Time-Critical Collision Detection Using an Average-Case Approach

Jan Klein
University of Paderborn
Germany

Gabriel Zachmann
University of Bonn
Germany

Osaka, Japan ñ October 2003

Motivation ñ Collision Detection

Fundamental operation:

ï Virtual prototyping
ï Haptic rendering
ï Interaction in VR
ï Rigid body simulation

Controlling the Errors

ï Factors that contribute to imprecision in a model and simulation:
- ï Numerical error in computation
- ï Approximations in the abstraction
- ï Inaccuracy in the data
- ï Inaccuracy in the computation of collision (point of collision, normals, Ö)

ï Barzel et al., 1996: physically plausible simulation.

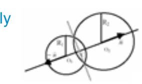
Set of ìplausible pathsî for a cannonball.

Our goal: Decrease the accuracy of collision detection in a controlled way to speed up the simulation.

Related Work

ï Approximating Polyhedra with Spheres for Time-Critical Collision Detection [Hubbard, 1996]

ï Graceful Degradation of Collision Handling in Physically Based Animation [Dingliana, OíSullivan, 2000]



We concentrate on collision detection between rigid bodies.

Our Average-Case Approach

Main Idea

ï Our approach is applicable to most BV hierarchies.

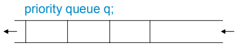
ï We estimate the probability of intersection for a set of polygons.

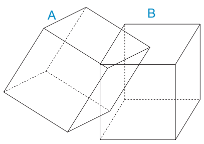
ï The estimation is also done for inner nodes.

ï We do not need any polygons during runtime, but only parameters describing the polygon distribution.

1. The application can control the runtime by specifying the desired quality (accuracy).
2. The probabilities can guide the BV traversal into those parts that allow for faster convergence.

Basic Traversal Algorithm



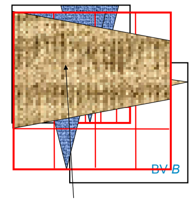



```
traverse(A, B)
{
  priorityQueue q;
  q.insert(A, B, 1);
  while q is not empty do
  {
    A, B := q.top;
    q.pop;
    for all children A[i] and B[j] do
      p := estimateProbability(A[i], B[j])
      if p > p_min return "collision found"
      if p > 0 then q.insert(A[i], B[j], p);
  }
  return "no collision";
}
```


Estimating Probabilities

- Conceptually, partition A ∩ B into a grid.
- Probability of collision in cell is high, if it contains "enough" polygons.
- Estimate probability of collision by
  - Pr[c(A ∩ B) ≥ x]: probability that at least x collision cells exist in A ∩ B.
  - LB(A ∩ B): lower bound for the probability that a collision really takes place in a collision cell. (for detail → paper)
- estimateProbability(A,B):

$$\max_{valid\ x} \{Pr[c(A\cap B) \geq x] \cdot (1-(1-LB(A\cap B))^x)\}$$



collision cell contains "enough" polygon area from both objects

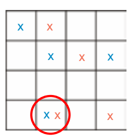
Necessary Parameters

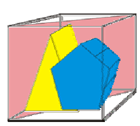
What parameters are necessary to compute Pr[c(A ∩ B) ≥ x]?

s = # cells contained in A ∩ B

s_A = # possible collision cells from A contained in A ∩ B

s_B = # possible collision cells from B contained in A ∩ B



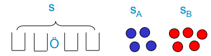


possible collision cell

Average-case approach: probability of being a possible collision cell is evenly distributed among all cells!

Probability Computations

Given *s*, s_A, s_B, compute Pr[c(A ∩ B) ≥ x] by *balls into bins model*.



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

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

s_A and s_B can only be *estimated*

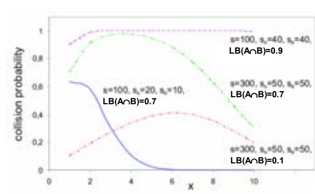
Average-Distribution Trees (ADB Trees)

Partitioning of A ∩ B is too expensive during runtime!

- ADB tree:
  - Conceptually, partition each BV of the hierarchy into a fixed number of cells.
  - Store # possible collision cells with nodes
- Use precomputed possible collision cells to determine s_A and s_B during runtime.
  - BVs A and B are equally sized.
  - BVs A and B are differently sized.
  - → please look at our paper

Lookup Tables

- estimateProbability(A,B): $\max_{x \leq \min\{s_A, s_B\}} \{Pr[c(A\cap B) \geq x]\cdot(1-(1-LB(A\cap B))^x)\}$
- The equation assumes the maximum at a small x → bound x
- Store probabilities in lookup table.
- Number of possible collision cells is bounded by preprocessing.



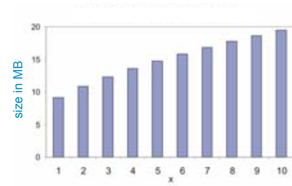
Results

Preprocessing

HEINZ NIXDORF INSTITUTE
University of Paderborn
Theoretical Computer Science
Jan Klein

ï ADB-tree based on AABBs.

ï 8≥cells per node.



Benchmark Scenario

HEINZ NIXDORF INSTITUTE
University of Paderborn
Theoretical Computer Science
Jan Klein

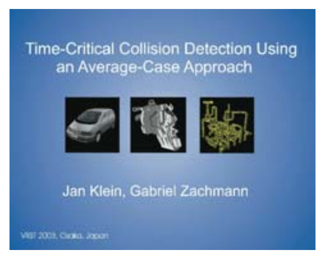
ï Objects are scaled uniformly → cube of size 2≥

ï Average collision detection time for a complete revolution (5000 steps).

ï Pentium-IV, 2GHz, 1 GB main memory.

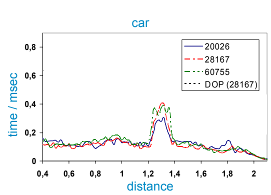
Video

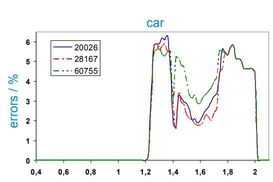
HEINZ NIXDORF INSTITUTE
University of Paderborn
Theoretical Computer Science
Jan Klein



Time and Quality vs Complexity

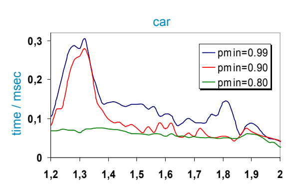
HEINZ NIXDORF INSTITUTE
University of Paderborn
Theoretical Computer Science
Jan Klein

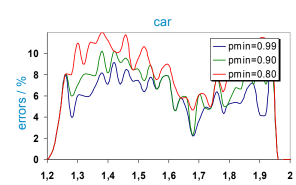




Time vs Quality

HEINZ NIXDORF INSTITUTE
University of Paderborn
Theoretical Computer Science
Jan Klein





Runtime increases if pmin increases.

Error rate decreases if pmin increases.

Conclusion & Future Work

HEINZ NIXDORF INSTITUTE
University of Paderborn
Theoretical Computer Science
Jan Klein

Conclusion

- Average-case approach.
- 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!

Heinz Nixdorf Institute
University of Paderborn
Institute of Computer Science

Germany

E-Mail: janklein@uni-paderborn.de