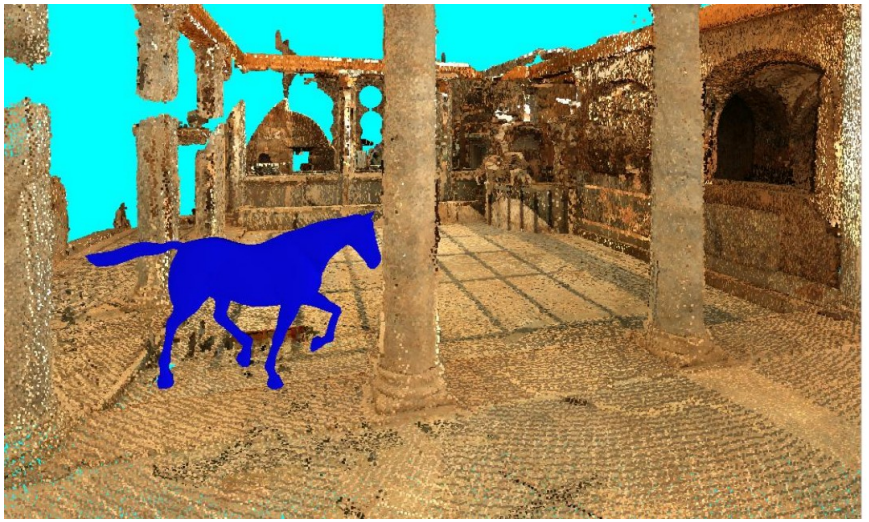
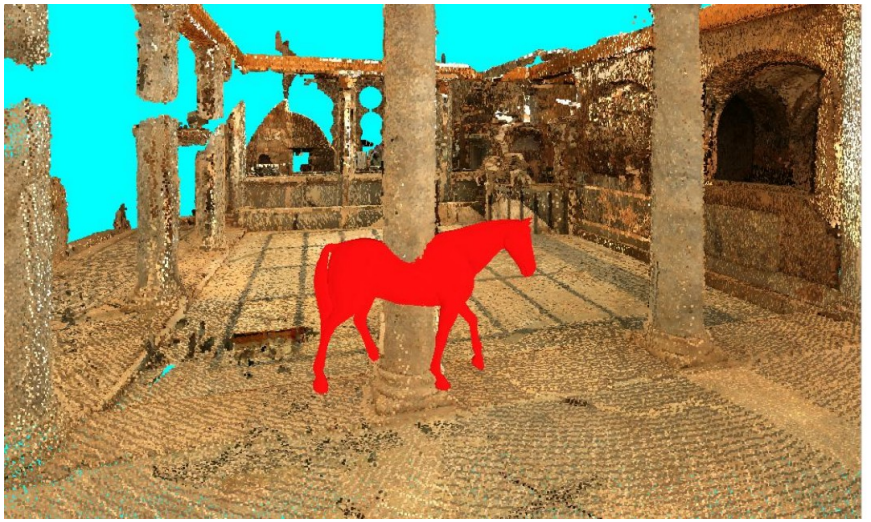
Efficient Collision Detection While Rendering Dynamic Point Clouds



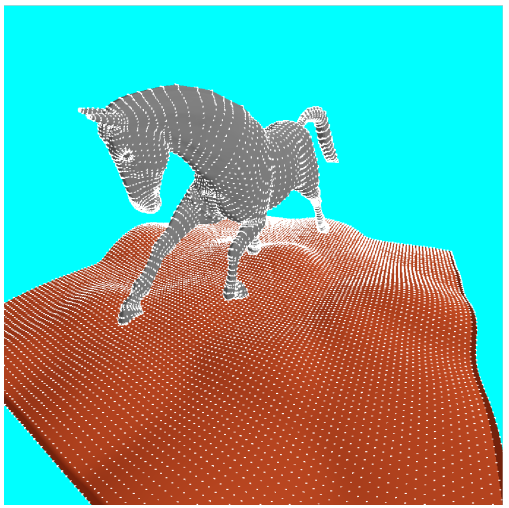


M. Radwan, S. Ohrhallinger and M. Wimmer

Vienna University of Technology, Austria

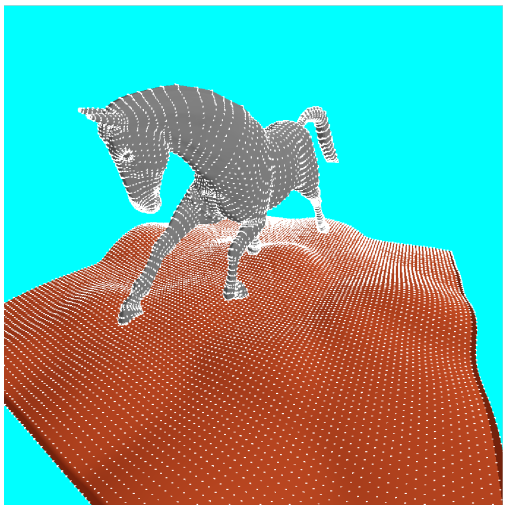
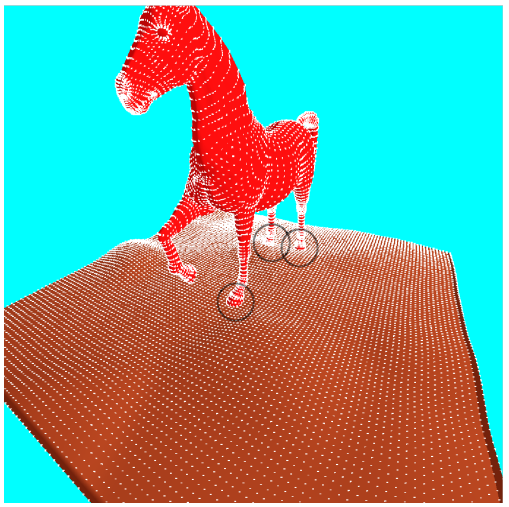
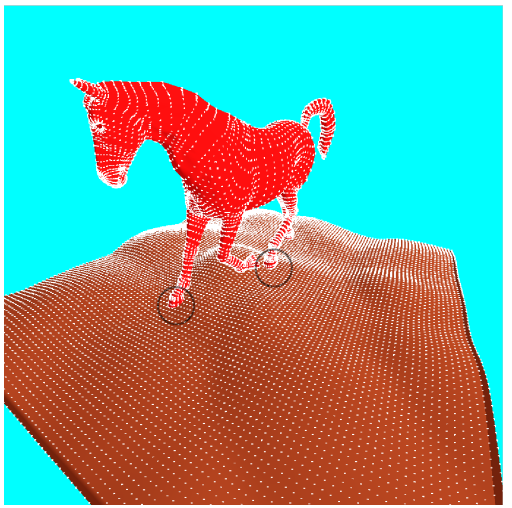
Motivation

- Point clouds are queried using bounding hierarchy
- Construction: O(N log N), query time: O(log N)



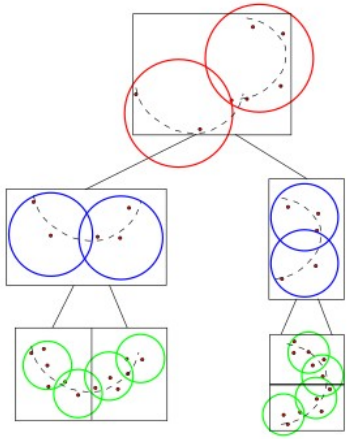
Motivation

- Point clouds are queried using bounding hierarchy
- Construction: O(N log N), query time: O(log N)

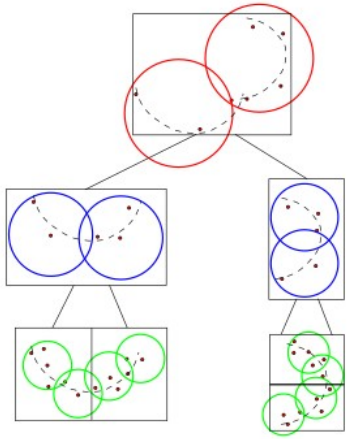
- Dynamic points without any time coherency:
  per-frame construction → too slow for N=1000000

Related work

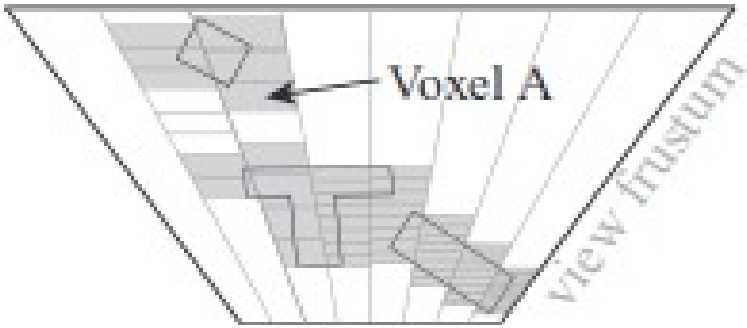


BVH [Klein et al '04]

Related work

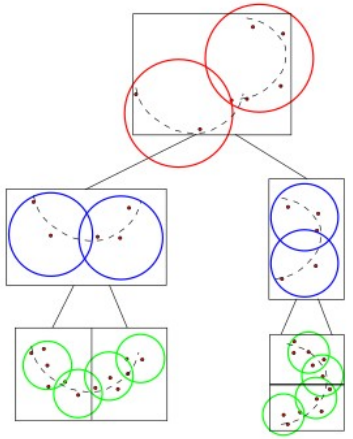


BVH [Klein et al '04]

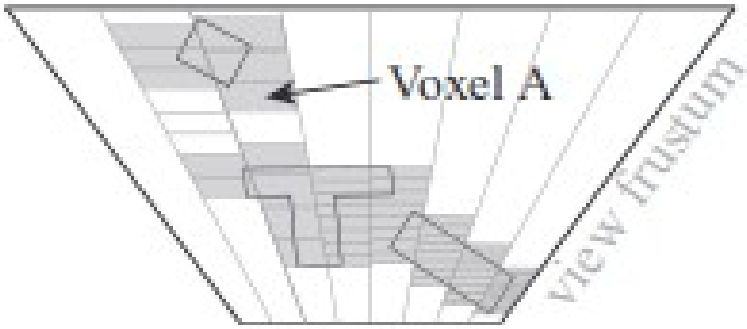


Voxels [Eisemann et al '06]

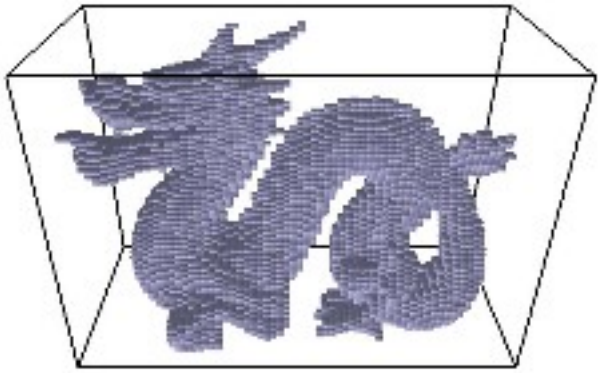
Related work



BVH [Klein et al '04]

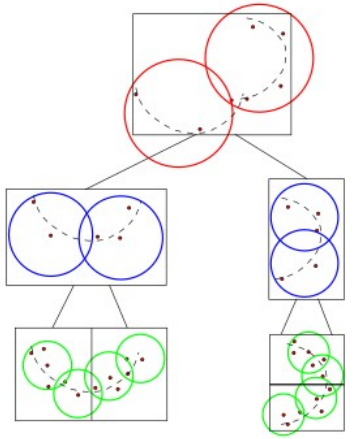


Voxels [Eisemann et al '06]

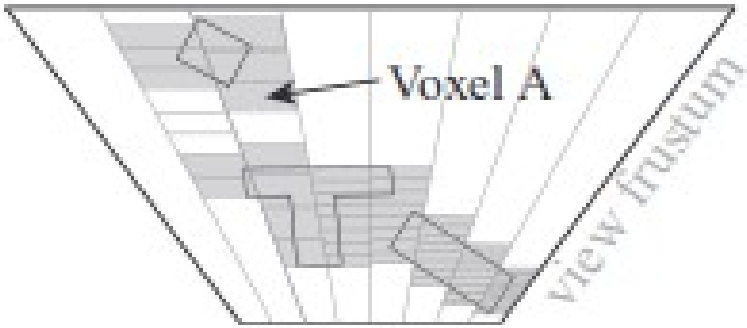


LDI [Heidelberger et al '04]

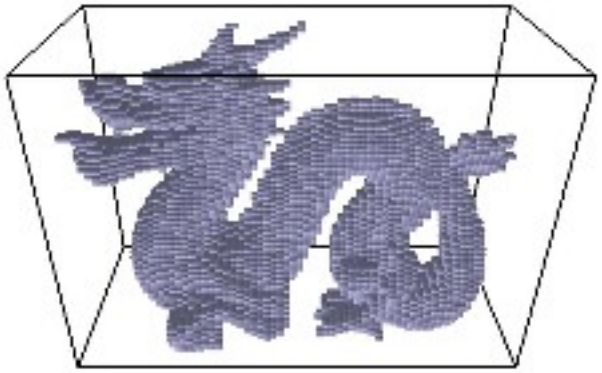
Related work



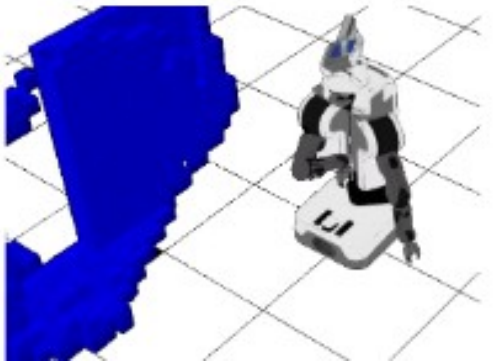
BVH [Klein et al '04]



Voxels [Eisemann et al '06]



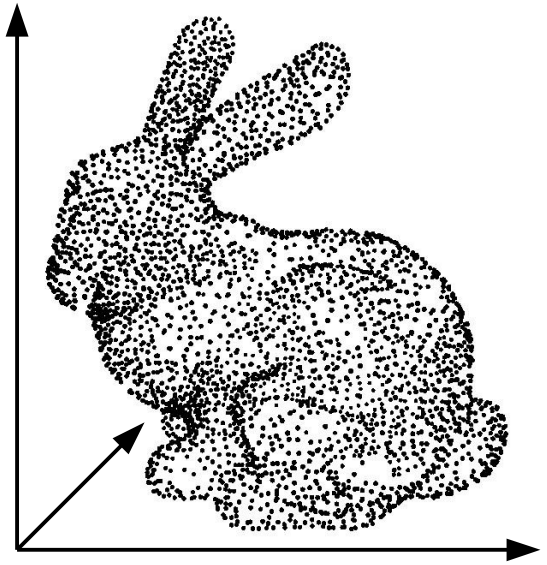
LDI [Heidelberger et al '04]



Dynamic [Pan et al '13]

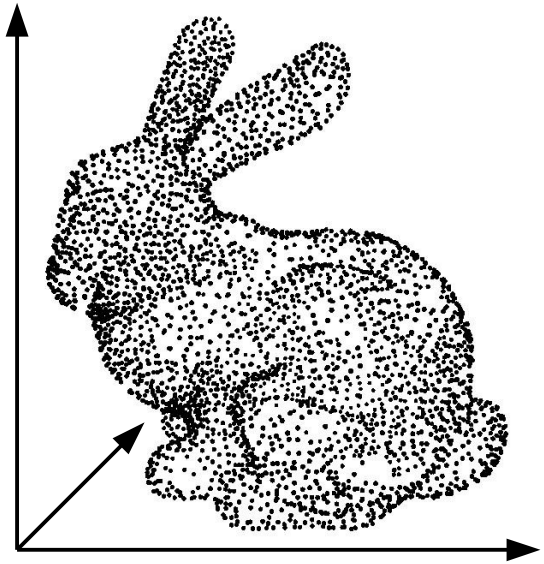
Proposed solution

- 3D point cloud is really sampled on 2D surface



Proposed solution

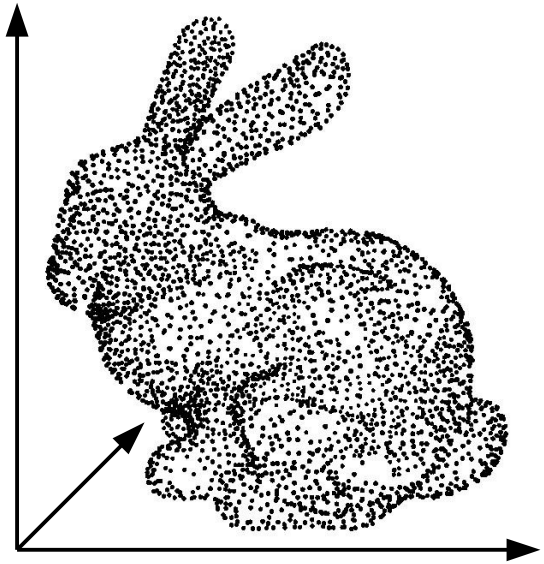
- 3D point cloud is really sampled on 2D surface



