Stochastic Methods

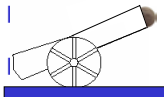
Gabriel Zachmann

Universität Bonn

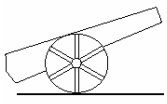
Motivation

- Absolute exactness not always necessary
- Real-time more important

→ Approximate collision detection



- Whenever only qualitative result matters, e.g.,
  Set of „plausible paths" for a cannonball.
- Games, virtual clothes prototyping, medical training, …



Overview

1. ADB-Trees [Klein & Zachmann, 2003]
2. Stochastic Closest Features Tracking [Raghupathi et al., 2004; Debunne & Guy, 2004]

ADB-Trees

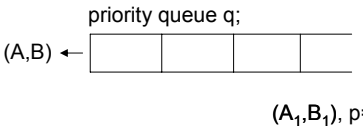
- ADB = "Average Distribution Trees"
- Average-case appraoch:
  - Estimate probability of intersection of 2 sets of polygons
- Applicable to almost any BV hierarchy
- Augment BVH by simple description of polygon distribution at inner nodes
- Probability-guided BVH traversal (p-queue)

Probability-Guided BVH Traversal

Traverse(A,B)

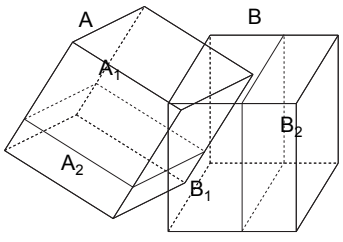
```
q.insert(A,B,1)
while q not empty
   A,B ← q.pop
   forall A_i, B_j
      p ← Pr[ collision in A_i, B_j ]
      if p ≥ pmin
         return "collision"
      if p ≥ 0
         q.insert(A_i, B_j, p)
 return "no collision"
```


Probability-Guided BVH Traversal



(A,B)

(A_1,B_1), p=0,9
(A_1,B_2), p=0
(A_2,B_1), p=0,5
(A_2,B_2), p=0

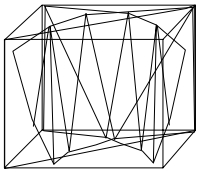


Traverse(A,B)

p-queue q

```
q.insert(A,B,1)
while q not empty
   A,B ← q.pop
   forall A_i, B_j
      p ← Pr[ collision in A_i, B_j ]
      if p ≥ p_min
         return "collision"
      if p ≥ 0
         q.insert(A_i, B_j, p)
 return "no collision"
```

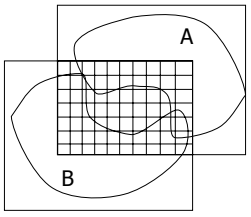
Well-filled Cells and Collision Cells



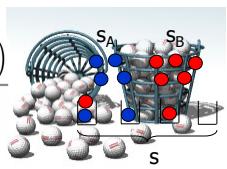
"well-filled" cell

Computing the Probability of Intersection

1. Partition A ∩ B by grid with s cells
2. Determine number of "well-filled" cells from BV A: s_A
3. Dito for B: s_B
4. Compute probability that x cells are well-filled from A *and* from B:



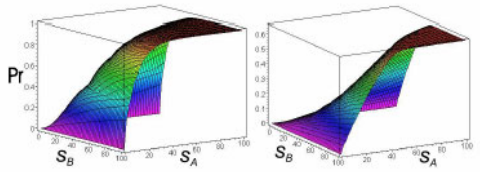
$$Pr[c(A \cap B) \geq x] = 1 - \sum_{t=0}^{x-1} \frac{\binom{s_B}{t}\binom{s-s_B}{s_A-t}}{\binom{s}{s_A}}$$



- Take curvature within cell into account:

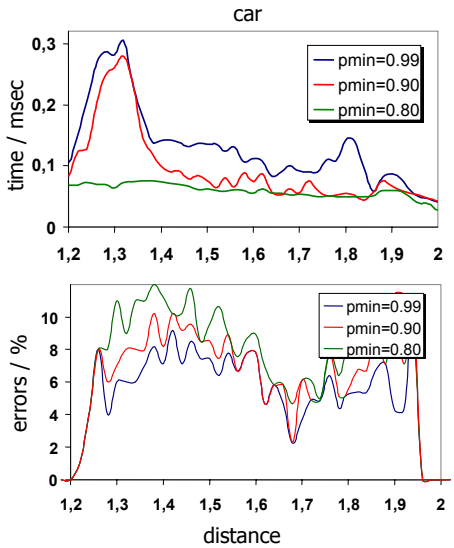
$$\max_{x \le \min\{s_A, s_B\}} \{Pr[c(A \cap B) \ge x] \cdot (1 - (1 - LB(A \cap B))^x)\}$$

- Preprocessing
- Estimate parameters
- Lookup-tables for probability functions:



Result

- Time vs. error:



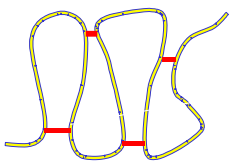
Stochastic Closest Features Tracking

- Based on Lin-Canny (only for convex objects)
  - Steepest descent for single pair of features
  - Accelerated by generalized Voronoi diagram
  - Temporal coherence
- Extension to non-convex, deformable objects:
  - Non-convex → multiple pairs of (locally) closest features
  - Deformable → feature pairs come and go
  - Voronoi diagram not really feasible
- Idea
  - Stochastically create pairs of features
  - Converge them to locally closest features

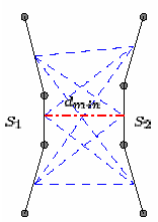
Details

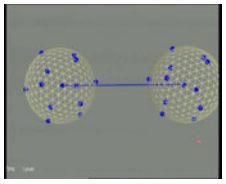
- Algorithm:

  Do animation step
  Add random pairs to list of "active feature pairs"
  For each feature-pair:
  Update features by local search
  Remove "unwanted" pairs
  If collision:
  apply response to local collision area



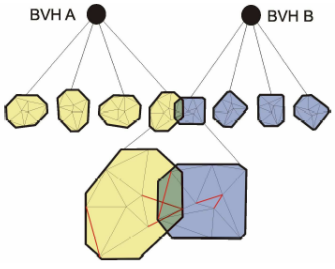
- Updating of feature pair:
  - No Voronoi diagram
  - Compute pairwise distance of all neighbor pairs
- Removal of feature pairs:
  - Distance too large (not likely closest feature)
  - Both features of two pairs too close (redundant)
- Creation of feature pairs:
  - Importance-driven (e.g., velocity-based)
  - Supported naturally by multires model

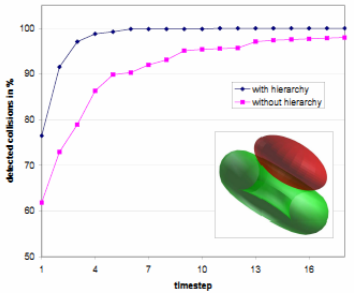




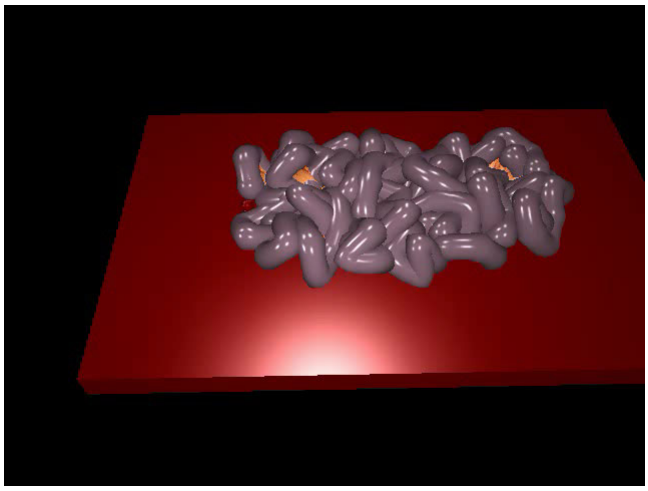
Acceleration by BV Hierarchy

- Use incomplete BVH to find "interesting" regions
- Stochastically sample those regions

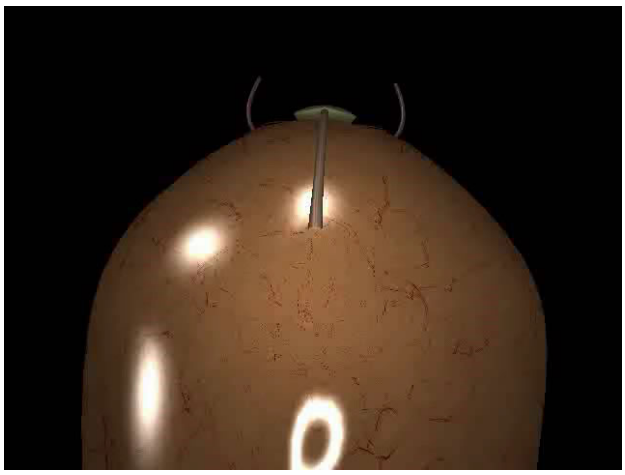




Example Application



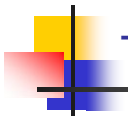
INRIA



INRIA

Conclusions

- Stochastic methods are not always error-free
- Good for plausible & fast simulations
- Interesting alternative to BVHs in specific cases
- Naturally yield time-critical collision detection
- Future work:
  - Continuous stochastic methods
  - Precise error bounds



Thank you for your attention …

… we continue with …

Arnulph Fuhrmann

presenting

Distance Fields