3D Mapping with OctoMap

Armin Hornung

University of Freiburg, Germany

octomap.sf.net

Thanks to: K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard

Robots in 3D Environments

Mobile manipulation

Outdoor navigation

Humanoid navigation

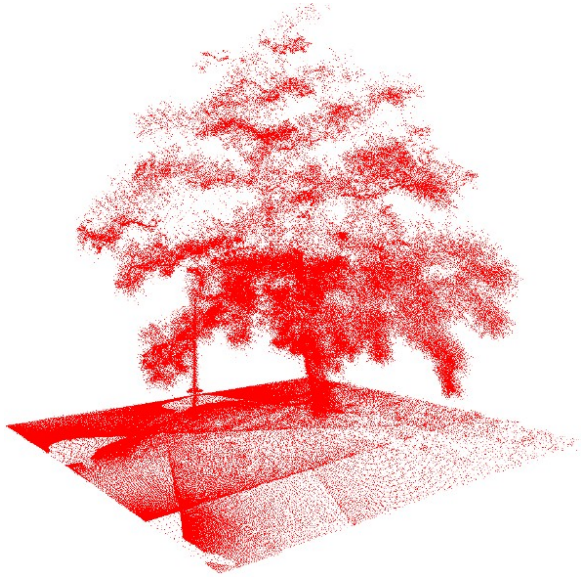
Rescue scenarios

3D Map Requirements

- Probabilistic representation
  - Map remains updatable
  - Handle sensor noise and dynamic changes
  - Fuse multiple sensors or data from multiple robots
- Represent free and unmapped areas
  - Collision-free navigation only in free space
  - Exploration of unmapped areas
- Efficiency
  - Compact in memory and on disk
  - Efficient map access and queries

Map Representations

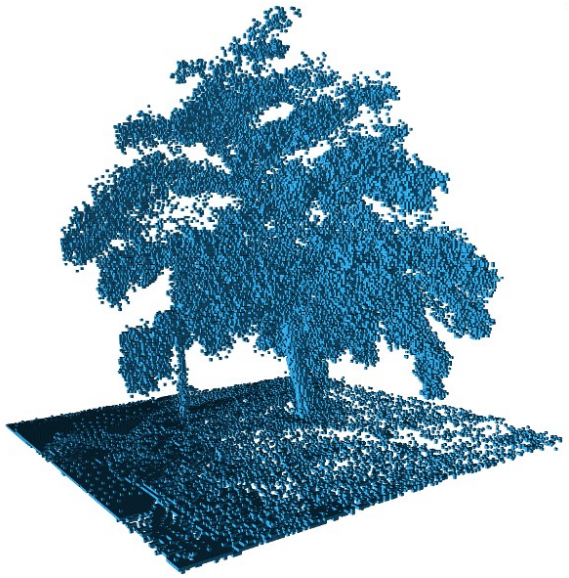
Point clouds



- **Pro:**
  - No discretization of data
  - Mapped area not limited
- **Contra:**
  - Unbounded memory usage
  - Not representing free or unknown space
  - Not possible to handle sensor noise

Map Representations

3D voxel grids

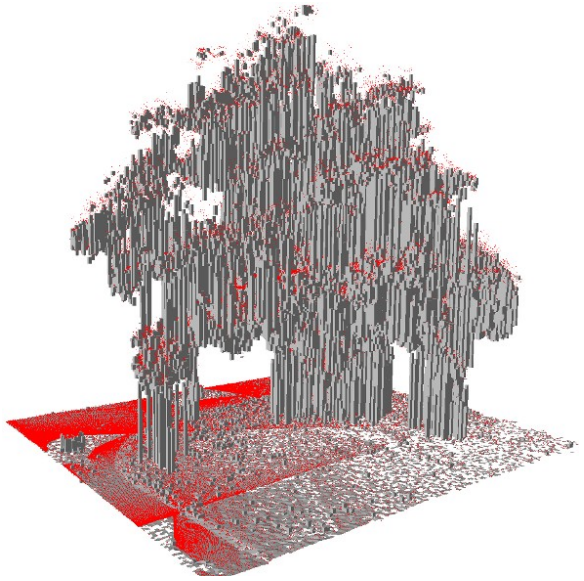


- **Pro:**
  - Probabilistic update
  - Constant access time
- **Contra:**
  - Memory requirement
    - Complete map is allocated in memory
    - Extent of map has to be known

Map Representations

2.5D (elevation) Maps

- 2D grid
- Height value(s) in each cell

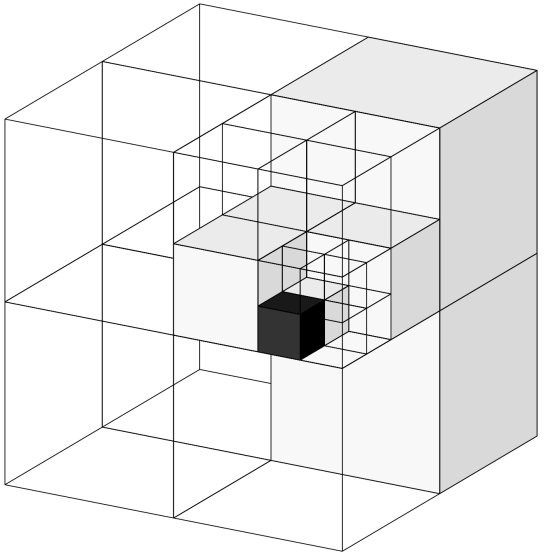


- **Pro:**
  - Memory-efficient

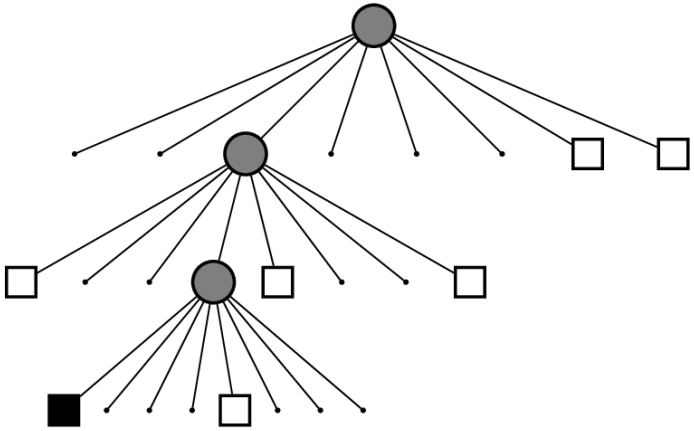
- **Contra:**
  - Not completely probabilistic
  - No distinction between free and unknown space

Map Representations

Octrees

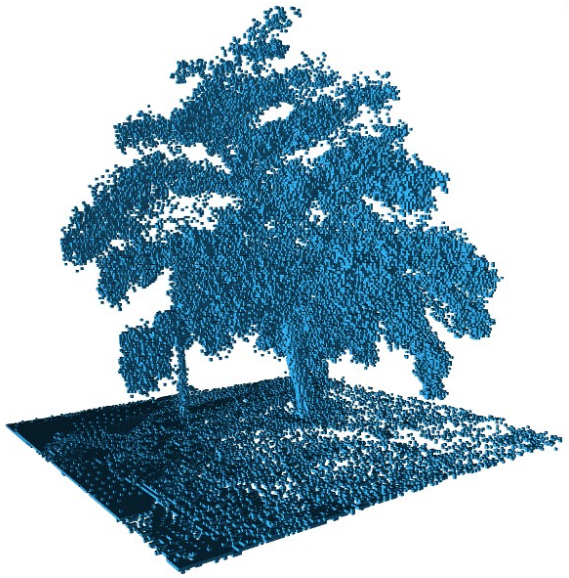


- Tree-based data structure
- Recursive subdivision of space into octants
- Volumes allocated as needed
- Multi-resolution



