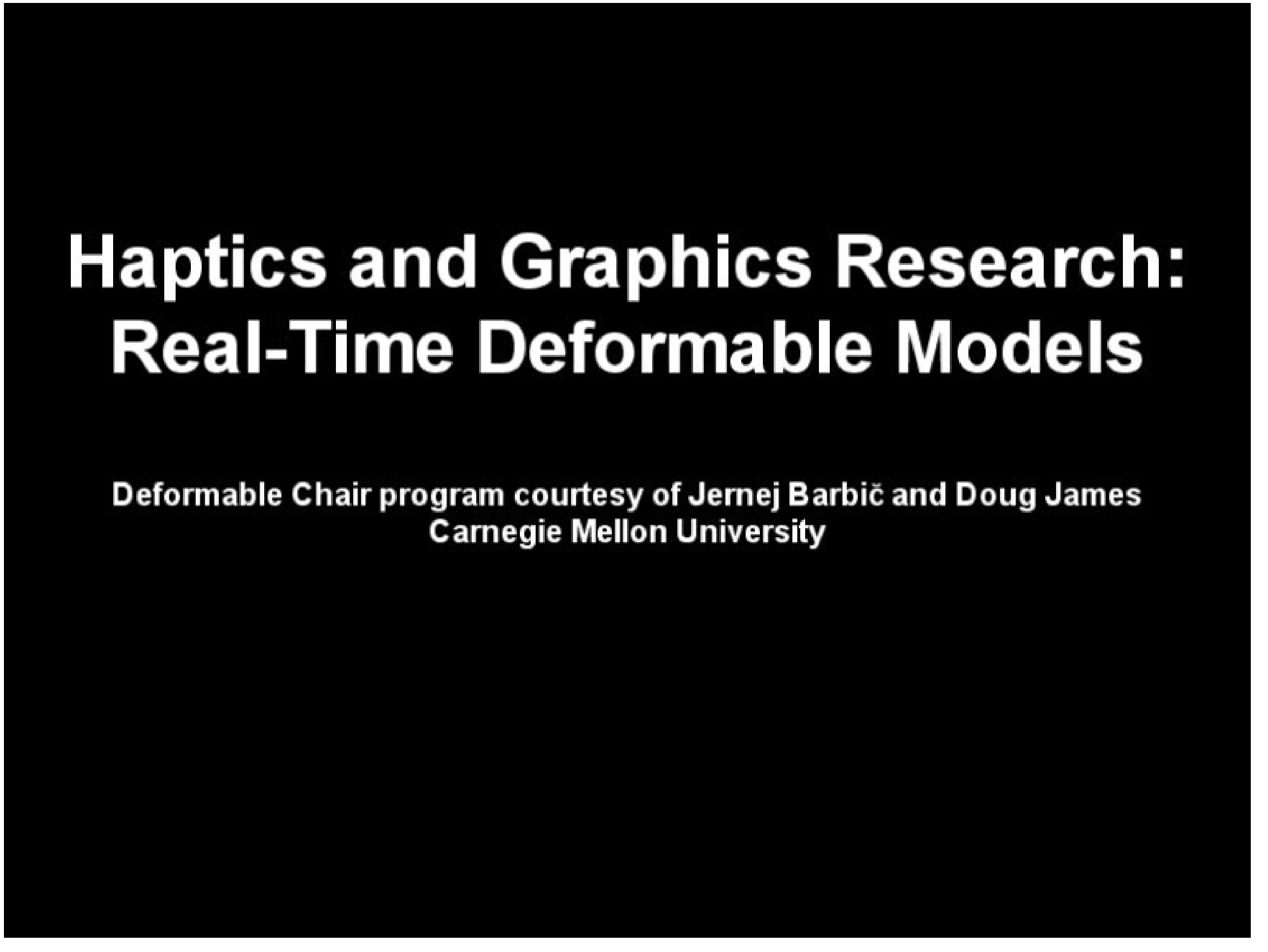
CD of deformable models

- Needed for animation of skin, cloth, tissue, and other soft bodies
- Pre-built BVHs can be used in many cases
  - Non-breakable objects
- Expected performance
  - For arbitrary deformation we aim at linear performance
  - For specific types of bounded deformations, we aim at sub-linear performance

Deformable Collision Detection

- Consider spatial subdivision and hierarchical methods for now
  - Others... graphics hardware
- Similar to nondeformable case
  - fast query times, e.g., O(lgN) for hierarchies
- But must update structure after deformation
  - various trade-offs
  - typically involves O(N) work for N simplices
- Methods can be specialized for cloth-like and strand-like objects (more on that later)

