3D Perception for Mobile Manipulation with OctoMap

http://octomap.github.io

Armin Hornung

UNI FREIBURG

Humanoid Robots Lab
University of Freiburg

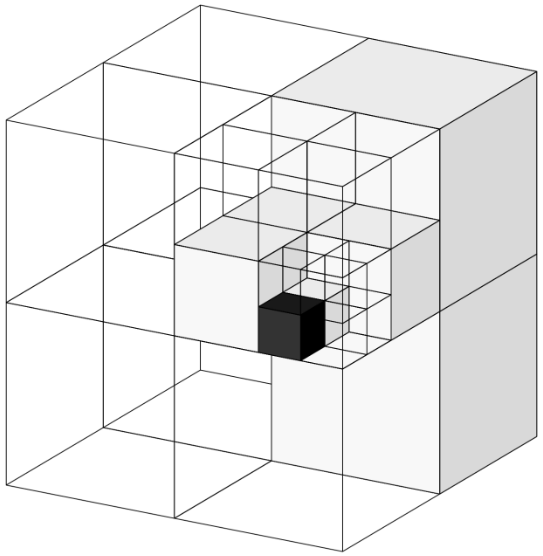
Joint work with K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard

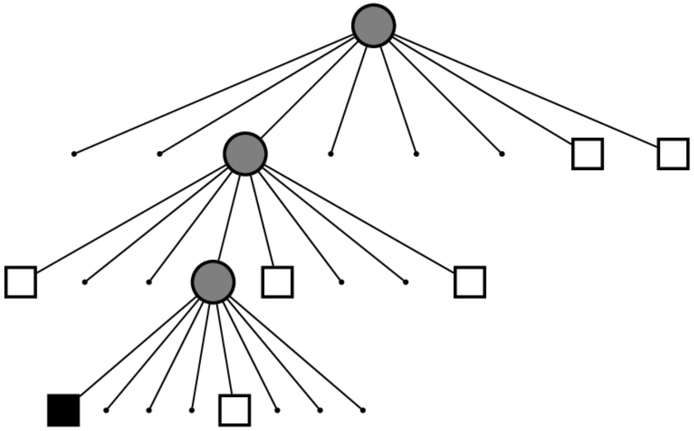
3D Environment Representation for Mobile Manipulation

- Integrate and store multiple measurements
- Update map during manipulation
- Reason about free and unseen areas
- Memory-efficiency

Octree

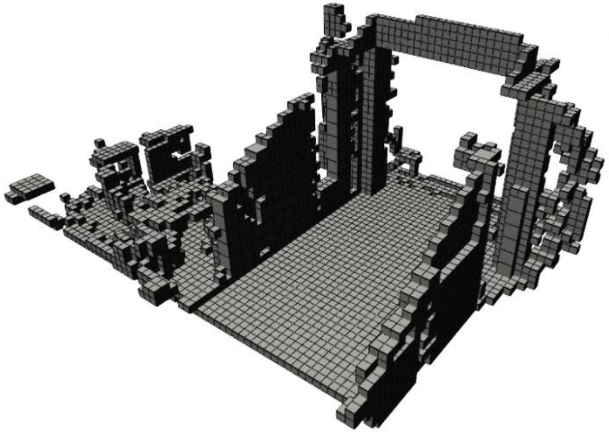
- Tree-based data structure
- Recursive subdivision of space into octants
- Volumes allocated as needed
- Multi-resolution

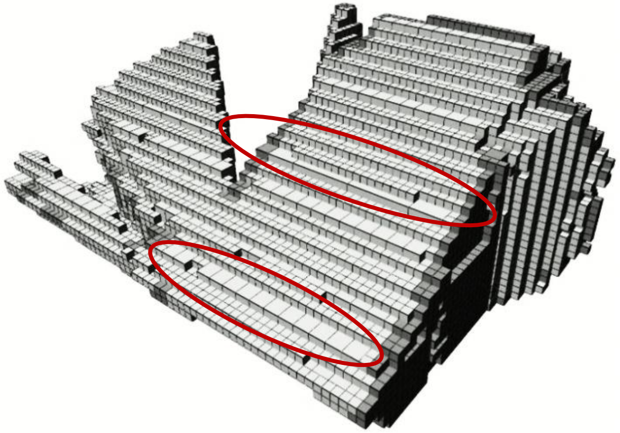




OctoMap Framework

- Based on octrees
- Probabilistic representation of occupancy
- Volumetric model of occupied and free space
- Supports multi-resolution map queries
- Lossless compression
- Compact map files





