Acceleration Structures

Benedict Brown
CIS 277 Spring 2015
University of Pennsylvania

Ray-Intersection on Large Models

- v1: Intersect ray with everything
- Slow: most scenes have *lots* of elements
- *Awakening* (right) has 100s of millions of triangles!

- Better: test groups of elements

Grid Acceleration

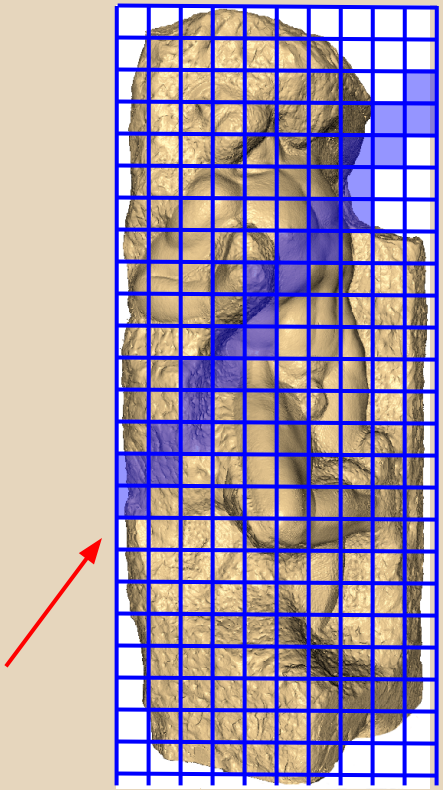
- Divide space into grid of cubes
- Each cube stores pointers to elements inside it
- Test cubes in front-to-back order

- Elements that cross a grid cell?
  - ➢ Store in both cells, or dice

Grid Acceleration

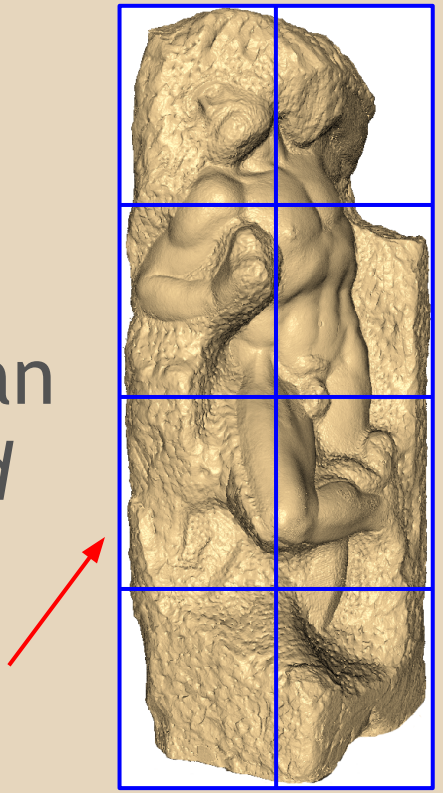
+ Simple to build
+ Easy to traverse

- What size should grid be?
- What if everything falls in one cell?



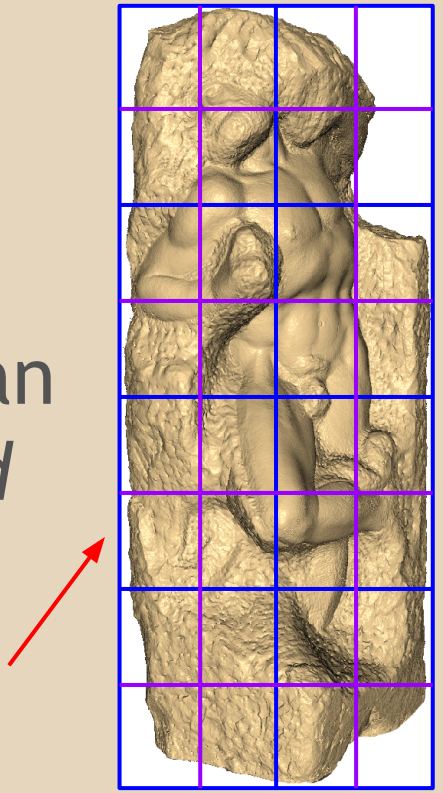
Octree Acceleration

- 3-d version of quadtree
- Divide bounding box into eight equal sub-boxes
- Recursively subdivide if more than *n* elements in cell, and depth < *d*



