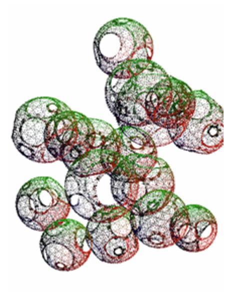
Point Cloud Collision Detection

- Modern acquisition methods lead to modern object representations.
- Efficient rendering (splatting & ray-tracing).
- Only little work on interaction.



Goals

- Fast collision detection between point clouds.
- No polygonal reconstruction.

Surface Definition

- Approximate surface by implicit function

  $$S = \{\mathbf{x} : f(\mathbf{x}) = 0, \mathbf{x} \in \mathbb{R}^3\}$$

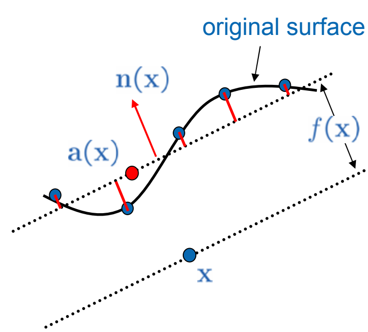
- Define $\mathbf{n}(\mathbf{x})$ by weighted least squares.

- Weight

  $$\theta(\mathbf{x}, \mathbf{p}) = e^{-\frac{d(x,p)^2}{h^2}}$$
  $$d(\mathbf{x}, \mathbf{p}) = \text{some distance measure}$$

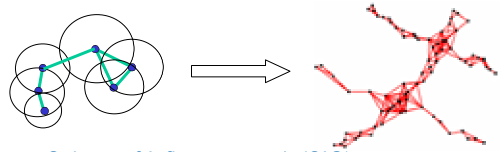
- $f(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{a}(\mathbf{x}) - \mathbf{x})$

- Which distance measure to use?



[Klein & Zachmann, 2004]

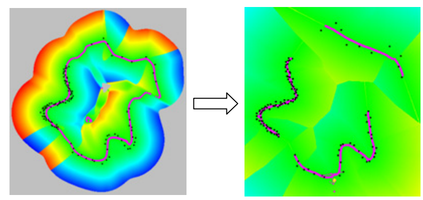
- Geometric proximity graph:
  - nodes = points
  - edges = "neighboring" points



Sphere-of-Influence graph (SIG)

- Approximate geodesic distance by shortest path.

- Properties:
  - Nice surface
  - Efficient evaluation
  - Implicit function throughout space
  - Surface with boundaries allowed



Contributions

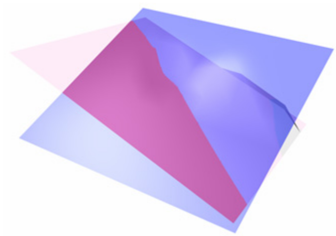
- Novel, fast intersection computation for point clouds
- Utilizes proximity graph
- Runtime O(log log N), if constant number of intersection points is sufficient.
- Quality/resolution of output is adjustable.

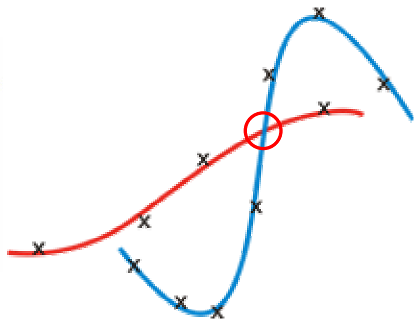
Problem Statement

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

- Given two point clouds A and B (or subsets thereof),
  - decide if there is an intersection
  - construct a sampling of

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

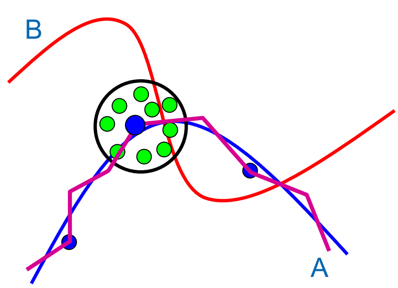




Overview

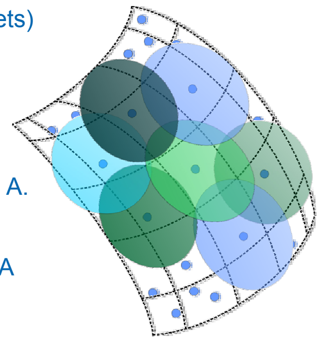
HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

1. Bracket intersections by pairs of points.
2. Find approximate intersection point (AIP) by interpolation search.
3. Refine AIP by (randomized) sampling.



