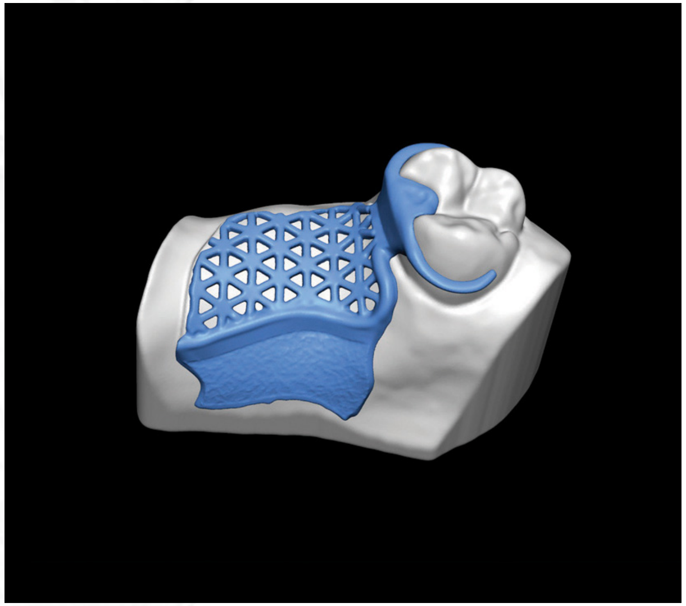
Collision Detection for Medical Applications

Gabriel Zachmann
Clausthal University, Germany
zach@tu-clausthal.de

Eurographics, Crete, 16. February 2008

Applications of Collision Detection

- Physically-based simulation of rigid anatomical parts ...
- ... and of deformable parts

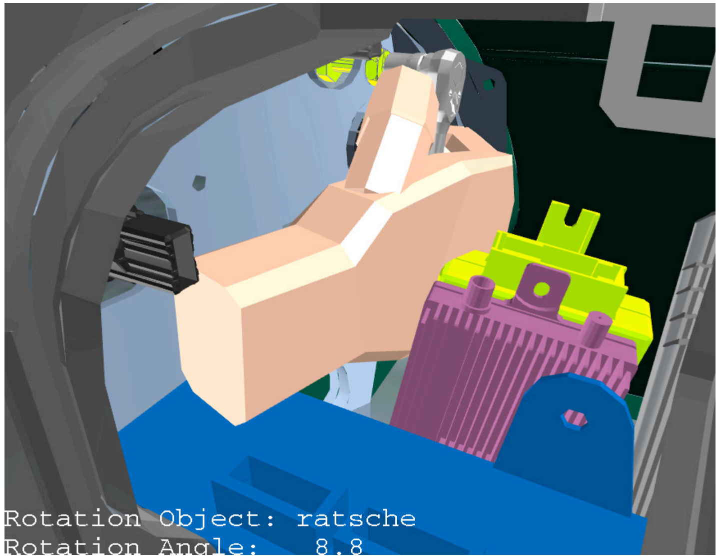


SensAble

Courtesy Raghupathi et al., INRIA

- Force feedback (e.g. in training simulators)
- And numerous apps outside of the medical domain

