

Ray Distribution to Parallel Batching-based Updates

Youngsun Kwon and Sung-eui Yoon

KAIST, South Korea

ICRA 2017 Workshop on
Robotics and Vehicular Technologies for Self-Driving Cars



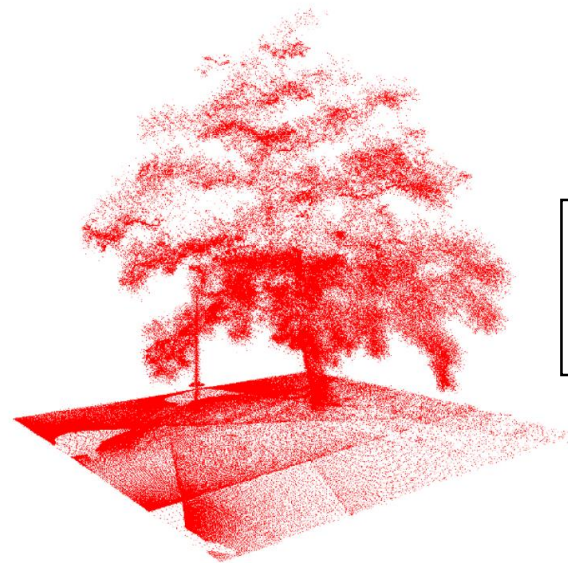
ICRA2017

May 29 - June 3, 2017 • Singapore

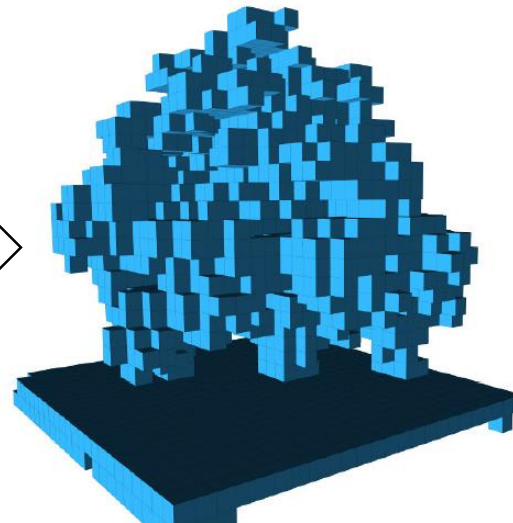
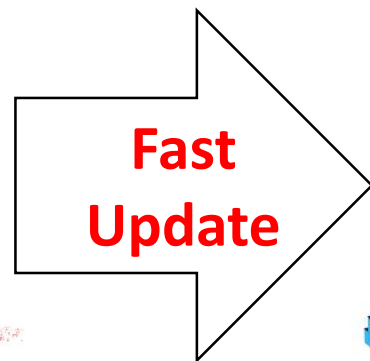
KAIST

Motivation

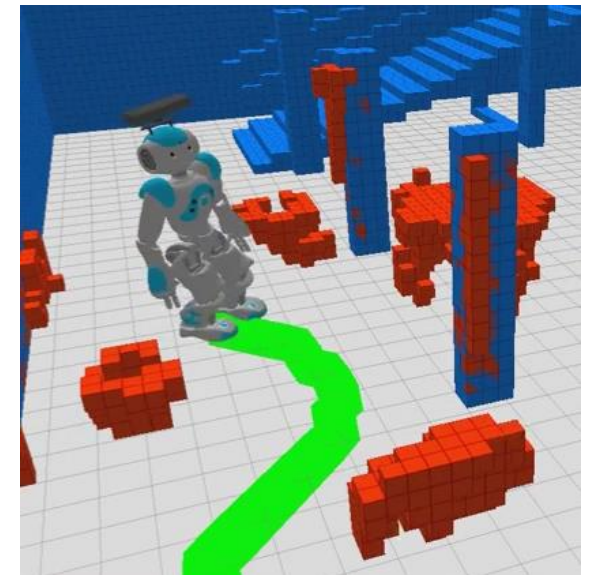
- It is essential to update map representation in real-time
 - Robot as well as vehicle should react to dynamic environment



Point clouds



Map representation
(grids or octrees)

