OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems

Kai M. Wurm Armin Hornung Maren Bennewitz Cyrill Stachniss Wolfram Burgard

***Abstract*— In this paper, we present an approach for modeling 3D environments based on octrees using a probabilistic occupancy estimation. Our technique is able to represent full 3D models including free and unknown areas. It is available as an open-source library to facilitate the development of 3D mapping systems. We also provide a detailed review of existing approaches to 3D modeling. Our approach was thoroughly evaluated using different real-world and simulated datasets. The results demonstrate that our approach is able to model the data probabilistically while, at the same time, keeping the memory requirement at a minimum.**

I. Introduction

Several robotic applications require a 3D model of the environment. Three-dimensional models are relevant in many airborne, underwater, or extra-terrestrial missions and may also be needed in domestic scenarios, for mobile manipulation tasks, or for navigation in multi-level environments.

In the past, various approaches for modeling environments in 3D have been proposed. Figure 1 depicts a tree observed in 3D laser range scans and modeled in three commonly used representations, namely point clouds, elevation maps [7], and multi-level surface maps [19]. It also shows the representation of the tree using the structure proposed in this paper which has been designed to meet the following four requirements:

Full 3D model. The map should be able to model arbitrary environments without prior assumptions about it. The representation should model occupied areas as well as free space. If no information is available about an area (commonly denoted as "unknown" areas), this information should be encoded as well. While the distinction between free and occupied space is essential for safe navigation, information about unknown areas is important for the autonomous exploration of an environment.

Updatable. It should be possible to add new information or sensor readings at any time. Modeling and updating should be done in a *probabilistic* fashion. This will account for sensor noise or measurements which result from dynamic changes in the environment. Furthermore, multiple robots should be able to contribute to the same map and a previously recorded map should be extendable when new areas are explored.

All authors are with the University of Freiburg, Department of Computer Science, D-79110 Freiburg, Germany.

This work has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and by the EC within the 7th framework programme under grant agreement no FP7-IST-213888-EUROPA and FP7-IST-248258-First-MM.

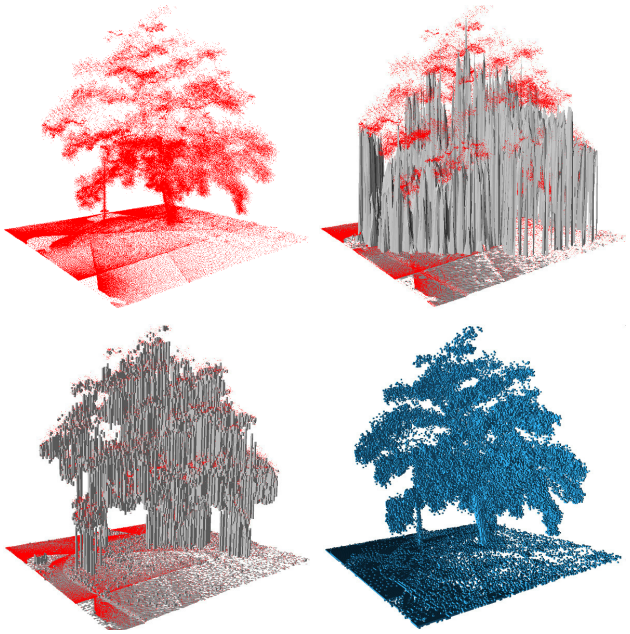

Fig. 1. 3D representation of a tree as a point cloud (top left), elevation map (top right), multi-level surface map (bottom left), and using our approach (bottom right).

Flexible. The extent of the map should not have to be known in advance. Instead, the map should be dynamically expanded as needed. The map should be multi-resolution so that, for instance, a high-level planner for navigation will be able to use a coarse map, while a local planner, e.g. for manipulation tasks, may operate using a fine resolution. This will also allow for efficient visualizations which scale from coarse overviews to detailed close-up views.

Compact. The map should be stored efficiently, both in memory and on disk. It should be possible to generate compressed files for later usage or convenient exchange between robots even under bandwidth constraints.

Although 3D mapping is an integral component of many robotic systems, there exist very few readily available implementations. Recently, the European Commission identified the lack of available software modules for robotic applications as a limiting factor both in research and in industrial applications, leading to the BRICS (Best Practice in Robotics) project.

In this paper, we present an integrated mapping system based on octrees for the representation of the three-dimensional structure of the environment. The goal is to

combine the advantages of previous approaches to 3D environment modeling to meet the requirements specified above. The advantage of our approach is that it allows for efficient and probabilistic updates while keeping the memory consumption at a minimum. We implemented our approach and thoroughly evaluated it on various simulated and real datasets of both indoor and large-scale outdoor environments. As a major contribution, our implementation in form of a self-contained C++ library is freely available at `http://octomap.sf.net/` as open source with the aim of facilitating future development of systems operating in three-dimensional environments.