Stol, ki se deformira



Example: Morphing models

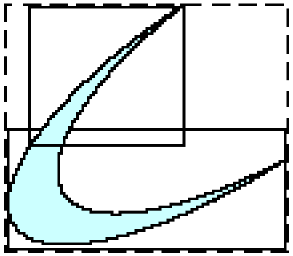
- A model is deformed by blending a set of *n* reference meshes
- We can update the bounding volumes in the bounding volume hierarchy as well as they are encountered during collision traversals
  - Suitable BVs are for example AABBs, k-DOPs, and Spheres

CD of deforming models

- Consider models deformed by arbitrary vertex repositioning
  - Hierarchies can still be pre-built in many cases
  - AABB Hierarchies have been suggested
    - Efficient update by merging AABBs

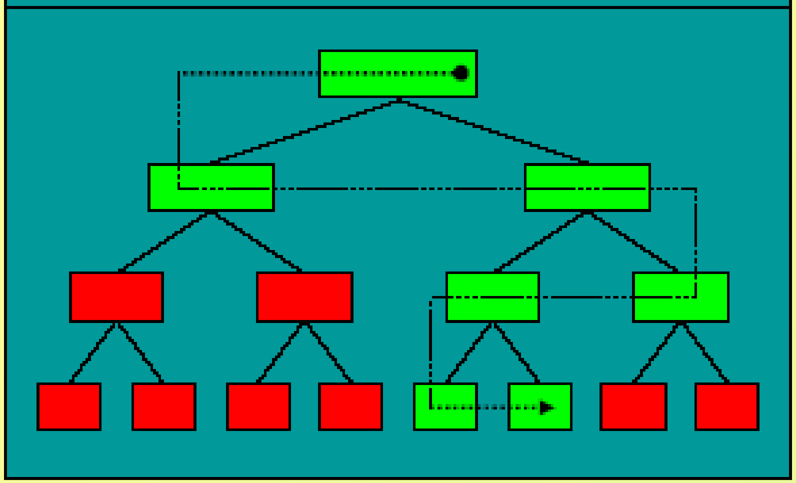
AABB refit property:

$$b(b(g_1) \cup b(g_2)) = b(g_1 \cup g_2)$$



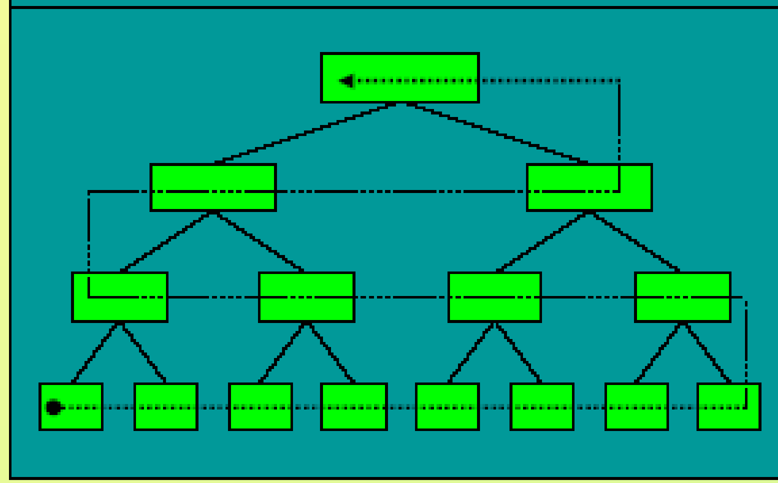
Hierarchy Update Methods

Top-down update



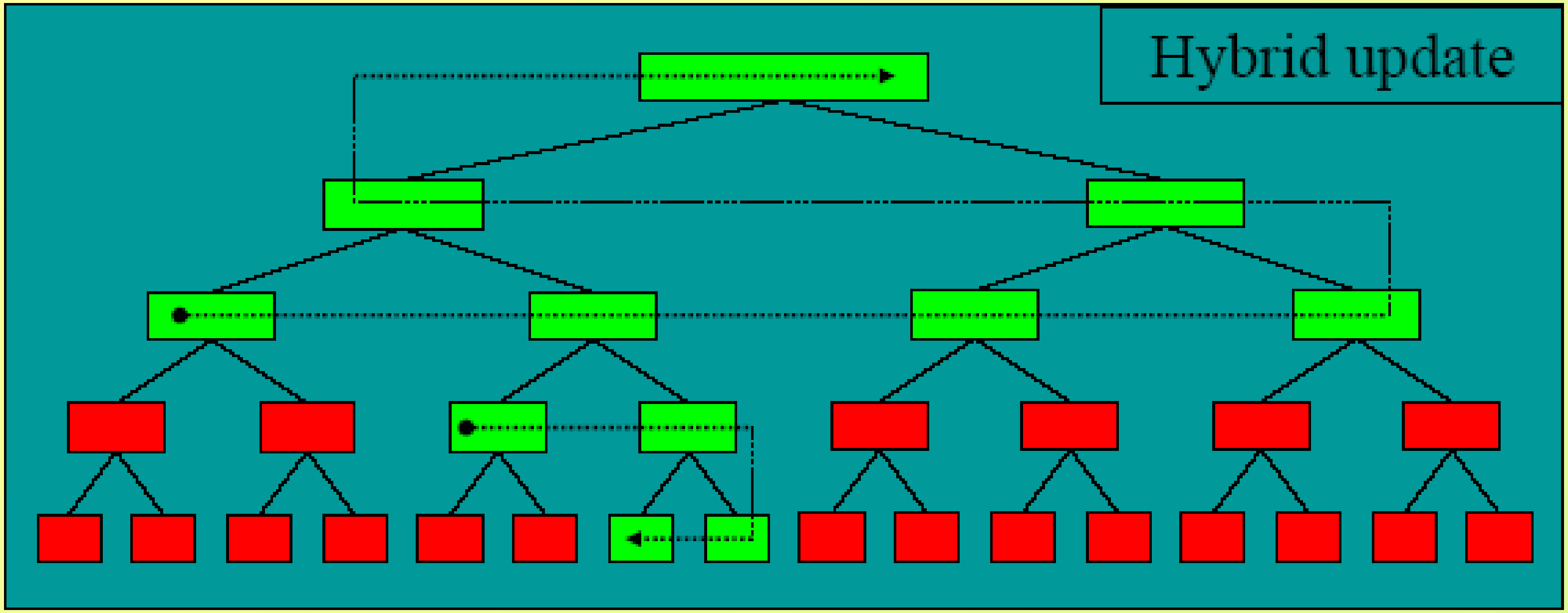
Updating as few BVs as possible top-down during collision traversals!

Bottom-up update



Updating every BV in the hierarchy bottom-up before collision traversals

Hybrid Update Method



Updating the upper levels bottom-up before collision traversals and other BVs when needed during collision traversals.

Hybrid AABB-Tree Updating

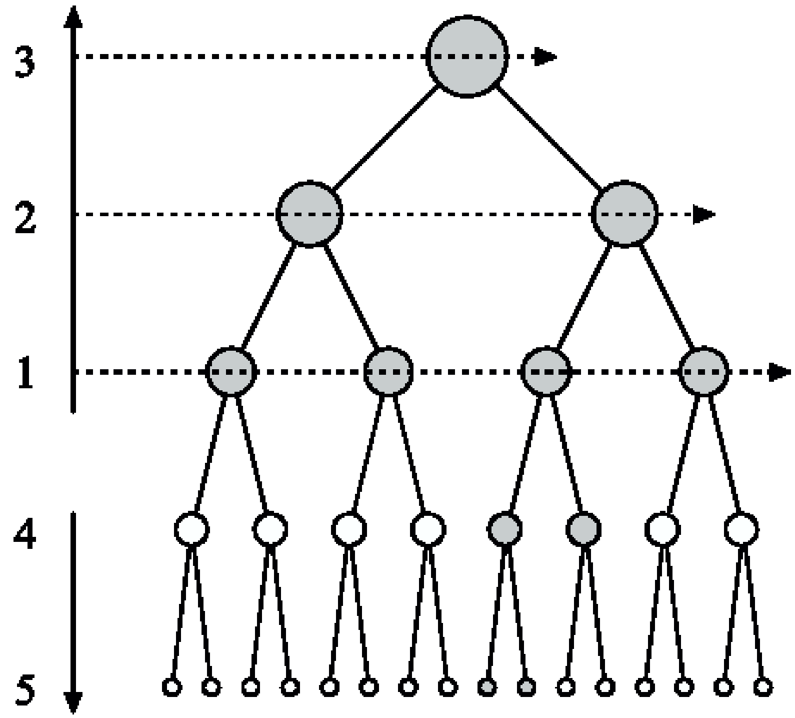


Figure 2: *Example of a hybrid tree update method, combining the bottom-up and top-down strategy.*

Local deformations

- If deformation occurs locally only it suffices if we update the affected paths in the hierarchy bottom-up
  - Update performance if m primitives (leaf nodes) are affected by the deformation
    - O(m + log n), affected leaf nodes are neighbors
    - O(m log n), affected leaf nodes are spread out
    - O(n), worst case, m == n; implement according to the bottom-up update shown on slide 20

Building hierarchies on-the-fly

- Bodies are regarded as deforming unconnected polygon soups
- In this way, we can handle body cutting naturally
- Approaches
  - Re-calculate AABB of models each frame. Then find all polygons in AABB/AABB overlap regions and pass them on to an on-the-fly octree building/pruning stage.
  - Re-insert all primitives in a pre-allocated uniform grid or octree each frame. Then we can search for close primitive pairs efficiently.