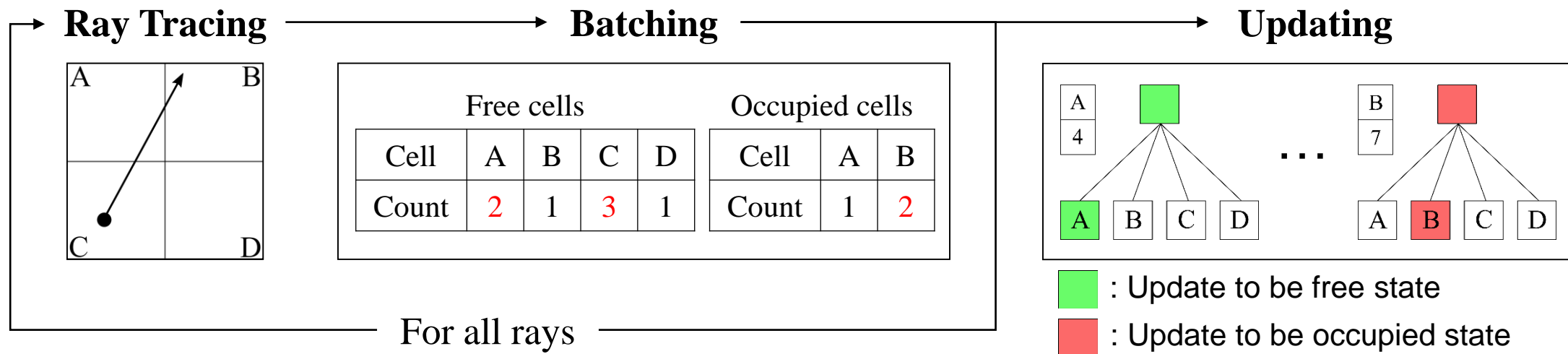


Applications:
e.g. Motion Planning

Prior Work

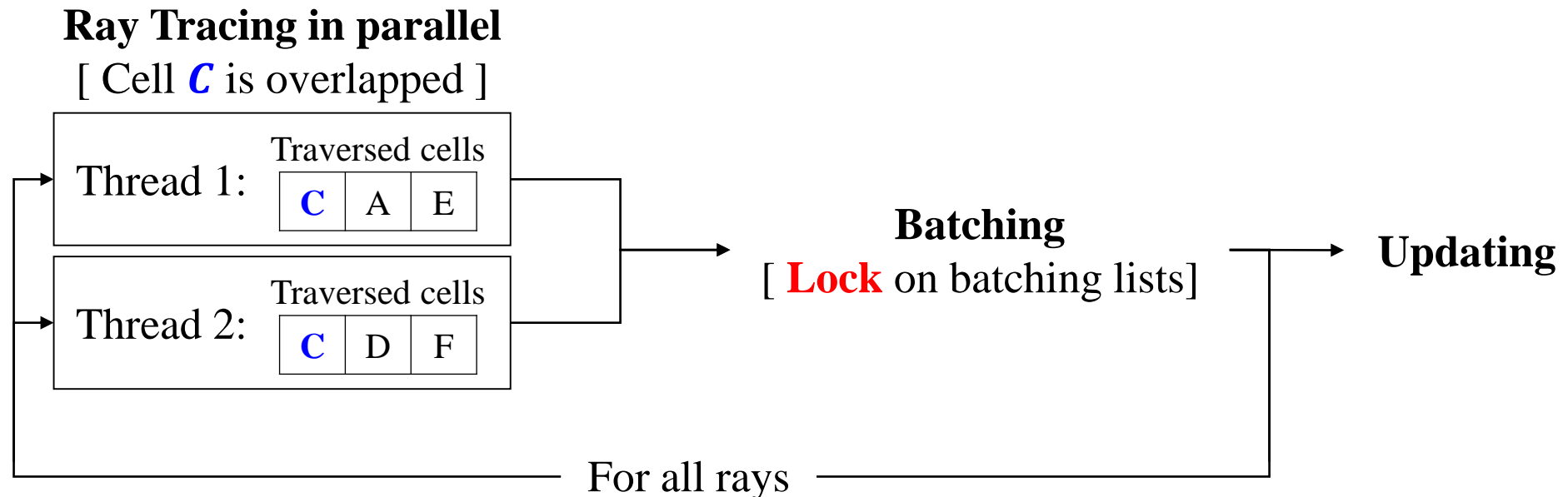
- **Batch based updates**

- **Ray Tracing** is to find a set of cells that a ray traverses on map
- **Batching** is to count how many rays traverse a cell
- **Updating** is to update occupancy probabilities of batched cells as well as their parent nodes of maps