This paper is organized as follows. After providing a detailed discussion of related work in this area, we present our multi-resolution map structure that is able to model arbitrary three-dimensional environments including their free and unknown areas in Sec. III. In Sec. IV we then evaluate our approach in different scenarios including large-scale outdoor maps, as well as small-scale indoor environments.

II. Related Work

A popular approach to modeling environments in 3D is to use a grid of cubic volumes of equal size (*voxels*) to discretize the mapped area. Roth-Tabak and Jain [15] as well as Moravec [10] presented early works using such a representation. A major drawback of rigid grids is their large memory requirement. The grid map needs to be initialized so that it is at least as big as the bounding box of the mapped area, regardless of the actual distribution of map cells in the volume. In large-scale outdoor scenarios or when there is the need for fine resolutions, the memory consumption becomes prohibitive. Furthermore, the extent of the mapped area needs to be known beforehand.

A discretization of the environment can be avoided by using point clouds. In such maps, the endpoints returned by range sensors such as laser range finders or stereo cameras are used to model the occupied space in the environment. Point clouds have been used in several 3D SLAM systems such as [2], [12]. The drawbacks of this method are that neither free space nor unknown areas are modeled and that sensor noise and dynamic objects cannot directly be dealt with. Thus, point clouds are only suitable for high precision sensors. Furthermore, the memory consumption of this representation increases with the number of measurements. This is problematic, as there is no upper bound.

If certain assumptions about the mapped area can be made, 2.5D maps are sufficient to model the environment. Typically, a 2D grid is used to store the measured height for each cell. In its most basic form, this results in an elevation map where the map stores exactly one value per cell [7]. One approach in which such maps have been demonstrated to be sufficient is the outdoor terrain navigation method described in [6]. In fact, in most outdoor settings, only one level for driving the vehicle exists. By ignoring all objects higher than the vehicle, an elevation map can be used for navigation. Elevation maps, however, are limited to one surface and are not able to model bridges, underpasses, tunnels, or multi-level buildings. This strict assumption can be relaxed by allowing multiple surfaces per cell [19] or by using classes of cells which correspond to different types of structure [5]. A general drawback of most 2.5D maps is the fact that they cannot store free or unknown areas in a volumetric way, which limits their use for localization or exploration. A related approach was proposed by Ryde and Hu [16]. They store a list of occupied voxels for each cell in a 2D grid. Although this representation is volumetric it does not differentiate between free and unknown volumes.

Tree-based representations such as octrees have been used in several previous approaches. They avoid one of the main shortcomings of grid structures by delaying the initialization of map volumes until measurements need to be integrated. In this way, the extent of the mapped environment does not need to be known beforehand. If inner nodes of a tree are updated properly, the tree can also be used as a multi-resolution representation since it can be cut at any level to obtain a coarser subdivision. The use of octrees for modeling was originally proposed by Meagher [9]. Early works mainly focused on modeling one boolean property such as occupancy [20]. Payeur *et al.* [14] used octrees to adapt occupancy grid mapping from 2D to 3D and thereby introduced a probabilistic way of modeling occupied and free space. A similar approach was used in [4] and [13]. In contrast to the approach presented in this paper, however, the authors did not explicitly address the problems of memory consumption or over-confidence in the map.

An octree-based 3D map representation was also proposed by Fairfield *et al.* [3]. Their map structure called *Deferred Reference Counting Octree* is designed to allow for efficient map updates and for copying, especially in the context of particle filter SLAM. In contrast to our approach, lossless compression of trees is not described. Instead, a lossy maximum-likelihood compression is performed periodically. Furthermore, the problem of overconfident maps and multi-resolution queries are not addressed.

Yguel *et al.* [22] presented a 3D map based on the Haar wavelet data structure. This representation is also multi-resolution and probabilistic. However, the authors did not evaluate applications to 3D modeling in-depth. In their evaluation, unknown areas are not modeled and only a single simulated 3D dataset is used. Whether this map structure is as memory-efficient as octrees is hard to assess.

Finally, to the best of our knowledge, no implementation of a 3D mapping system which meets the requirements specified in the introduction is freely available.

III. OctoMap Mapping Framework

The approach proposed in this paper uses a tree-based representation to offer maximum flexibility with regard to the mapped area and resolution. It performs a probabilistic occupancy estimation to ensure updatability and to cope with sensor noise. Furthermore, a lossless compression method ensures the compactness of the resulting models.