OctoMap Framework

- Open source (BSD) implementation as C++ library available at octomap.github.io

- Pre-built debian packages for ROS *electric* to *hydro*, see www.ros.org/wiki/octomap

- ROS integration in packages octomap_ros, octomap_msgs, and octomap_server

- Collision checks in FCL / MoveIt!

Map Update

- Occupancy modeled as recursive binary Bayes filter [Moravec '85]

$$P(n \mid z_{1:t}) = \left[1 + \frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}$$

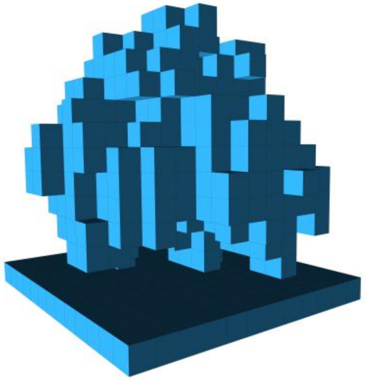
- Efficient update using log-odds

$$L(n \mid z_{1:t}) \;=\; L(n \mid z_{1:t-1}) + L(n \mid z_t)$$

Map Update

- Clamping policy ensures updatability [Yguel '07]

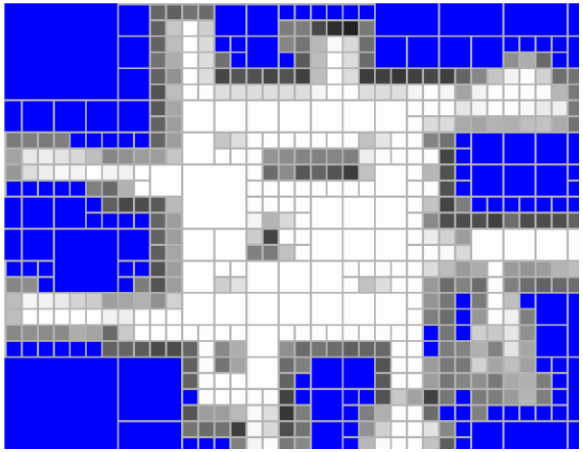
  $L(n) \in [l_{\text{min}}, l_{\text{max}}]$



- Update of inner nodes enables multi-resolution queries

  $L(n) = \max_{i=1..8} L(n_i)$

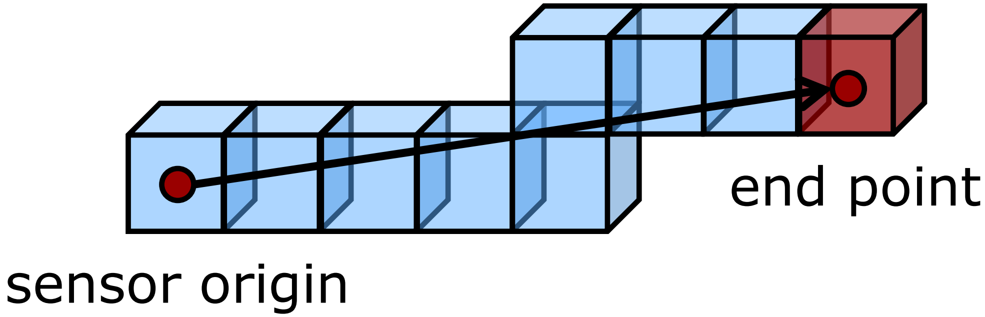
- Compression by pruning a node's identical children



[Kraetzschmar '04]