Map Representations

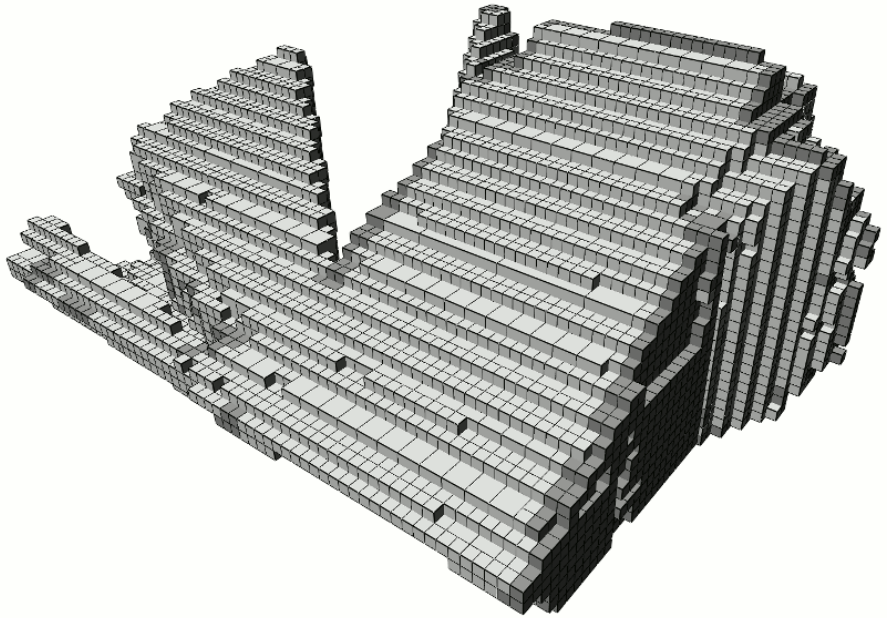
Octrees



- **Pro:**
  - Full 3D model
  - Probabilistic
  - Flexible, multi-resolution
  - Memory-efficient
- **Contra:**
  - Implementation can be tricky (memory, update, map files, …)

OctoMap Framework

- Based on octrees
- Probabilistic representation of occupancy including free and unknown areas
- Supports multi-resolution map queries
- Lossless compression
- Compact map files



OctoMap Framework

- Open source implementation as C++ library available at octomap.sf.net
- Stand-alone, self-contained library for Linux, Mac, and Windows
- Fully documented
- Pre-built debian packages at ros.org, see www.ros.org/wiki/octomap
- ROS integration in packages octomap_ros, octomap_msgs, and octomap_server

ROS.org

Probabilistic 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 notation

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

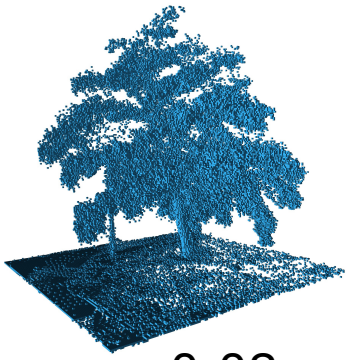
Probabilistic Map Update

- Clamping policy ensures updatability [Yguel '07]

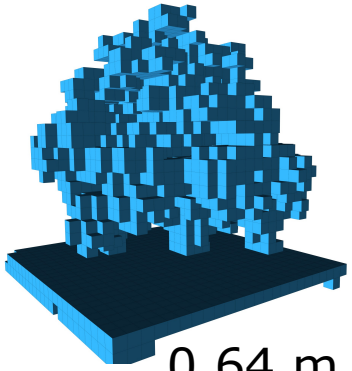
$$L(n) \in [l_{\text{min}}, l_{\text{max}}]$$

- Update of inner nodes enables multi-resolution queries

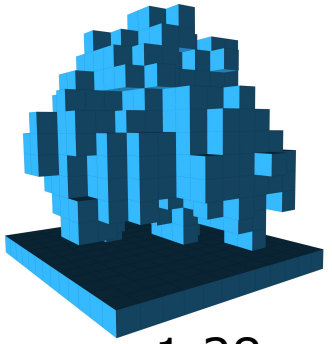
$$L(n) = \max_{i=1..8} L(n_i)$$



0.08 m



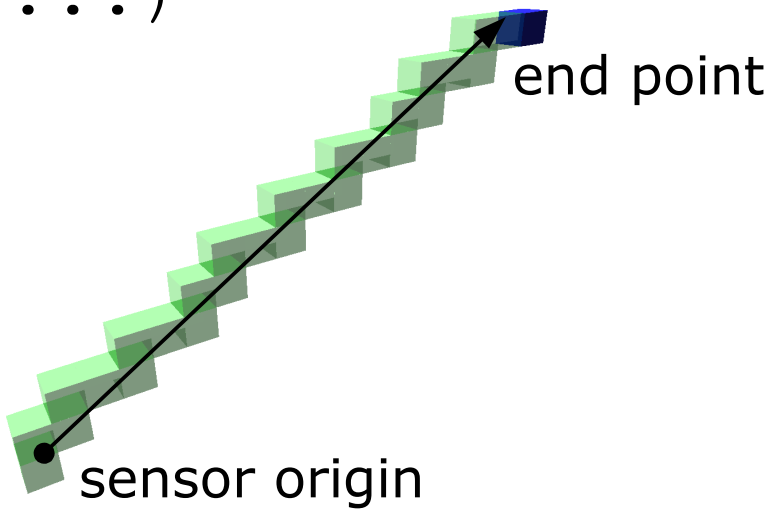
0.64 m



1.28 m

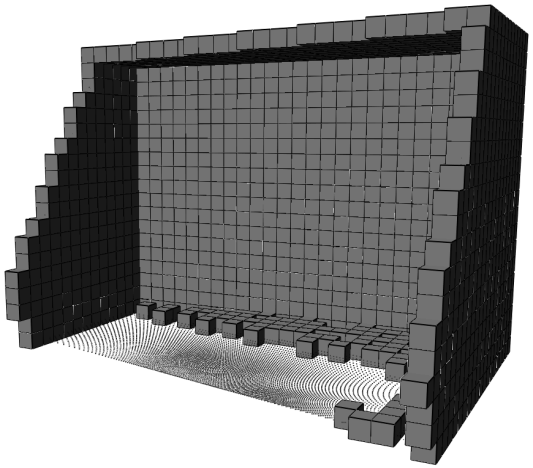
Sensor Model for Single Rays

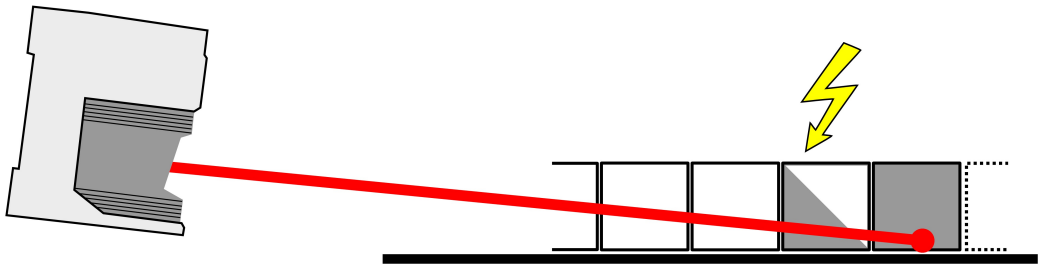
- Ray casting from sensor origin to end point
- Mark last voxel as occupied, all other voxels on ray as free
- Measurements are integrated probabilistically
- Implemented in `OcTree::computeRay(...)` and `OcTree::insertRay(...)`



Sensor Model for 3D Scans

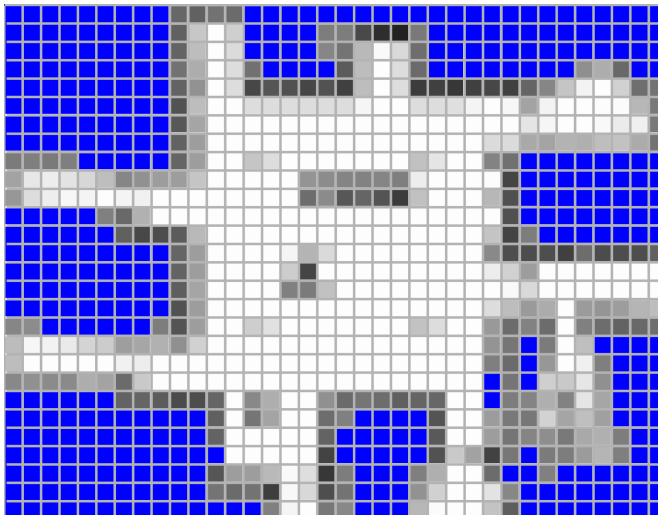
- Sweeping sensor, discretization into voxels
- Planes observed at shallow angle may disappear in a volumetric map
- Solution: Treat complete sweep as one point cloud with a preference on occupied voxels
- Implemented in `OcTree::insertScan(...)`

