Efficient collision detection for VR and simulation software

Gabriel Zachmann

Bonn University

zach@cs.uni-bonn.de

www.gabrielzachmann.org

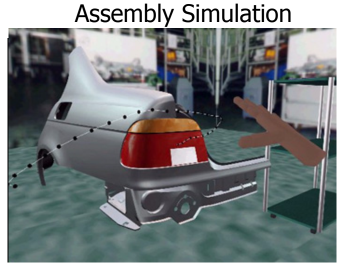
cg.cs.uni-bonn.de

Virtual Prototyping

Ergonomics Investigation

[illegible]

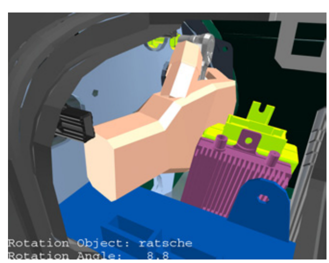
Assembly Simulation



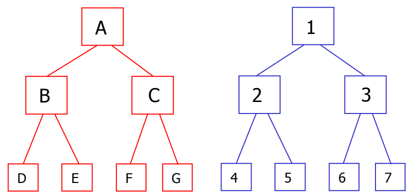
IGD / BMW

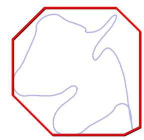
Virtual Cities

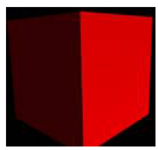
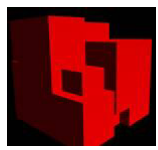
Bonn & Universität Lissabon

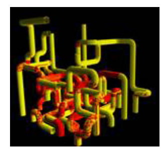


Hierarchichal Collision Detection



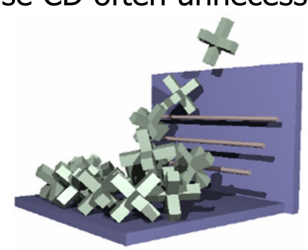
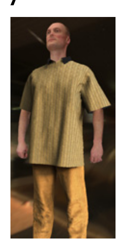

k-DOPs



Time-Critical Collision Detection

- Large time variations in conventional CD
- Observation: precise CD often unnecessary

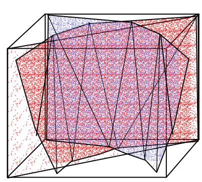



- Goal: achieve continuous and controlled balancing between run-time and precision

Basic Idea

- Average-case approach:
  - Estimate probability of intersection of 2 *sets* of polygons
- Guide traversal by probabilities (p-queue)
- Can be applied to almost any BV hierarchy

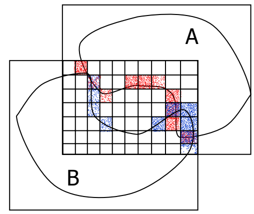
Thought Experiment ("Gedankenexperiment")



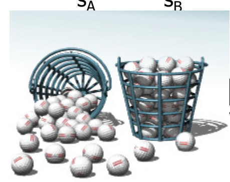
Estimation of the Probability

1. Partition A ∩ B by s cells
2. Compute s_A = #cells well-filled from A
3. Dito for s_B = #cells from B
4. c(A ∩ B) = # collision cells
5. Compute probability that c ≥ x:

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



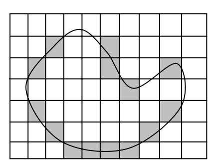
"Balls into Bins"



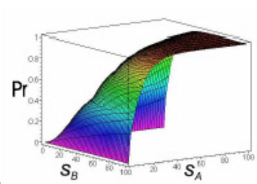
Efficient Estimate

- Preprocessing:
  - Partition each BV of BVH by grid
  - Count number s_A of well-filled cells
  - Store with each node of BVH
- At runtime estimate s_A and s_B:

$$s'_A = s_A \frac{\text{Vol}(A)}{\text{Vol}(A \cap B)}$$



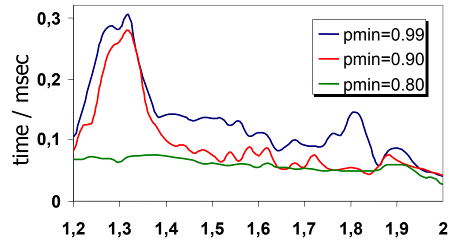
- Precompute LUT for function *Pr* for all possible input values

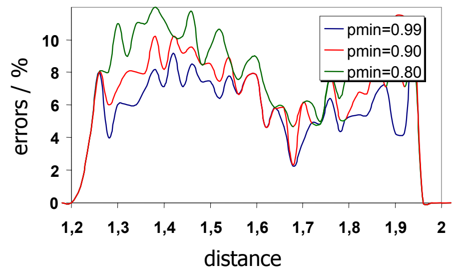


Results

- Time vs. error:

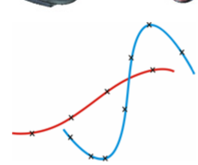
60,000 polygons each





Point clouds

- Modern acquisition techniques (laser scanners) lead to modern object representation
- Efficient rendering techniques (Splatting & Ray-Tracing)
- Basically no interaction
- Goal:
  - Fast collision detection between 2 given point clouds
  - No polygonal reconstruction

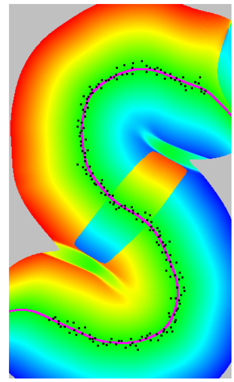


Definition of the surface

- Implicit:

$$S = \{\mathbf{x} \mid f(\mathbf{x}) = 0\}$$

$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x}))$$

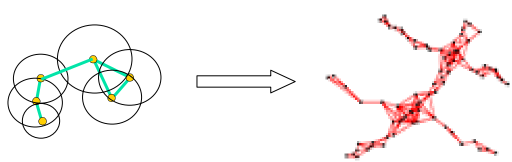


- Definition of $\mathbf{n}$ by Weighted Least Squares
- Weighting function, e.g.:

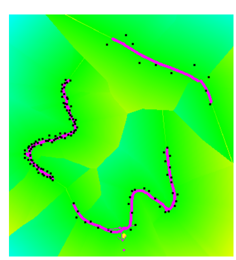
$$\theta(d) = e^{-d^2/h^2}\ , \quad d = |\mathbf{x} - \mathbf{p}_i|$$

Geometric Proximity Graph

- Here: Sphere-of-Influence graph



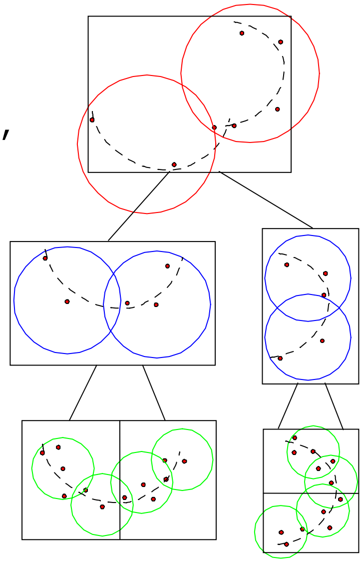
- Benefits:
  - Reduced artifacts (geodesic dist. approx.)
  - Boundary detection
  - Localized sampling density estimation



Point Cloud Hierarchy

1. Build BVH according to some local criterion (.e.g., volume of child BVs)
2. Construct subsampling and sphere covering at inner nodes

→ Efficient storage

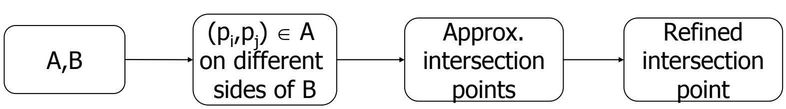


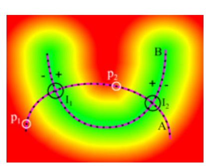
Intersection of Point Clouds

- Given two point clouds A and B (or subsets thereof), construct a sampling of

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

- Overall method:



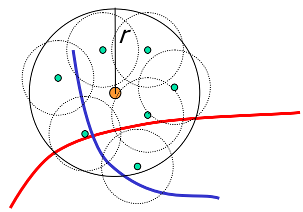


1. Construct root brackets (p_i,p_j) by randomly drawing $O(a \cdot \ln a + c \cdot a)$ many points, where a = "desired density of brackets".
2. Find $\hat{p}$ along shortest path $\overline{p_i p_j}$ in proximity graph, such that $|f(\hat{p})| = \min$ (by interpolation search)
3. Sample sphere around $\hat{p}$ by

$$s \ln s + cs$$

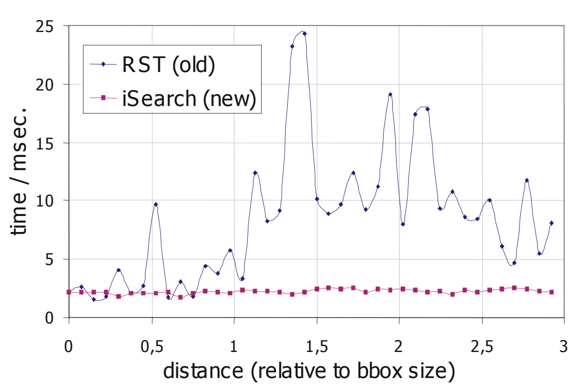
many points, where

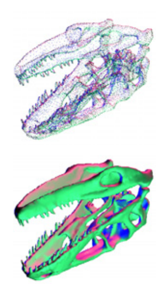
$$s = \lceil \sqrt{3} \cdot r / \varepsilon \rceil^3$$



Results

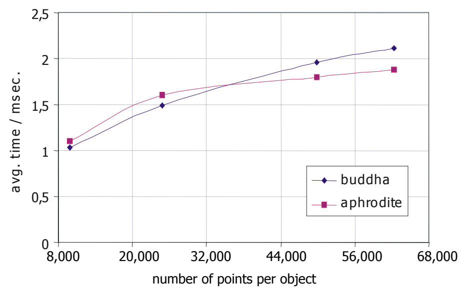
- Benchmarking old vs. new method





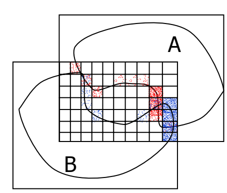
28,000 points

- Theoretical complexity: $O(\log \log N)$
  - Assumptions: f(x) monotone along paths $\overline{p_i p_j}$; and, evenly sampled point cloud.
- Experimental complexity:

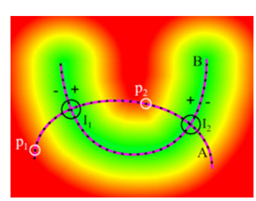


Summary

- Method to turn hierarchical collision detection into time-critical collision detection



- Collision detection for point clouds



Future Work

- Collision detection for deformable point clouds
- Approximate contact computation
- Time-critical separation or penetration distance

Acknowledgements

- Jan Klein, Paderborn University, Germany;
  and Prof. Meyer auf der Heide, Head of Theor. CS, Paderborn University.
- DFG, grant ZA 292/1-1 ("Aktionsplan Informatik")

Thanks a lot for the attention!