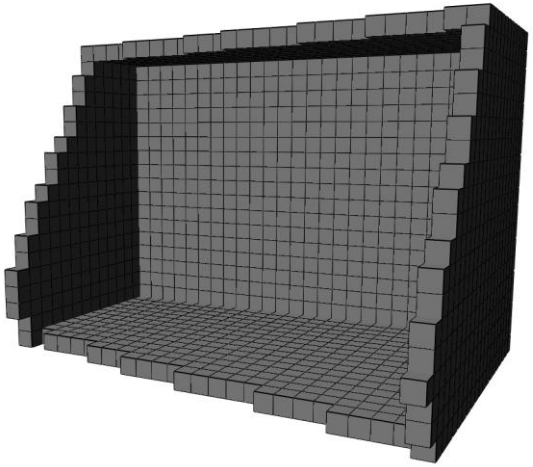
Sensor Model for Single Rays

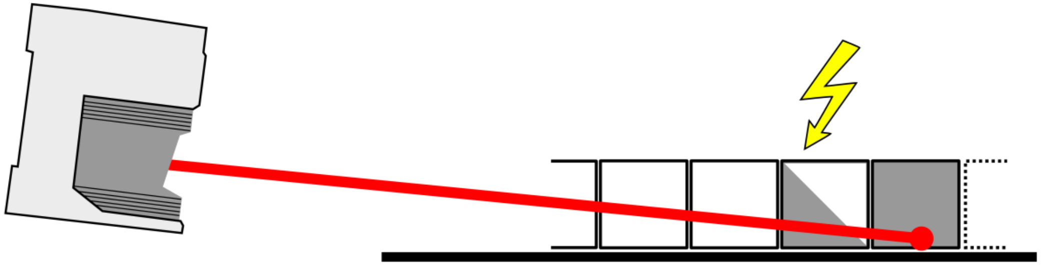
- Ray casting from sensor origin to end point
- Mark last voxel as occupied, all other voxels on ray as free
- Measurements are integrated probabilistically
- Implemented in `OcTree::computeRay(...)` and `OcTree::insertRay(...)`



Sensor Model for 3D Scans

- Sweeping sensor, discretization into voxels
- Planes observed at shallow angle may disappear in a volumetric map
- Solution: Update each voxel of a point cloud at most once, preferring occupied endpoints
- Implemented in `OcTree::insertScan(...)`





Accessing Map Data

- Traverse nodes with iterators

```
for(OcTree::leaf_iterator it = octree.begin_leafs(),
             end=octree.end_leafs(); it!= end; ++it)
{  // access node, e.g.:
   std::cout << "Node center: " << it.getCoordinate();
   std::cout << " value: " << it->getValue() << "\n";
}
```

- Ray intersection queries
  - `octree.castRay(...)`
- Access single nodes by searching

```
OcTreeNode* n = octree.search(x,y,z);
if (n){
   std::cout << "Value: " << n->getValue() << "\n";
}
```

Occupancy and Sensor Model

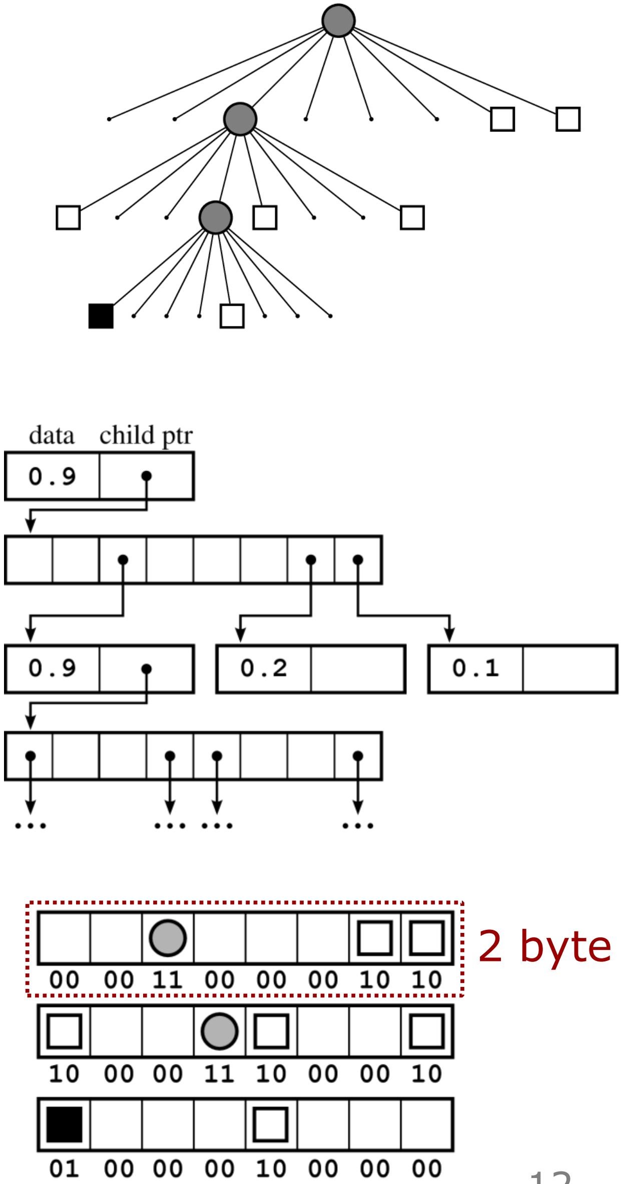
- Set occupancy parameters in octree
  - `octree.setOccupancyThres(0.5);`
  - `octree.setProbHit(0.7); // ...setProbMiss(0.3)`
  - `octree.setClampingThresMin(0.1); / ...Max(0.95)`