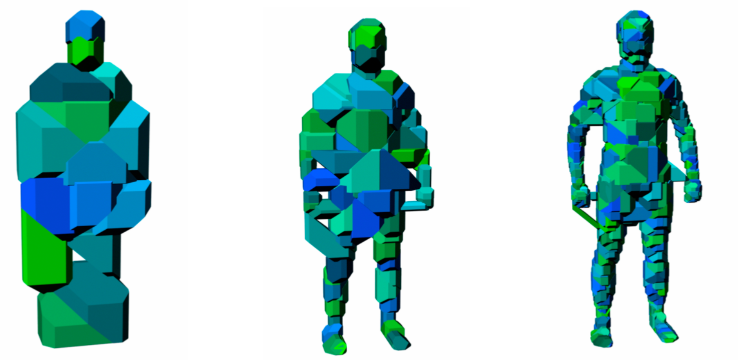
Courtesy FhG-IGD



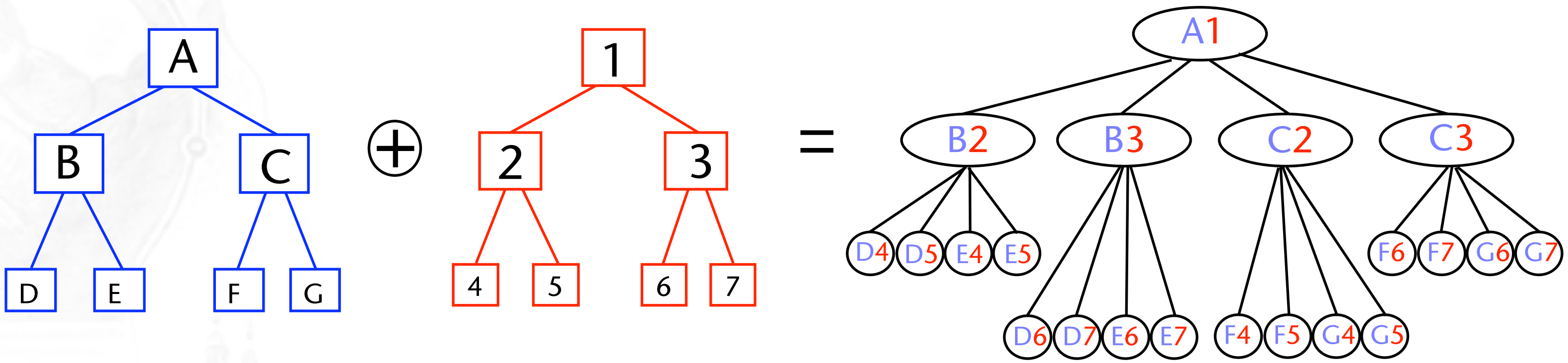
Courtesy FhG-IGD

Rigid Objects

- Standard method: bounding volume hierarchies (BVH)

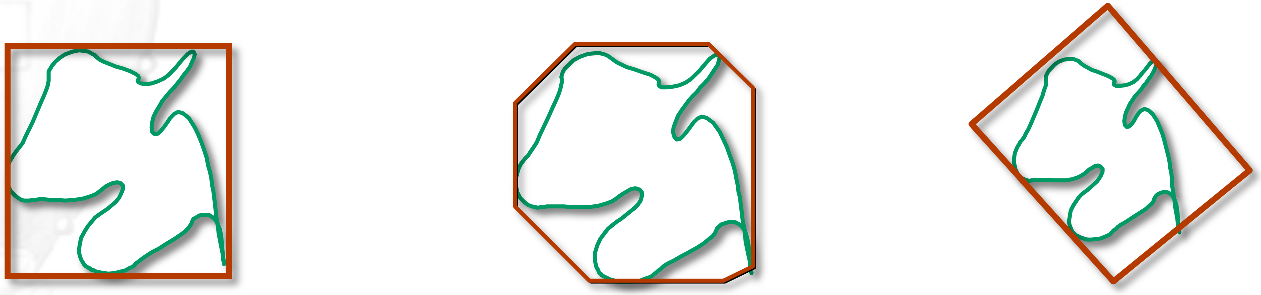


- Simultaneous traversal of two BVHs = single traversal of one conceptual BV test tree (BVTT)