1. Root Bracketing

- Goal:
  - The pairs should evenly sample the surface.
  - The two points should not be too far apart.
  - Do it without explicit spatial data structure!
- Task: construct a pairs of points (to be root brackets)

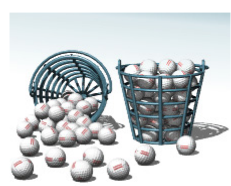


- Thought experiment:
  - Assume surface is covered by a surfels.
  - Cover each surfel with at least one point from A.
    (candidate points for root brackets)
  - For each point: try to find another point from A
    lying on *the other side* of B.
    (completing the brackets)

Covering the Surfels

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

Avoid spatial data structure $\rightarrow$ pursue probabilistic approach: occupy all a surfels with high probability!



- Assumption: A is uniformly sampled.
- Lemma from paper $\rightarrow$
  draw $O(a \ln a)$ random and independent points
  from $A \cap Vol(A \cap B)$.
  Proof: see paper.

Premise: number of intersection points should be bounded by a constant.

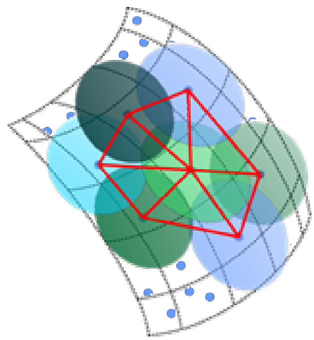
Consequence: choose a constant, or choose a depending on surfels size and surface area

Completing the Brackets

- Use $f_B(p_i) \cdot f_B(p_j) < 0$ as an indicator.
- Test only points p_j that
  - belong to the randomly chosen points
  - are close to each other

  Only a few points have to be tested!
- Solution: SIG

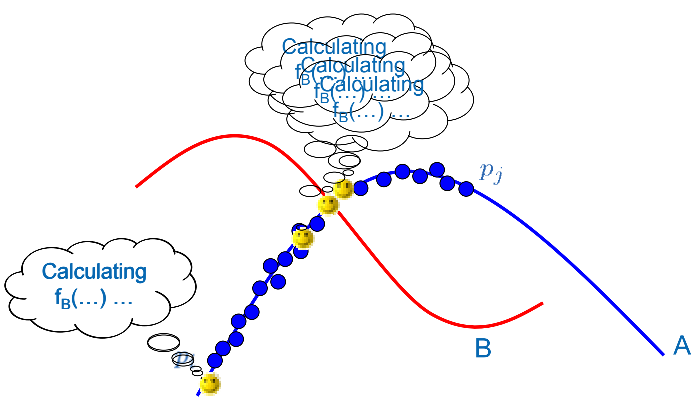
Finding brackets:
O(a ln a * d),
where d = max. out-degree;
average-ase: O(1)



2. Interpolation Search

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

- Find $\hat{\mathbf{p}} \in A$ along shortest path $\overline{p_i p_j}$ in the geometric proximity graph, such that $|f_B(\hat{\mathbf{p}})|$ is minimal.
- Utilize *interpolation search!* → O(log log *m*), m = # elements



Interpolation Search

- Assumptions:
  - Shortest path are precomputed and stored in LUT.
  - f_B is monotone along shortest path.

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}

$P_i \in A$

$\uparrow l = 1$ $\qquad \uparrow r = 11$

- Interpolation parameter: $x = l + \lceil \frac{-f_B(P_l)}{f_B(P_r) - f_B(P_l)} (r - l) \rceil$

- Large point clouds:
  - Memory consumption could be too high.
    $\rightarrow$ compute paths on-the-fly.
  - In practice: runtime still behaves sublinear.

3. Precise Intersection Points

- Refine approximate intersection point.

$\rightarrow$ Details: see paper…

Runtime: O(*a* ln *a*)

Complexity Considerations

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

- constant number of brackets, as a is constant
- Interpolation search: O(a ln a log log *m*) = O (log log *N*)
  (m= length of paths, is not constant!)
- Precise intersection points: O(*a* ln *a*) = O(1).
- f(x) can be evaluated in O(1).