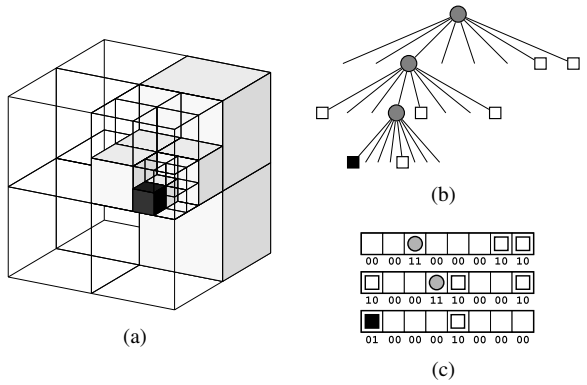

Fig. 2. Example of an octree storing free (shaded white) and occupied (black) cells (a), the corresponding tree representation (b), and the corresponding bitstream for compact storage in a file (c). The complete octree structure can be stored using only six bytes, 2 bits per child of a node.

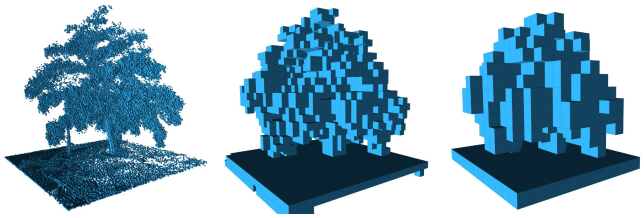

Fig. 3. By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. The occupied cells are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

A. Octrees

An octree is a hierarchical data structure for spatial subdivision in 3D [9], [20]. Each node in an octree represents the space contained in a cubic volume, usually called a voxel. This volume is recursively subdivided into eight subvolumes until a given minimum voxel size is reached, as illustrated in Fig. 2. This minimum voxel size determines the resolution of the octree. Since an octree is a hierarchical data structure, the tree can be cut at any level to obtain a coarser subdivision. An example of an octree map queried for occupied voxels at several depths is shown in Fig. 3.

In its most basic form, octrees can be used to model a boolean property. In the context of robotic mapping, this is usually the occupancy of a volume. If a certain volume is measured as occupied, the corresponding node in the octree is initialized. Any uninitialized node could be free or unknown in this boolean setting. To resolve this ambiguity, free cells can be explicitly represented as free nodes in the tree. Subvolumes which are not initialized implicitly model unknown areas. An illustration of a laser scan and the corresponding octree map can be seen in Fig. 4. Using boolean occupancy states (or *labels*) allows for compact representations of the octree: If all children of a node are occupied (or all are free) they can be pruned. This leads to a substantial reduction in the number of nodes that need to be maintained in the tree.

In robotic systems, one typically has to cope with sensor noise and temporarily or permanently changing environments. In such cases, a discrete occupancy label will not be sufficient to perform sensor fusion. Instead, occupancy has to be modeled probabilistically. Occupancy grid mapping [11] can be used to represent occupancy as a binary random variable. However, such a probabilistic model lacks the possibility of lossless compression by pruning.

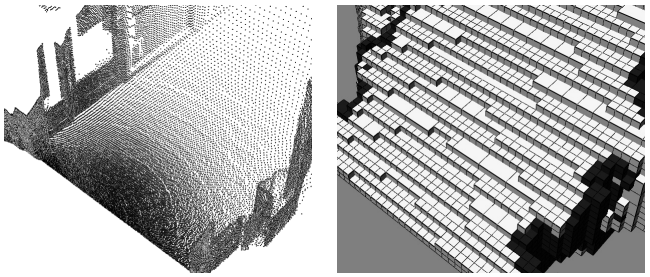

Fig. 4. A single 3D scan of the corridor dataset recorded with a tilting scanner (left) is converted to a maximum-likelihood map (right). Free cells are shown in white, occupied cells in black.

The approach presented in this paper offers a means of combining the compactness of octrees that use discrete labels with the updatability and flexibility of probabilistic modeling as we will discuss in Sec. III-C.

B. Sensor Fusion

Sensor readings are integrated using occupancy grid mapping as introduced by Moravec and Elfes [11]. The probability $P(n \mid z_{1:t})$ of a leaf node n being occupied given the sensor measurements $z_{1:t}$ is estimated according to

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

The inverse sensor model $P(n \mid z_t)$ is specific to the sensor used for mapping. Under the common assumption of a uniform prior ($P(n) = 0.5$) and by using the $logOdds$ (L) notation, Eq. 1 can be simplified to

$$\mathrm{L}(n \mid z_{1:t}) = \mathrm{L}(n \mid z_{1:t-1}) + \mathrm{L}(n \mid z_t). \quad (2)$$

Note that $logOdds$ values can be directly converted into probabilities and vice versa [11]. The $logOdds$ formulation typically allows for faster updates in case of precomputed sensor models.