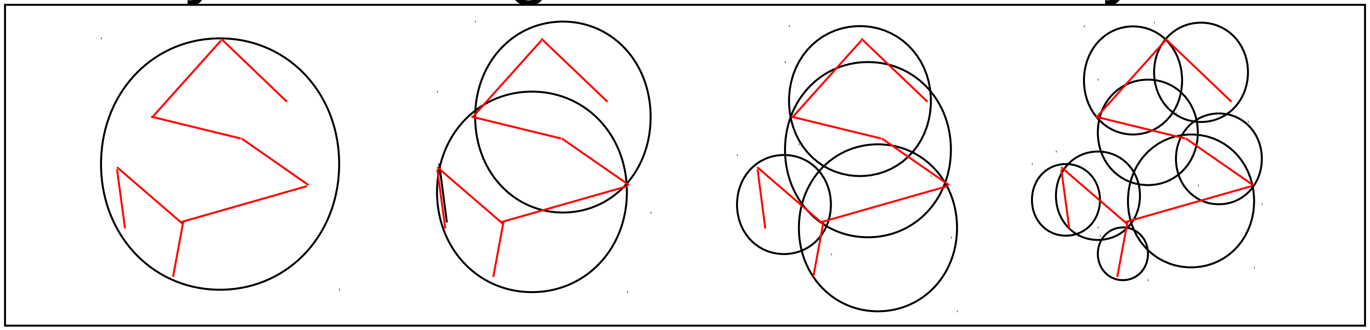
Hierarchical Representations

- Two Common Types:
  - Bounding volume hierarchies – trees of spheres, ellipses, cubes, axis-aligned bounding boxes (AABBs), oriented bounding boxes (OBBs), K-dop, SSV, etc.
  - Spatial decomposition - BSP, K-d trees, octrees, MSP tree, R-trees, grids/cells, space-time bounds, etc.
- *Many are inappropriate for deformable models*
- Do very well in “rejection tests”
- Performance may slow down when the two objects are in close proximity and can have multiple contacts

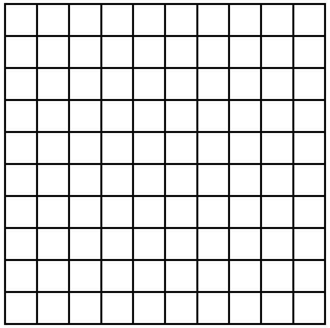
(modified from Ming Lin’s notes)

Bounding Volume Hierarchies

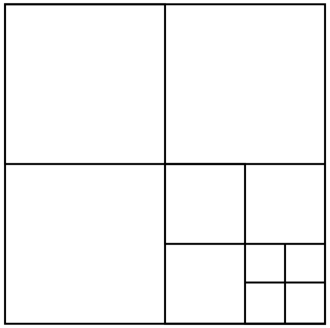
- Model Hierarchy:
  - each node has a simple volume that bounds a set of triangles
  - children contain volumes that each bound a different portion of the parent's triangles
  - The leaves of the hierarchy usually contain individual triangles
- A binary bounding volume hierarchy:



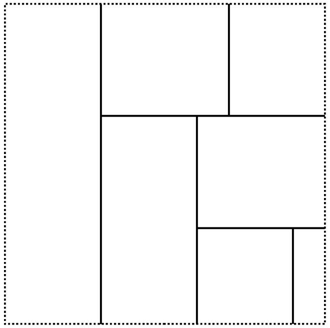
Spatial Data Structures & Subdivision



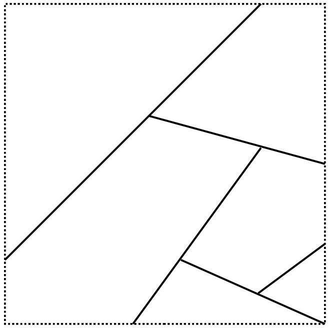
Uniform Spatial Sub



Quadtree/Octree



kd-tree



BSP-tree

Uniform Spatial Subdivision

- Simple but effective idea for real-time deformable collision detection
- Space-time trade-off