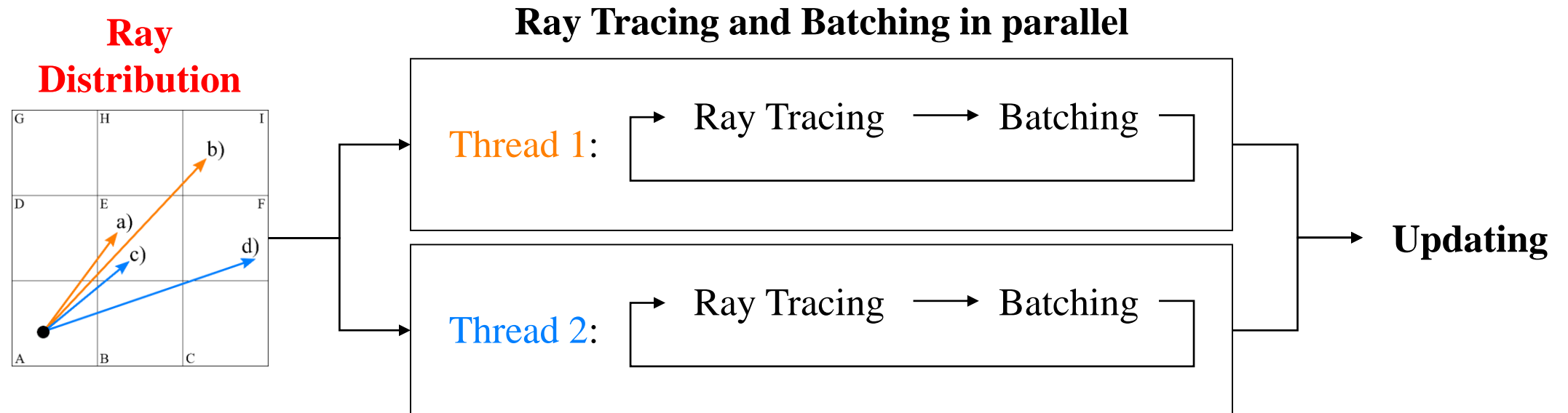
Prior Work

- **Batch based updates in parallel manner**
 - Ray tracing can be processed in multi-threads, but batching requires **lock** on batching list to prevent a concurrent counting from different threads
 - **“Locking”** is a bottleneck of multi-threading



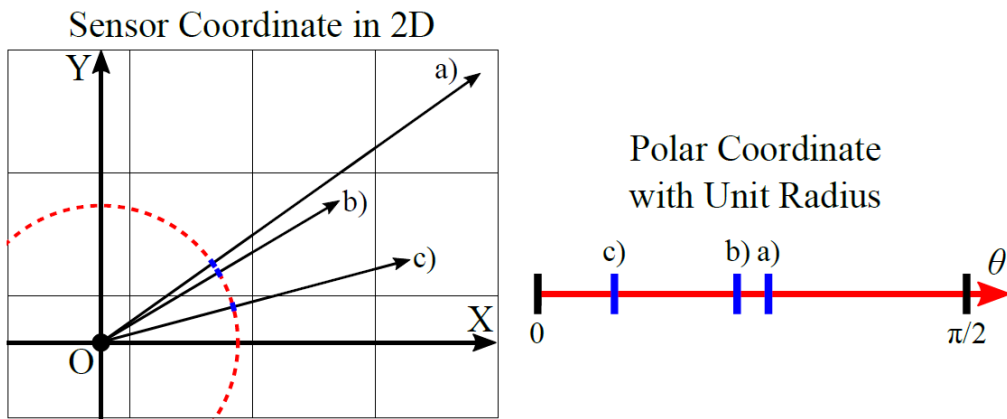
Our approach

- **A novel ray distribution method to parallel batching**
 - Our method distributes the rays into threads for exploiting the high performance of multi-threading [**Lock-free**]
 - Each thread has own Ray Tracing and Batching for rays distributed to it = we approve of batching some overlapped cells among threads