→ flatten to depth images (in screen space): O(N)

Proposed solution

- 3D point cloud is really sampled on 2D surface

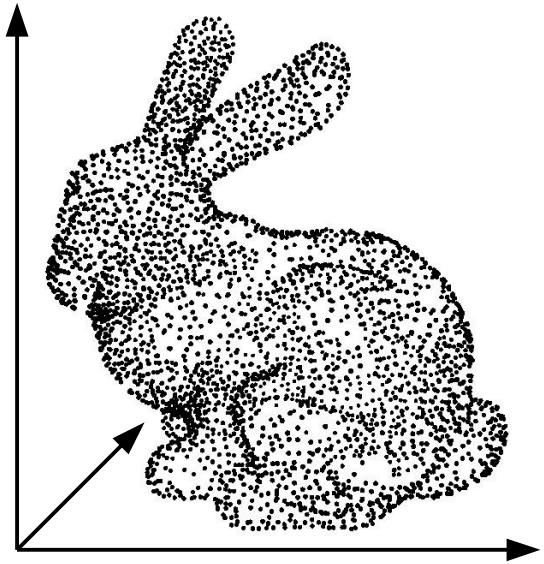


→ flatten to depth images (in screen space): O(N)

- Incidental benefits of our method:

Proposed solution

- 3D point cloud is really sampled on 2D surface

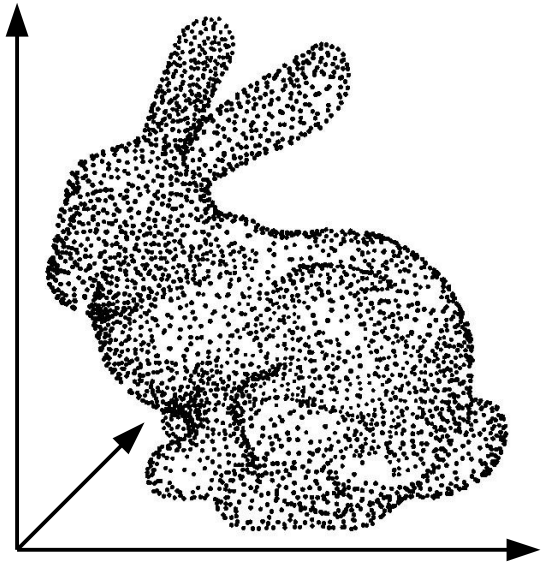


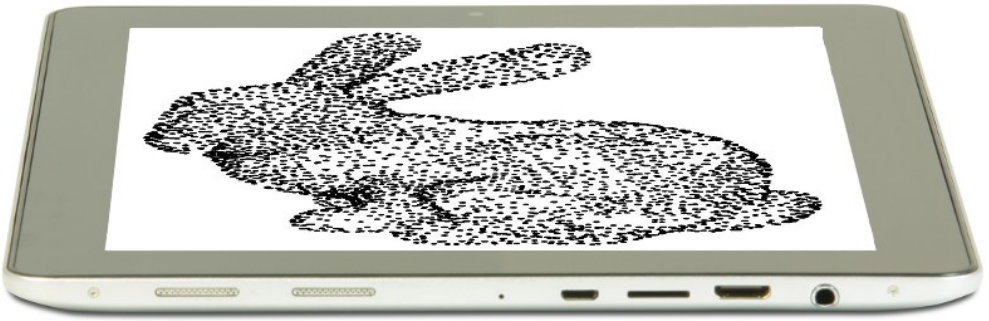
→ flatten to depth images (in screen space): O(N)

- Incidental benefits of our method:
  Superior accuracy

Proposed solution

- 3D point cloud is really sampled on 2D surface

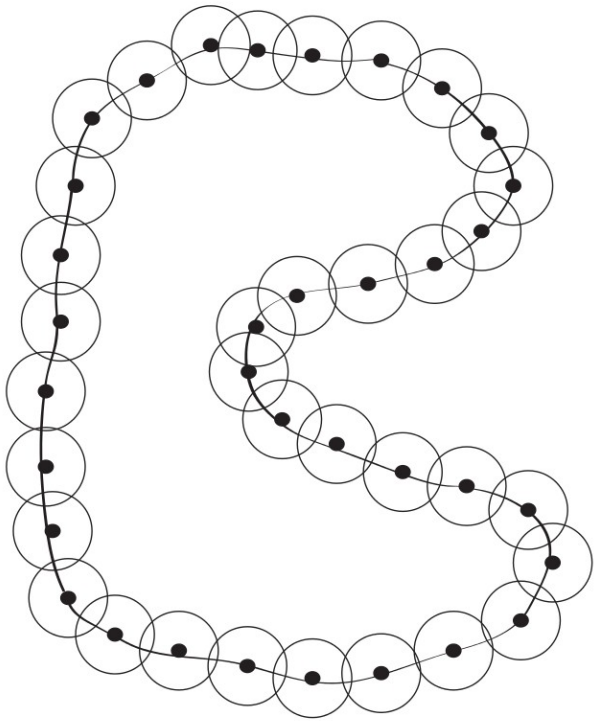




→ flatten to depth images (in screen space): O(N) ✓

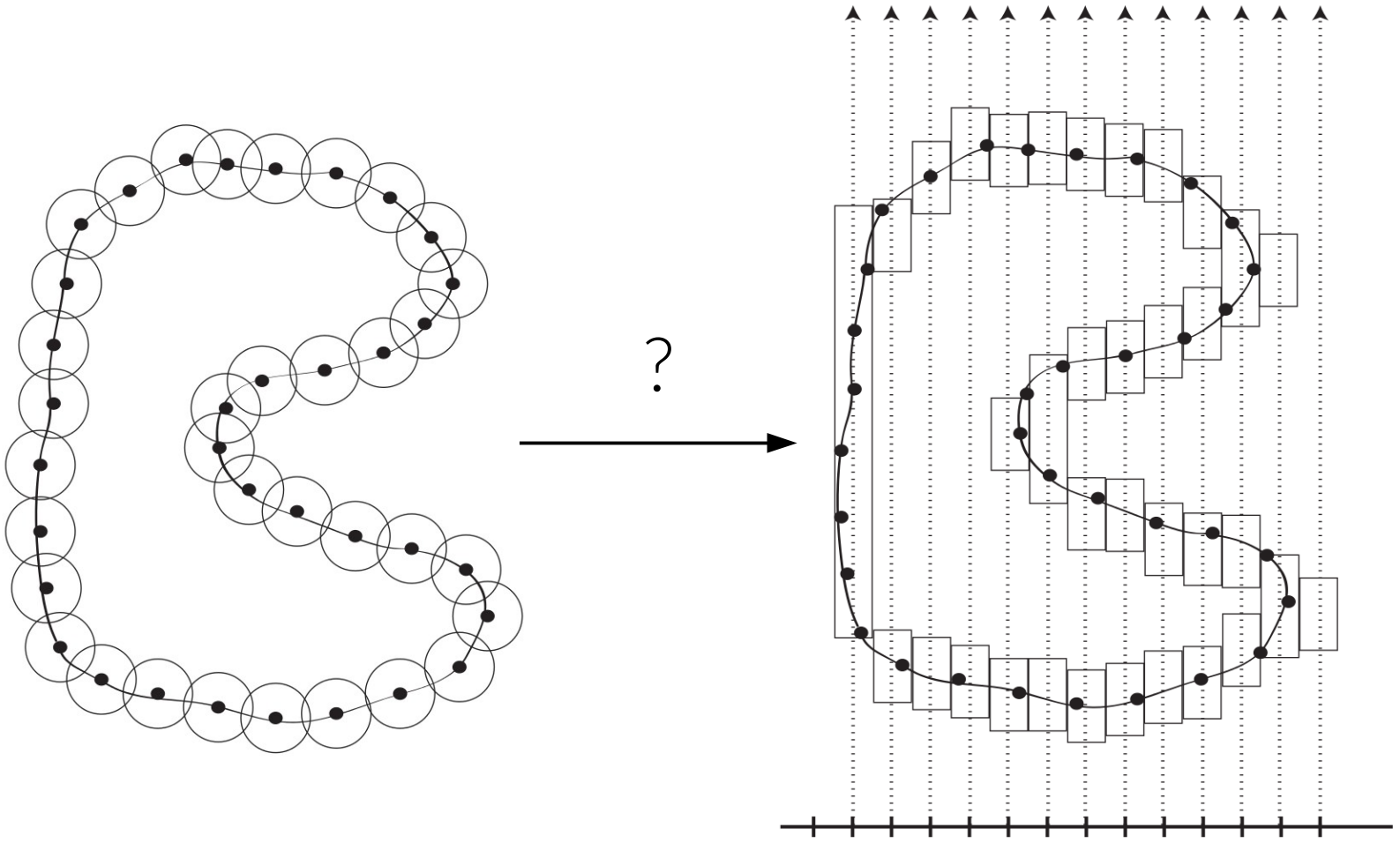
- Incidental benefits of our method:
  Superior accuracy ✓ Robustness to sensor noise ✓

Bounding the points



R^3: spherical cover

Bounding the points

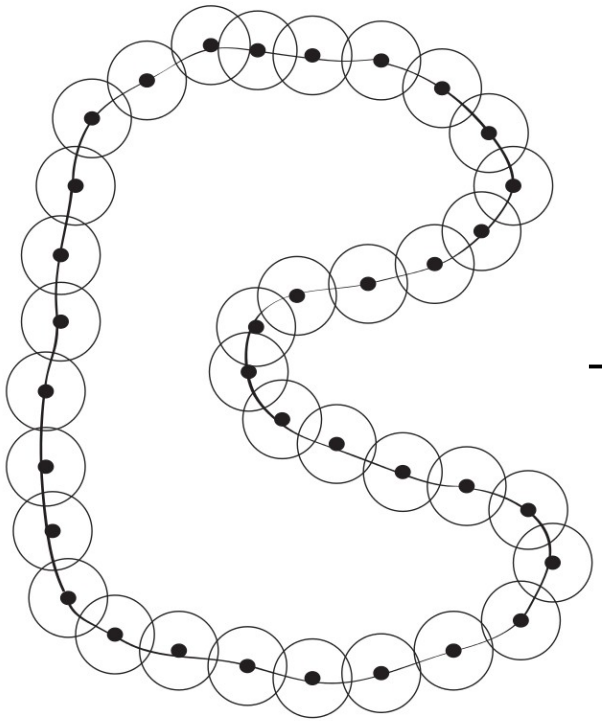


R^3: spherical cover

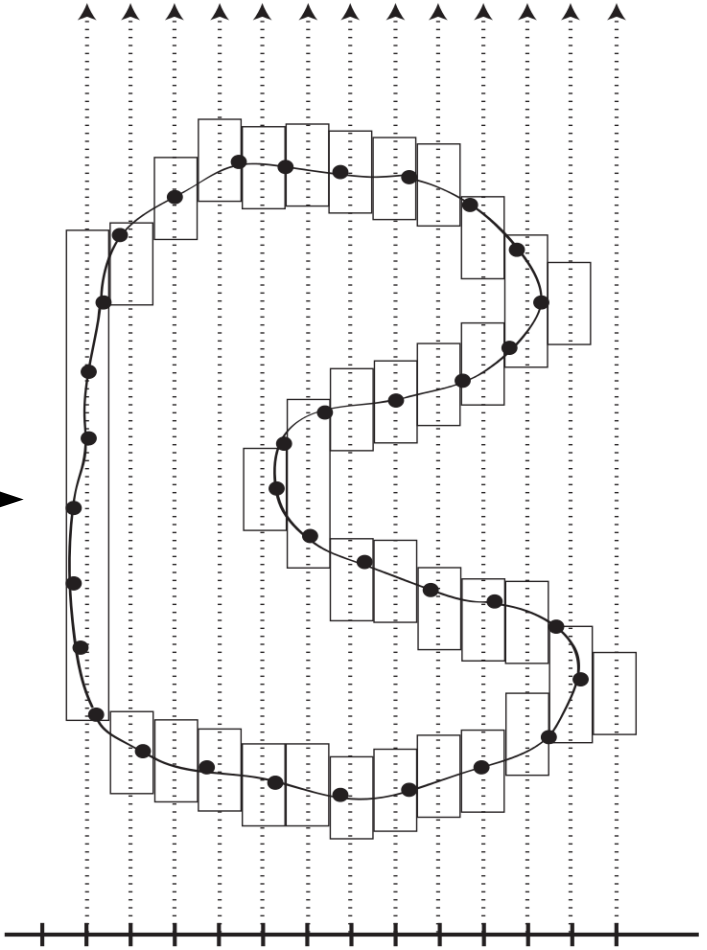
view ray depth intervals

Bounding the points

inequal boundary thickness



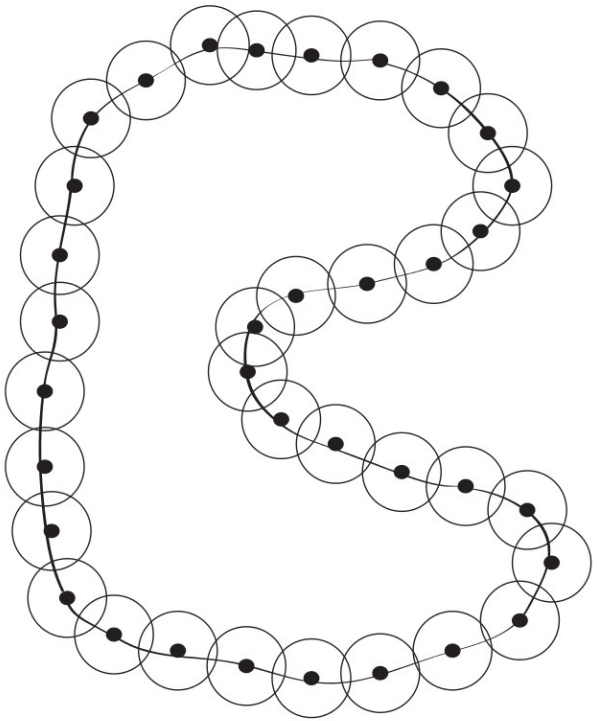
?