- Check if a node is free or occupied
  - `octree.isNodeOccupied(n);`

- Check if a node is “clamped”
  - `octree.isNodeAtThreshold(n);`

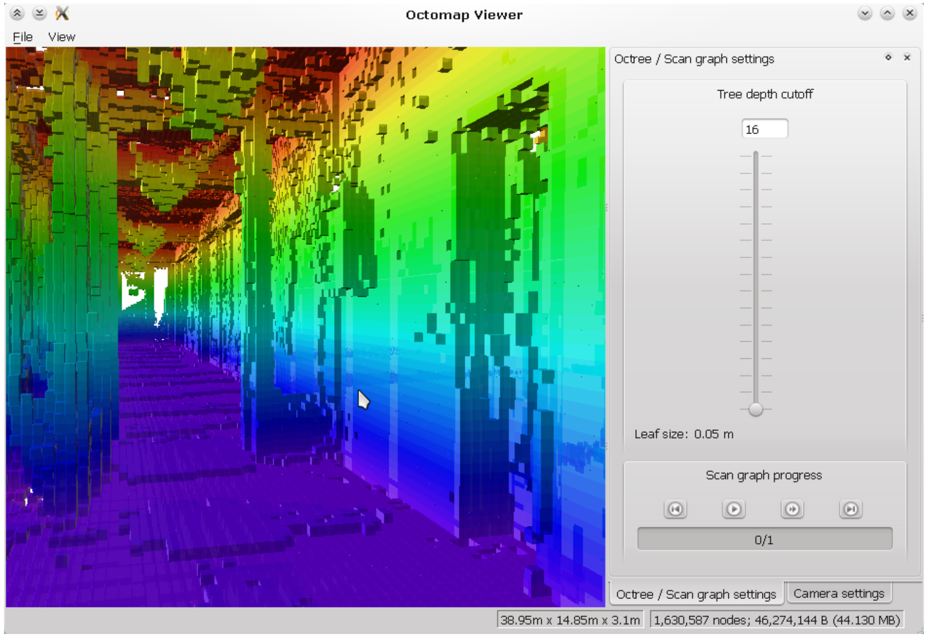
Map File Format

- Full probabilities encoded in .ot file format
- Maximum-likelihood map stored as compact bitstream in .bt file
- Exchange as ROS message: octomap_msgs package