Variations

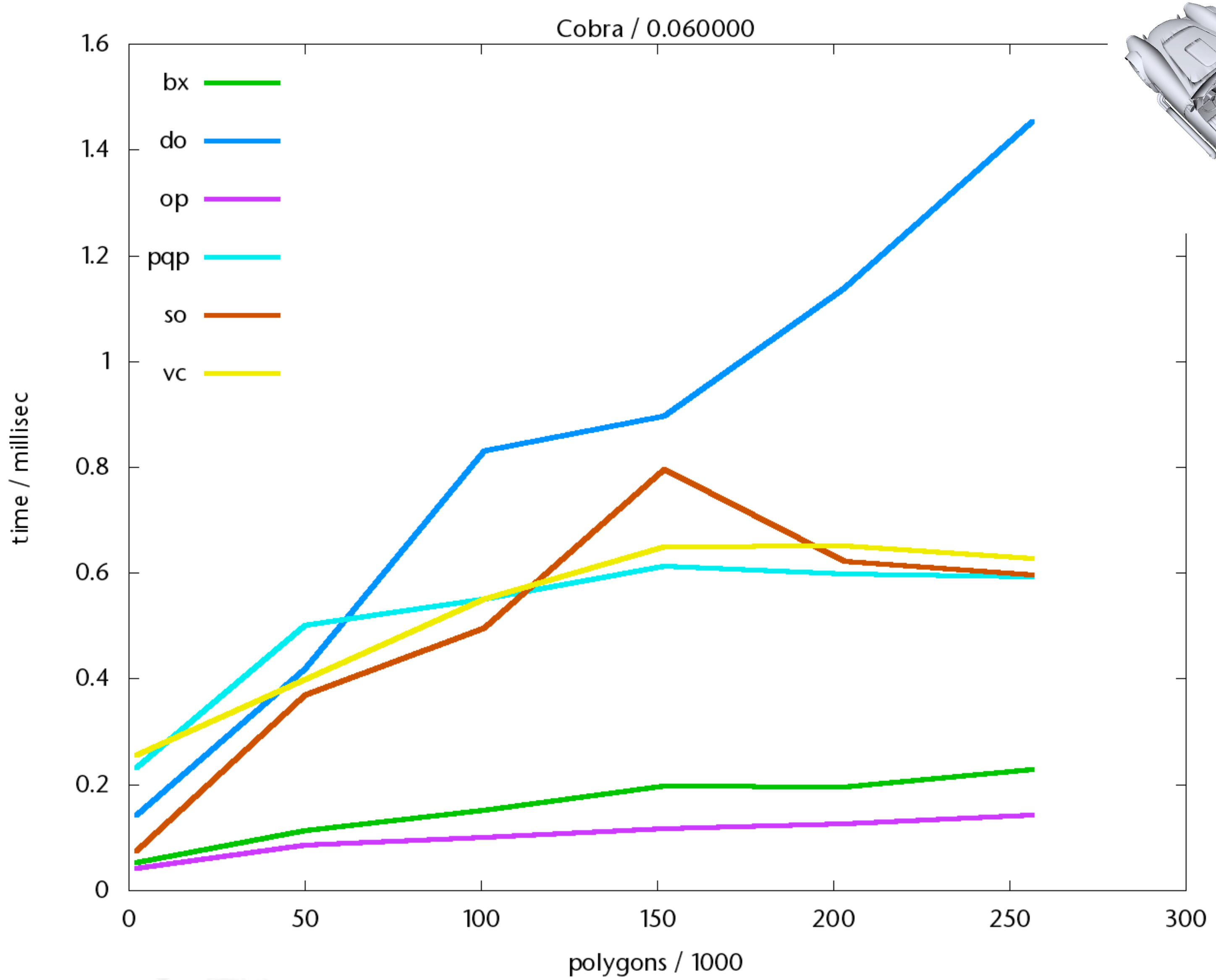
- Type of bounding volumes:

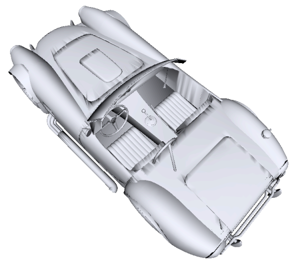