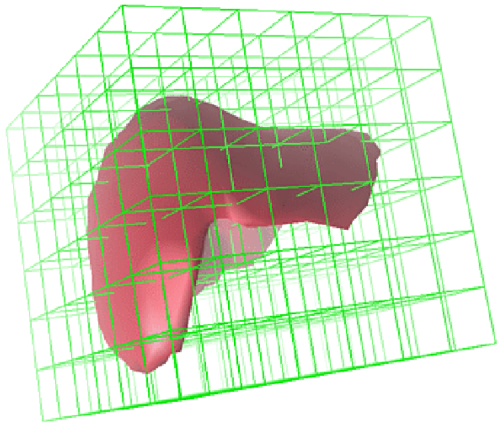


Fig. 11. The bounding box of the mesh is decomposed into a set of buckets associated to a hash table.

BVH vs. Spatial Partitioning

BVH:

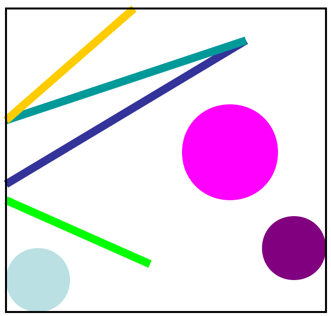
- Object centric

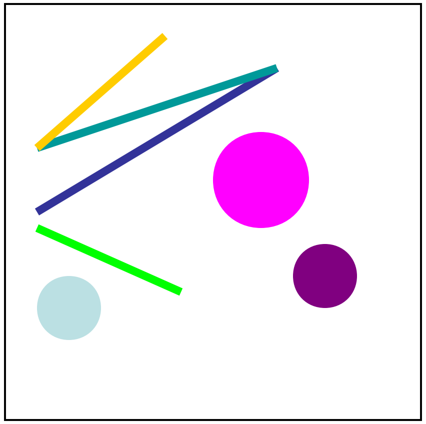
- Spatial redundancy

SP:

- Space centric

- Object redundancy





BVH vs. Spatial Partitioning

BVH:

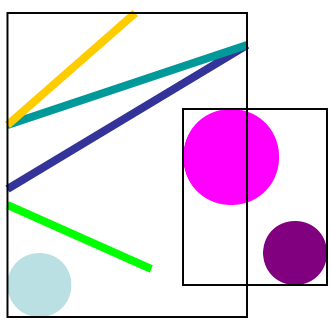
- Object centric

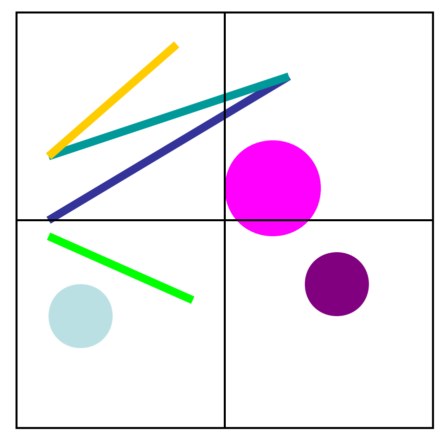
- Spatial redundancy

SP:

- Space centric

- Object redundancy





BVH vs. Spatial Partitioning

BVH:

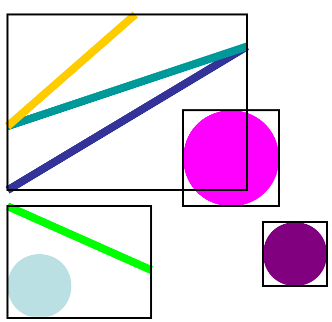
- Object centric

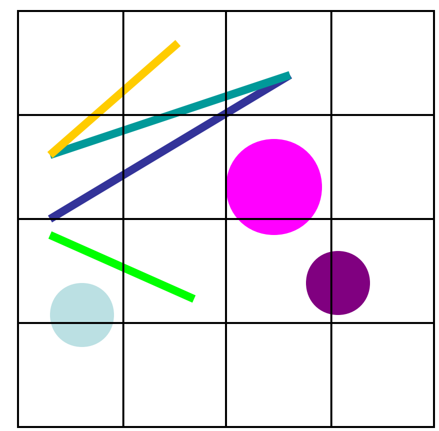
- Spatial redundancy

SP:

- Space centric

- Object redundancy





BVH vs. Spatial Partitioning

BVH:

- Object centric

- Spatial redundancy

SP:

- Space centric

- Object redundancy