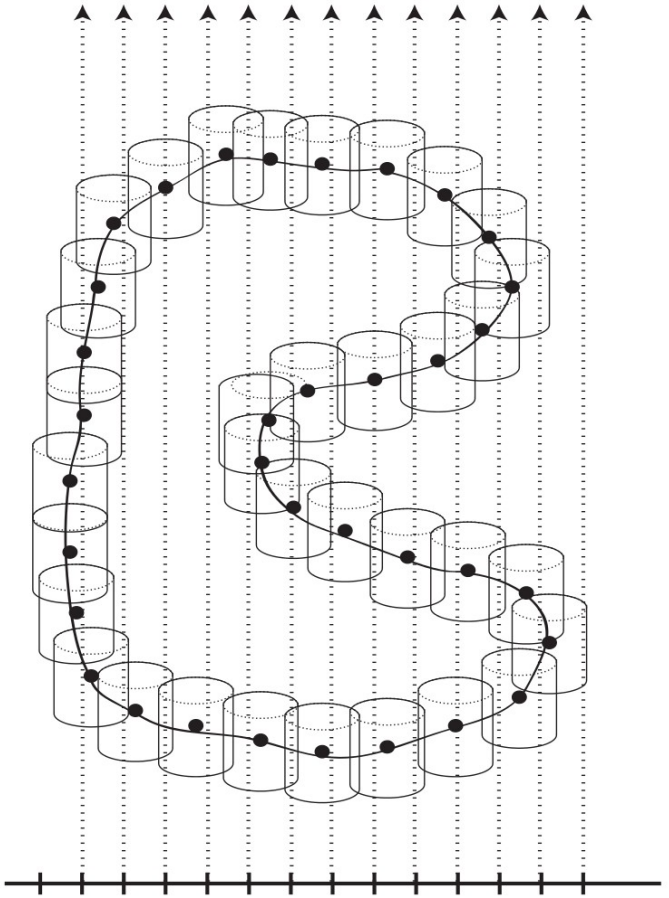
R^3: spherical cover

view ray depth intervals

Equalize boundary thickness

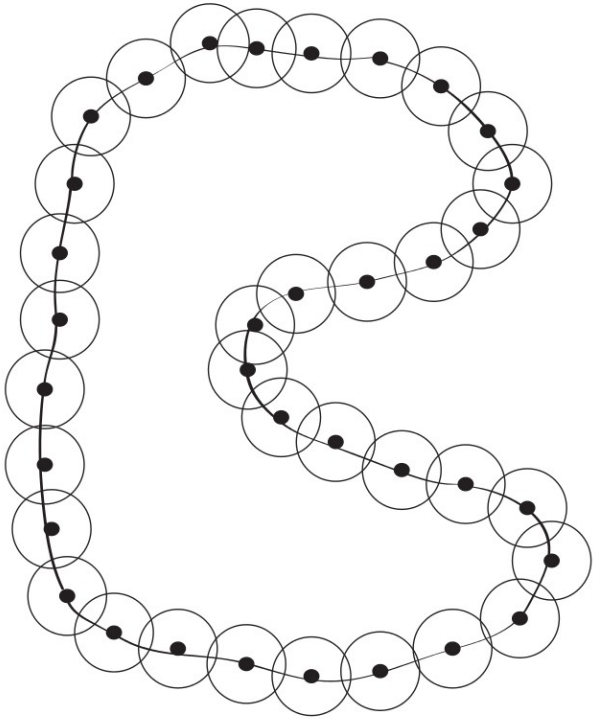


R^3: spherical cover

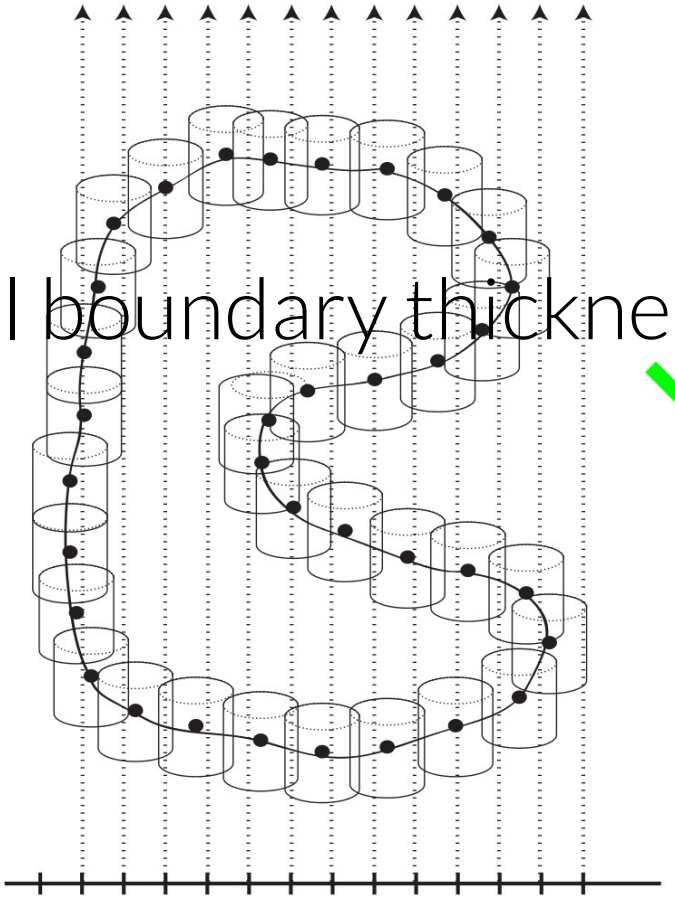


R^3: cylindrical cover

Equalize boundary thickness



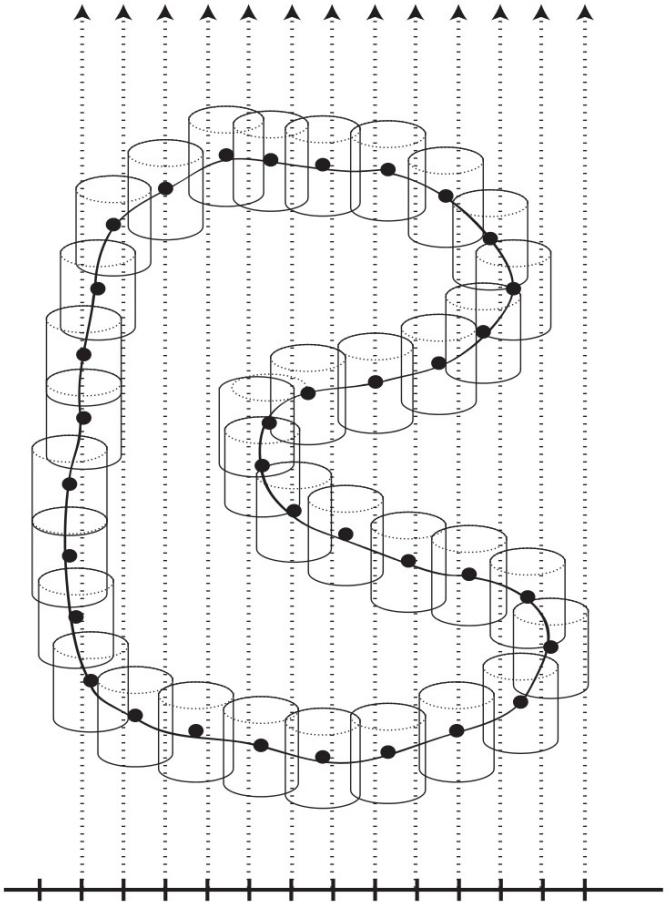
R^3: spherical cover



equal boundary thickness ✓

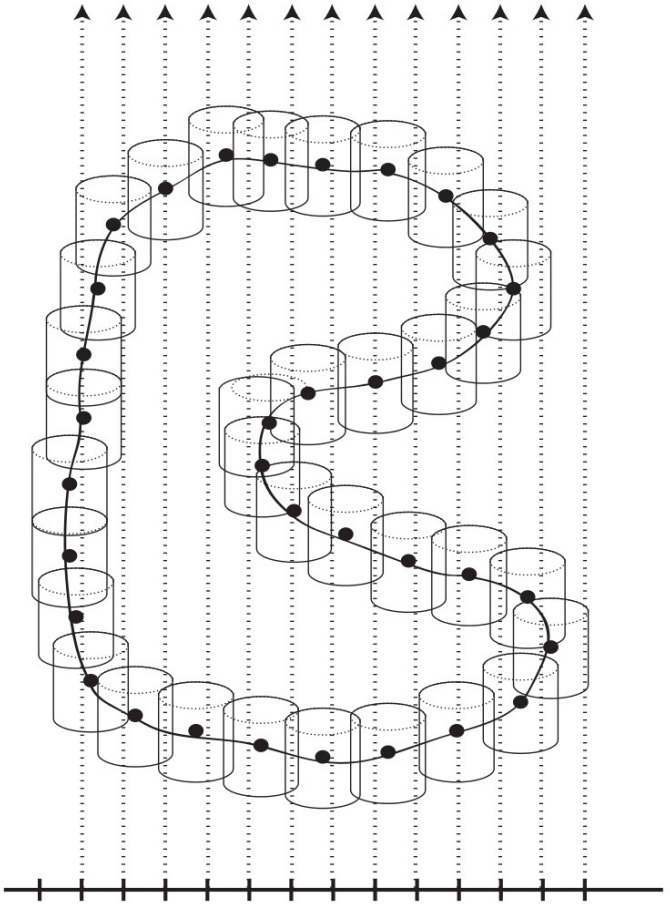
R^3: cylindrical cover

Blending view rays

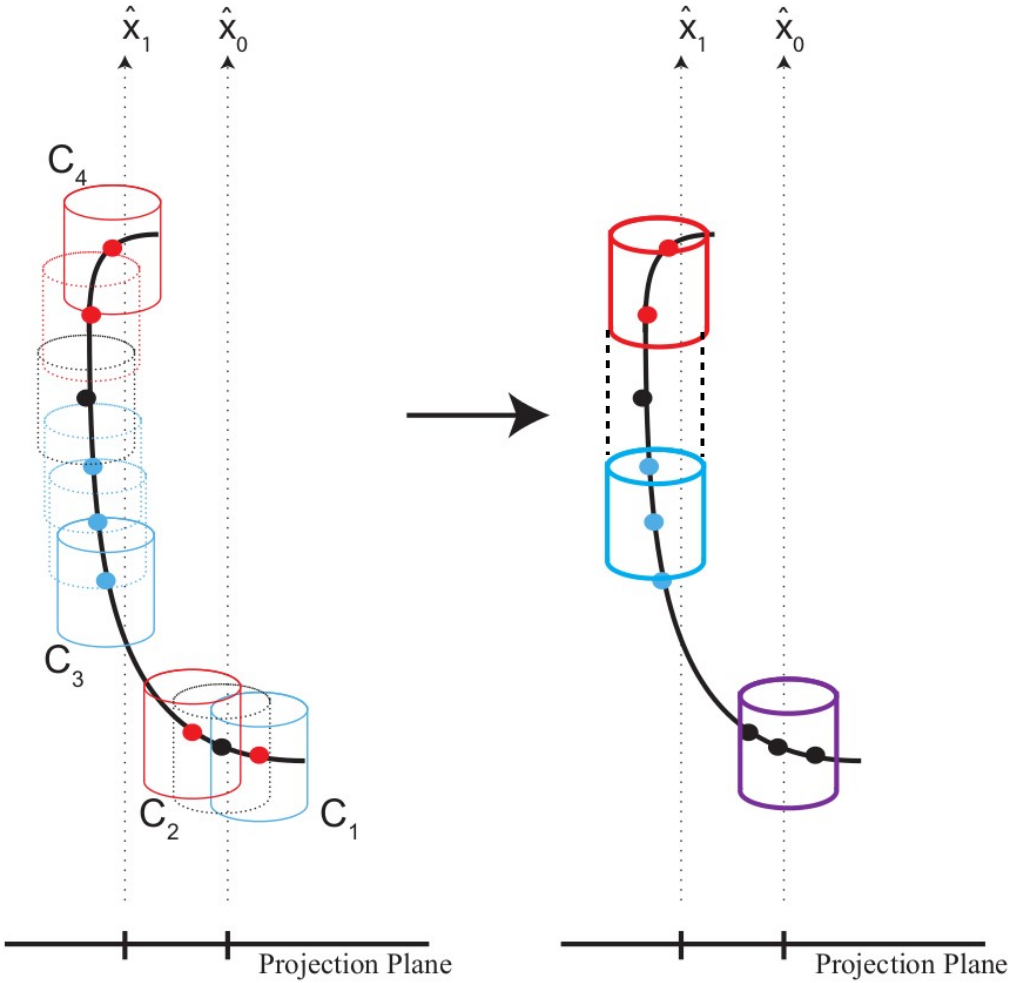


R^3: cylindrical cover

Blending view rays

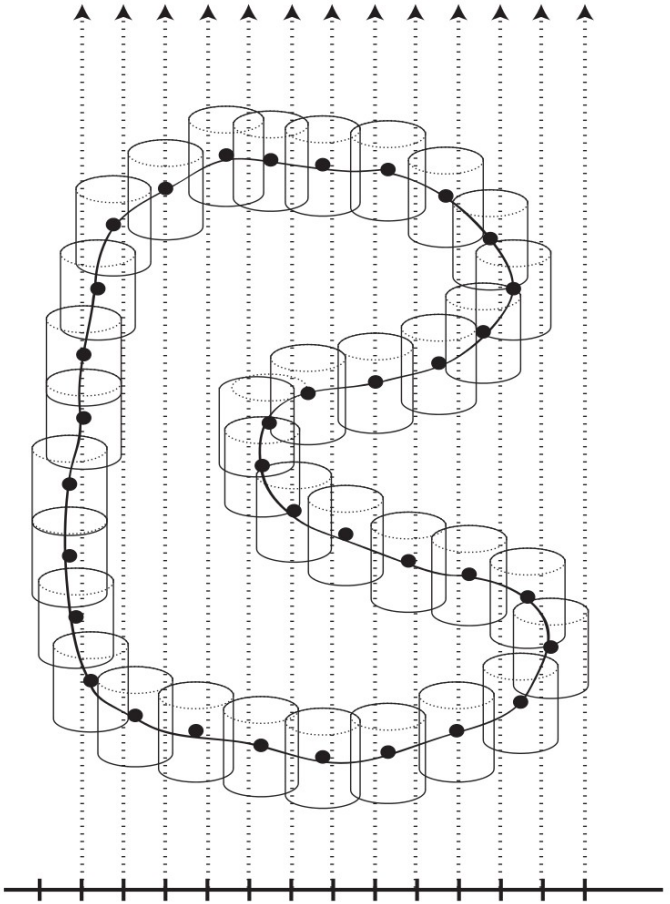


R^3: cylindrical cover

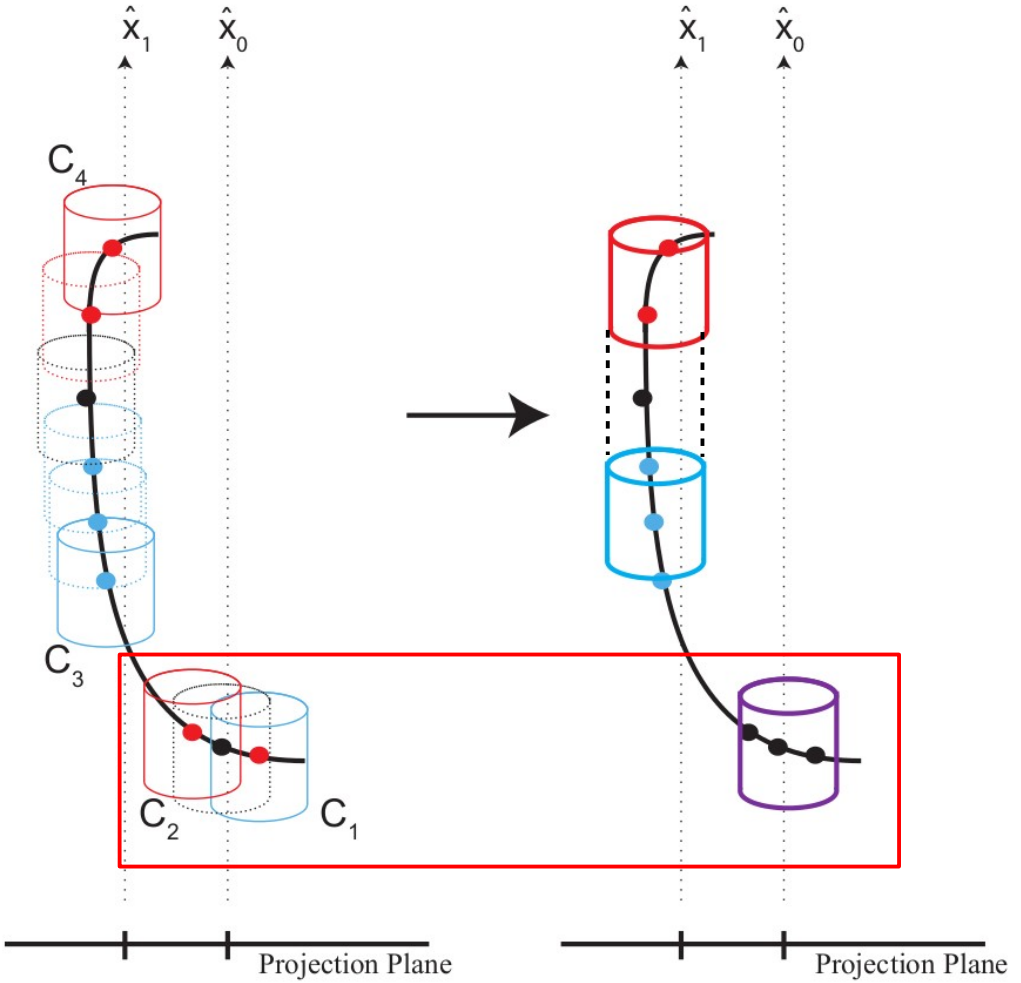