Octree Acceleration

- 3-d version of quadtree
- Divide bounding box into eight equal sub-boxes
- Recursively subdivide if more than *n* elements in cell, and depth < *d*



Octree Acceleration

- 3-d version of quadtree
- Divide bounding box into eight equal sub-boxes
- Recursively subdivide if more than *n* elements in cell, and depth < *d*

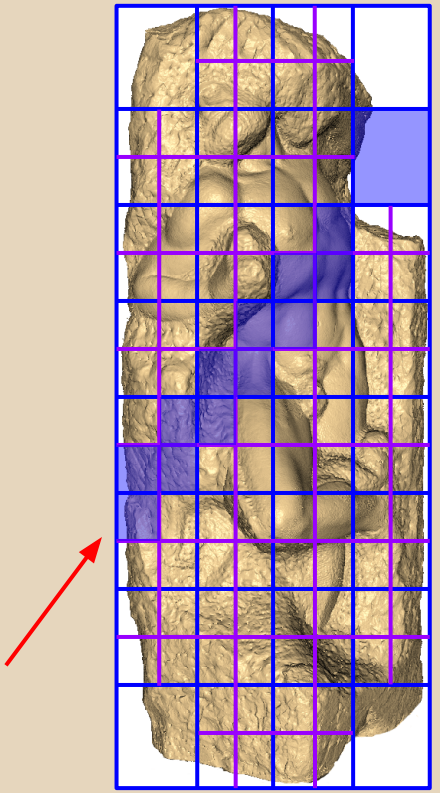
Octree Acceleration

- Traverse octree using a stack
- Leaf node? Check all elements for intersection
- Next node? Walk up hierarchy to find sibling, then walk down to next leaf

\+ Still pretty simple, adapts to geometry

\+ Easy to do dynamic updates

\- Handles large objects poorly



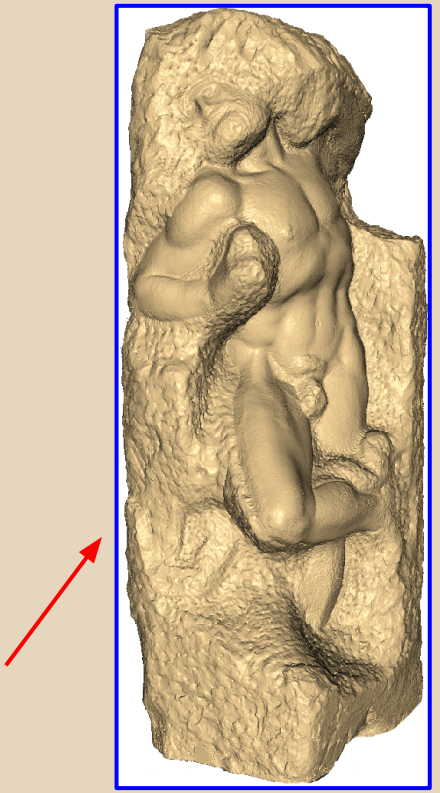
k-D Tree Acceleration

- Split along one axis at a time
- Divide wherever you want (e.g.: half of elements on each side)

\+ More flexible than octree

\+ Works great for point sets

\- Extra complexity



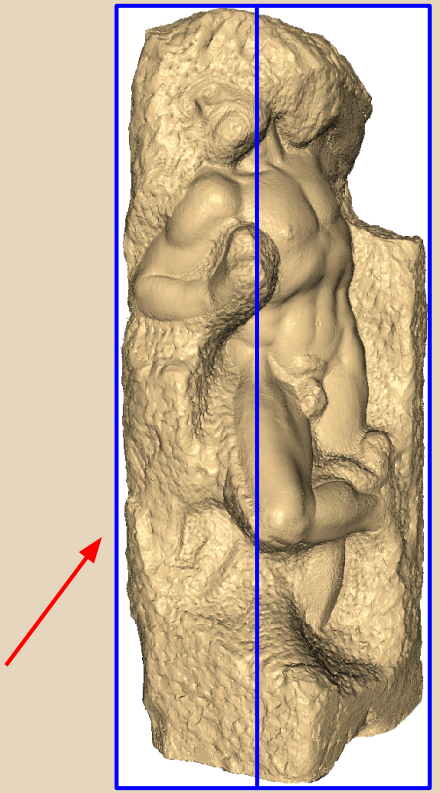
k-D Tree Acceleration

- Split along one axis at a time
- Divide wherever you want (e.g.: half of elements on each side)

