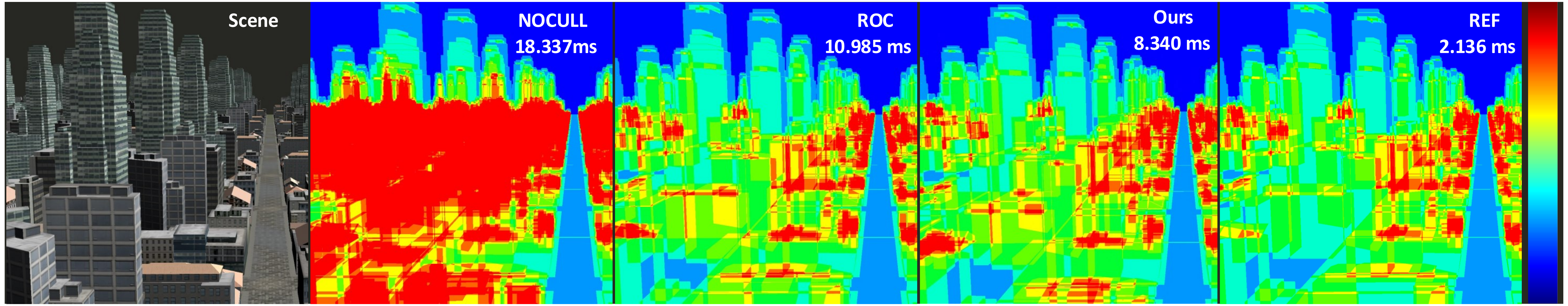
Iterative GPU Occlusion Culling with BVH

Gi Beom Lee Sungkil Lee
Sungkyunkwan University



Motivation

Raster occlusion culling (ROC) requires to render the bounding boxes of all objects not seen in the previous frame to test the visibilities of the occludees. ROC works well up to medium-size scenes, but does not scale well with massive scenes due to the excessive rasterization overhead for the occlusion test.

Challenges

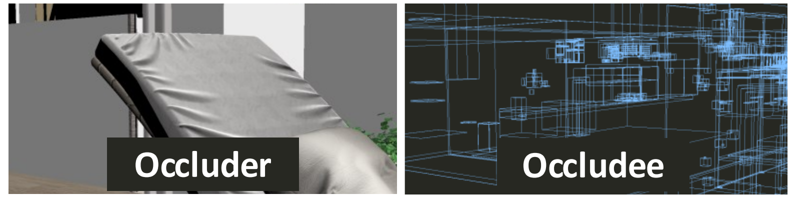
Scalable raster occlusion culling with light-weight test

- Fewer draw calls to determine the visibility of many objects
- Reduction of redundant rasterization for the occlusion culling

Occlusion culling?

A technique that bypasses hidden objects in rendering and thereby accelerates performance.

- Previously visible objects are chosen as *occluders*
- Bounding boxes of the occludee are occlusion-tested



Previous Approaches

CHC++ [Mattausch 2008]

- Hierarchical GPU occlusion query

Raster occlusion culling [Kubisch 2015, NVIDIA 2014]

- Direct writing to a GPU buffer with early Z
- Indirect multidraw to hide read-back latency

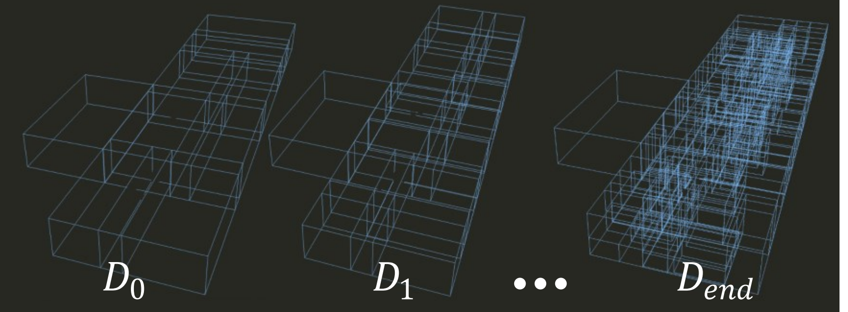
Our Approach

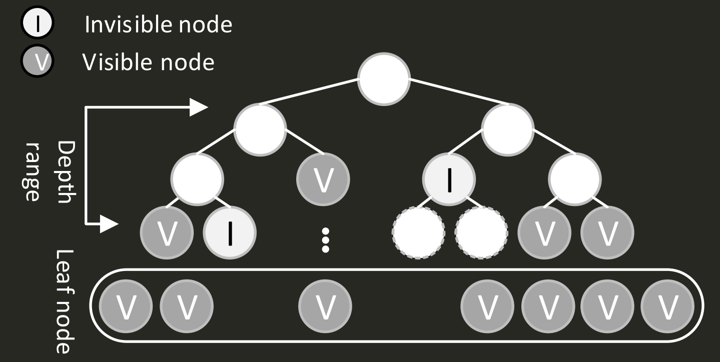
We use a geometry-level BVH to batch-test the visibilities of the occludees with iterative visibility tests through top-down traversal. This allows us to avoid brute-force tests for the individual occludees, achieving real-time performance even for large-scale scenes (more than dozens of thousands objects).

Batch raster culling with fewer drawcalls:

Avoid too many per-object occlusion tests

- Top-down traversal within a pair of top and bottom depths in BVH
- Significant reduction in the number of effective occlusion culling





Algorithm

- The traversal halts for the culled node
- Leaf nodes are marked as a potentially visible occludee
- Visible nodes are tightly packed for the next occlusion test
- All the children in the subtress of visible nodes are marked as potentially visible occludee at the bottom level

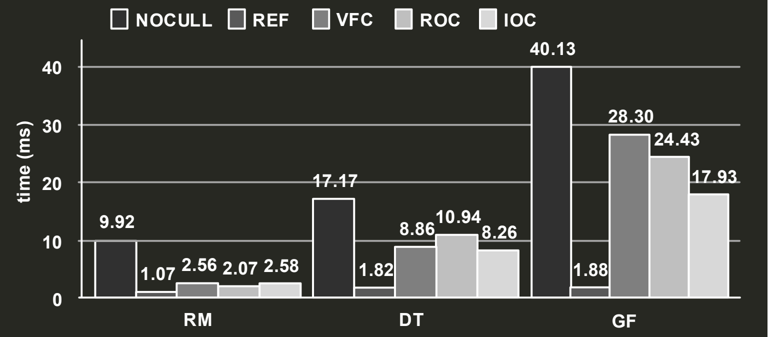
Results

- Testbed:
  Intel i7 machine on GTX 1080 Ti at 1920×1080
- Evaluated techniques:
  no culling (NOCULL), ideal culling (REF), view frustum culling (VFC), raster occlusion culling (ROC), Iterative occlusion culling (IOC)

scene		occlusion test (ms)	occludee rendering (ms)
RM	ROC	0.41	0.40
	IOC	0.61	0.40
DT	ROC	3.47	3.50
	IOC	0.78	3.58
GF	ROC	7.96	7.81
	IOC	1.05	8.04

(a) Culling performance comparison

- 4.5 times and 7.6 times faster in DT and GF in the box test



(b) Rendering performance comparison

- Better performance in the large scale scene with the same culling rate of ROC

limitations

- Manual configuration of the pair of the top and bottom levels for the iteration