Collision detection for Point Clouds

Gabriel Zachmann

Bonn University

zach@cs.uni-bonn.de

Motivation

- 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

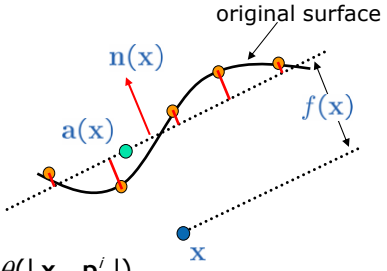
Definition of the Surface

- Implicit:

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

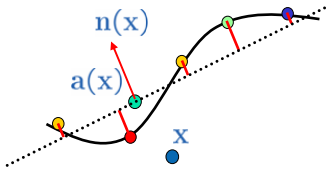
$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{a}(\mathbf{x}))$$

- Definition of **f/n** by Weighted Least Squares (WLS):

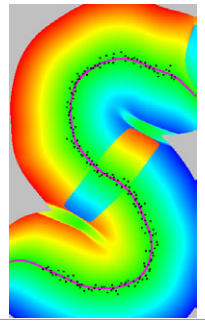


$$\arg\min_{\mathbf{n},|\mathbf{n}|=1} \sum_{i=1}^{N} \left(\mathbf{n}(\mathbf{x}) \cdot (\mathbf{p}^i - \mathbf{a}(\mathbf{x}))\right)^2 \cdot \theta(|\mathbf{x} - \mathbf{p}^i|)$$

- Weighting function (kernel), e.g.:



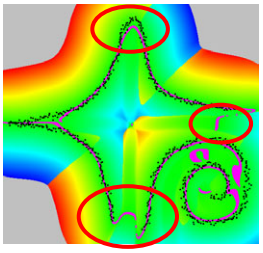
$$\theta(d) = e^{-d^2/h^2},$$
$$d = |\mathbf{x} - \mathbf{p}_i|$$

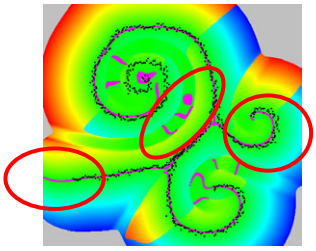


- Which distance measure?

Problems with Euclidean distance

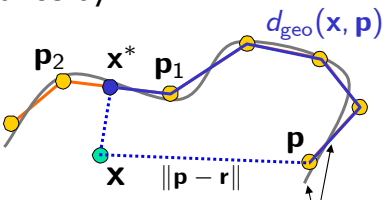
- Leads to artifacts, such as
- Bias / Noise
- Spurious zero sets
- "Fuzzy" boundaries





Solution: Proximity Graph

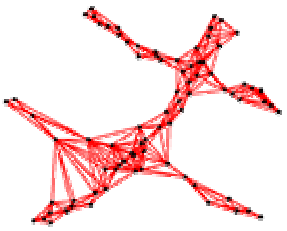
- Geometric proximity graph:
  - Points = nodes of graph
  - Edges = "neighboring" points
- Approximate geodesic distance by shortest path:
  - compute closest point x* on edge,



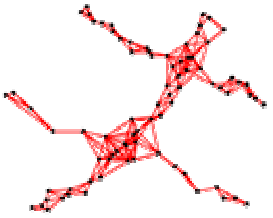
- $d_{geo}(\mathbf{x}, \mathbf{p}) = \min_{\mathbf{r}=\mathbf{p}_1,\mathbf{p}_2} \{ d(\mathbf{r}, \mathbf{p}) + \|\mathbf{p}^* - \mathbf{r}\| \}$ (proximity graph)

Sphere-of-Influence Graph (SIG)

- k-SIG:

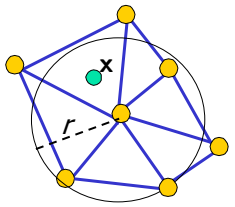


- With simple outlier pruning (3rd quartile + interquartile):