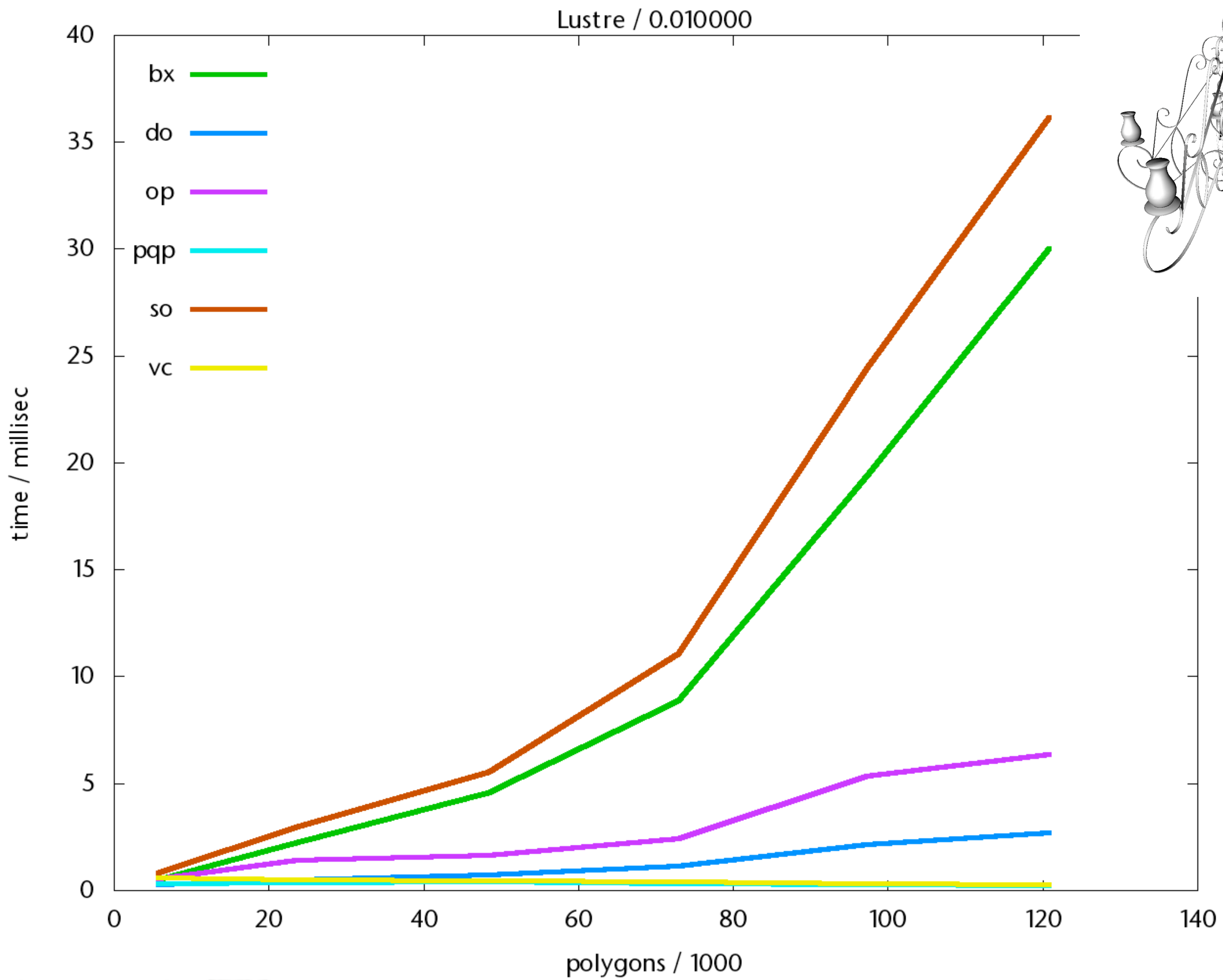
AABB (axis-aligned b.-box) (R*-trees) | k-DOP (discretely oriented polytope) | OBB (oriented bounding box)