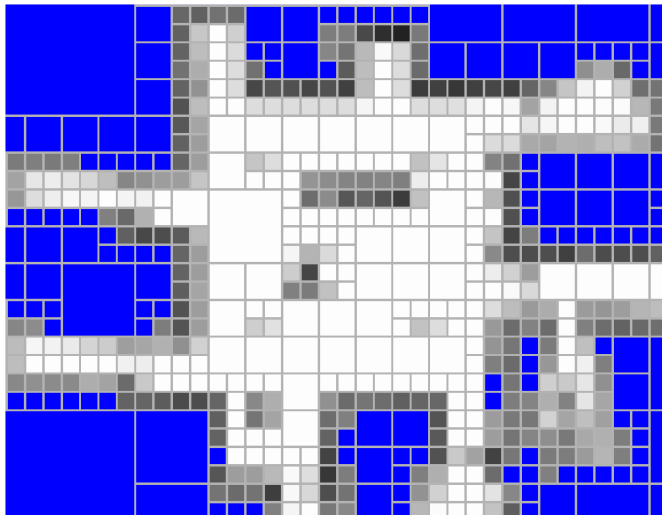




Lossless Map Compression

- Lossless pruning of nodes with identical children
- High compression ratios esp. in free space
- Call `OcTree::prune()` after updates





[Kraetzschmar '04]

Accessing Map Data

- Traverse nodes with iterators

```
for(OcTree::leaf_iterator it = octree.begin_leafs(),
  end=octree.end_leafs(); it!= end; ++it)
{
  //manipulate 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";
}
```

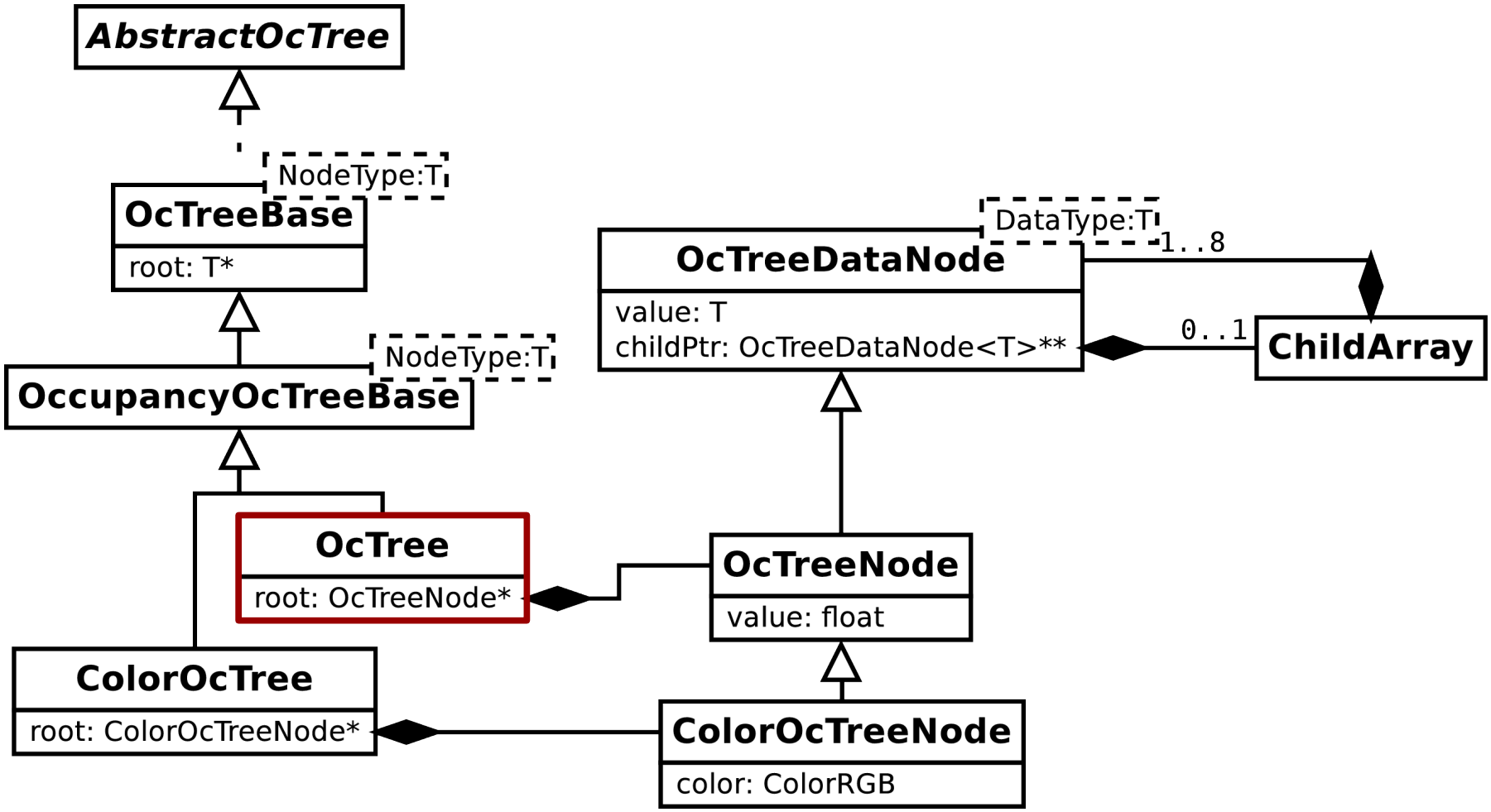
Occupancy and Sensor Model

- 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)`

Implementation Details

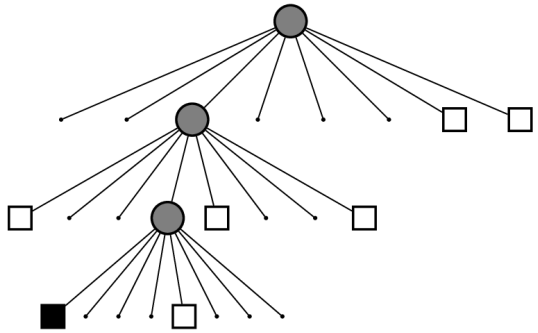
- Nodes store only necessary information for memory-efficiency:
  - Pointer to child nodes
  - Payload: float value for occupancy
  - 40 byte for inner nodes, 8 byte for leafs (32 bit architecture)
- All other information reconstructed while traversing the tree (position, size, ...)
- Flexible for own extensions:
  - OcTree implementation templated over nodes
  - OcTreeNode implementation templated over stored data
- Depth currently limited to 16

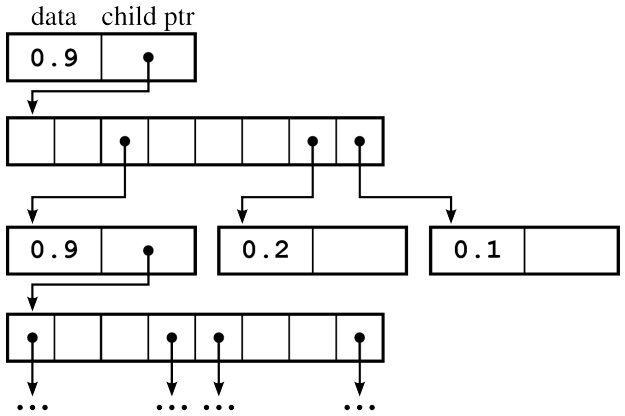
Implementation Details

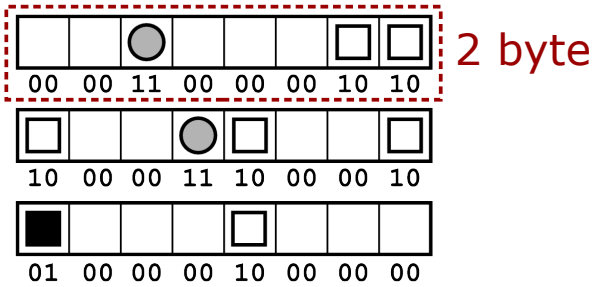


Map Files (Serialization)