Map Visualization

- Native OctoMap visualization: octovis



Tree depth cutoff: 16

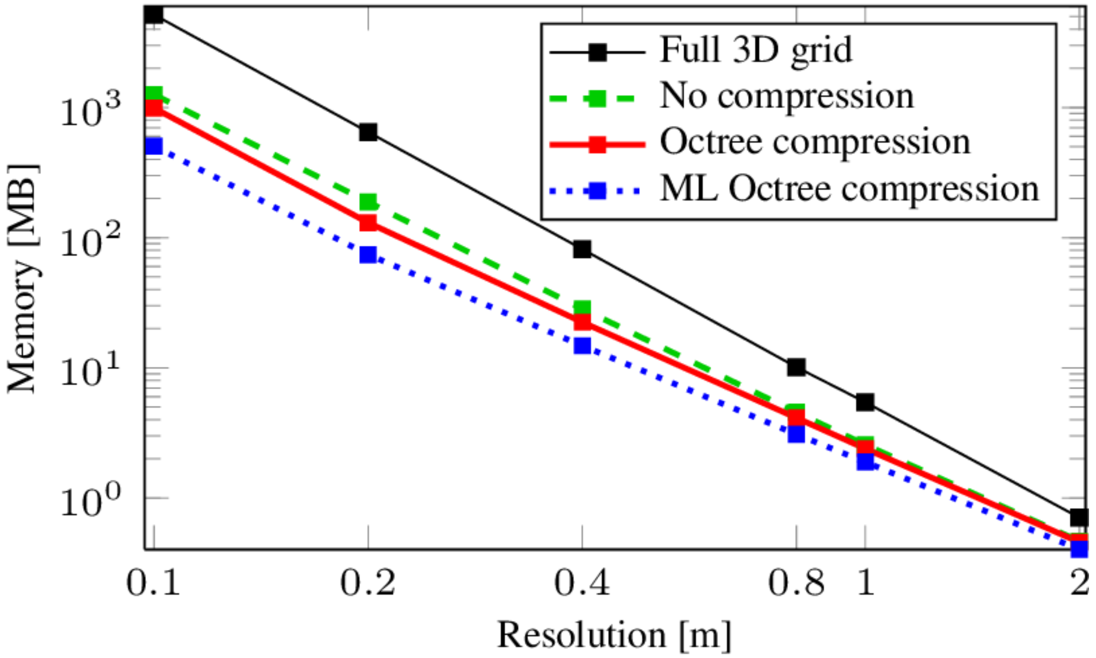
Leaf size: 0.05 m

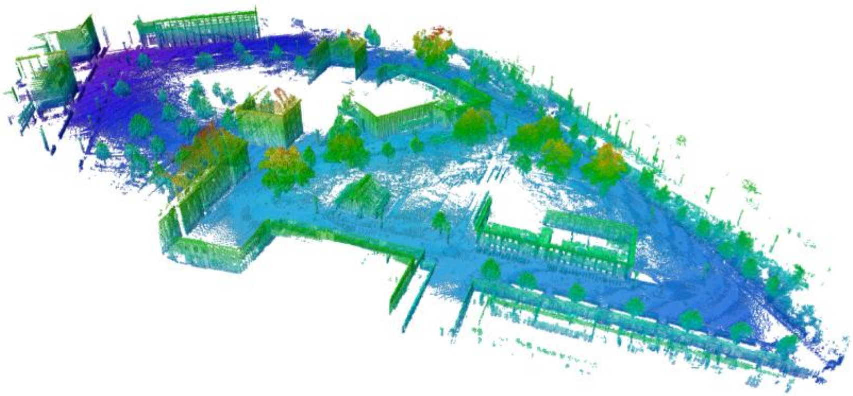
38.95m x 14.85m x 3.1m

1,630,587 nodes; 46,274,144 B (44.130 MB)

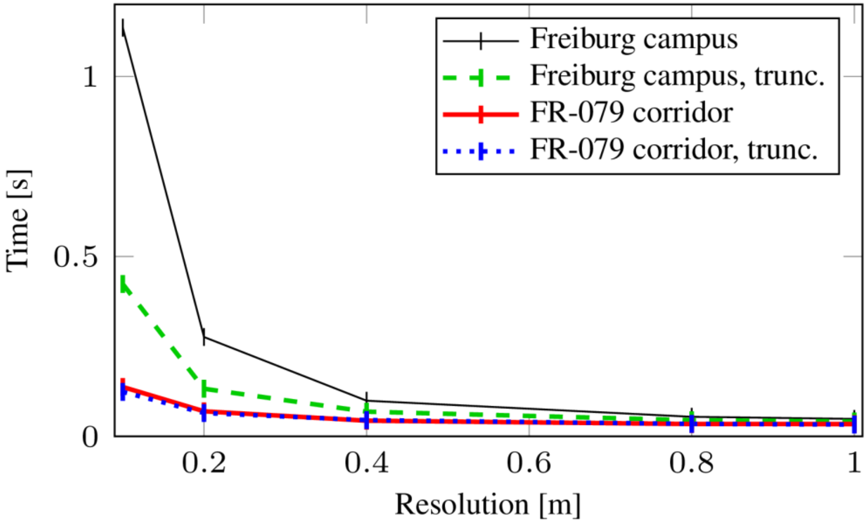
- RViz:
  - MarkerArray display from octomap_server
  - octomap_rviz_displays
  - MoveIt planning scene