cylinders → blended view rays

Blending view rays

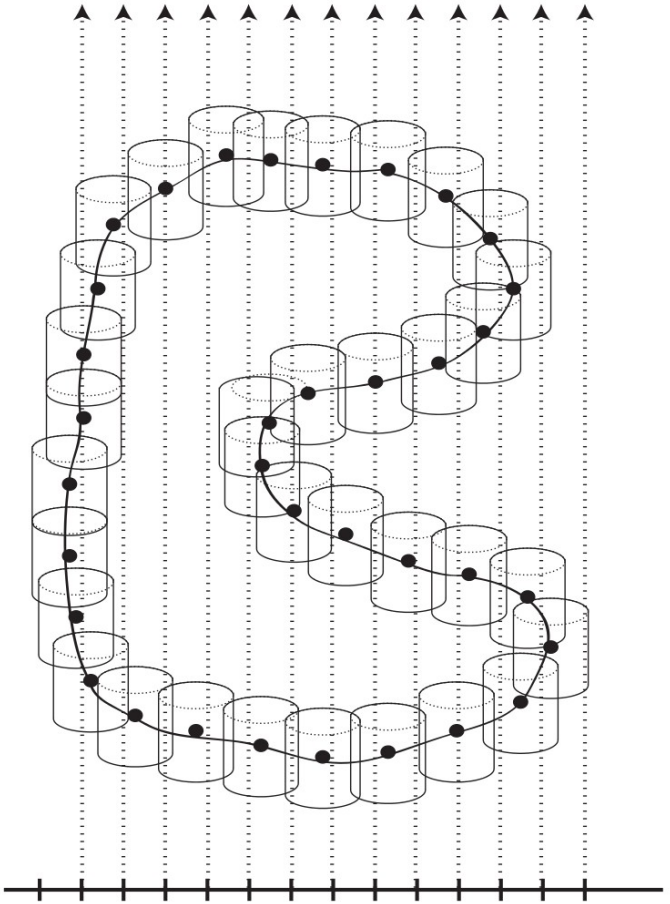


R^3: cylindrical cover

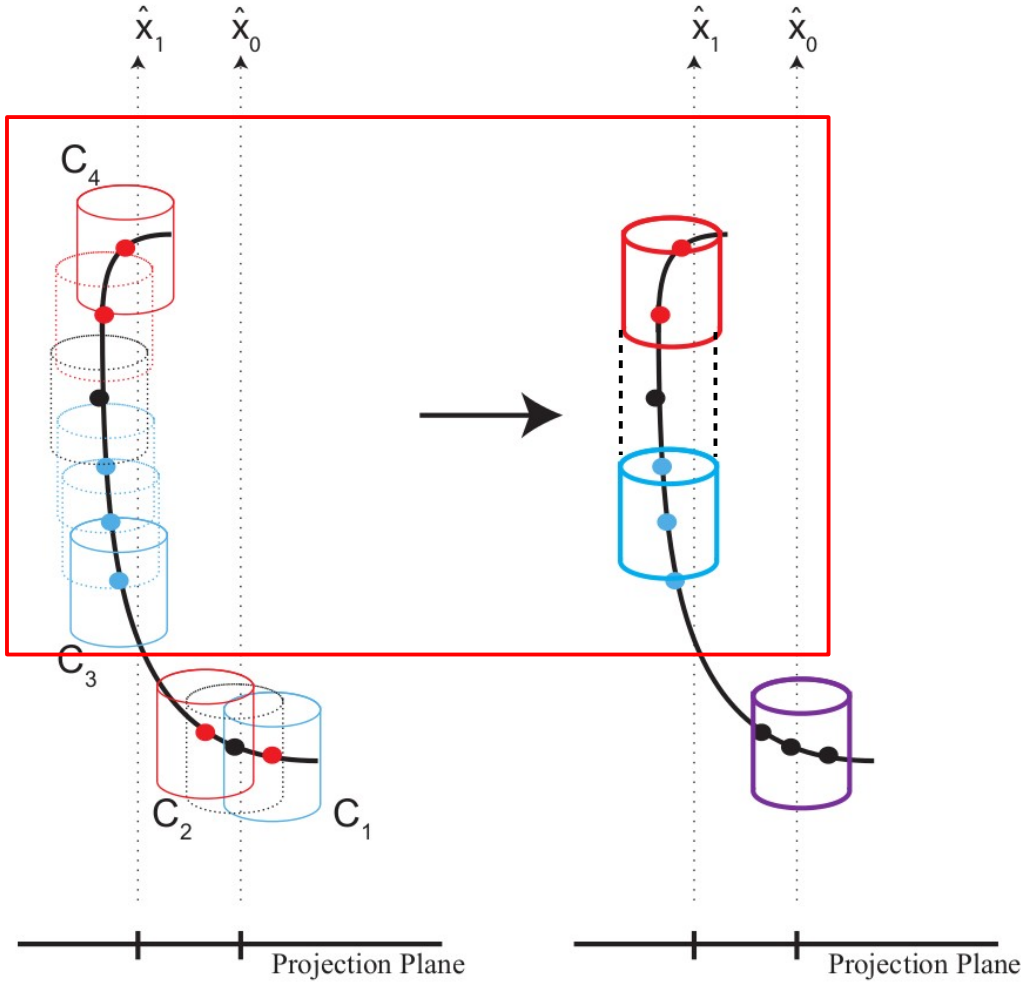


cylinders → blended view rays

Blending view rays

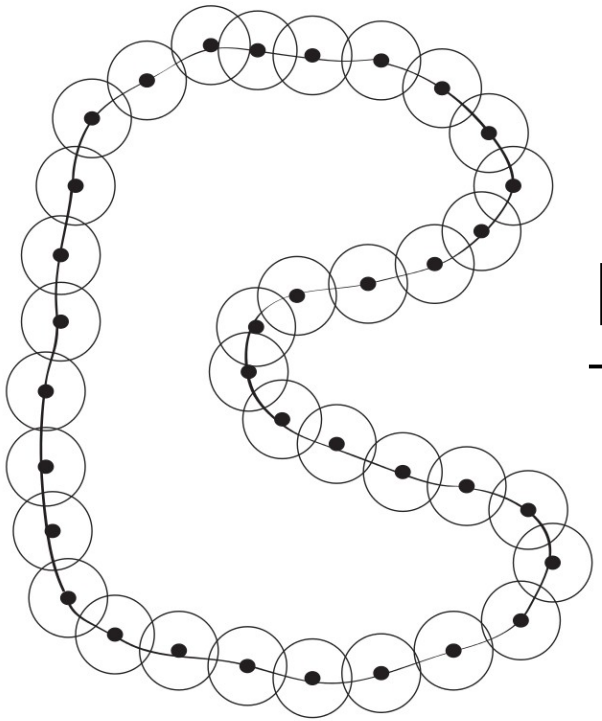


R^3: cylindrical cover

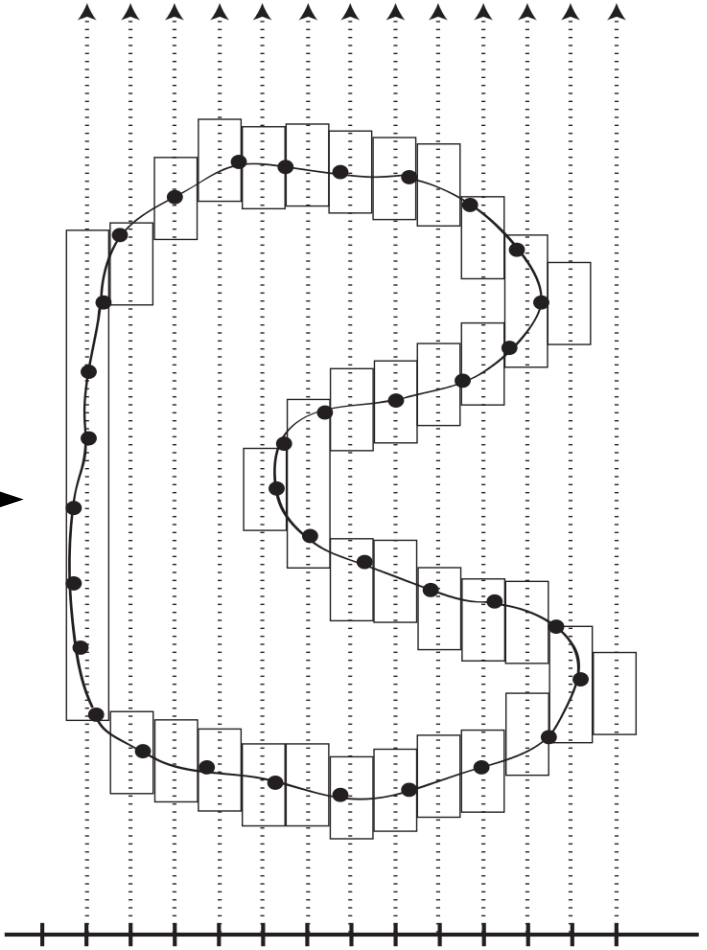


cylinders → blended view rays

Discretize in screen space



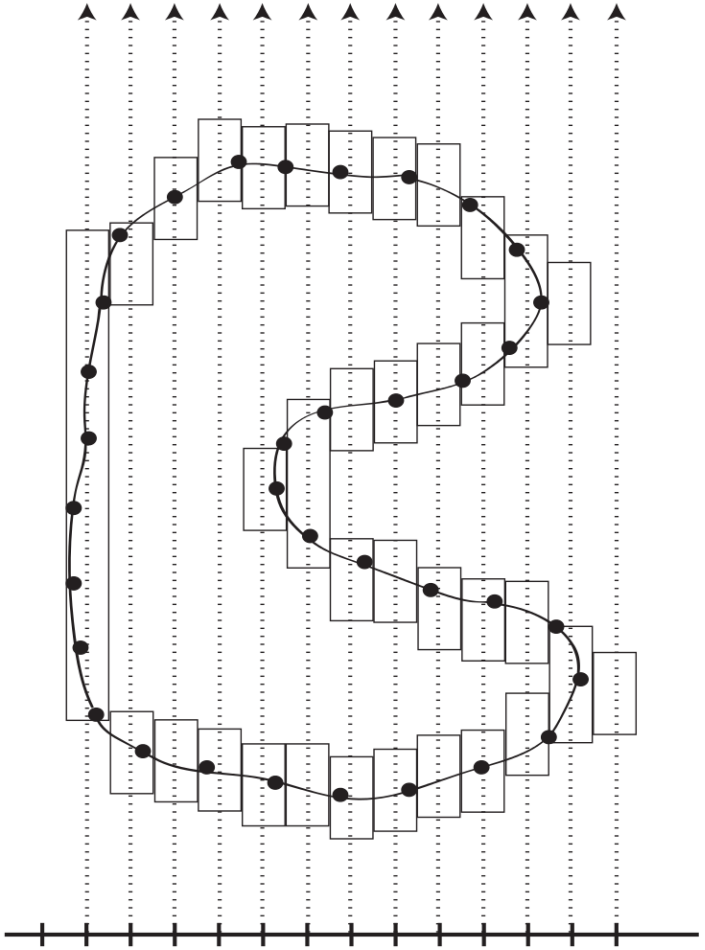
blended



R^3: spherical cover

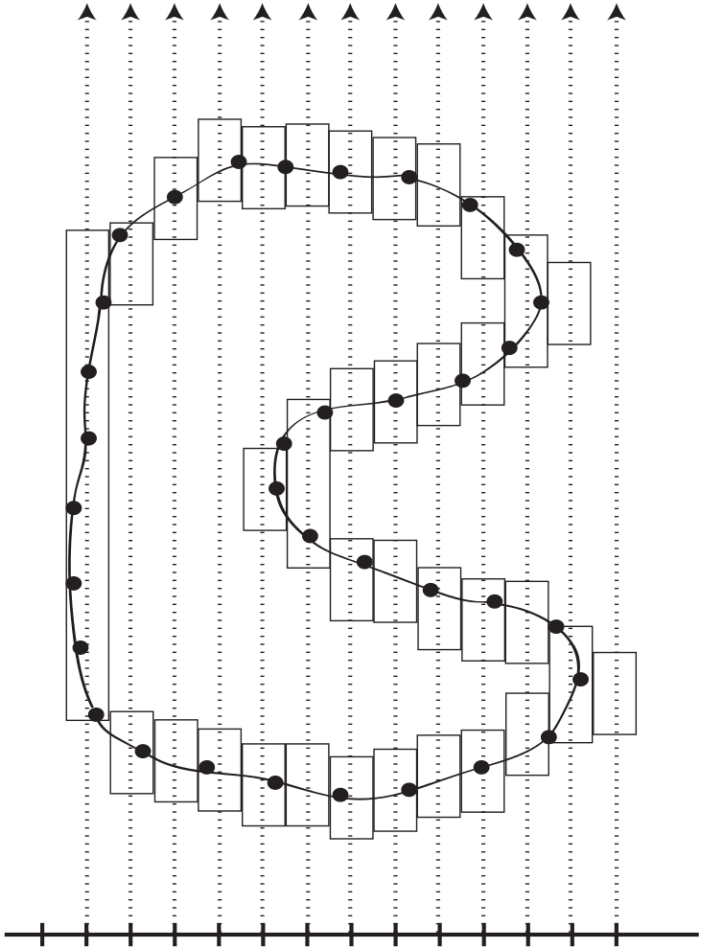
view ray depth intervals

Thickened LDI (layered depth images)

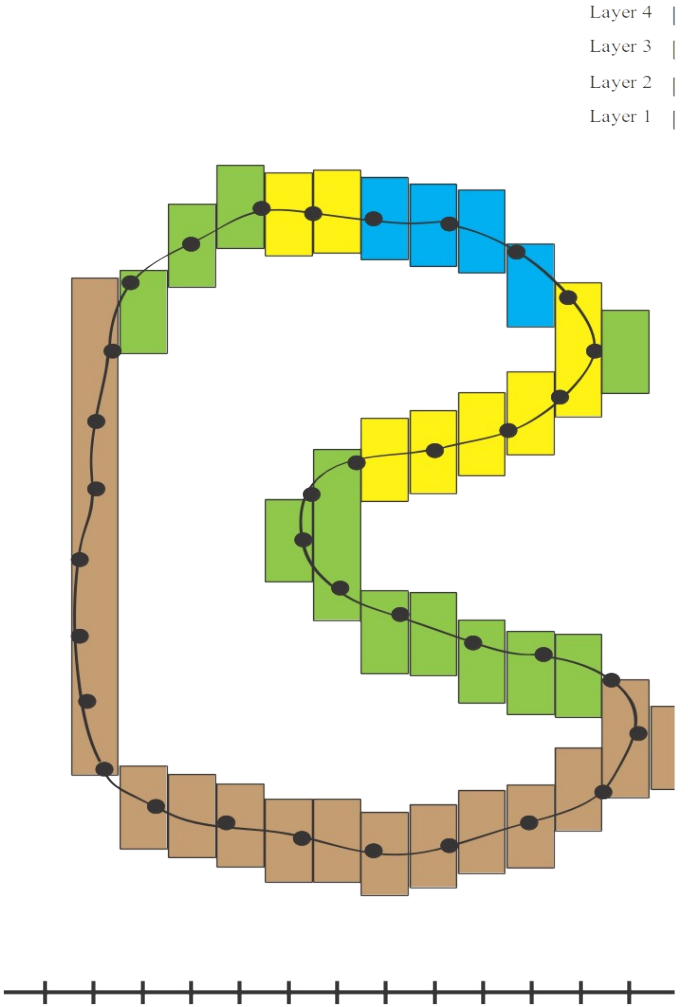


depth intervals

Thickened LDI (layered depth images)