- Full probabilities encoded in .ot file format
  - `OcTree::write(file)`
- Maximum-likelihood map stored as compact bitstream in .bt file
  - Occupied, free, and unknown areas
  - Very small file sizes
  - `OcTree::writeBinary(file)`

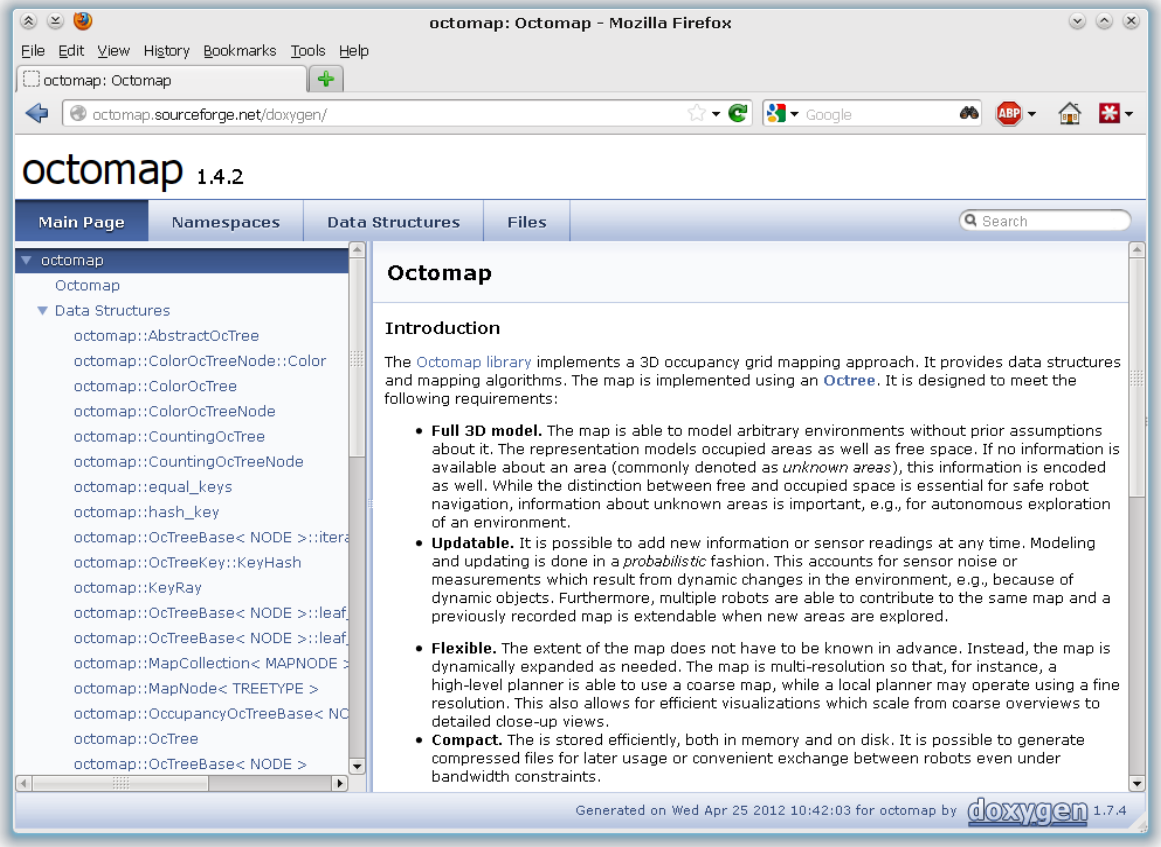






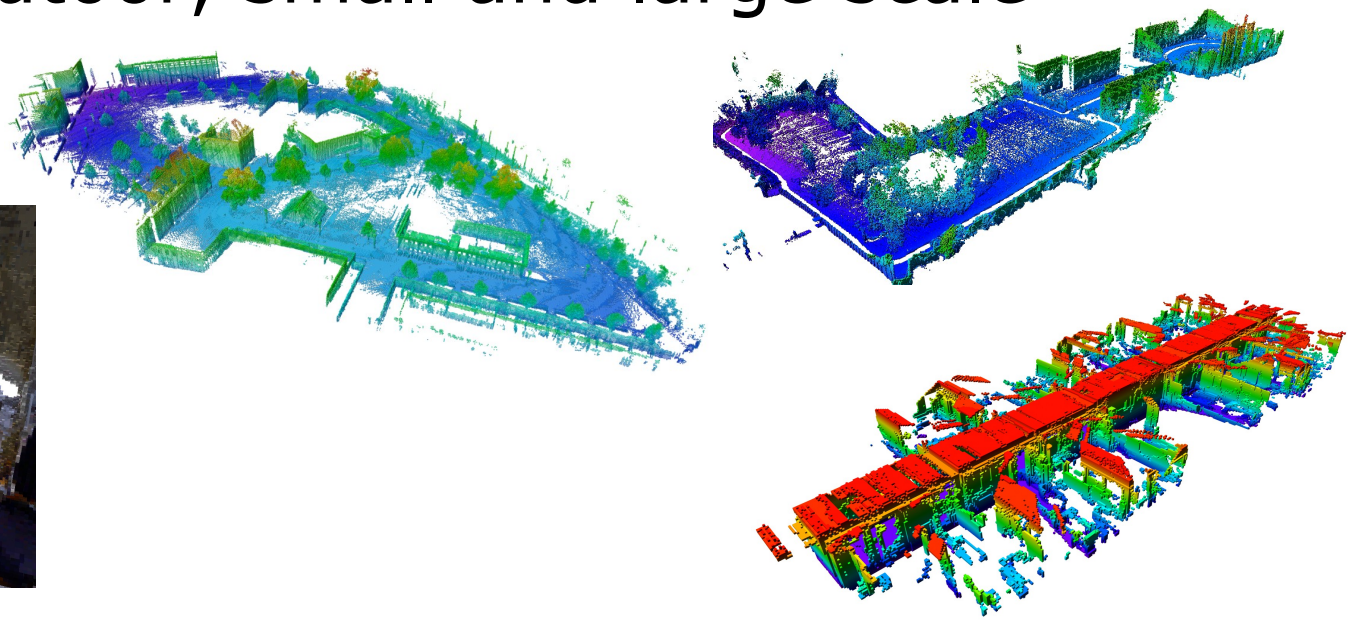
API Documentation

- Latest released version online: http://octomap.sf.net/doxygen
- Generate from source: “make docs”