\+ More flexible than octree

\+ Works great for point sets

\- Extra complexity



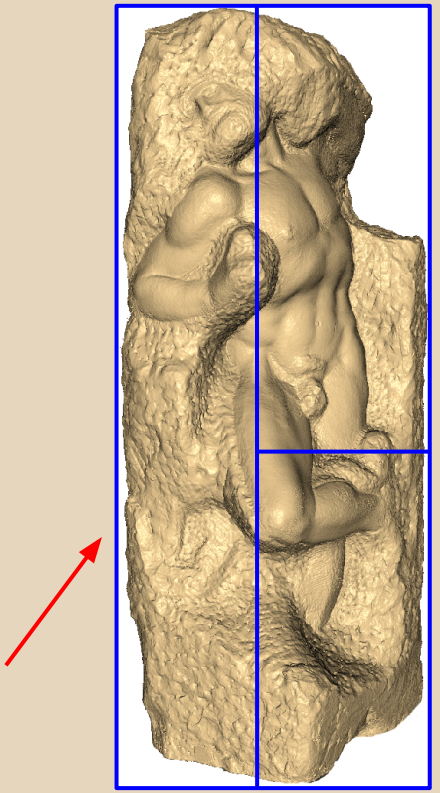
k-D Tree Acceleration

- Split along one axis at a time
- Divide wherever you want (e.g.: half of elements on each side)

\+ More flexible than octree

\+ Works great for point sets

\- Extra complexity



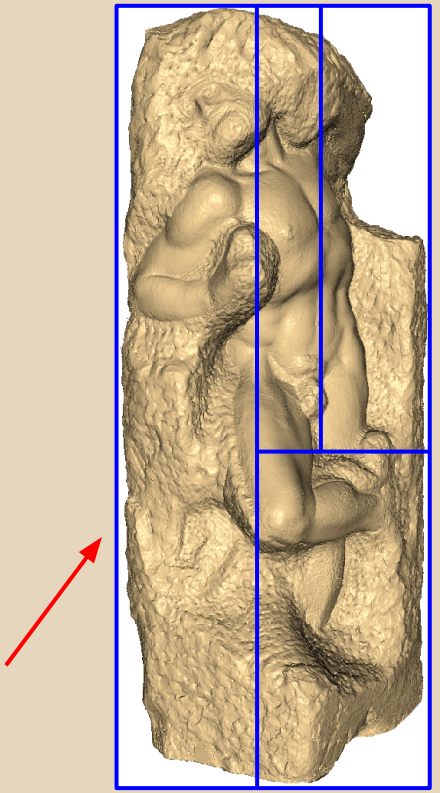
k-D Tree Acceleration

- Split along one axis at a time
- Divide wherever you want (e.g.: half of elements on each side)

\+ More flexible than octree

\+ Works great for point sets

\- Extra complexity



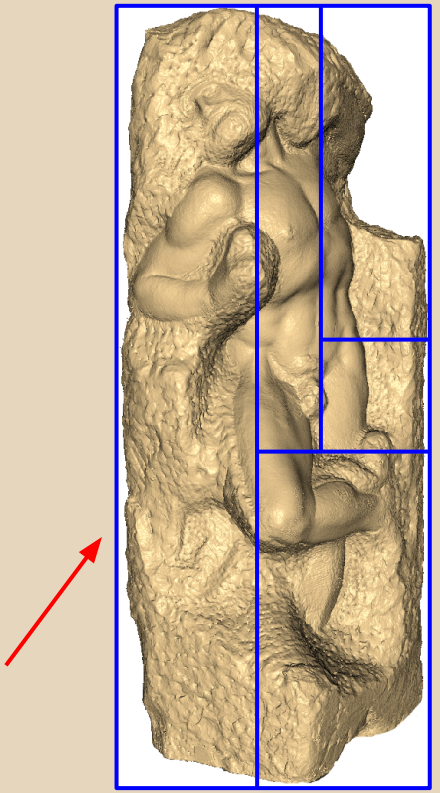
k-D Tree Acceleration

- Split along one axis at a time
- Divide wherever you want (e.g.: half of elements on each side)