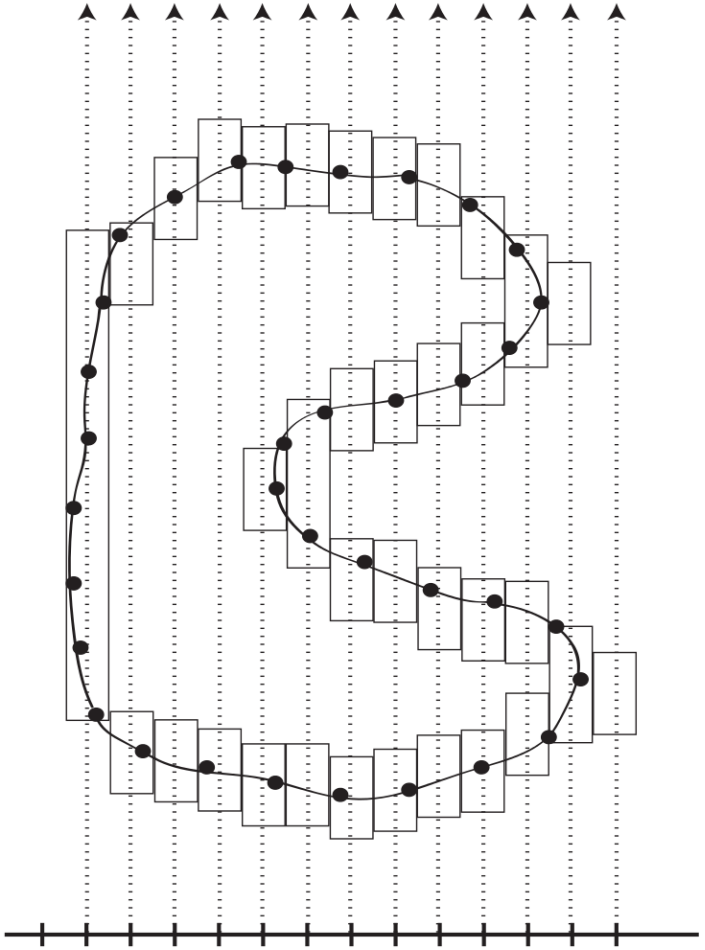


depth intervals

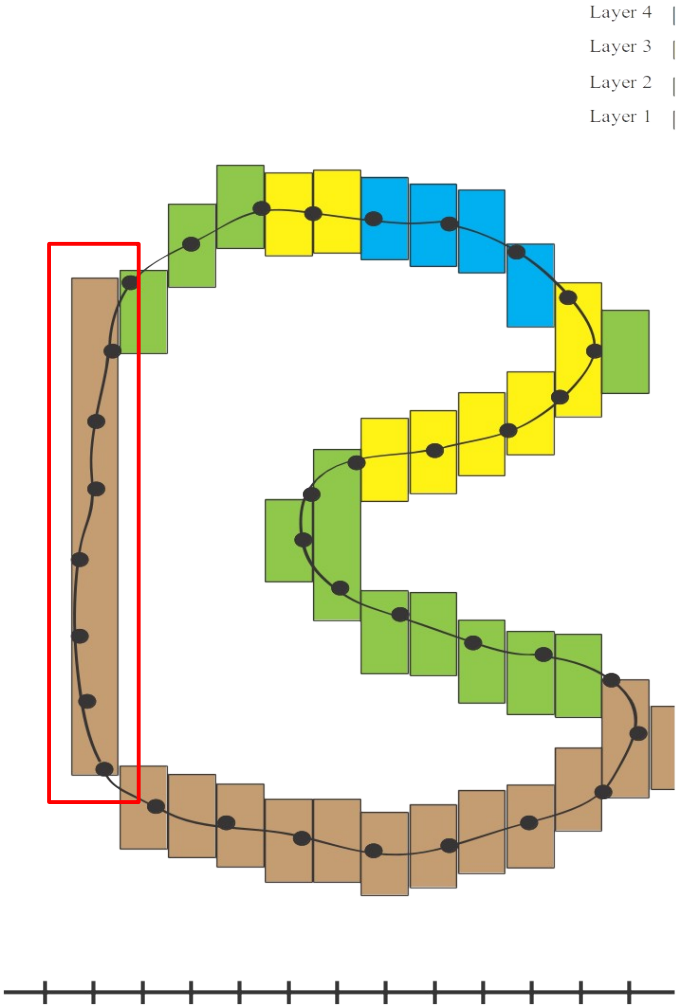


depth image layers

Thickened LDI (layered depth images)