Graph yields Automatic Bandwidth

- Critical parameter h in $\theta(d) = e^{-d^2/h^2}$:
  - Too small → too much variance
  - Too large → too much bias
- Should be adapted to *local* sampling density
- Use graph to determine **h** = **h**(**x**) :
  - Estimate local sampling density **r**(**x**)
  - Compute $h(\mathbf{x}) = \frac{\eta\, r(\mathbf{x})}{\sqrt{-\log \theta_\varepsilon}}$

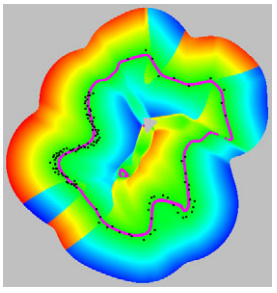


Boundary detection

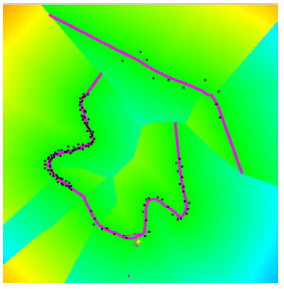
- Observation: $\mathbf{a}(\mathbf{x})$ is always inside convex hull
- Idea: $\mathbf{x}$ not on S if "too far" away from $\mathbf{a}(\mathbf{x})$, *relative* to local sampling density
- New implicit function

$$\hat{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{if } |f(\mathbf{x})| > \varepsilon \vee \|\mathbf{x} - \mathbf{a}(\mathbf{x})\| < 2r(\mathbf{x}) \\ \|\mathbf{x} - \mathbf{a}(\mathbf{x})\|, & \text{else} \end{cases}$$

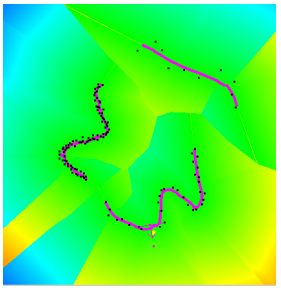
Example Surfaces



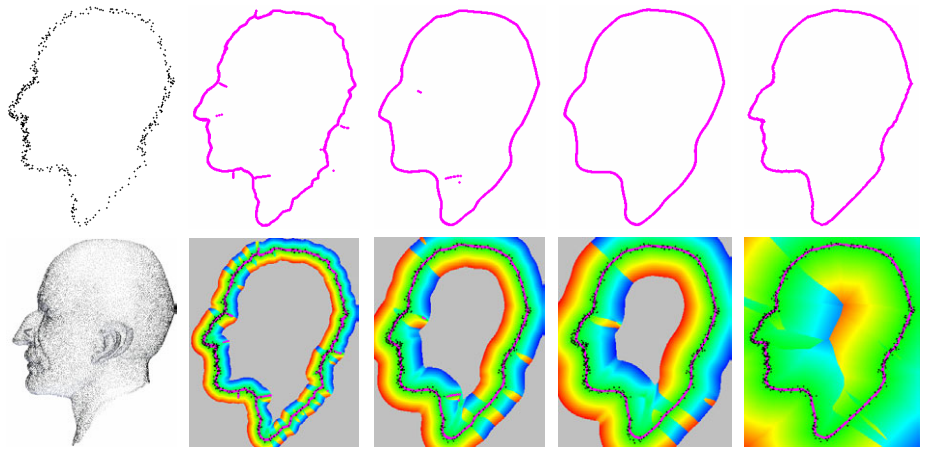
Plain WLS



With automatic bandwidth det.

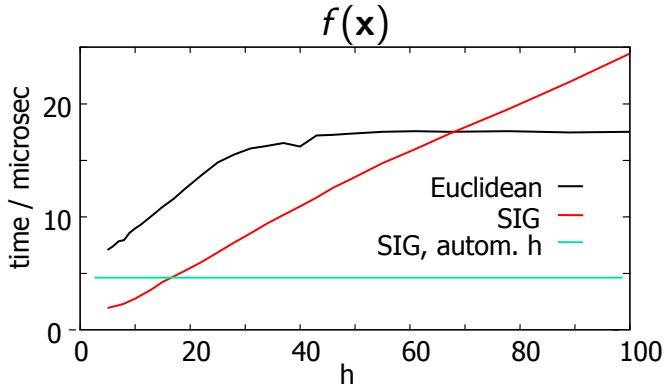


Plus automatic boundary det.