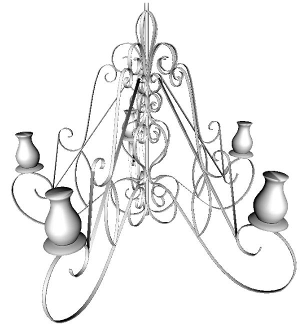
- Arity of the BVHs:
  - Most prefer 2-ary or 4-ary
  - Particularly well-suited for SSE implementations
- Kind of traversal:
  - Depth-first or breadth-first

Current Performance



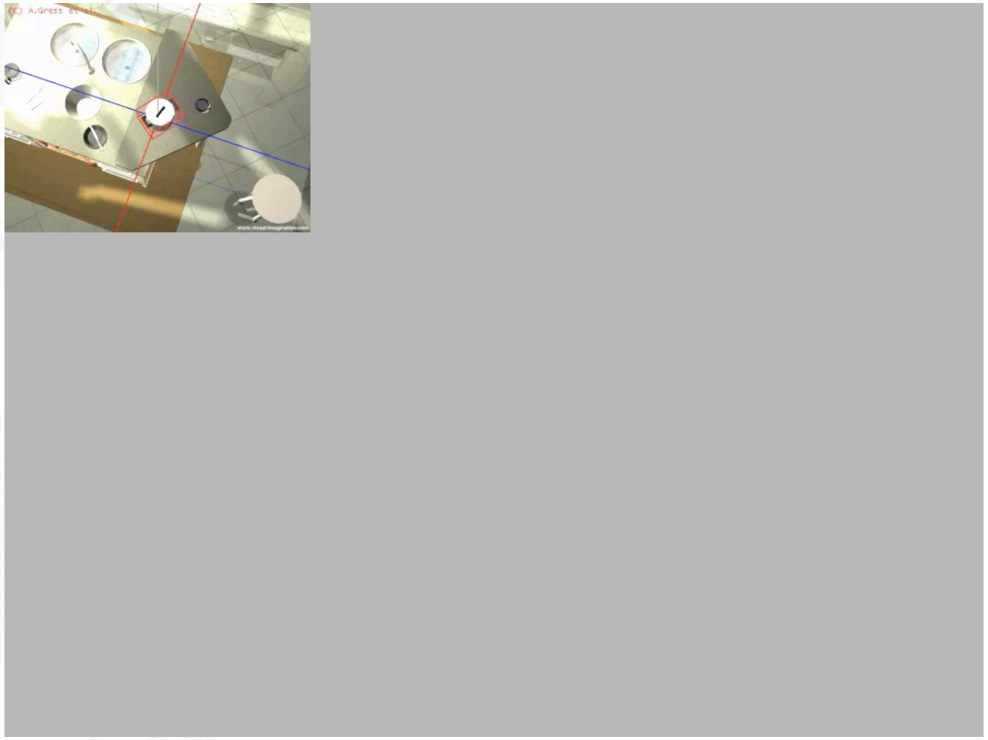


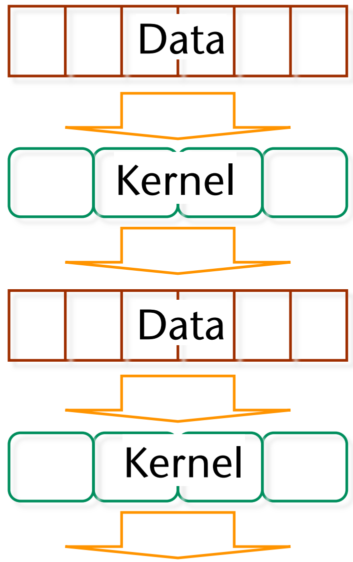

Lustre / 0.010000
time / millisec
polygons / 1000
bx
do
op
pqp
so
vc
0
5
10
15
20
25
30
35
40
0
20
40
60
80
100
120
140



Object-Space Coll. Detection on the GPU

- Implementation:
  - List of BVs = stream → texture
  - BV intersection test = kernel → fragment program