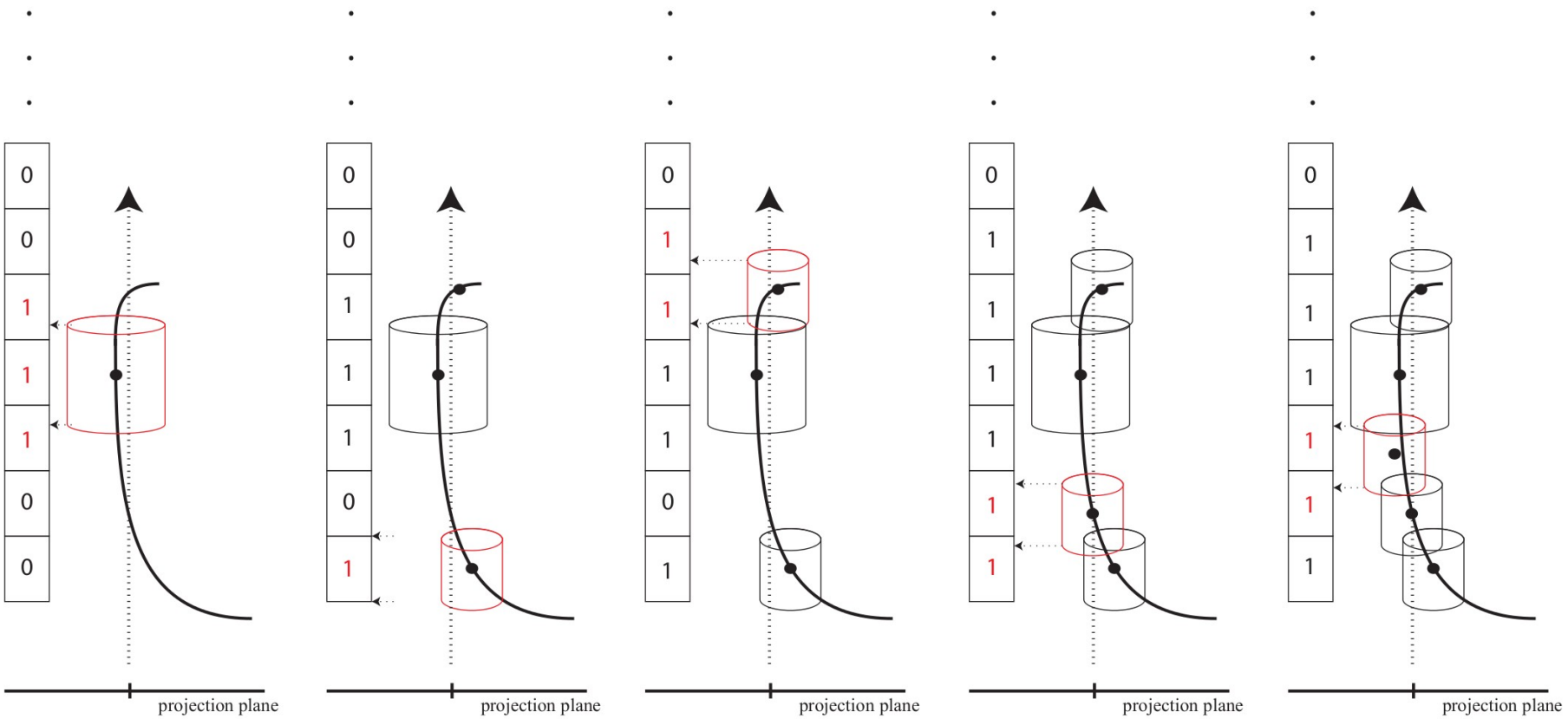


depth intervals



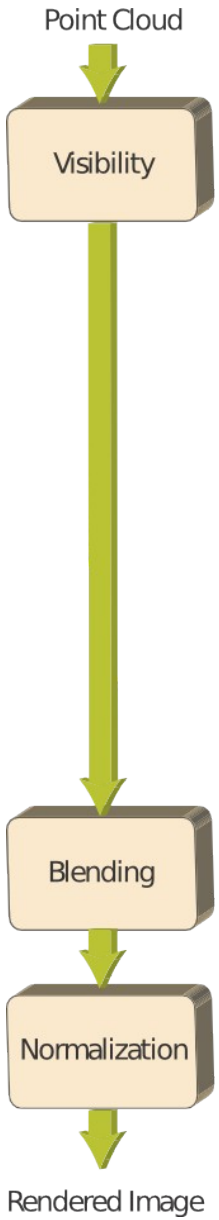
depth image layers

Stacking cylinders into a depth layer

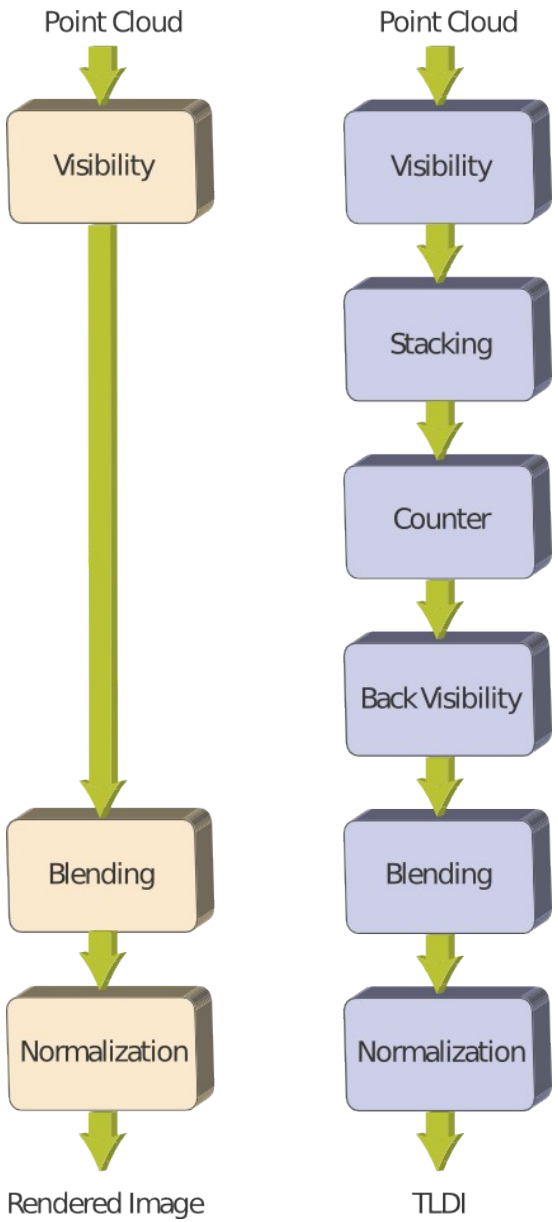


Detect layer connectivity with bit array occupancy

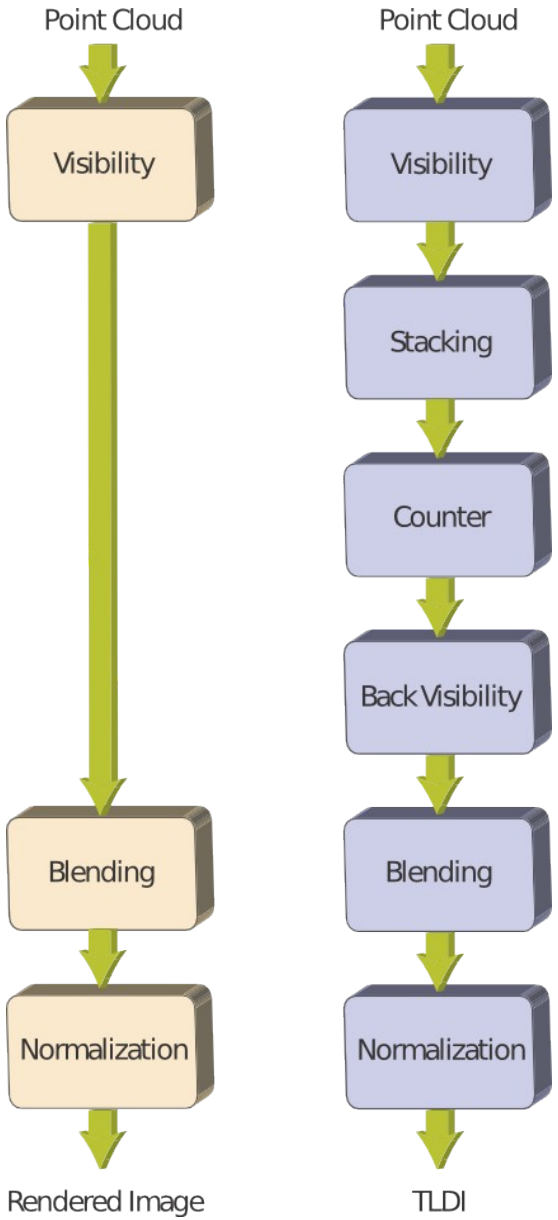
Reuse of point-based rendering pipeline

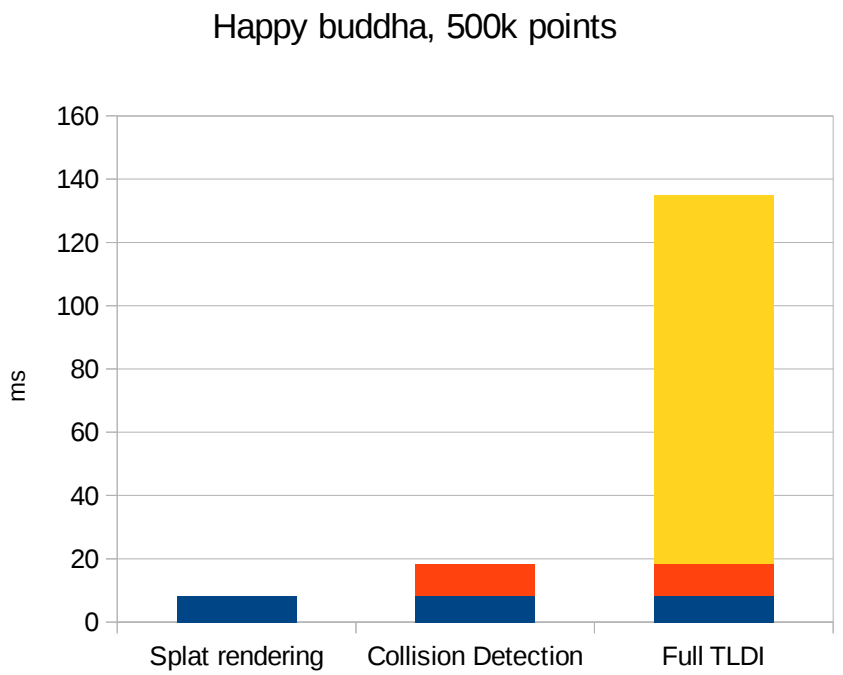


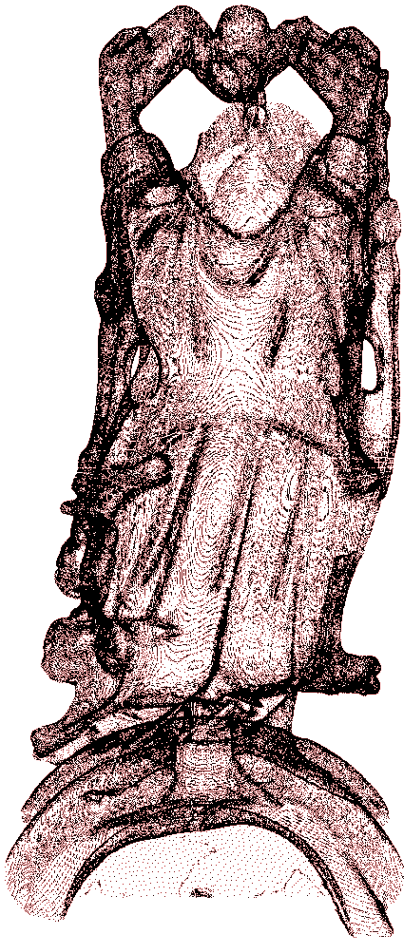
Reuse of point-based rendering pipeline



Reuse of point-based rendering pipeline

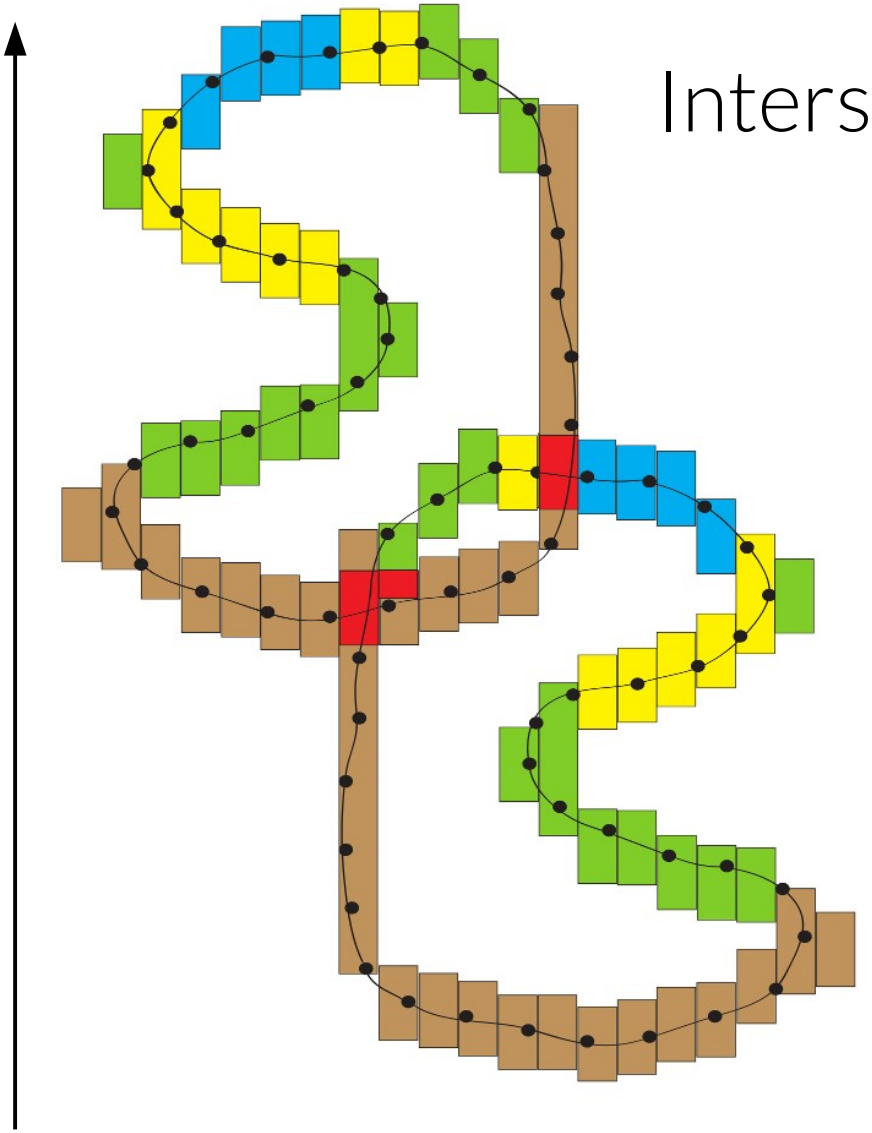






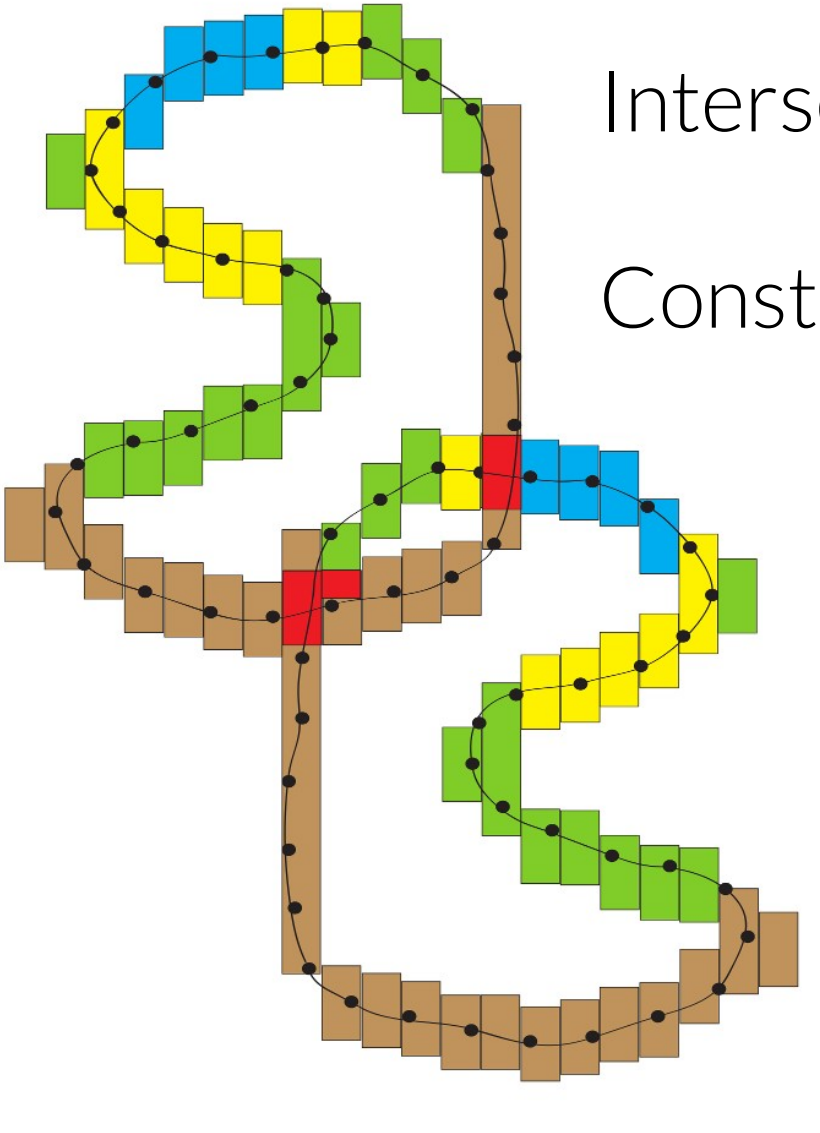
Almost half of run time reused

Application: Collision detection



Intersection of view ray intervals

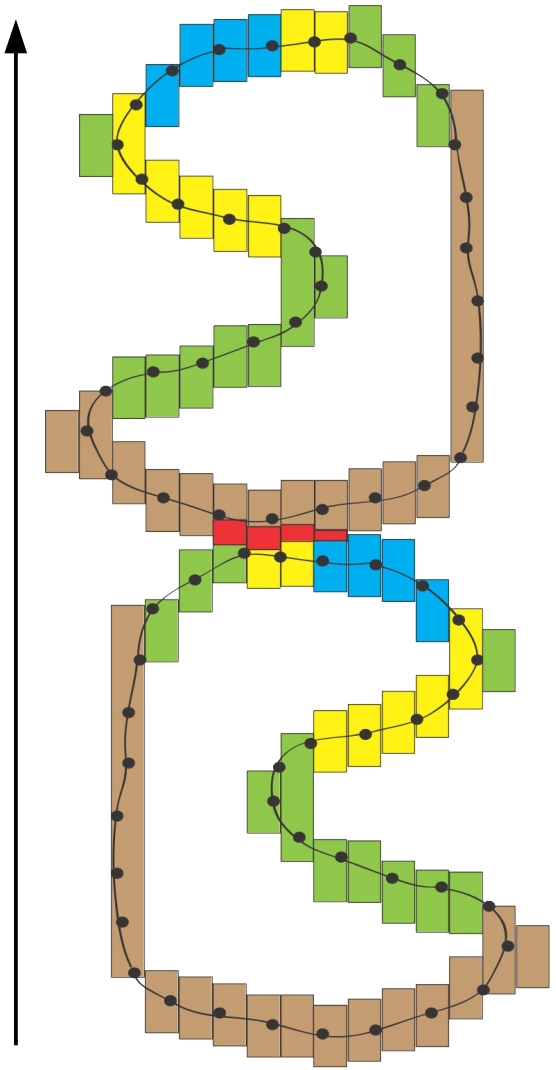
Application: Collision detection



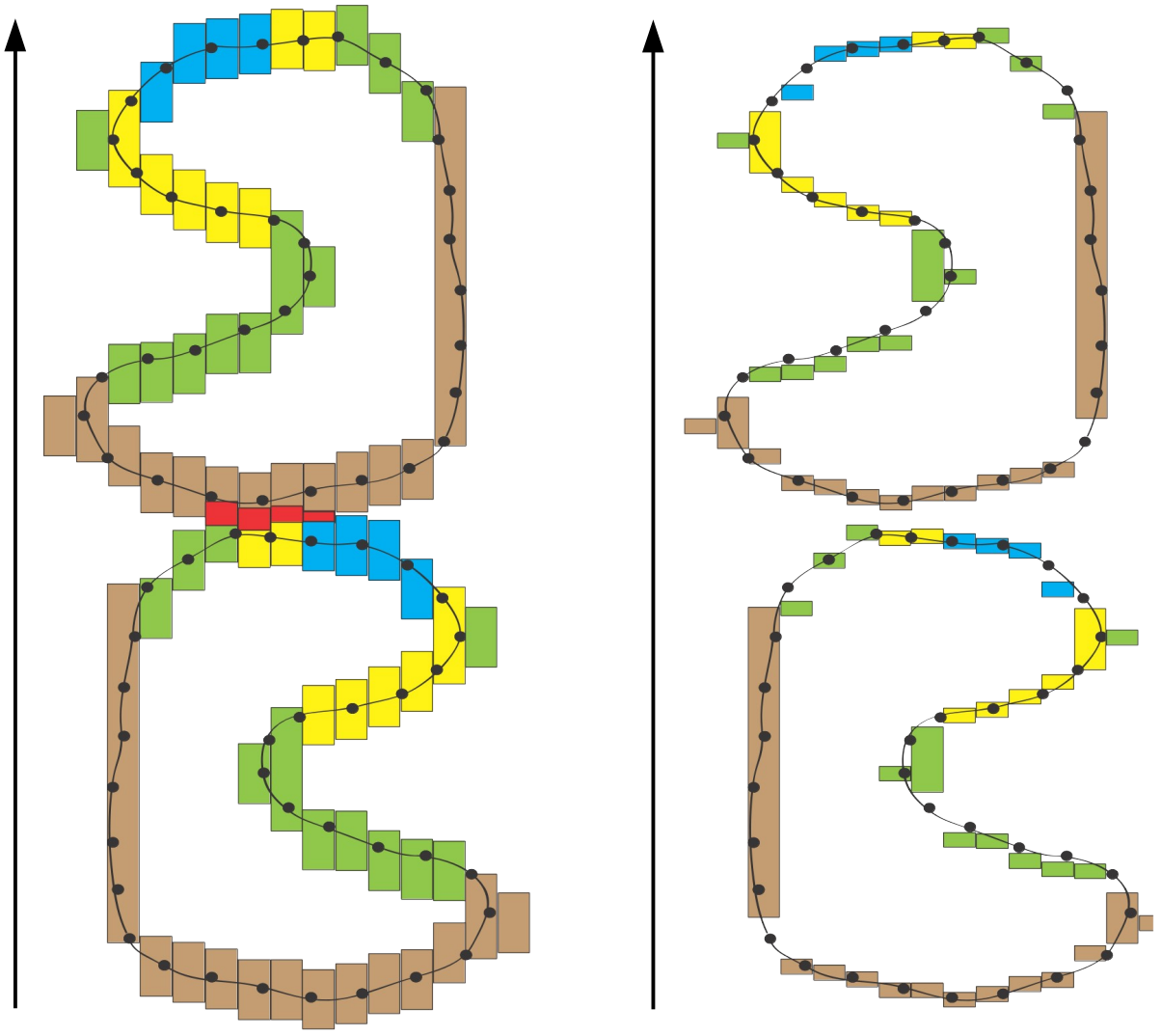
Intersection of view ray intervals

Construct depth layers as needed

False positives

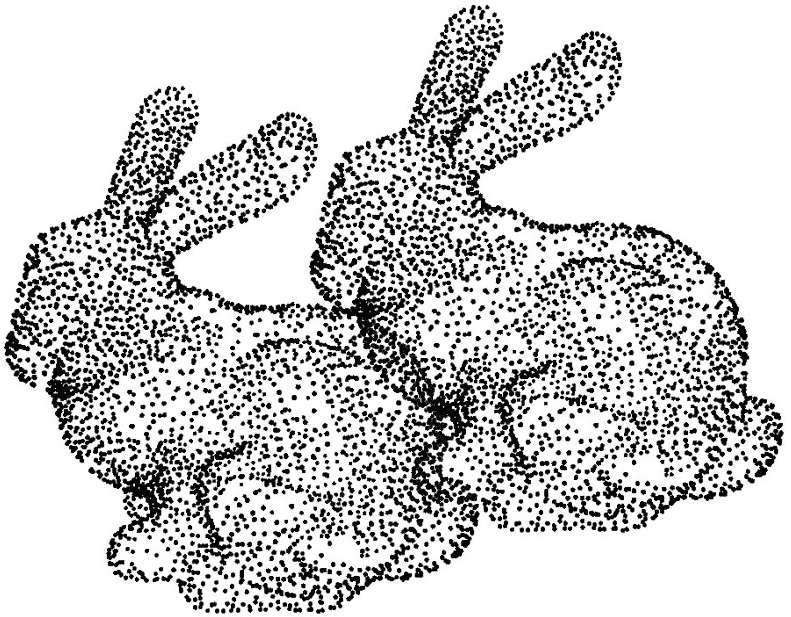


False positives



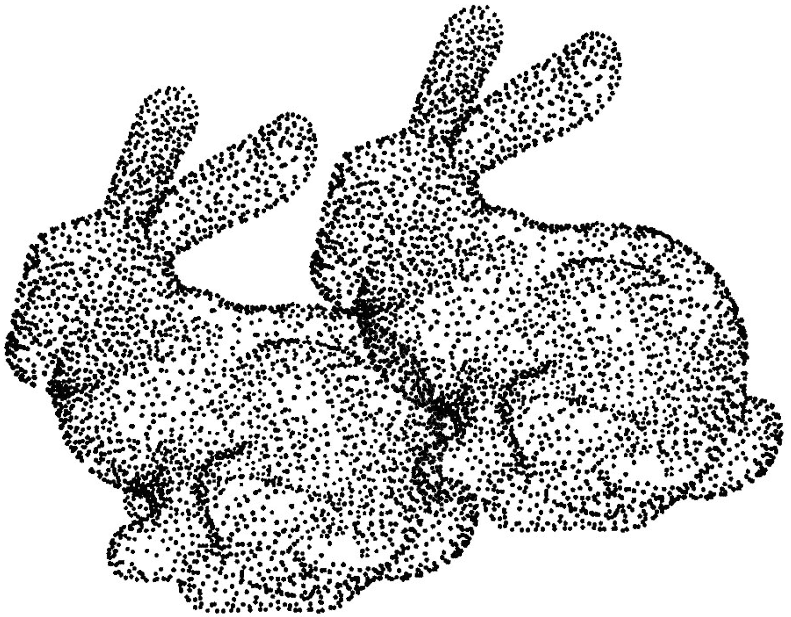
Squash in view direction – but how much?