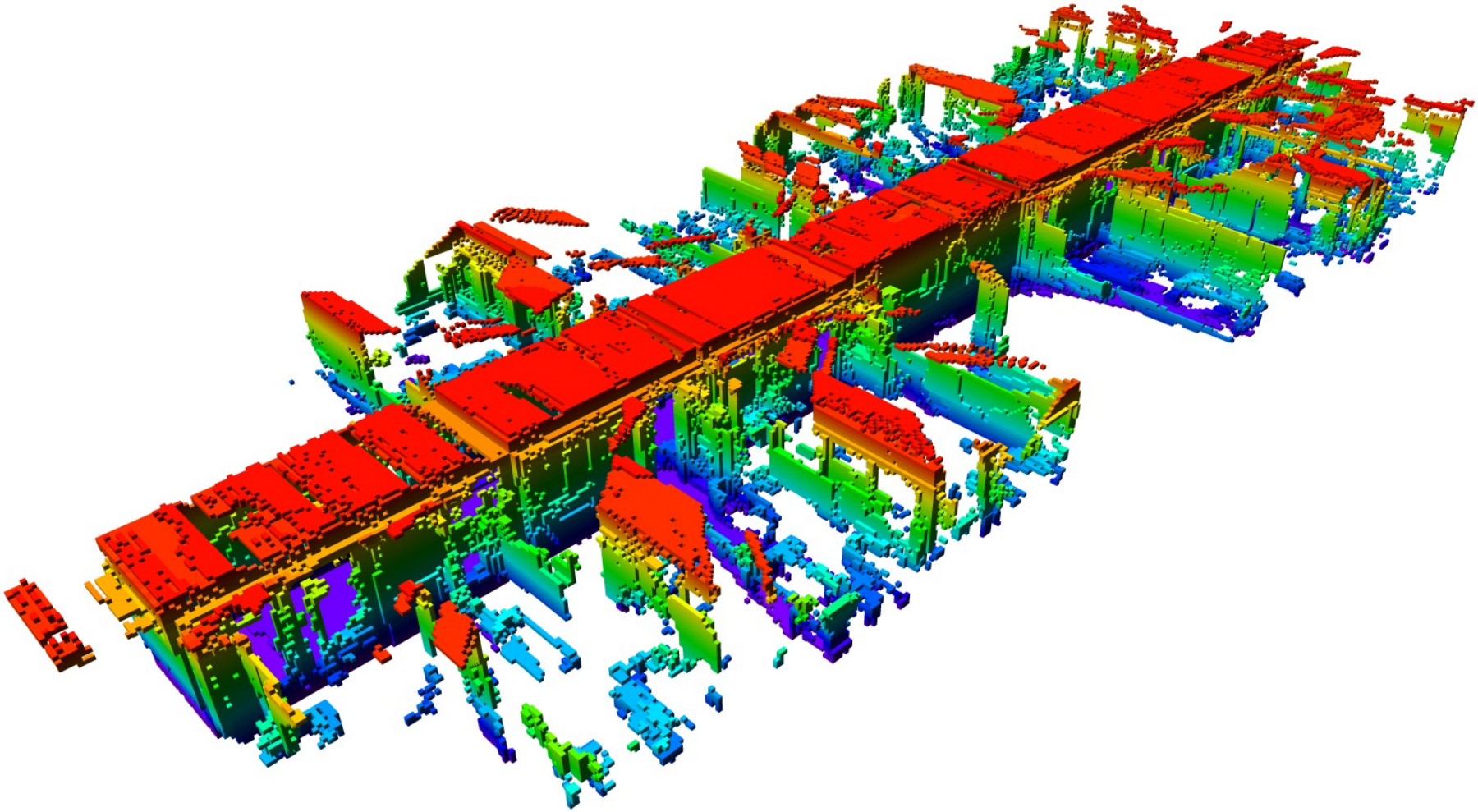
Example Data Sets

- Data set repository at http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/
- Source data (3D laser scans) and final occupancy maps
- In- and outoor, small and large scale



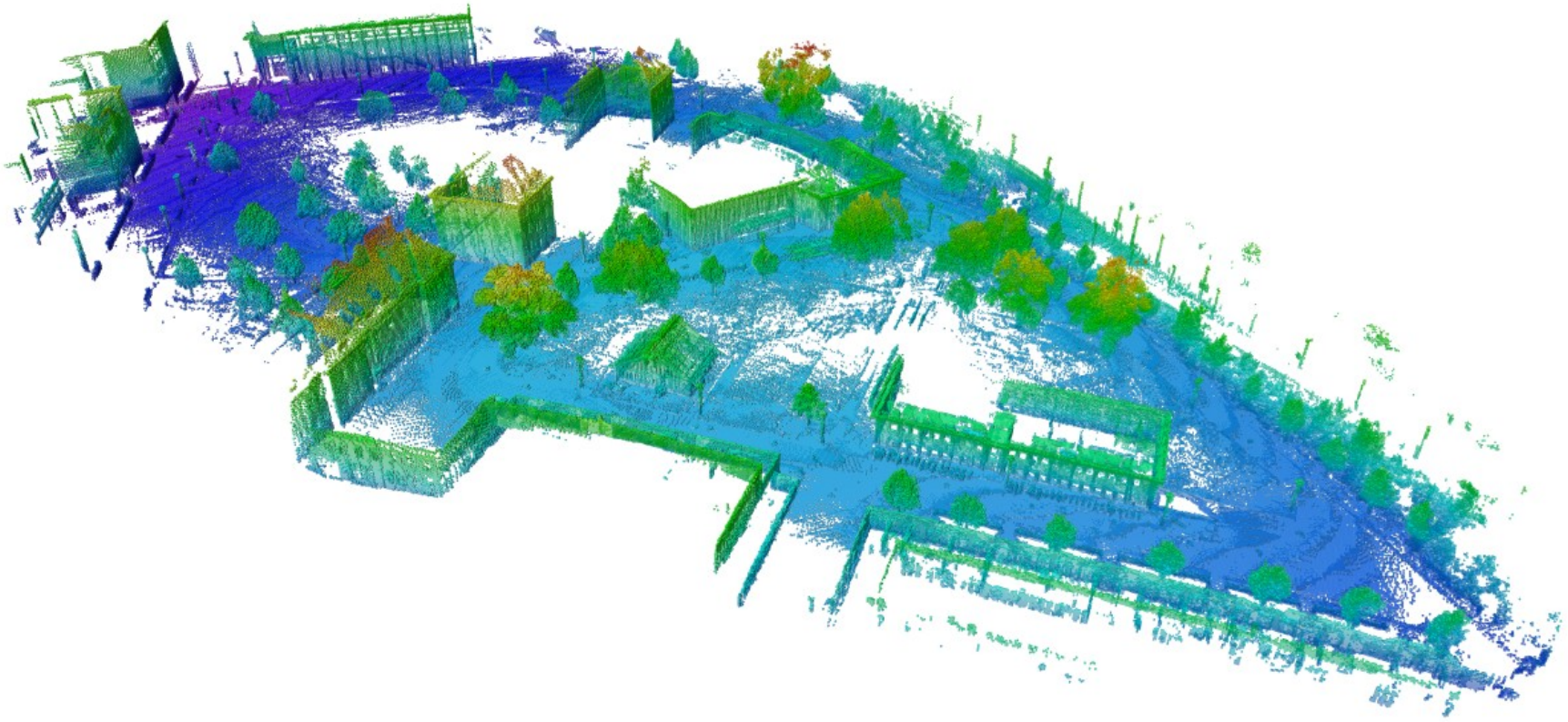
Examples: Office Building

- FR-079 corridor (44 x 18 x 3 m³, 5 cm resolution)