WLS, h=5 WLS, h=10 WLS, h=14 SIG, autom. h

Evaluation Time of Implicit Fct

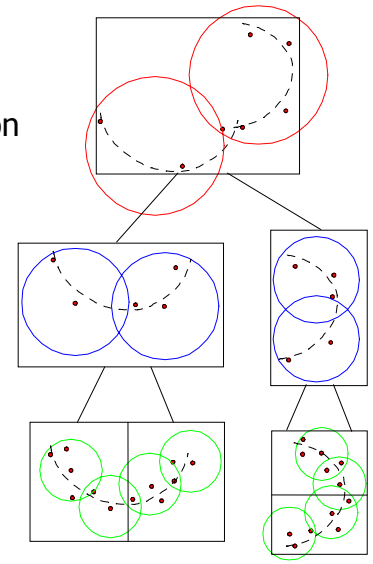


Pentium III, 1 GHz

Point Cloud Hierarchy

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

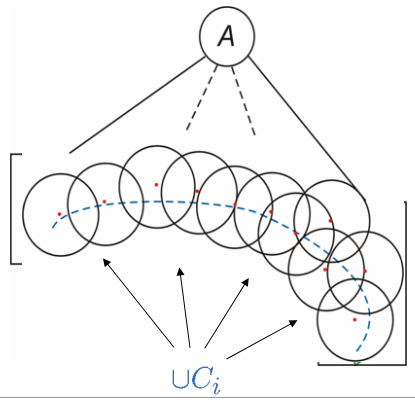
→ Efficient storage

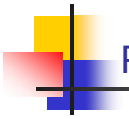


Construction of the Sphere Covering

- Observation: surface stays (usually) within the set of convex hulls C_i of the leafs underneath A

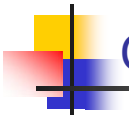
→ *c* sample points and 1 radius per node





Randomized Technique

- Basic operation:
  - Construct random point inside $\bigcup C^i$
- With basic opeation, draw set of samples from orig. point set for sphere centers, such that they …
  - are more likely close to the middle of $\bigcup C^i$
  - have approx. same distance from each other
- Grow common radius of all spheres simultaneously
  - Check for covering property like Monte-Carlo integration

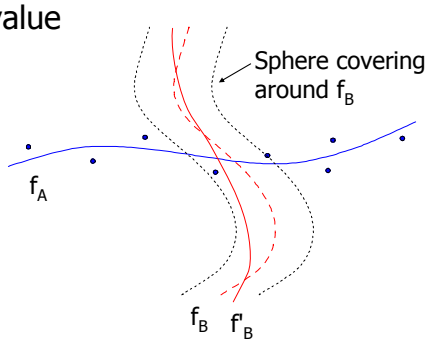


Collision Detection Algorithm

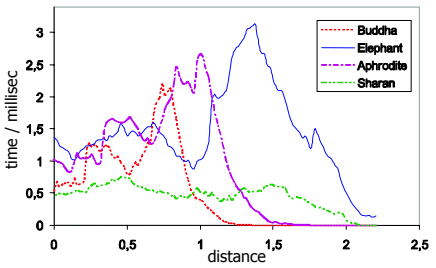
- Just like standard hierarchical CD using BVHs:
  1. Check BVs for overlap
  2. Check sphere coverings for overlap
  3. Descend into relevant child pairs
     - At leaves: check implicit surfaces for intersection
- Time-ciritical version ...