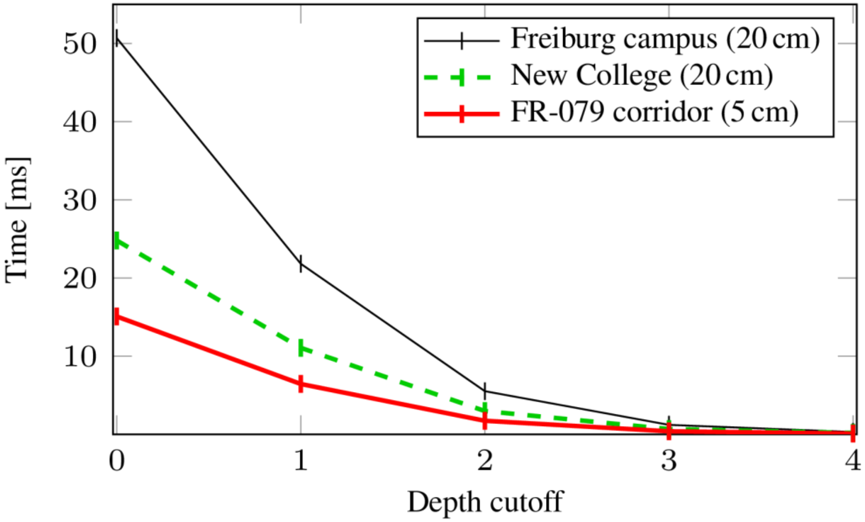
Memory Usage (Freiburg campus)





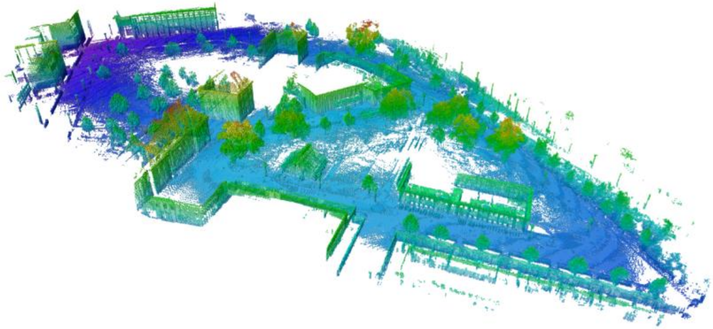
Update and Query Times

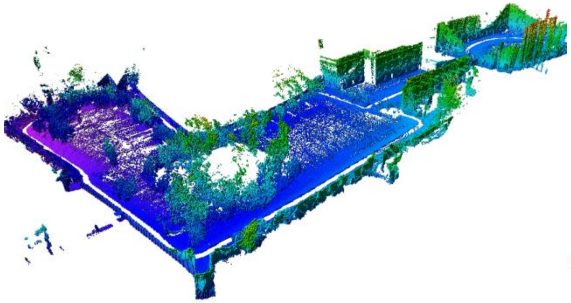


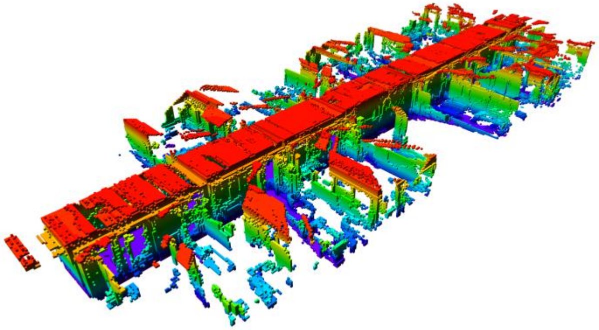


Map update
(Avg. over 100000 points)

