Lecture 16: Space Partitioning

Rui Wang
Topics

- Voronoi Diagram
- Uniform Grid
- KD-Tree
- kNN Search
Given a set of points $S$ in Euclidean space, find a partitioning of space into cells, such that any point within a cell is closer to one point in $S$ than any other point in $S$.

Example:
Voronoi Diagram

- The dual graph of a Voronoi diagram is a Delaunay triangulation

- Black dots: point set S
- Red lines: Voronoi diagram
- Black lines: Delaunay triangles
Delaunay Triangulation

- A triangulation of point set $S$ in Euclidean space, such as no point in $S$ is inside the circumcircle of any triangle.
Voronoi Diagram

- Computing Voronoi Diagram:
  - Bowyer-Watson algorithm
  - Fortune's algorithm
  - Lloyd's algorithm
Voronoi Diagram

- Computing Voronoi Diagram:
  - Bowyer-Watson algorithm
  - Fortune's algorithm
  - Lloyd's algorithm
- Let's think about a brute-force numerical solution:
  - Focus on 2D, and compute the solution on an NxN image (solution space)
  - For every pixel in the image, classify it to its closest Voronoi center.
Voronoi Diagram

- Computing Voronoi Diagram:
  - Bowyer-Watson algorithm
  - Fortune’s algorithm
  - Lloyd’s algorithm
- Let's think about a brute-force numerical solution:
  - Focus on 2D, and compute the solution on an NxN image (solution space)
  - For every pixel in the image, classify it to its closest Voronoi center.
- But this sounds very expensive, especially if the point set is large!
Space Partitioning

- Question: given a point set $S$, and an arbitrary point $p$ in space, how to quickly find the closest point in $S$ to $p$?
- This is called the nearest neighbor (NN) search problem.
Space Partitioning

- Question: given a point set $S$, and an arbitrary point $p$ in space, how to quickly find the closest point in $S$ to $p$?
- This is called the nearest neighbor (NN) search problem.
- A key to make it efficient is to quickly eliminate groups of points that cannot possibly be the solution.
- Use space partitioning data structures!
Uniform Grid

- Uniformly partition the space into cells
  - Keep track of the list of points inside each cell.
Uniform Grid

- Given a query point, how would this help?
Uniform Grid

- Start from the cell that the query point belongs to
Uniform Grid

- Find the closest point
Uniform Grid

- Are we done yet? Why?
Uniform Grid

- Search is continued in concentric circles centered around the query point. All cells touching the circle must be searched.
Uniform Grid

- GPU Implementation Steps
  1) Build uniform grid
  2) Search
KD-Tree

- Short for *K-Dimensional Tree*. Use axis-aligned planes/lines to recursively split the original point set into subsets.
- Just like 'split', except this is in k-dimensional, so the each split is with respect to a possibly different dimension.
- Result in a binary tree.
KD-Tree

- Where to put the splitting plane?
  - Mid-point Split: center of bounding box
KD-Tree

- Where to put the splitting plane?
  - *Mid-point Split*: center of bounding box
    - Does not work so well for non-uniform points
KD-Tree

- Where to put the splitting plane?
  - **Median Split**: median of the current group of points
    - More expensive to compute
KD-Tree

- Search Algorithm
  - Given a kd-tree, how do we use it to search for the closest point to the query point?
• GPU Implementation Notes

1) Build kd-tree
   - Conceptually similar to quick sort, except much easier if only using middle-point split
   - Utilize segmented scan

2) Search
   - Recursive, can use a manual stack maintained in shared memory or register space
kNN-Search

- Question: given a point set $S$, and an arbitrary point $p$ in space, find for $p$ the $k$ closest points in $S$.
- NN search is equivalent to 1NN-search
- Sometimes also imposes a maximum search radius $r_{max}$. 