\+ More flexible than octree

\+ Works great for point sets

\- Extra complexity



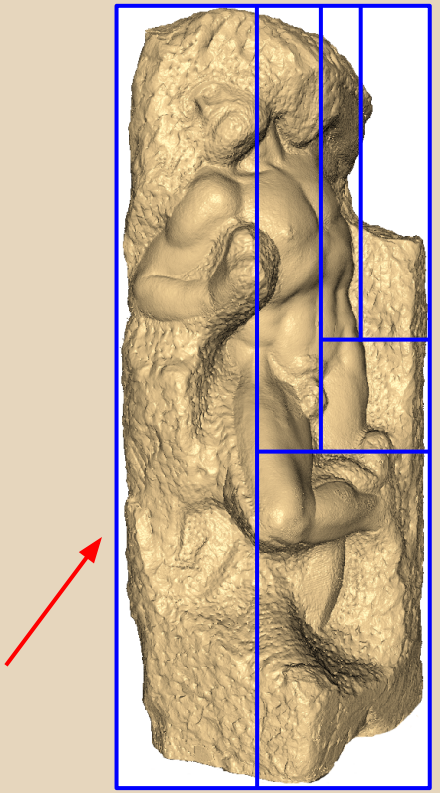
k-D Tree Acceleration

- Split along one axis at a time
- Divide wherever you want (e.g.: half of elements on each side)

\+ More flexible than octree

\+ Works great for point sets

\- Extra complexity

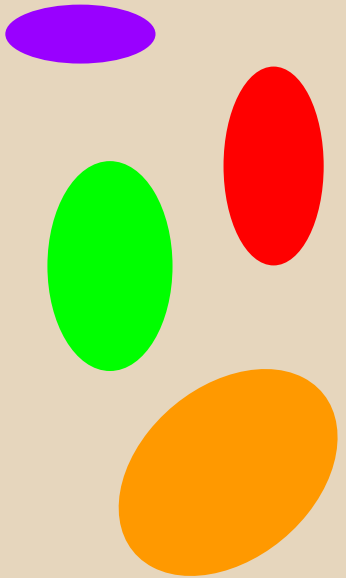


BSP Trees

- Better k-D: Binary Space Partition (BSP tree)
  - Split on arbitrary plane, not just axis-aligned
  - Tighter bounding volumes than k-D tree, still easy to traverse front-to-back
  - *Really hard to construct*
    - What is the best splitting plane
    - Numerical headaches abound
  - Doom, Quake, etc. (until Doom 3) build BSP tree based on walls. It's messy.
- BSP Demo

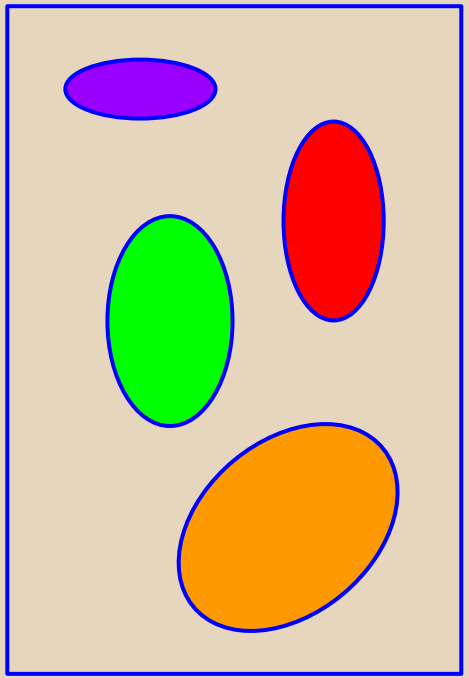
Bounding Volume Hierarchy

- Organize objects in a tree
- Bounding box at each tree node for all objects in subtree
- Root bbox is bbox for entire scene