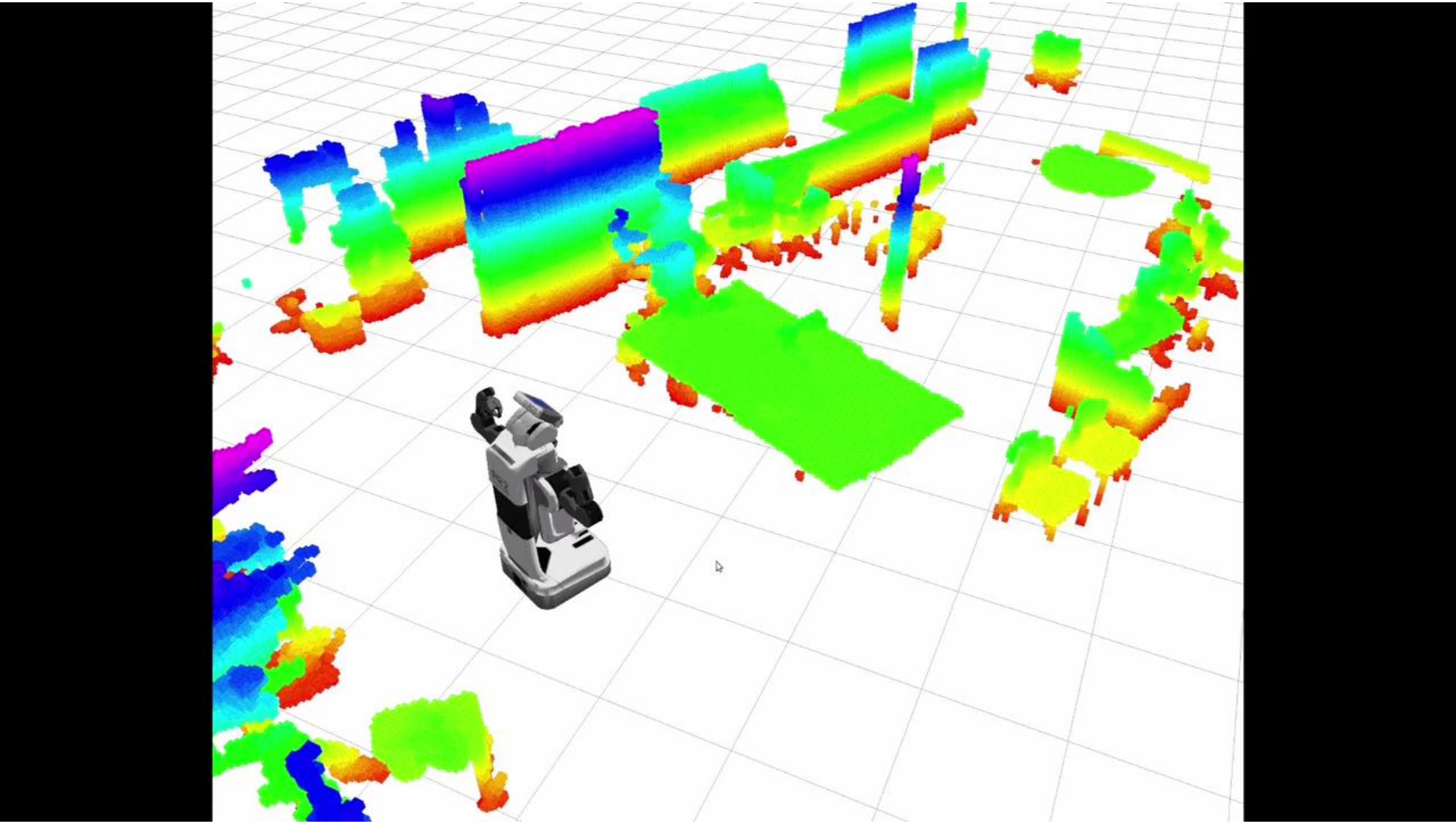
Traverse all leaf nodes





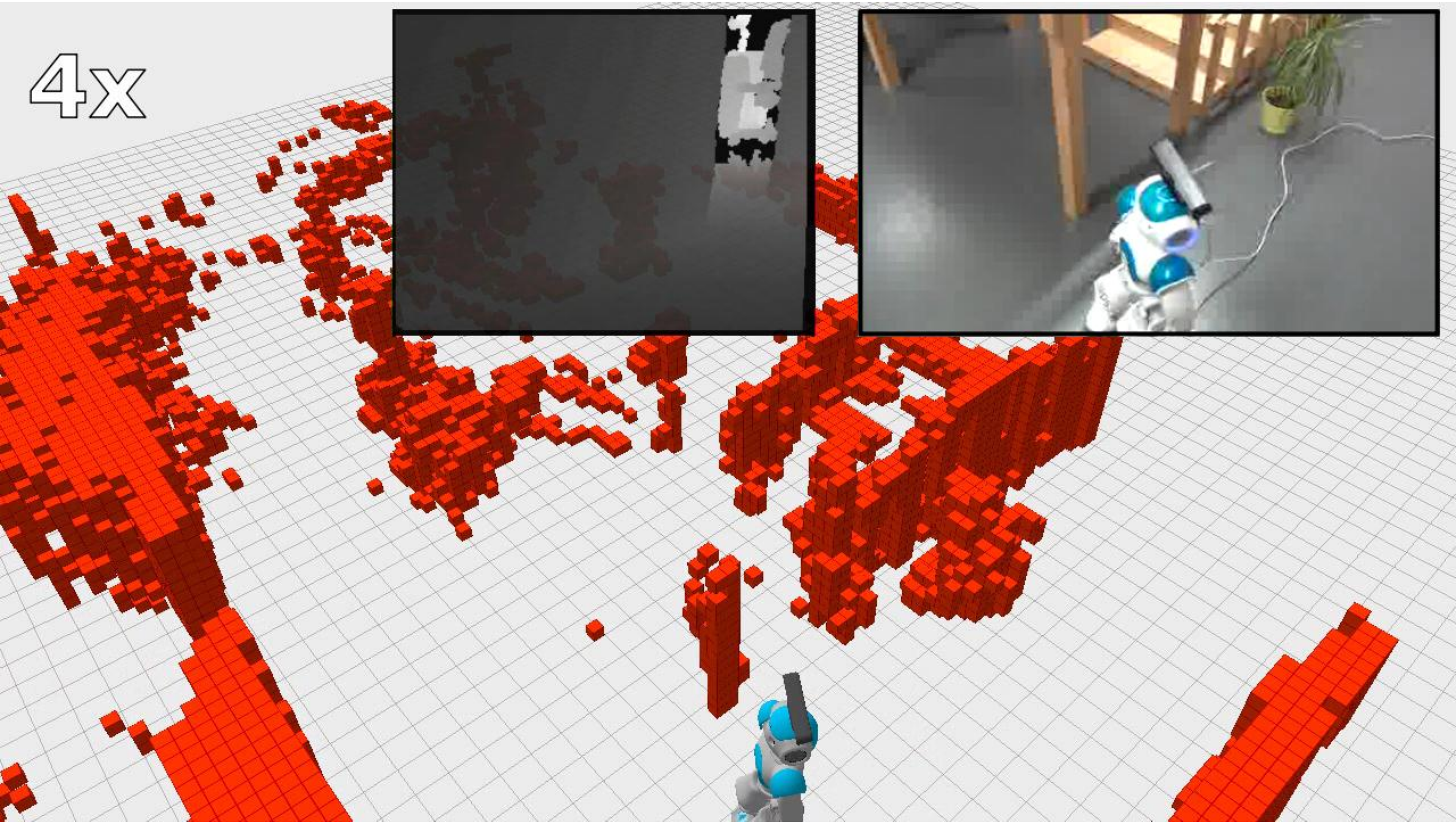


Example Use Case: Navigation in Clutter with the PR2



[Hornung et al., ICRA '12]

Example Use Case: Localization and Mapping with a Nao humanoid



[Maier et al., HUMANOIDS '12]

Conclusion

- Memory-efficient map data structure based on Octrees
- Volumetric representation of occupied, free, and unknown space
- Implementation of common map functionality: sensor updates, raycasting, ...
- Open source implementation with integration into ROS and MoveIt!
- Code, mailing list, and example data sets available at octomap.github.io

Thanks for your attention!