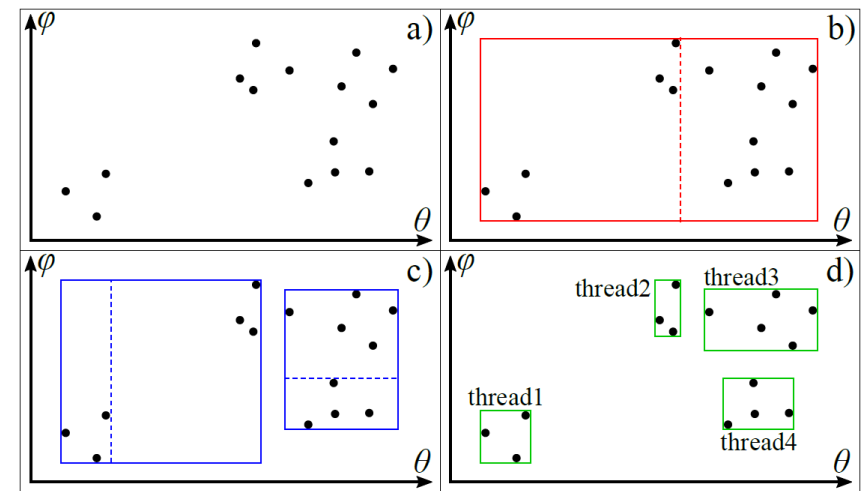


Our approach

- **K-D tree based distribution by considering workloads**
 - Minimize the number of overlapped cells among threads
 - = Cluster the points neighboring in spherical coordinate using K-D tree
 - Distribute rays as each threads has the same workload
 - = Apply our definition of workload to criterion for partitioning on k-d tree

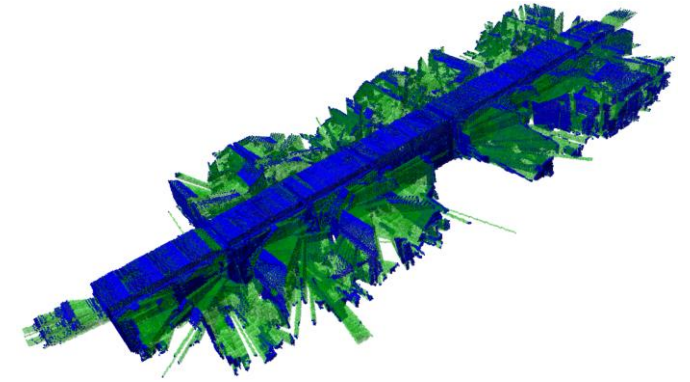


Map a ray in sensor coordinate onto spherical coordinate with unit radius



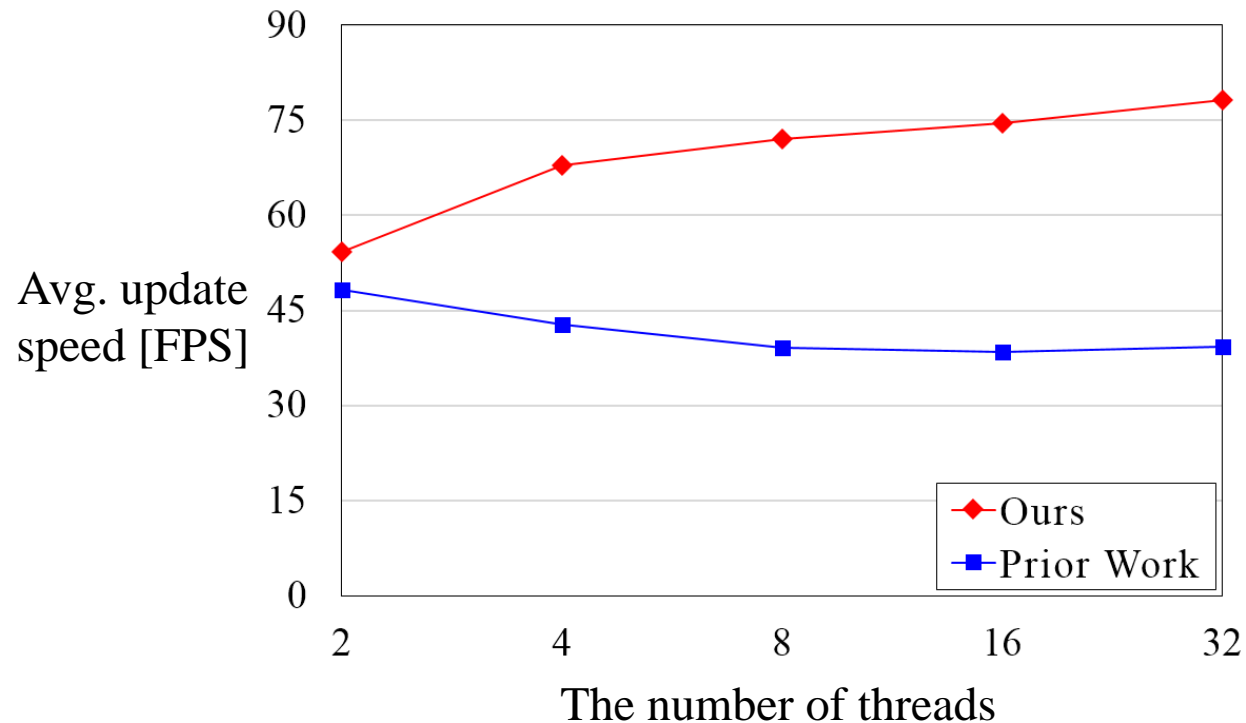
Cluster points in spherical coordinate using K-D tree partitioning with workload balancing