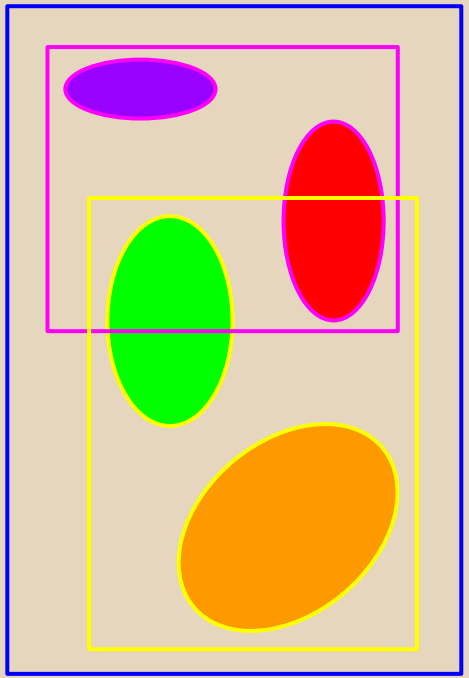
Bounding Volumes (Top-Down)

- Start with bbox of entire scene



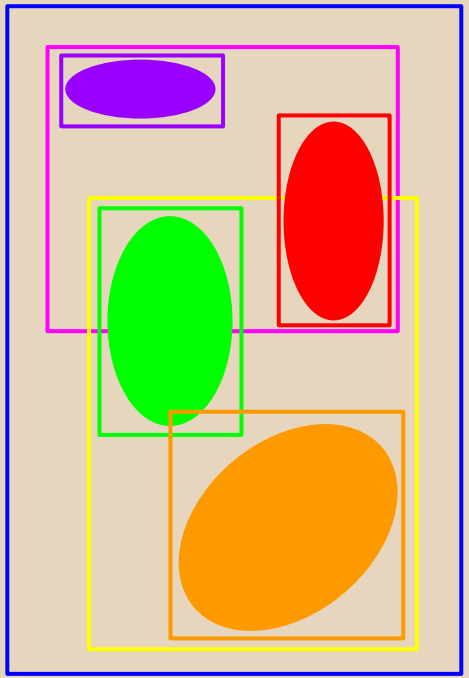
Bounding Volumes (Top-Down)

- Start with bbox of entire scene
- Split so half of objects in each child



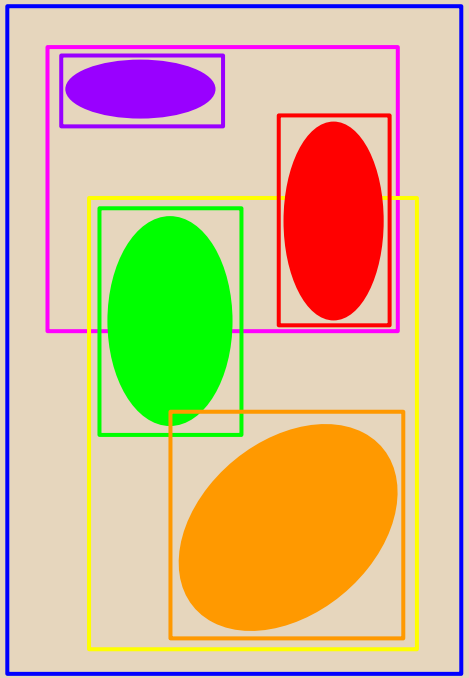
Bounding Volumes (Top-Down)

- Start with bbox of entire scene
- Split so half of objects in each child
- Split again on different axis...



Bounding Volumes (Top-Down)

+ Tighter bounding boxes than octrees and k-D trees (usually)
+ Easy to build
+ Each object in only one leaf
- Nodes have overlapping bboxes: harder to traverse



BVH: Best bounding volumes

Bounding volume not necessarily axis-aligned box

- Non-axis-aligned box is much tighter
  - Very hard to compute, so rarely bother
- Sphere has easy intersection tests
  - Also hard to compute
  - Not as tight for boxy elements (common)
  - QSplat: Group points using bounding spheres to get fast rendering + level-of-detail + progressive

BVH Variants

Lots of variants of BVH:

- Where to split
- Shape of bounding volume
- Build bottom-up instead of top-down
- Support changing geometry

BVH vs. k-D

- Top-Down BVH similar to k-D trees
  - BVH: overlapping bounding boxes
  - k-D trees: elements can be in more than one leaf node
- Where to recursively split:
  - Optimal tree structure is NP-hard problem
  - Split on mid-point: fast, but not much better than octree
  - Split on median: usually pretty good