Determining collision accuracy

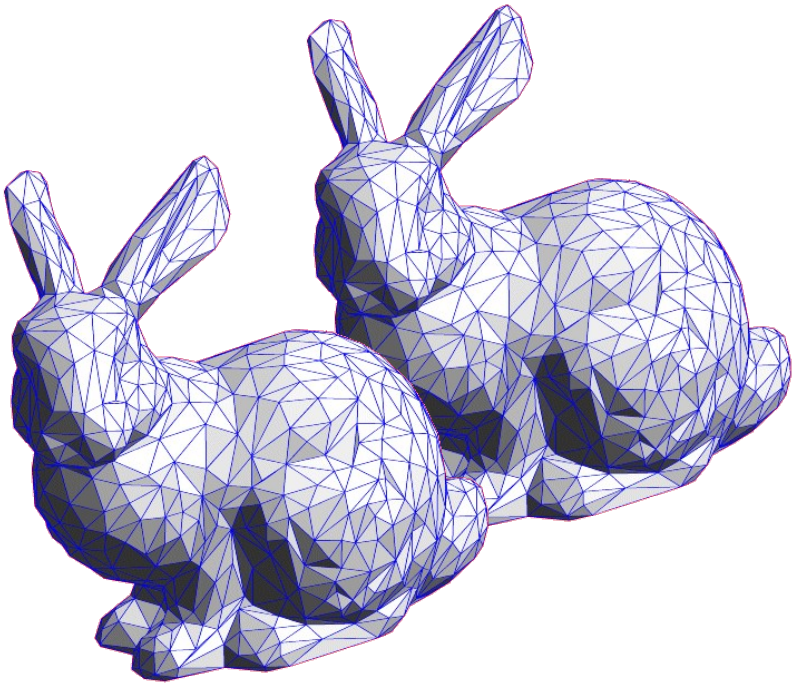


point clouds

Determining collision accuracy

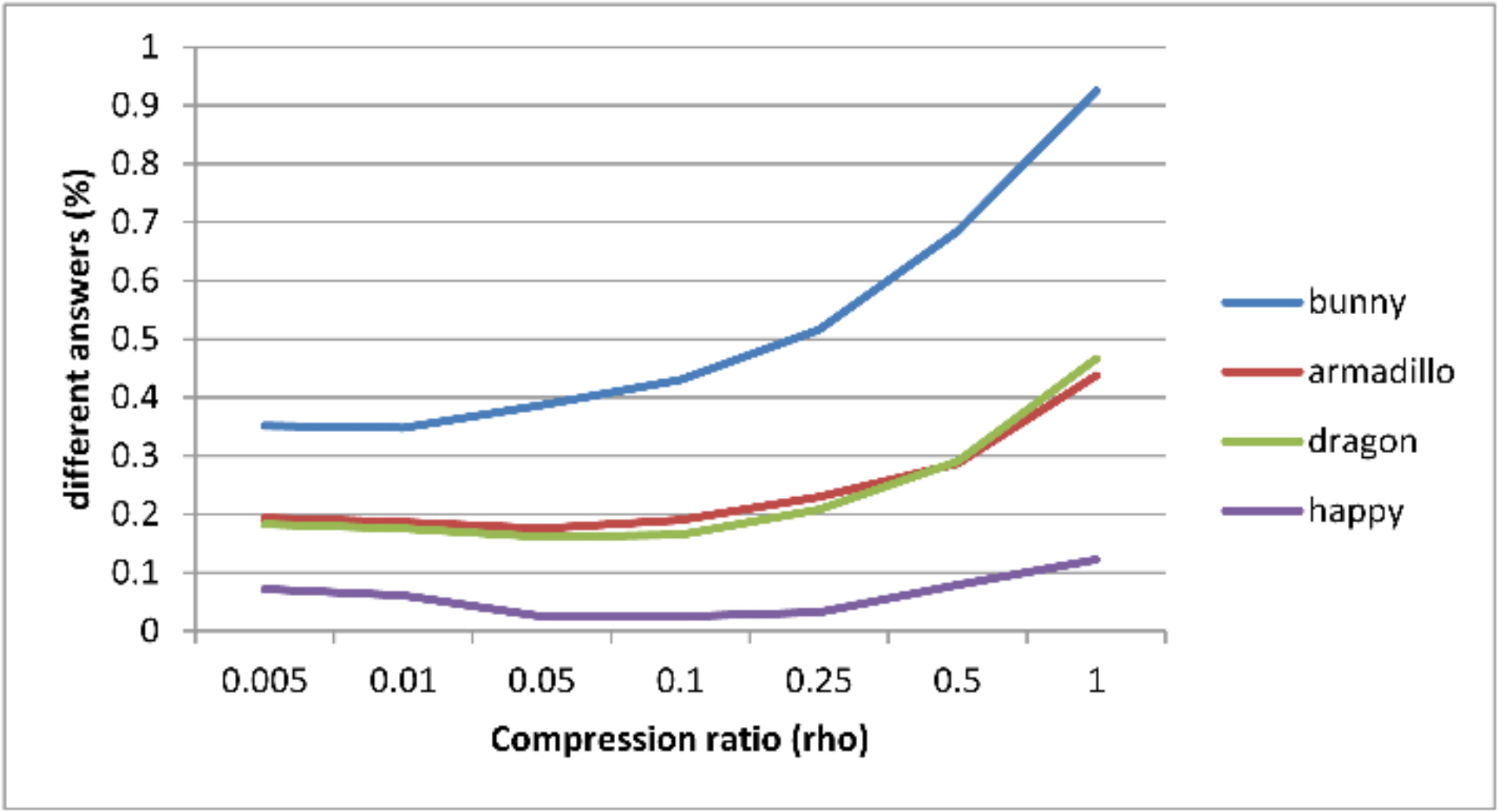


point clouds

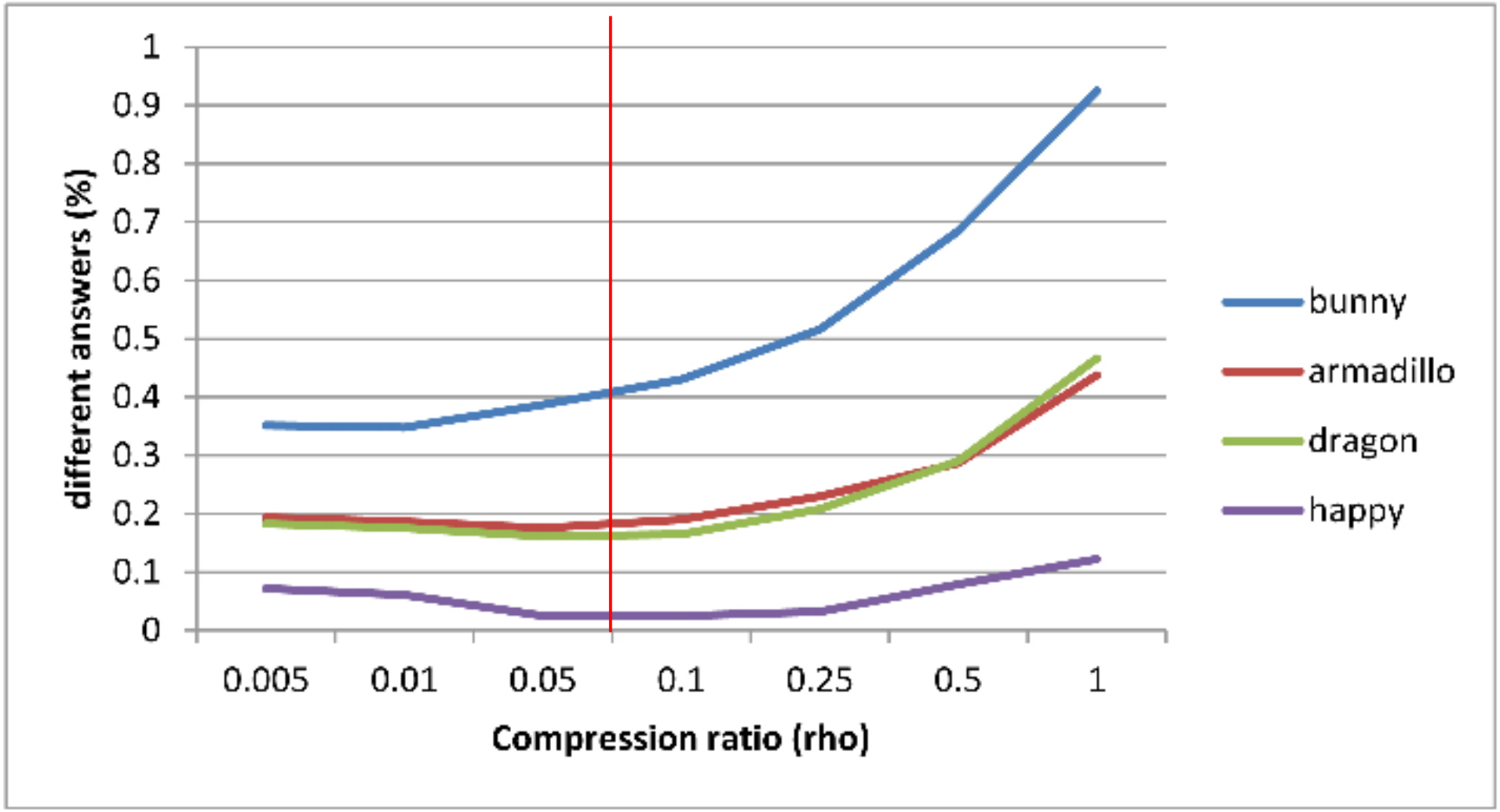


mesh = reference

Squashing the bounding volume

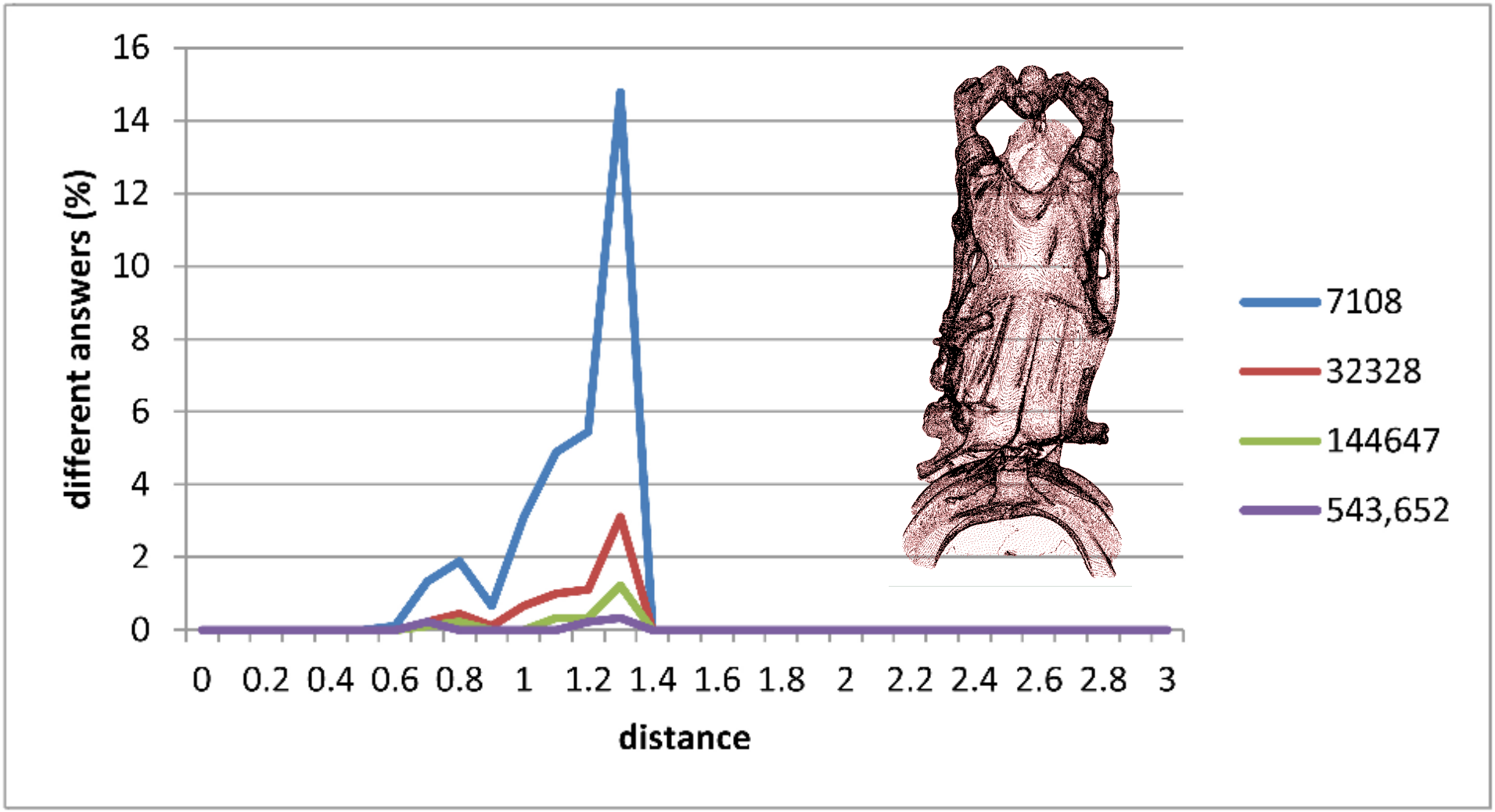


Squashing the bounding volume



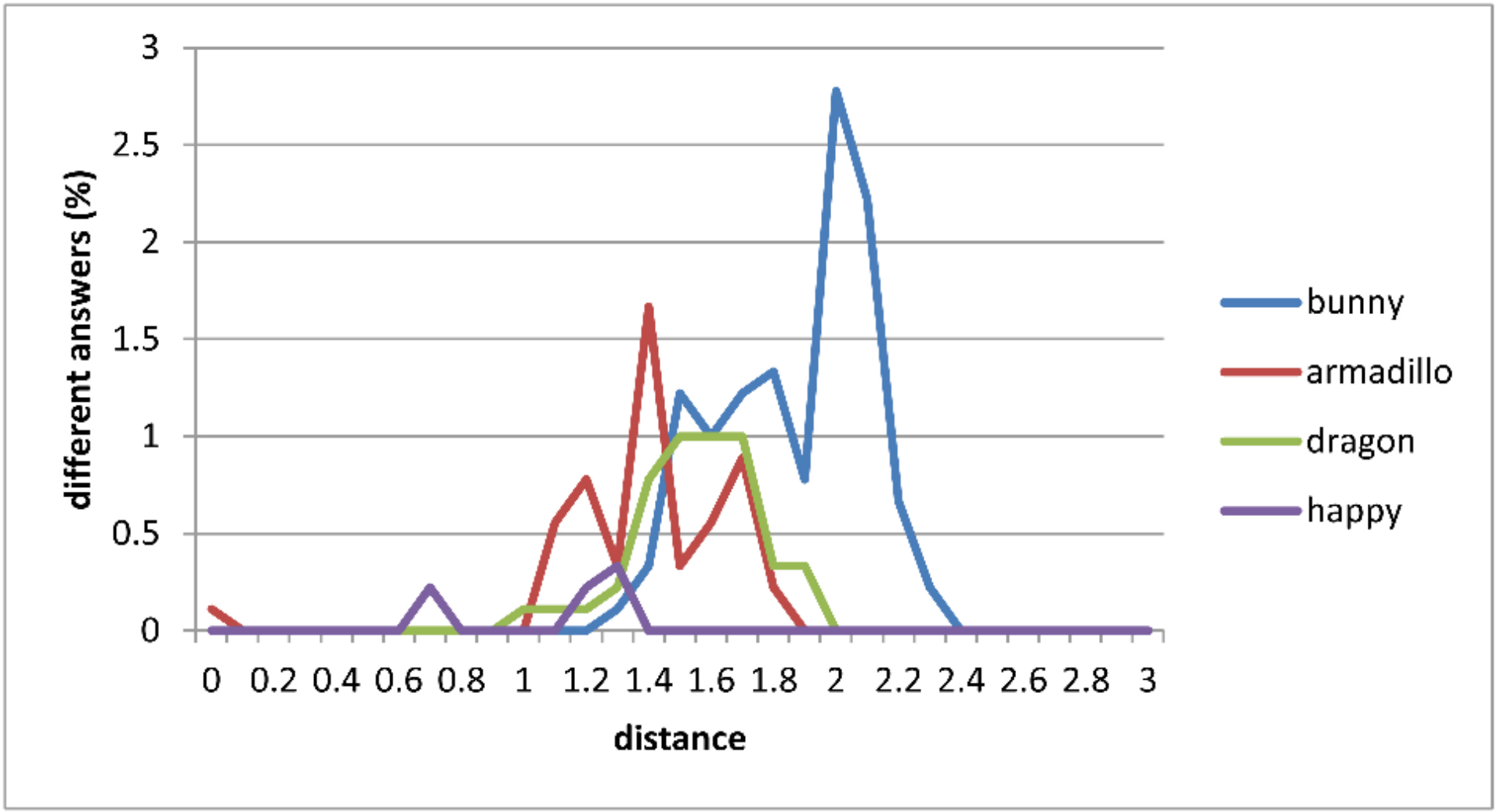
rho=0.05 is good → squashed by factor 20

