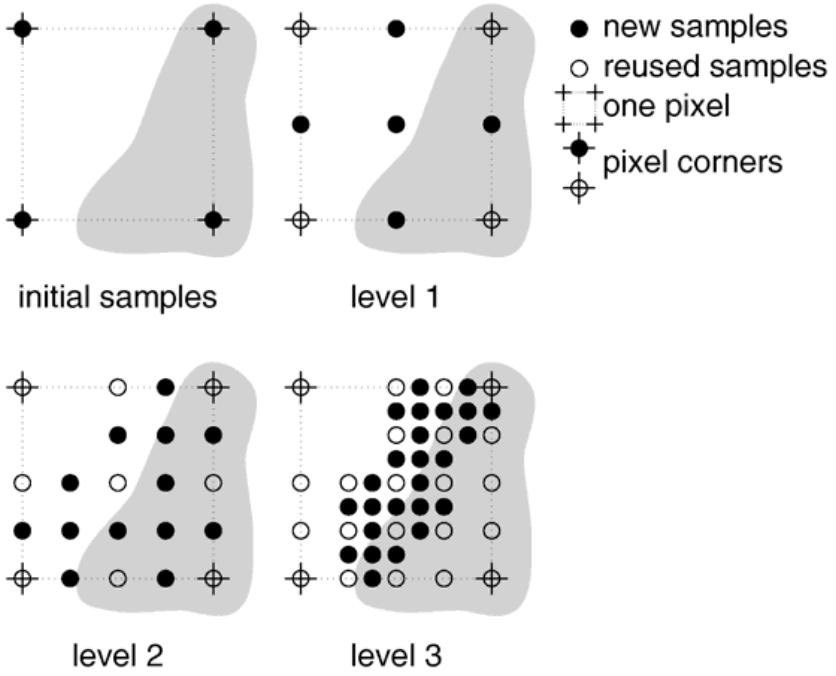
Eikonal Equation

- The numeric method for solving Eikonal equation is non-data-parallel
- Guarantees minimum cost path with no local minima



Example 3: Adaptive Super-Sampling





POV ray document