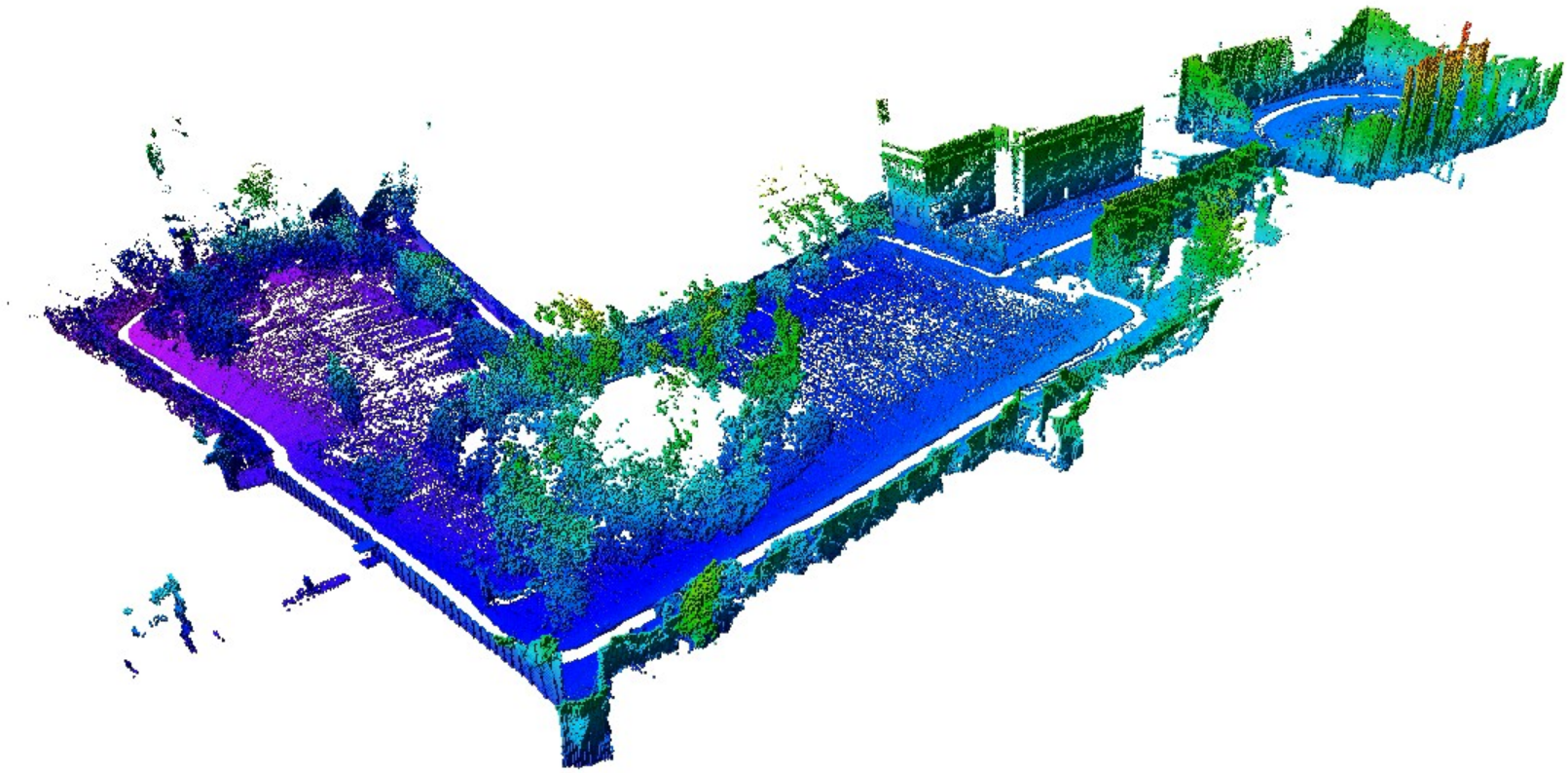
Examples: Large Outdoor Areas

- Freiburg campus (292 x 167 x 28 m³, 20 cm resolution)



Examples: Large Outdoor Areas

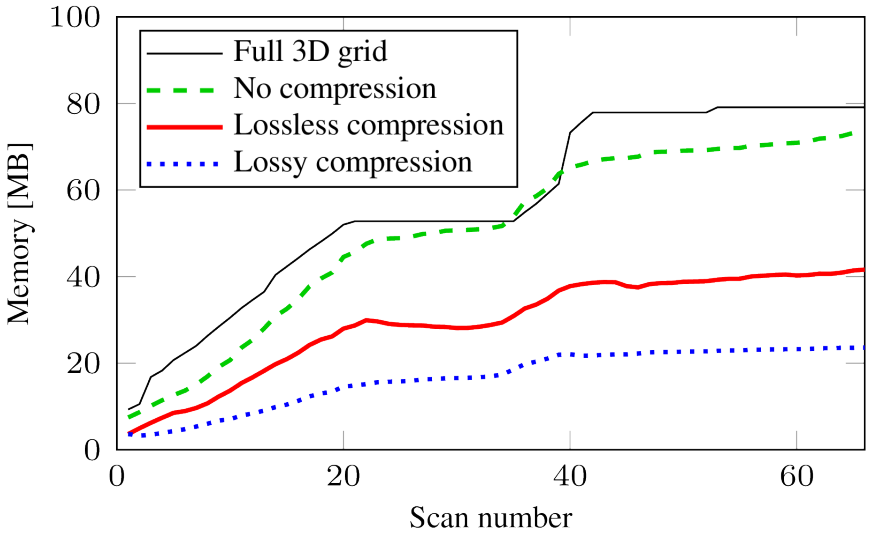
- New College (250 x 161 x 33 m^3, 20 cm resolution)



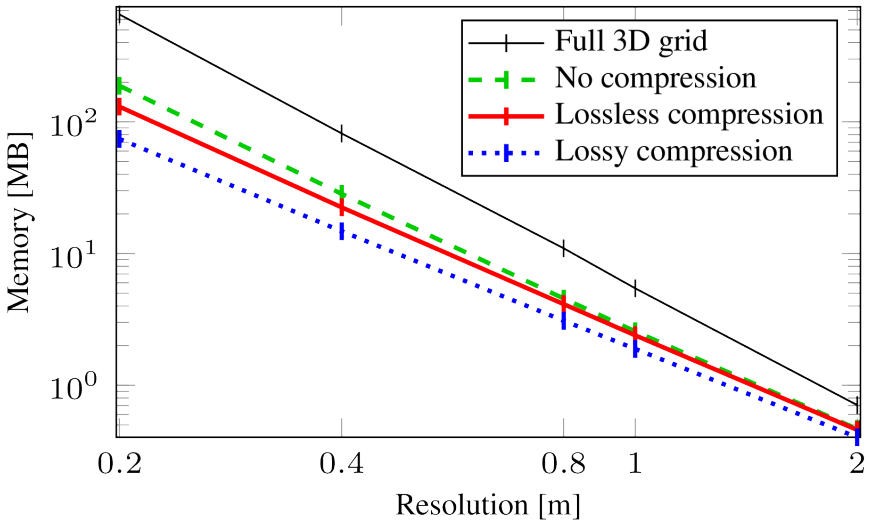
Examples: Indoor environment

- RGBD freiburg1_360 (8 x 7 x 5 m^3, 2 cm resolution)

Memory Usage

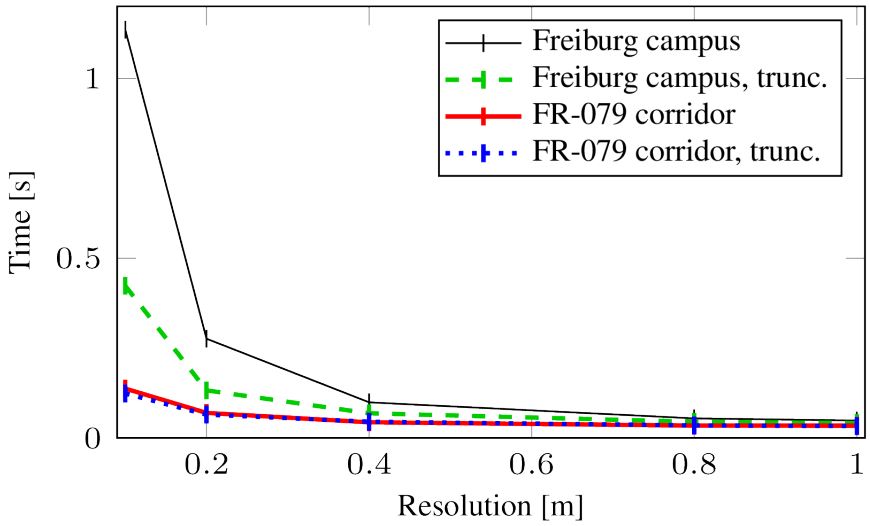


Memory usage over time
(FR-079 dataset)

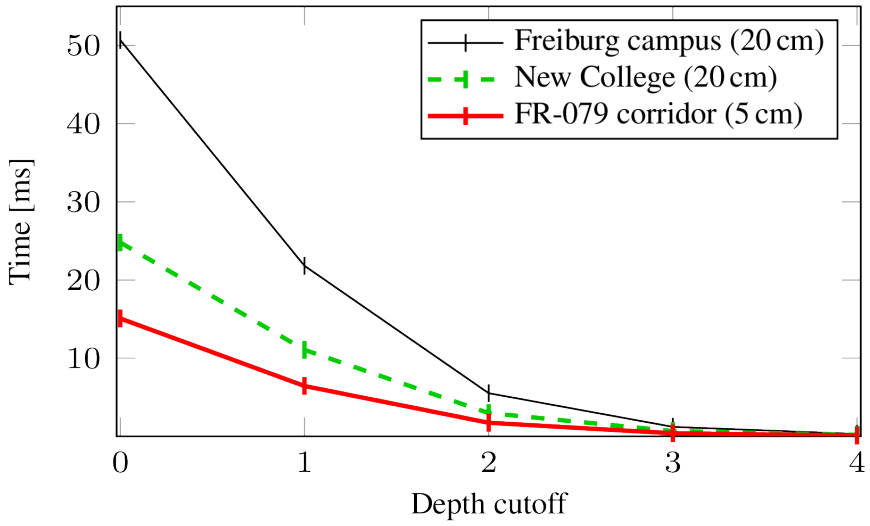


Effect of resolution on memory
(Freiburg campus)