Main Result – Indoor Scene

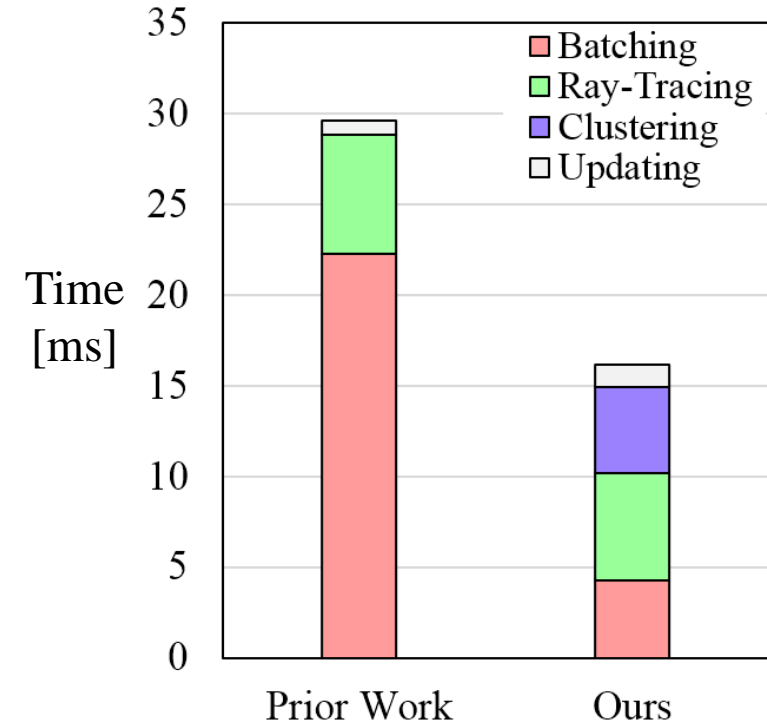


- Improve overall performance **1.8 times**
 - Thanks to **5.2 times** performance improvement on Batching process

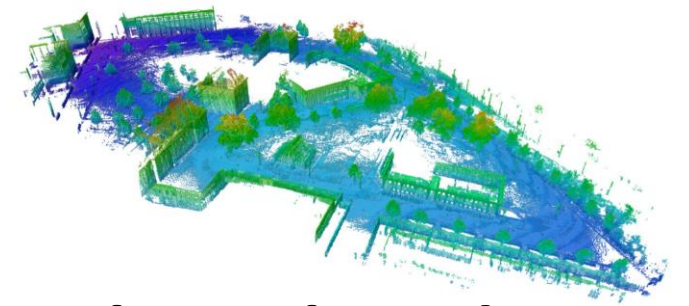
Overall performance on OctoMap
with 0.6m resolution