Result 1: Enhanced accuracy



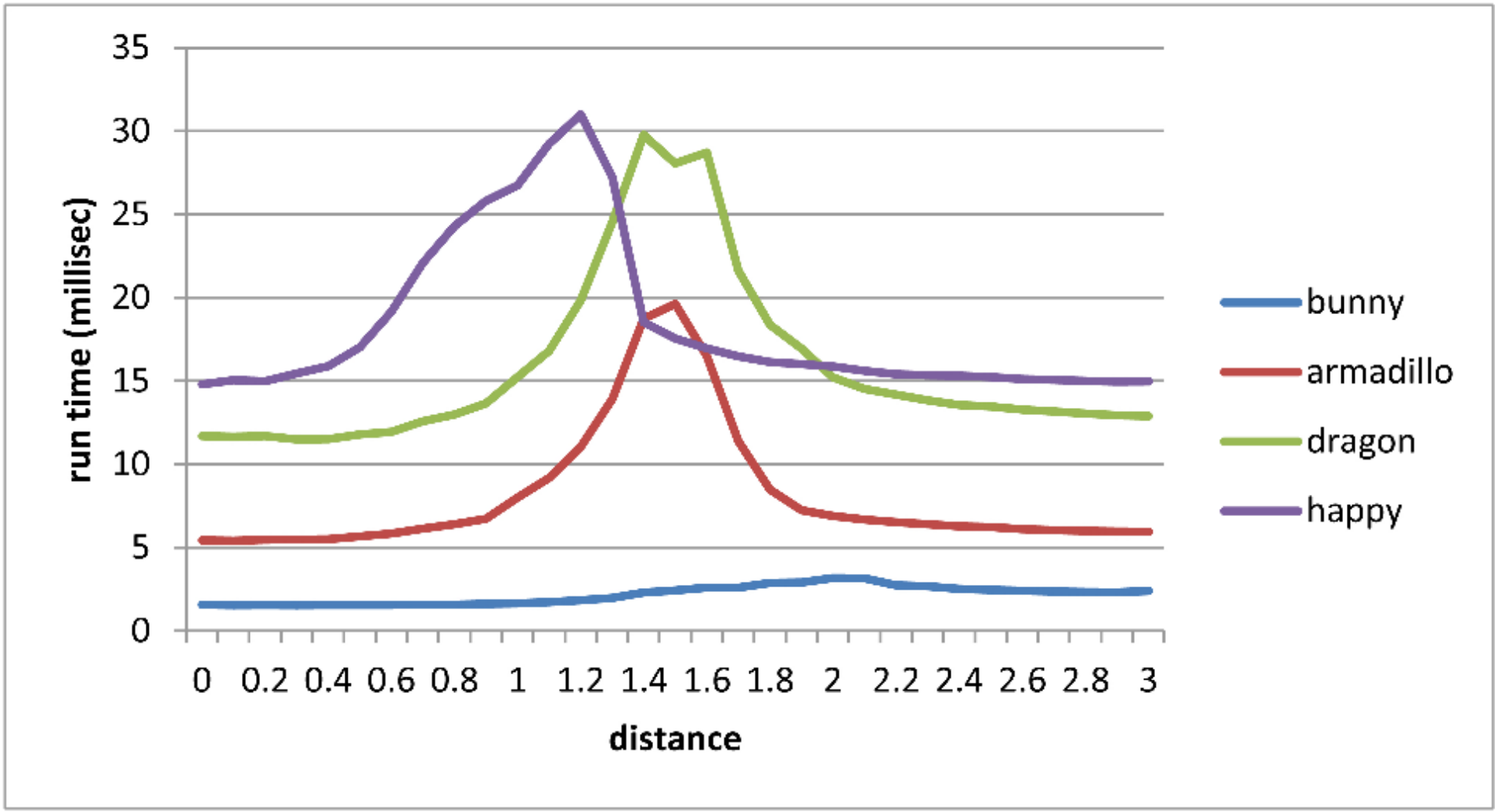
[Zachmann 2002] ~7% accuracy, ours: from 0.3%

Accuracy for different models



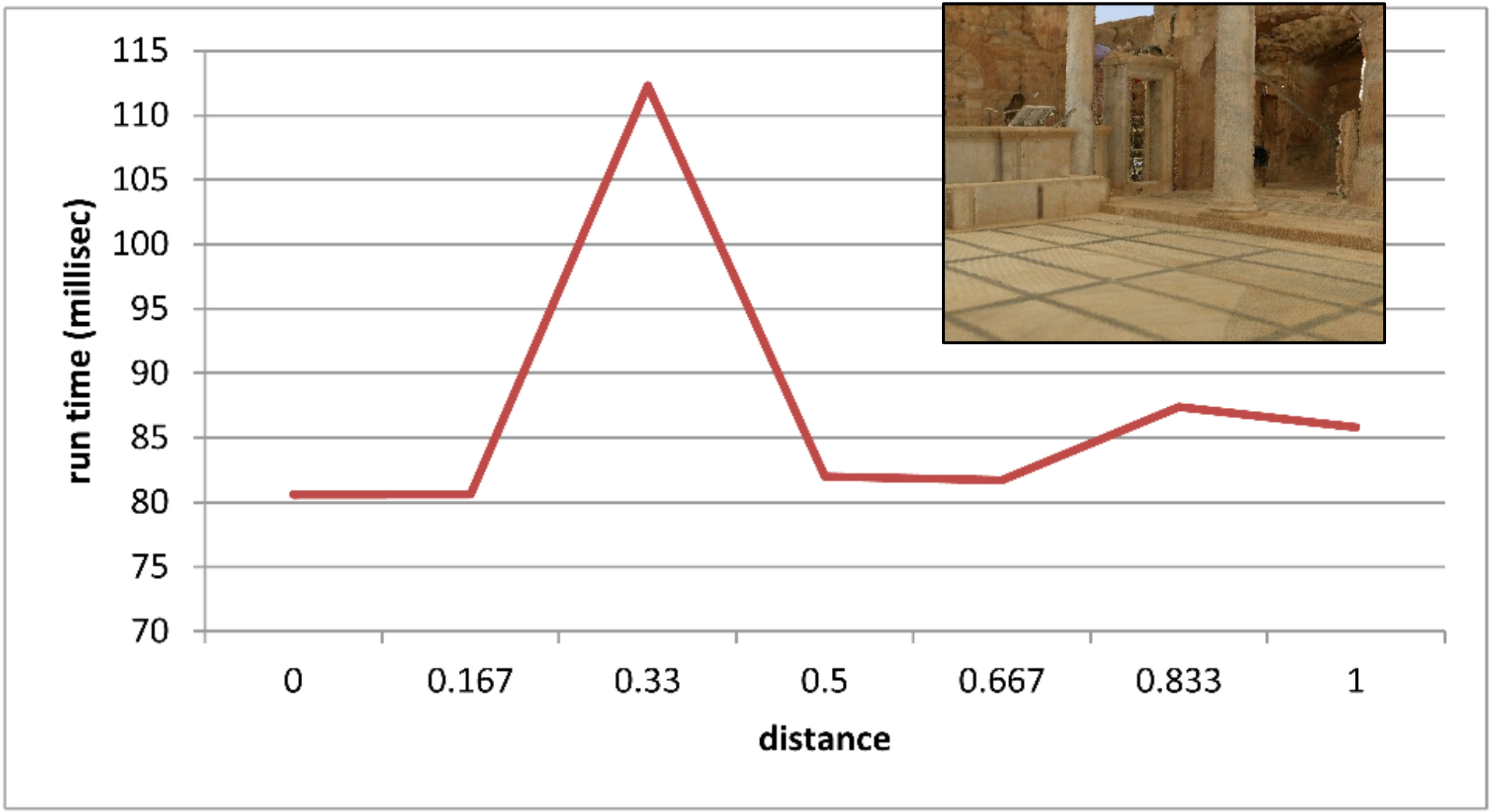
within 0.3-3% of reference

Result 2: Real-time collision detection

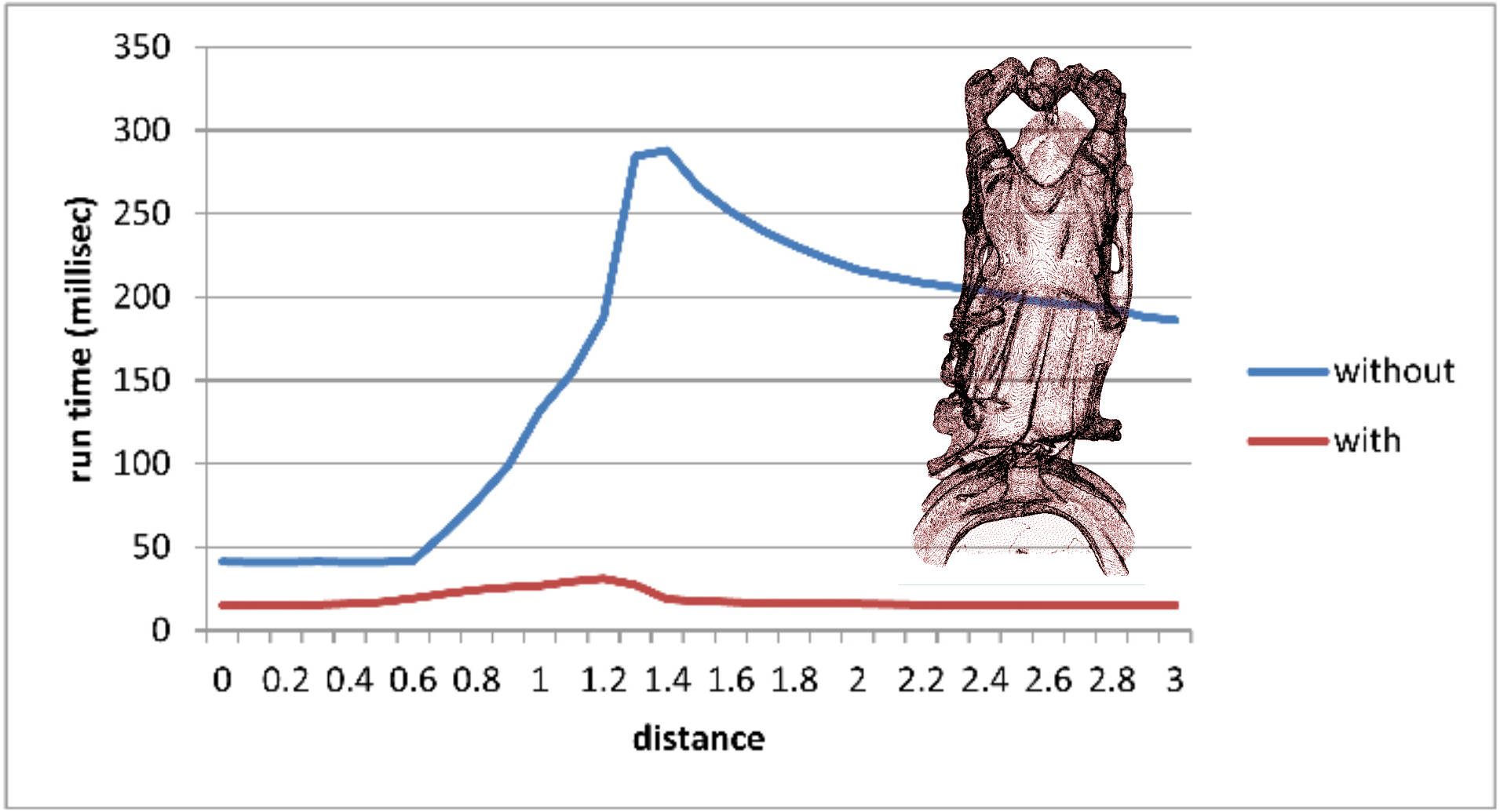


Previously only feasible using probability

Interactive for large models (5M points)



Early rejection test



Test first layer against last layer of other point cloud

Incidental distance queries

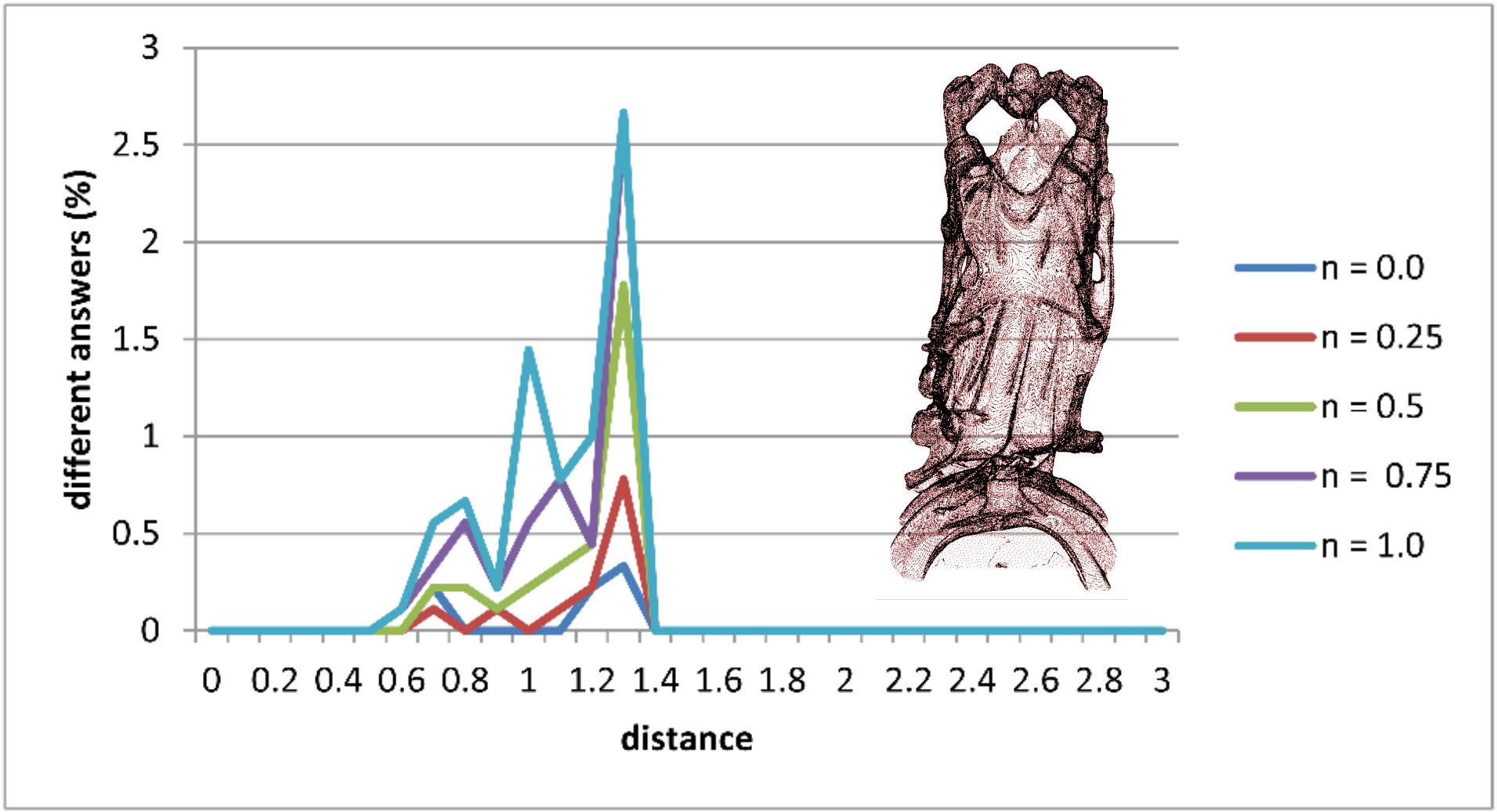
For collision detection:

- For each view ray (pixel), test if intervals overlap

In case of non-collision:

- Keep shortest distance between view ray intervals
  → separation distance in view direction

Result 3: Robust to noise



Test with added uniform Gaussian noise $\sigma = nr_{avg}$

Time complexity

- O(*L*N), *L* = number of layers (depth complexity)
- Very little output-sensitivity measured

Colliding *m*>2 objects, adds factor *m*^2

- But need only construct TLDIs once
- Can also combine small point clouds to reduce *m*

Conclusions

Novel structure bounds surface of dynamic points

- Real-time, accurate, robust to noise
- Potential other applications: everything which benefits from fast surface queries: GI, ray-tracing

Work in progress

- Compact TLDI data structure
- Speed up construction+queries