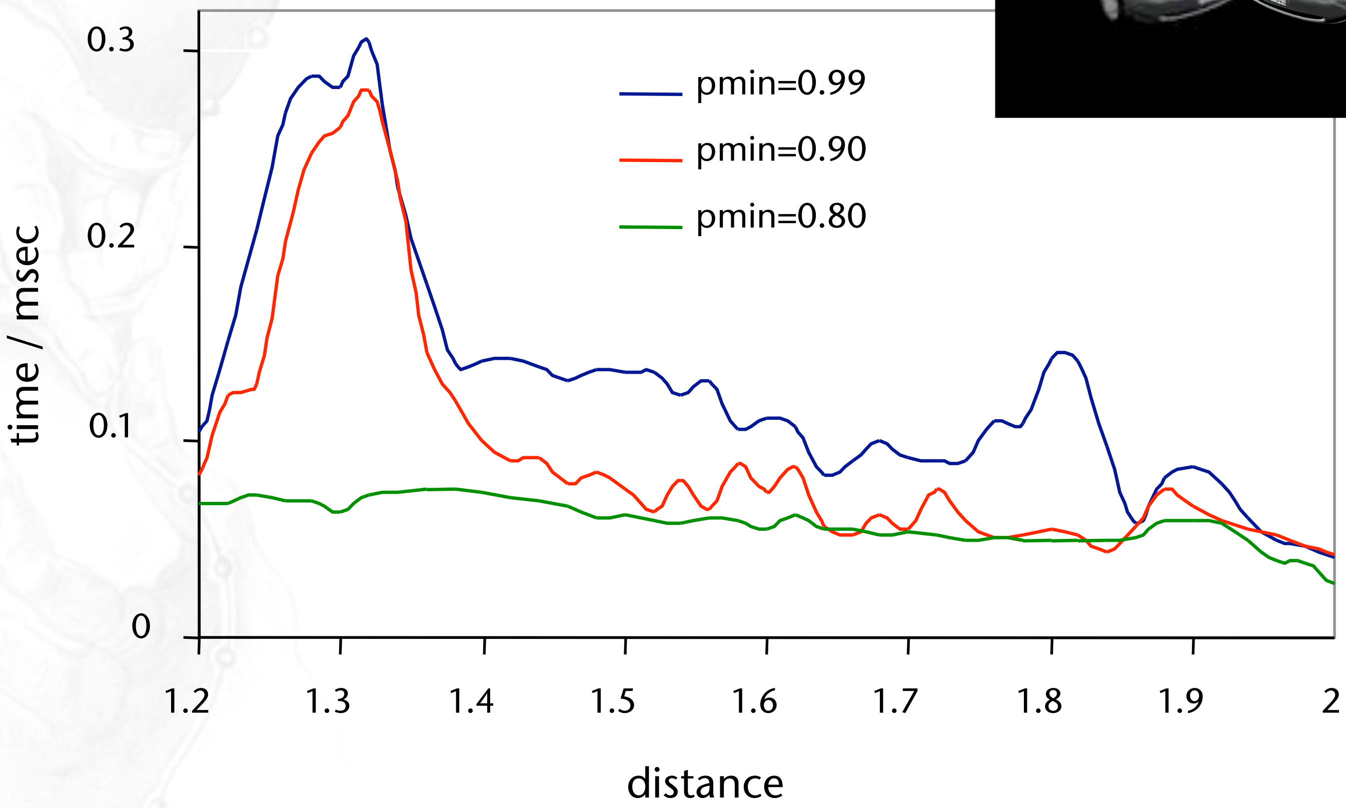




Time-Critical Collision Detection

- Goal:
  - Continuous and controlled balancing between running time and accuracy; i.e.,
  - Time-critical computation of collision detection queries
- Approach:
  - Stochastic, average-case approach
  - Idea: guide traversal of BVTT by probability (→ p-queue)
  - Modification of BVHs: store simple description → ADB trees

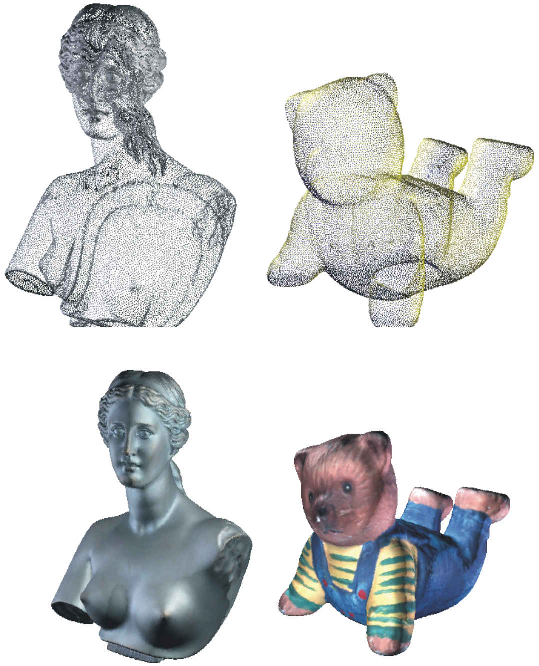
Result



0.3
0.2
0.1
0
time / msec
pmin=0.99
pmin=0.90
pmin=0.80
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
distance

Collision Detection on Point Clouds

- Motivation: renaissance of points as object representation because of 3D scanners



- Goal:
  - Fast collision detection between 2 given point clouds
  - No polygonal reconstruction

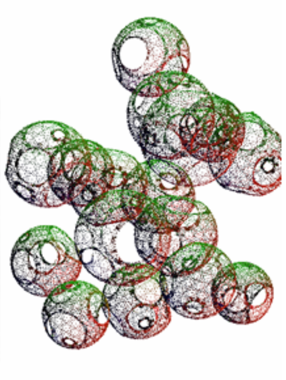
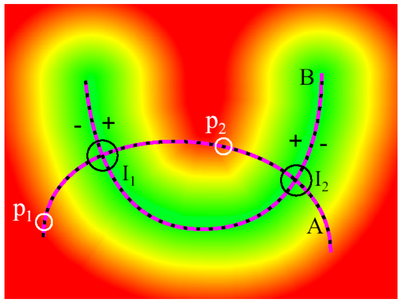
Approach

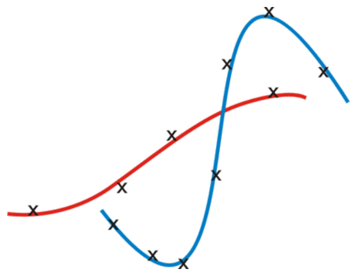
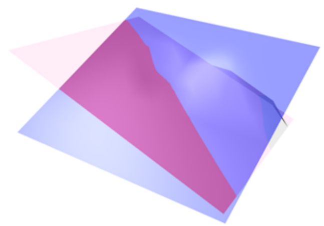
- Given two point clouds A and B, construct a stochastic sampling of

$$\mathcal{Z} = \{x \mid f_A(x) = f_B(x) = 0\}$$

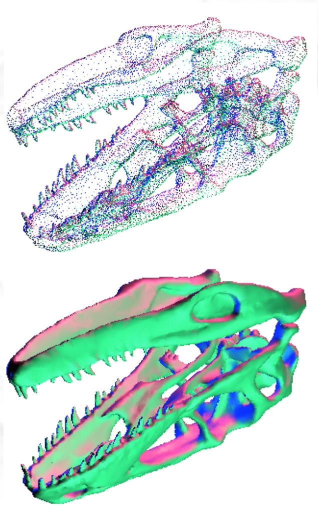
- Overall method:

A,B → $(p_i,p_j) \in A$ on different sides of B → Approx. intersection points → Refine intersection points

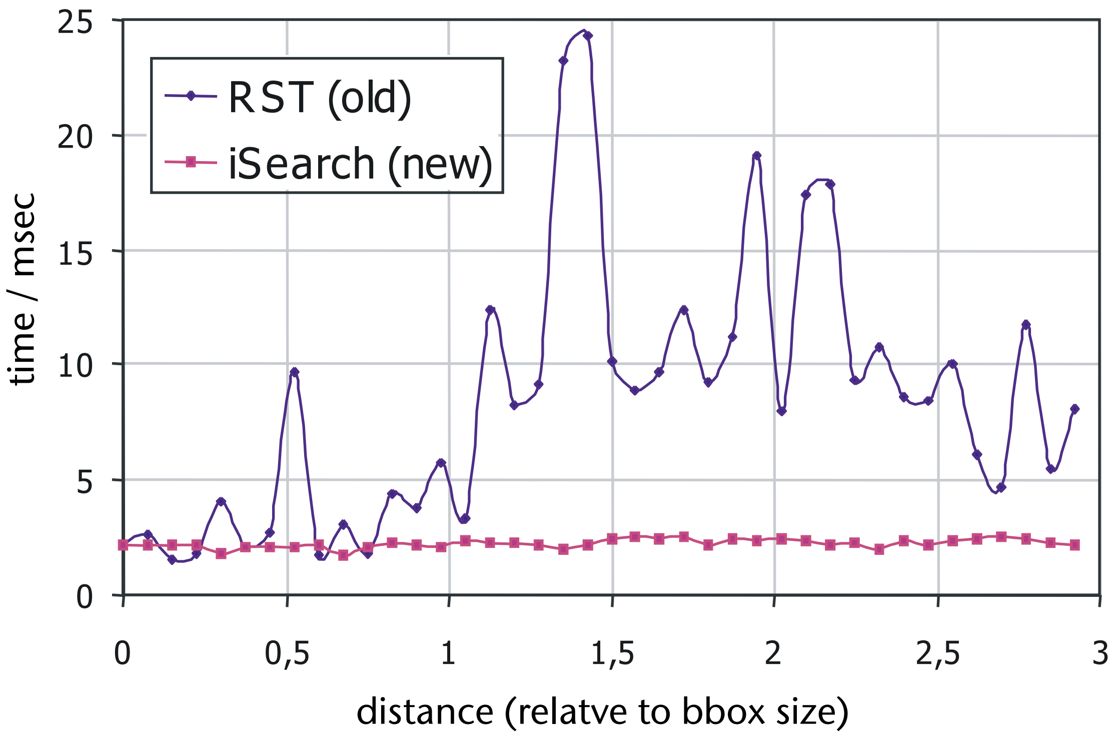



Results



28,000 points

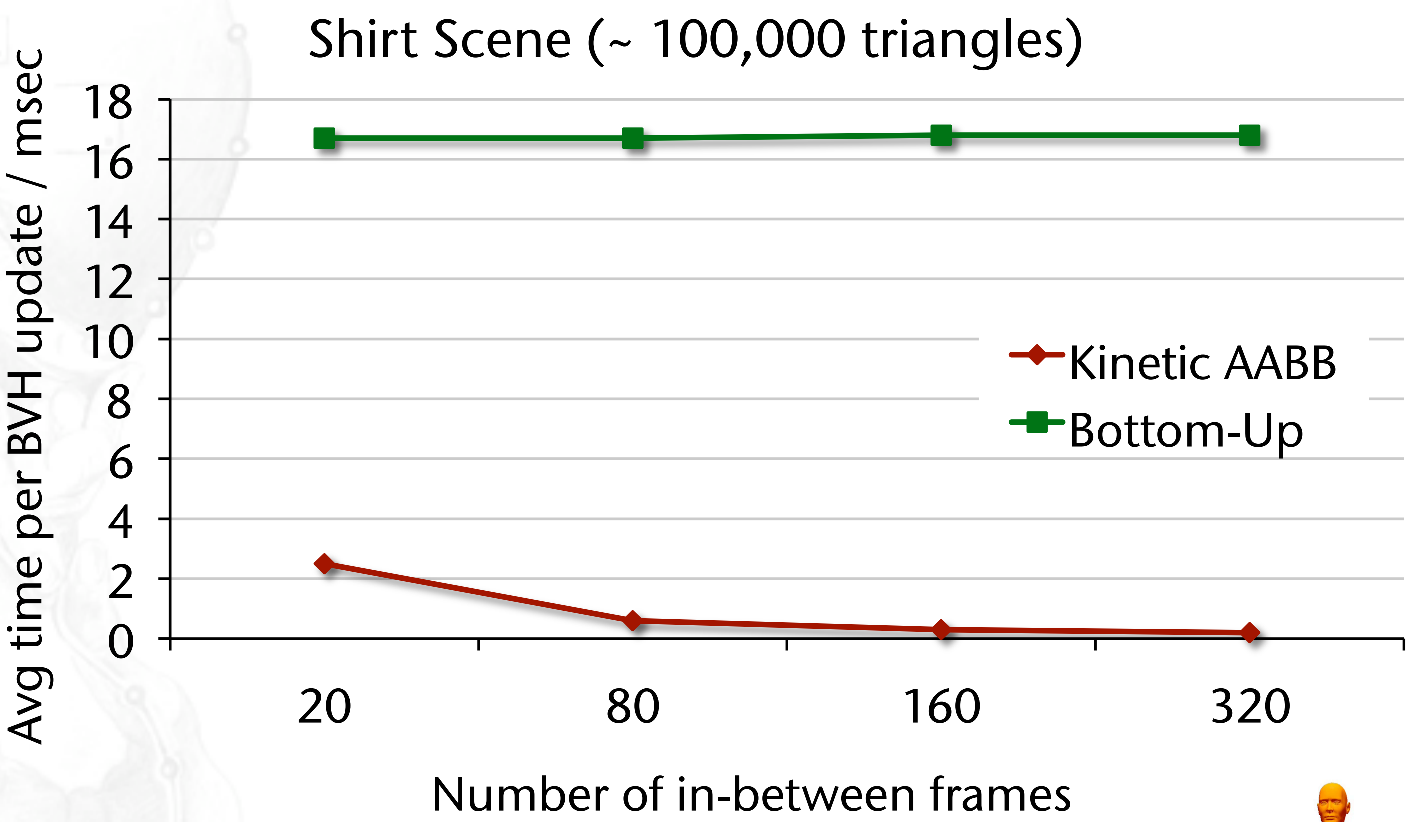


- Theoretical complexity: $O(\log \log N)$

Deformable Objects

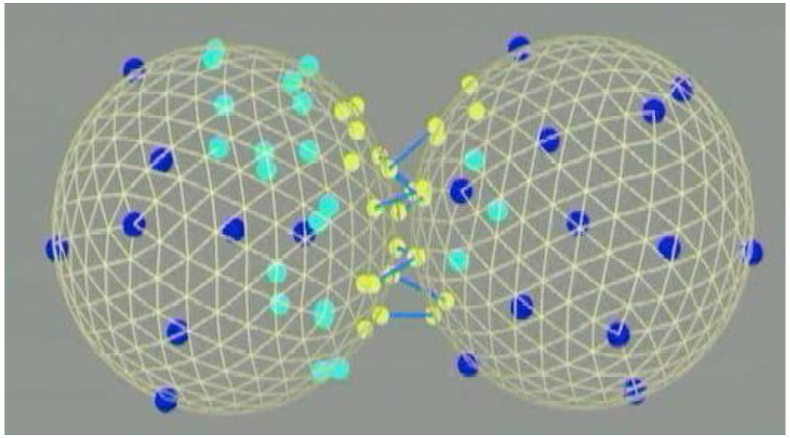
- Most objects in medical applications are (probably) deformable
- Use BVHs and update them somehow:
  - Brute-force update bottom-up
  - BV inflation with conservative estimate of motion of vertices
  - Kinetize the BVH
    - Augment data structure such that only combinatorial changes, which occur only at discrete points in time, need to be handled
    - Update time is $O(n \log n)$, independent from query frequency

Performance of Kinetic AABB



- Use "naked trees" and compute conservative BVs "as you go"
  - Only for special kinds of deformations, and with limited amounts
- Use BVHs and reconstruct every time
  - Use very simple construction algorithm
  - Reconstruct only the most deteriorated parts
- Use space partitioning scheme and update that
  - Most popular today: grid with hashing

- Don't use BVHs nor space partitioning schemes at all:
  - Use GPU, compute collision detection by "brute-force" in image space (e.g., clip edges against stencil buffer)
  - Use NURBS, tessellate and compute BVs on the fly
  - Sample mesh stochastically, update by closest features technique
  - Use point clouds with our stochastic approach



Conclusion

- Have not touched on continuous collision detection
- Collision detection for rigid bodies is fairly well researched
- For deformable bodies: still room for improvement