**Motivation**

- Collision detection is a fundamental operation in:
  - Games
  - Haptic rendering
  - Interaction in VR

- Important: responsive collision detection at all times.
- Time-critical: if algorithm cannot reach an exact answer during that time, return “best effort” result.
- “Best effort” result should be as good as possible.

**Related Work**

- Approximating Polyhedra with Spheres for Time-Critical Collision Detection [Hubbard, 1996]
- Graceful Degradation of Collision Handling in Physically Based Animation [Dingliana, O’Sullivan, 2000]
- No theoretical foundation concerning the error.
- No collision probability if time budget exhausted.

We concentrate on collision detection between rigid bodies.

**Our Contribution**

- New general framework for time-critical, hierarchical collision detection with error estimation.
- New probability-guided traversal of BV-hierarchies.
- Very small footprint of data structure (no polygons are needed).
Collision Detection using ADB-Trees

ADB-tree: hierarchical, object partitioning tree where BVs are augmented by density attributes.

Given two ADB-trees for two objects:
- traverse hierarchies simultaneously
- for each pair A,B: estimate Pr[A,B] that polygons in A and B collide
- give priority to pair with highest probability
- stop traversal, if Pr[A,B] > p_{min}

No polygon intersection tests are needed for "estimate Pr[A,B]."

Average-Distribution (ADB) Trees

ADB-tree:
- Construct any hierarchical, object partitioning tree (e.g., AABB tree).
- For each BV A in the hierarchy:
  - partition A into grid
  - count number of cells that contain "large" polygon area \( \rightarrow pc(A) \)
  - store only pc(A) with BV A
  - delete all polygons from tree

Given BVs A and B of different ADB-trees:
- estimate probability Pr[A,B] of collision based on pc(A), pc(B) and Vol(A ∩ B)

Main Idea

ADB-Trees: Assumption

\( s_A \) : number of pc of A lying in Vol(A ∩ B)

Can we quickly estimate \( s_A \) and \( s_B \) by pc(A) and pc(B)?

→ Yes, under the following assumption!

Assumption: each cell \( A \) has same probability of being a pc.

If Vol(A) = Vol(B) \( \Rightarrow \) estimate \( s_A \) and \( s_B \) proportional to Vol(A ∩ B).
If Vol(A) ≠ Vol(B) \( \Rightarrow \) please refer to our paper.

→ Average-case approach: it works well in the average case, where pc are evenly distributed.
Our Average-Case Approach

Compute probability that \( \geq x \) cells exist, which are pc with respect to \( A \) and \( B \) (called collision cells (cc)).

Probability Computations

Given \( s, s_A, s_B \), compute \( \Pr[\text{cc}(A \cap B) \geq x] \) by balls into bins model.

What is the probability that at least \( x \) bins get a red and a blue ball?

Estimating Probability of Collision

What can we do with \( \Pr[\text{cc}(A \cap B) \geq x] \)?

- This probability depends on the definition of a pc ("large" polygon area).
- cc contain "large" polygon area from both objects, but the polygons do not have to collide.

\( \rightarrow \) LB(\( A \cap B \)): lower bound for the probability that a collision really takes place in a cc (detail \( \rightarrow \) VRST’03 paper).

Estimate \( \Pr[A,B]: \)

\[
\max_{a \in A} \left( \frac{\text{Pr}[a \cap (A \cap B)]}{s_A - s_B} \right) \approx \left[ 1 - (1 - s_A s_B) s_B \right]^{1 / s_B}
\]

Results
Possible Collision Cell Distribution

- Given all BVs of an ADB tree + information which cell is a pc.
- Each node is partitioned into c cells.
- Identify corresponding cells of all nodes by $x$.
- Count over all nodes how often cells with the identifier $x$ are pc $\rightarrow n(x)$.

$n(x)$

Video

ADB-Trees: Controlling the Error of Time-Critical Collision Detection

Jan Klein, Gabriel Zachmann

Time and Quality vs Complexity

Error rate decreases if $p_{\min}$ increases.

Time vs Quality

Runtime increases if $p_{\min}$ increases.

Remember: if $p_{\min}$ is reached, our algorithm claims it has found a collision.

$(P(A \cup B) > x) \Rightarrow \text{stop traversal}$
Conclusion & Future Work

Conclusion

• General framework for time-critical hierarchical collision detection.
• It uses probability estimations to balance speed and quality.
• Results show speedup of about a factor 3 to 6 with only about 4% errors.

Future Work

• Non-polygonal geometry
• Broad phase of collision detection
• Deformable objects
• Other BV hierarchies (DOP tree, restricted boxtree)

Thank you!

Jan Klein
Heinz Nixdorf Institute and
Institute of Computer Science
University of Paderborn, Germany
E-Mail: janklein@uni-paderborn.de

Dr. Gabriel Zachmann
Dept. of Computer Graphics and
Virtual Reality
University of Bonn, Germany
E-Mail: zach@cs.uni-bonn.de