Query Times



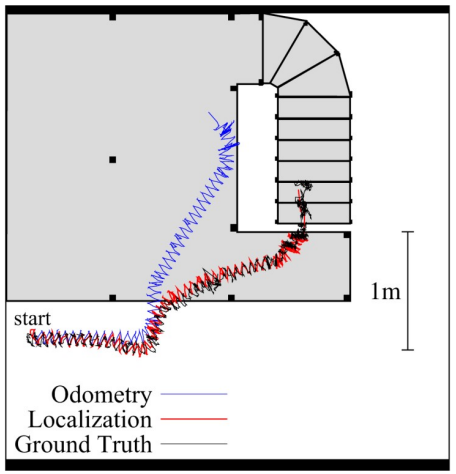
Map update
(Avg. over 100000 points)



Traverse all leaf nodes

Case Study: Localization

- 6D pose of a humanoid robot estimated in OctoMap as 3D environment model
- Monte Carlo localization based on laser, IMU, and joint angle data
- Sensor model: ray casting in OctoMap
- Augmented with local vision data for improved accuracy while climbing stairs



[Hornung et al., IROS '10]
[Oßwald et al., IROS '12 (to appear)]

Video: Nao Localization

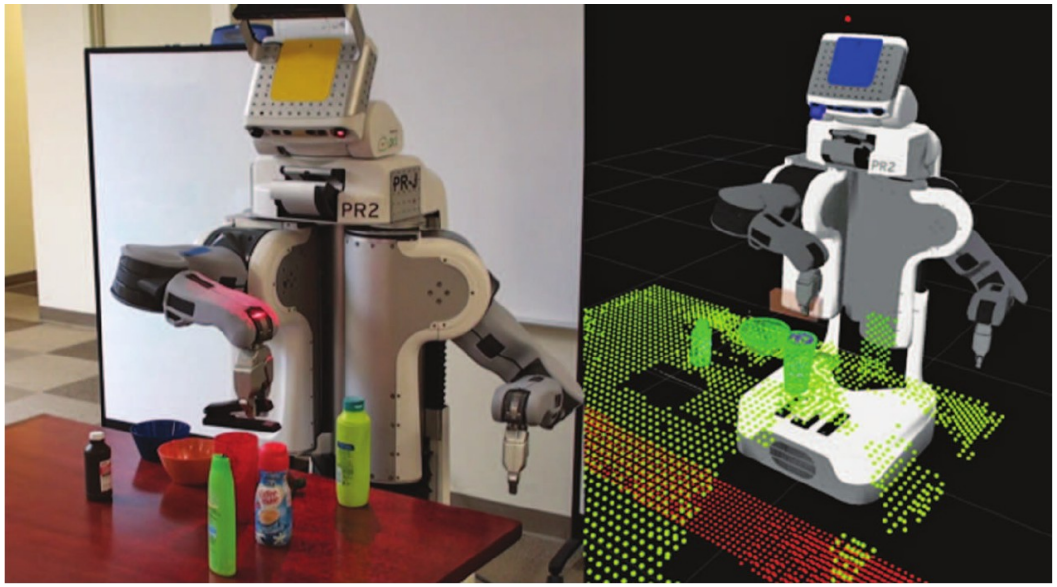
- http://www.youtube.com/watch?v=uiIi2rSKWAU

Video: Nao on Stairs

- http://www.youtube.com/watch?v=U9ll8y3Svkw

Case Study: Tabletop Manipulation

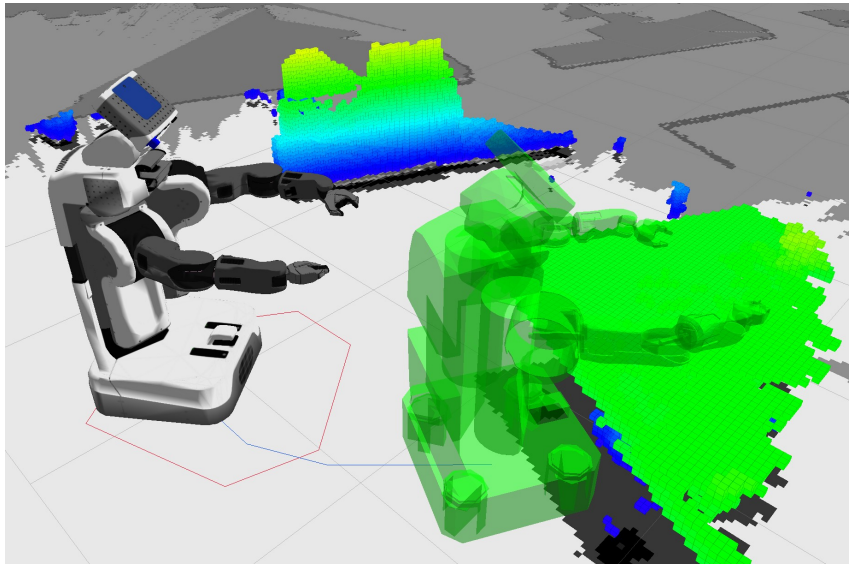
- collider package in ROS *fuerte*, new planning framework MoveIt! in ROS *groovy*
- OctoMap as probabilistic collision map
- Updates map from stereo and laser data
- Enables dynamic updates of the collision map



[Chitta et al., Robotics & Automation '12]

Case Study: Navigation in Clutter

- Collision map and obstacle avoidance for mobile manipulation
- Enables moving through narrow passages and docking tables
- Mapping in octomap_server: 3D OctoMap and downprojected 2D map layers
- Search-based planning with motion primitives and 2D / 3D collision checks in 3d_navigation stack



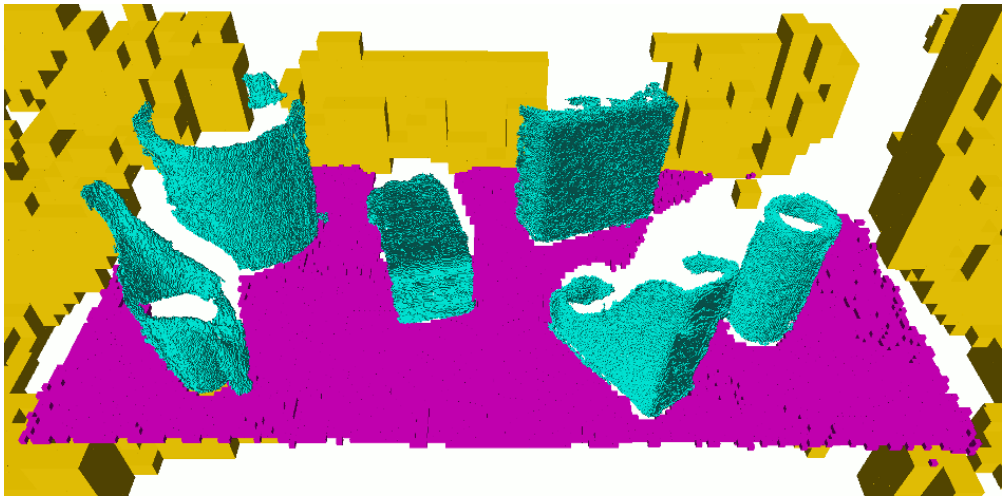
[Hornung et al., ICRA '12]