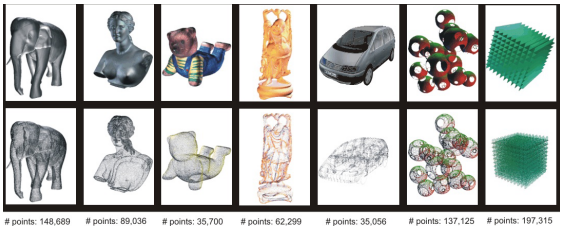
Prioritizing the Traversal

- Observation: if points from node A are on different sides of (simplified) surface in node B, then intersection is likely
- Interpret sign and value of f'_B as indicator of likelihood



Collision Detection Performance



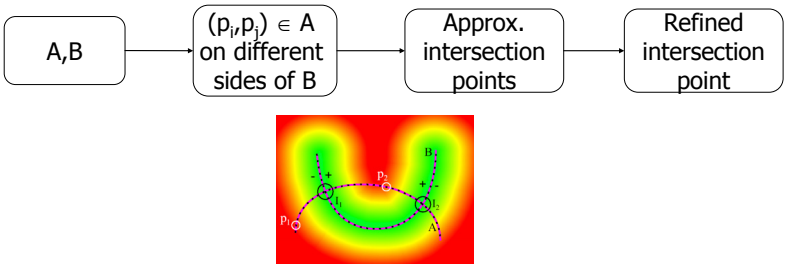


Intersection of Point Clouds

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

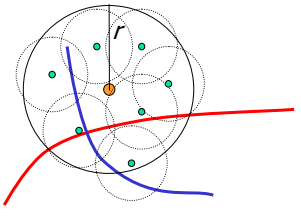
$$\mathcal{Z} = \{x \mid f_A(x) = f_B(x) = 0\}$$

- Overall method:



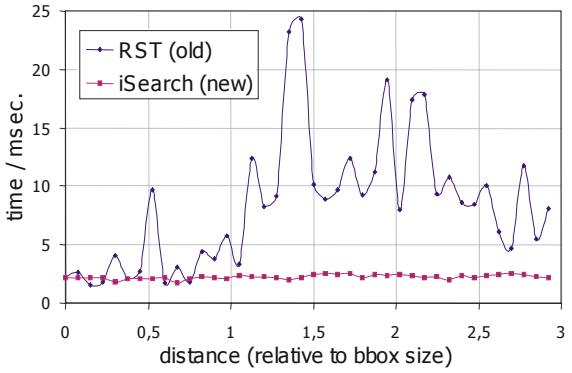
Randomized Technique

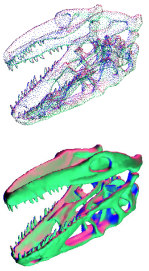
1. Construct root brackets (p_i,p_j) by randomly drawing $O(a \cdot \ln a + c \cdot a)$ many points, where a = "desired density of brackets".
2. Find $\hat{p}$ along shortest path $\overline{p_i p_j}$ in proximity graph, such that $|f(\hat{p})| = \min$ (by interpolation search)
3. Sample sphere around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



Results

- Benchmarking old vs. new method

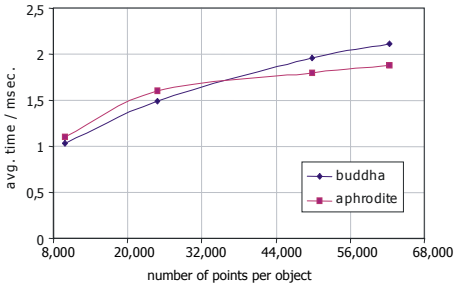




28,000 points

Results

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



References

- Boissonnat & Cazals, 2000 : Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions.
- Ohtake et al., Siggraph 2003: Multi-level Partition of Unity Implicits.
- Lee, CAGD 2000: Curve Reconstruction from Unorganized Points.
- Adamson & Alexa, SGP 2003: Approximating and Intersecting Surfaces from Points.
- Adamson & Alexa, SMI 2004: On Normals and Projection Operators for Surfaces Defined by Point Sets.
- Amenta & Kil, Siggraph 2004: The Domain of a Point Set Surface.
- Klein & Zachmann, EG 2004: Point cloud collision detection.
- Klein & Zachmann, SPBG 2004: Proximity graphs for defining surfaces over point clouds.
- Klein & Zachmann, WSCG 2005: Interpolation Search for Point Cloud Intersection.

Thanks a lot for the attention!