Overall runtime: O(log log *N*)

Benchmark Scenario

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

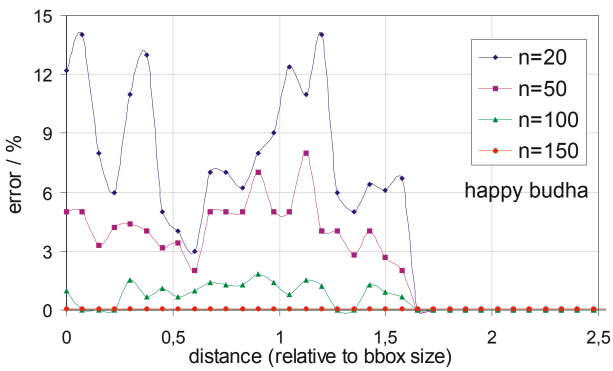
- Objects are scaled uniformly → cube size 2^3
- Perform a full tumbling turn by a fixed, large number (5000) of small steps.
- Average collision detection time for a complete revolution.

Minimal Bracket Density

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

- If number of surfels is too small → influencing spheres in the graph are too large
  → likelihood increases that
  n(x) flips its sign without x changing sides.

- Use boolean collision queries to measure error.



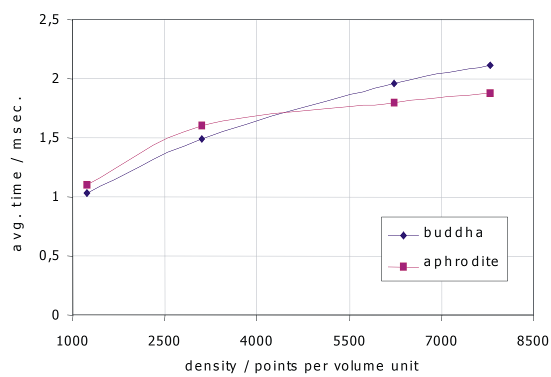
60,000 points

Complexity

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

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

- Experimental complexity:



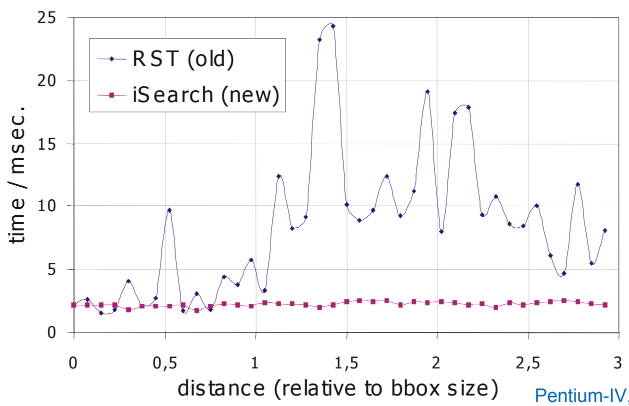
max. 60,000 points

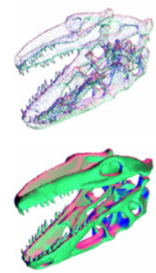
max. 90,000 points

Pentium-IV, 2.4GHz

Timings

- Benchmarking old vs. new method:
  – Old (RST) = brute-force sampling [EG'04]
  – iSearch = new





28,000 points

Pentium-IV, 2.4GHz

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

Conclusion

- Technique:
  - utilizes a proximity graph for collision detection and surface definition.
  - needs no BV hierarchies and no spatial partitioning data structure.
  - any BV hierarchy can be augmented by new technique to increase performence.

- Runtime:
  - fast (approximate) collision detection
  - overall runtime: O(log log *N*) in average case.
  - speedup of factor 5–10 compared to "old" technique.

- Quality/resolution of output (intersection points) can be adjusted ($\rightarrow$ surfel radius)

Future Work

- Deformable point clouds, SIG can be updated in O($\log^3 N$).
- More rigorous estimation of minimal bracket density.
- Consistency of n(x).
- Out-of-core collision detection.

HEINZ NIXDORF INSTITUTE
University of Paderborn
Algorithms and Complexity
Jan Klein

Thank you!

Jan Klein

Heinz Nixdorf Institute and
Institute of Computer Science
University of Paderborn, Germany

janklein@uni-paderborn.de

Dr. Gabriel Zachmann

Dept. of Computer Graphics and
Virtual Reality
University of Bonn, Germany

zach@cs.uni-bonn.de