Video: Navigation in Clutter

- http://www.youtube.com/watch?v=XzkC4Ez8GYE

Extension: Octree Hierarchies

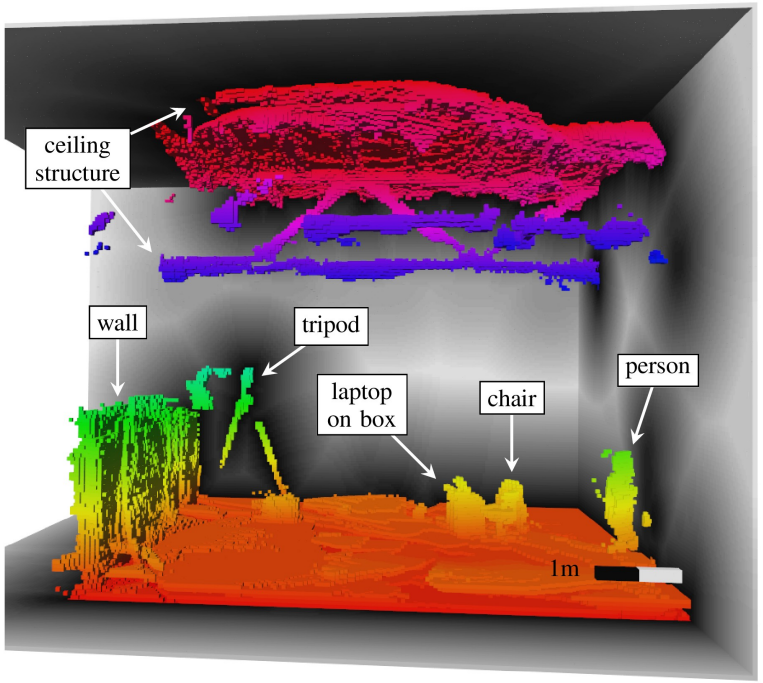
- Collection of local sub-maps in a tree structure
- Each sub-map has its own resolution and origin
  - Fine resolution for objects
  - Coarse resolution for environment and table
- More compact than a single map representing the complete scene



[Wurm et al., IROS '11]

Extension: Distance Maps

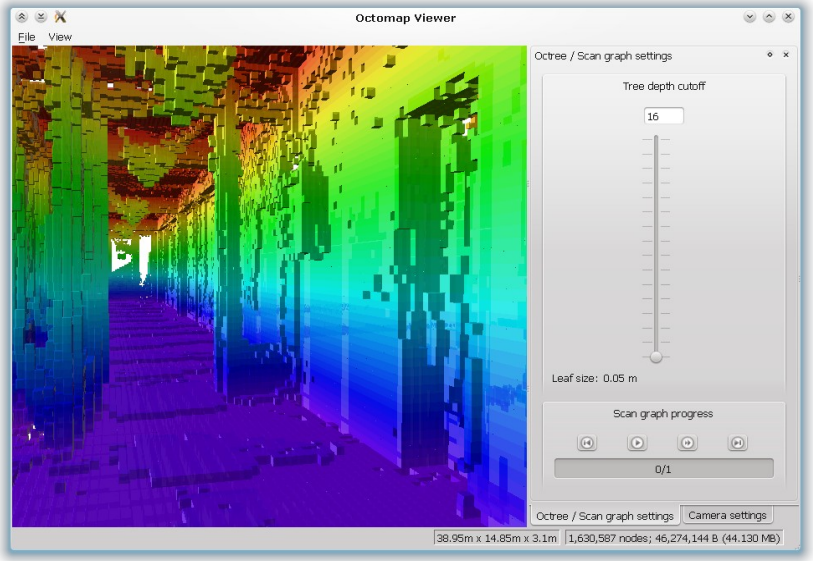
- Incremental updates of 3D distance maps
- Uses change detection in OctoMap
- Useful for path planning and collision avoidance
- Will be available in next OctoMap release (1.5): dynamicEDT3D



[Lau et al., Robotics and Autonomous Systems '12 (to appear)]

Get OctoMap Installed & Running

- Install octomap and octomap_server for ROS:
  - `sudo apt-get install ros-fuerte-octomap-mapping`
- Install octovis from build server: `http://goo.gl/XnST6`
- Download a dataset file from http://ais.informatik.uni-freiburg.de/projects/datasets/octomap/ and unzip, e.g.:
  - `gunzip fr_079.ot.gz`
- Change into ROS workspace
  - `source /opt/ros/fuerte/setup.bash`
- Start `octovis` and open a file (.graph / .ot / .bt)
  - Process source files scan by scan with "*Open graph incremental*"



Alternative: Install From Source

- Install prerequisites:
  - `sudo apt-get install build-essential cmake doxygen libqt4-dev libqt4-opengl-dev libqglviewer-qt4-dev`
- Extract source package:
  - `tar -xvzf octomap-1.4.2.tar.gz`
- Change to build directory:
  - `cd octomap-distribution/build`
- Configure and build:
  - `cmake .. && make`
- Run unit tests:
  - `make test`
- Binaries end up compiled in octomap-distribution/bin

Process and Analyze a Dataset

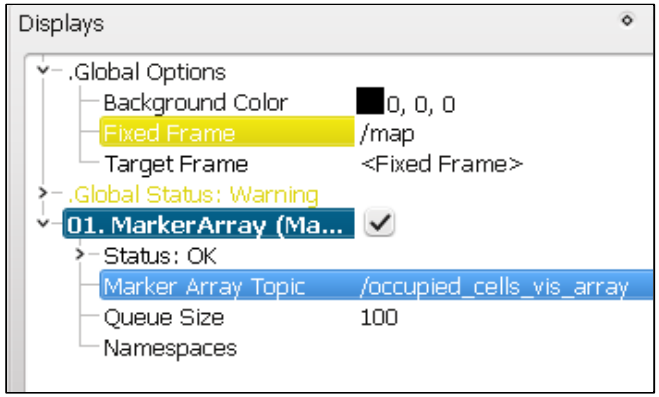
- `graph2tree -i input.graph -o map.bt -res 0.1`

  input file output file resolution

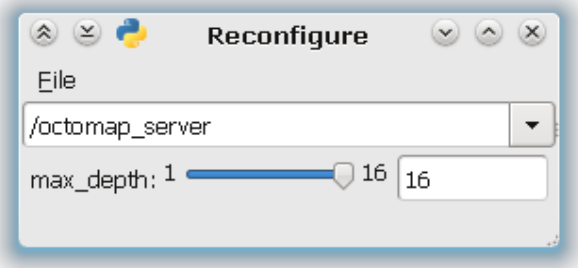
- Creates .bt and .ot files
- Output:
  - Processing time (total / average)
  - Memory consumption (compressed / uncompressed)
- Restrict sensor range: `-m 10.0`
- Further options: `graph2tree -h`

Map Visualization in ROS

- Start map server with a static map
  - `rosrun octomap_server octomap_server_node fr_079.bt`
- Start `rviz`
- Set fixed frame to "/map"
- Add MarkerArray display on topic "/occupied_cells_vis_array"



- Change resolution dynamically:
  - `rosrun dynamic_reconfigure reconfigure_gui`

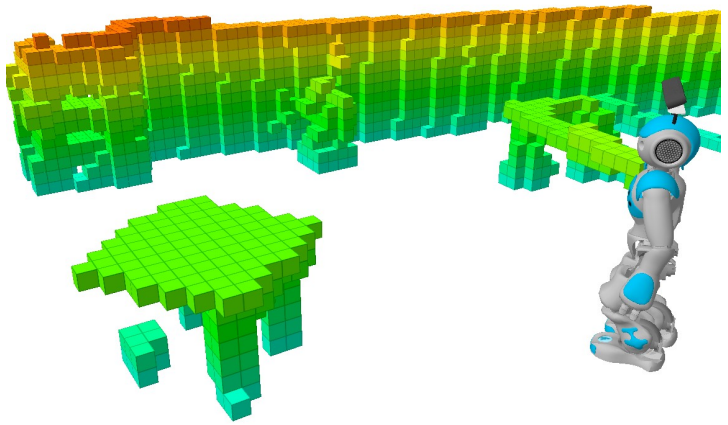


- Detailed documentation at www.ros.org/wiki/octomap_server

3D Mapping in ROS (Outline)

- Start octomap_server, remap topic "cloud_in" to a PointCloud2 of your sensor
- Requires tf from sensor to map frame

  (e.g. localization, SLAM, …)
- Example launch file: octomap_server/ launch/octomap_mapping.launch
- Build maps with ROS package rgbdslam



[Maier et al., HUMANOIDS '12 (subm.)]

Using OctoMap in Your Project

- Standard CMake (stand-alone or in ROS)
- In CMakeLists.txt:

```
find_package(octomap REQUIRED)
include_directories(${OCTOMAP_INCLUDE_DIRS})
link_directories(${OCTOMAP_LIBRARY_DIRS})
link_libraries(${PROJECT_NAME} ${OCTOMAP_LIBRARIES})
```

- For ROS, add to your manifest.txt:

```
<rosdep name="octomap" />
```

- Additional ROS packages for integration
  - octomap_msgs: ROS messages & serialization
  - octomap_ros: conversions from native ROS types

Thank you!