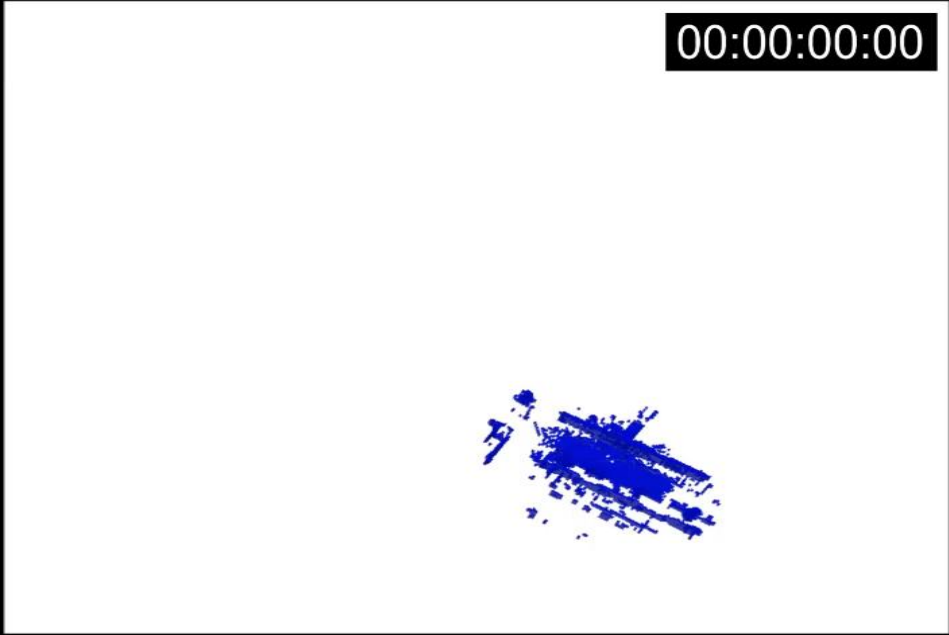
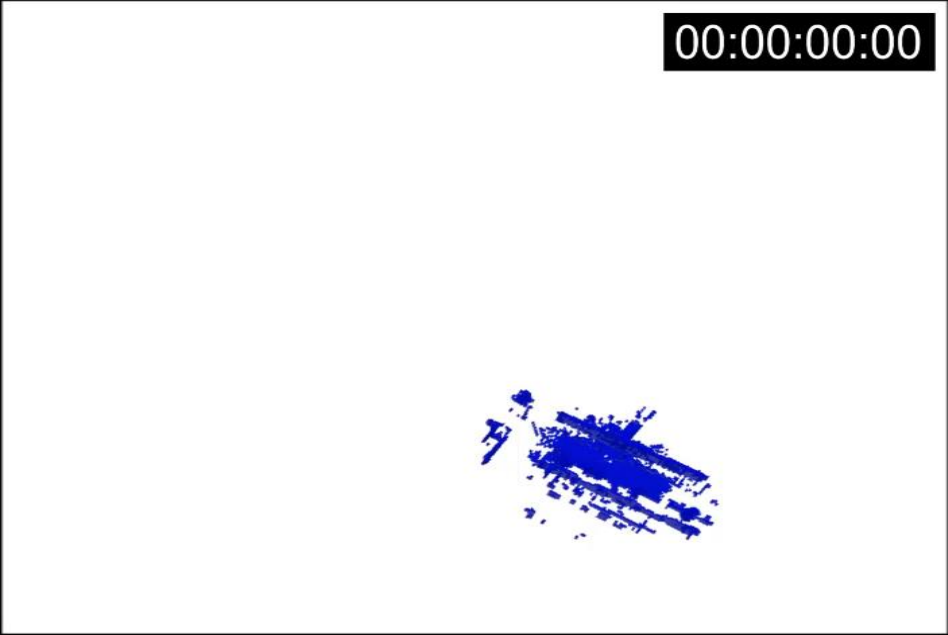
Time breakdown on OctoMap
with 0.6m resolution and 8-threads



Main Result – Outdoor Scene



- Enable **1.9 times** on performance improvement with 32-threads

Outdoor Dataset	
OctoMap with 0.6m resolution	
Prior Work (Batch, 32-threads)	Ours (32-threads)
 <p>00:00:00:00</p>	 <p>00:00:00:00</p>