As we stated in the introduction, we require the map to remain updatable in order to react to temporary or permanent changes in the environment. From Eq. (2) it is evident, however, that any change in the state of a node requires as many observation as were integrated to define its current state. To overcome this overconfidence in the map, Yguel *et al.* [21] propose a *clamping update policy*:

$$\mathrm{L}(n \mid z_{1:t}) = \max\left(\min\left(\mathrm{L}\left(n \mid z_{1:t-1}\right) + \mathrm{L}\left(n \mid z_t\right), l_{\max}\right), l_{\min}\right) \quad (3)$$

with the upper and lower bounds $l_{\max}$ and $l_{\min}$. Applying the clamping update policy in our approach ensures that the

confidence in the map remains bounded. As a consequence the model of the environment remains updatable.

Probabilistic updates are performed for the leaf nodes only. To obtain a multi-resolution map, however, inner nodes have to be updated as well. To determine the occupancy probability of a node n given its eight subvolumes n_i, several strategies could be pursued [8]. Depending on the application at hand, either the mean occupancy

$$\bar{l}(n) = \frac{1}{8} \sum_{i=1}^{8} \mathrm{L}(n_i) \tag{4}$$

or the maximum occupancy

$$\hat{l}(n) = \max_i \mathrm{L}(n_i) \tag{5}$$

could be used. Here, $\mathrm{L}(n)$ returns the current $logOdds$ occupancy value of a node n. Using $\hat{l}(n)$ to update inner nodes can be regarded as a conservative strategy which is well suited for robot navigation. By assuming a volume to be occupied if parts of it have been measured occupied, collision-free paths can be planned at coarser resolutions and thus computationally efficient. For this reason it is used in our system. Note that in an even more conservative setting, $\mathrm{L}(n)$ can be defined to return a positive occupancy probability for unknown cells as well.

C. Tree Compression

Whenever the $logOdds$ value of a node reaches either the threshold l_{min} or l_{max}, we consider the node *stable* in our approach. Intuitively, stable nodes have been measured free or occupied with high confidence. We combine the advantages of probabilistic occupancy mapping and octrees that use discrete labels by pruning stable parts of the tree. If all children of a node are stable leafs with the same occupancy state, then the children can be pruned. Should future measurements be integrated that contradict the node's state, its children will be re-generated. Applying this compression does not lead to a loss of information in the probabilistic model. It does, however, lead to a considerable reduction in the number of nodes as we will show in the experiments.

D. Memory-Efficient Implementation

In general, octree nodes need to maintain an ordered list of its children. This can be naively achieved by using eight pointers per node. If sparse data are modeled, the memory requirement of those pointers (8×4 byte $= 32$ byte on a 32 bit architecture) will lead to a significant memory overhead [20]. With an implementation trick, however, one can overcome this by using only one pointer per node that points to an array of eight pointers. This array is only allocated if children need to be initialized.

E. Map File Generation

Whenever maps need to be stored for later usage or have to be exchanged between robots, a compact representation is required in order to minimize the consumption of disk space and communication bandwidth.

The most compact files can be generated whenever a maximum likelihood estimate of the map is sufficient for the task at hand. In this case the per-node probabilities are discarded. As motivated above, volumes in which no information has been recorded can be of special interest in robotic systems, for example, during exploration. For this reason, we explicitly differentiate between free and unknown areas and encode nodes as either occupied, free, unknown, or as inner nodes in our map files. Using these labels, octree maps can be encoded as a compact bit stream. Each node is represented by the eight labels of its children. Beginning at the root node, each child that is not a leaf is recursively added to the bit stream. Leaf nodes do not have to be added since they can be reconstructed from their label during the decoding process. Fig. 2(c) illustrates the bitstream encoding. Each row represents one node with the upper row corresponding to the root node. The lower row only contains leafs so no further nodes are added.

In this maximum likelihood representation, each node occupies 16 bits of memory, 2 bits per child. In our experiments, file sizes never exceeded 2 MB even for fairly large outdoor environments with a size of $292\,\mathrm{m} \times 167\,\mathrm{m} \times 28\,\mathrm{m}$ (see Sec. IV-C).

There exist applications, in which all information in a map needs to be stored and maintained. This often requires the use of hard disk space as secondary memory, where maps are temporarily saved to disk until they need to be accessed again. Another important demand may be the storage of additional node data such as terrain information which would be lost in a maximum likelihood encoding as introduced above. In these cases, we encode nodes by storing their data (occupancy, terrain data, etc.) and eight bits per node which specify whether a child node exists. This, however, results in considerably larger files as we will show in the experiments.

IV. Experiments

The approach presented in this paper was evaluated using several real world datasets as well as simulated ones. The experiments are designed to verify that the proposed representation is meeting the requirements formulated in the introduction. More specifically, we demonstrate that our approach is able to adequately model various types of environments and that it is an updatable and flexible map structure which can be compactly stored.

A. Sensor Model for Laser Range Data

In general, the map representation introduced in the previous section can be used in conjunction with any kind of distance sensor. Since our real-world datasets have been acquired using laser range finders (SICK LMS and Hokuyo 30LX), we employ a beam-based inverse sensor model. To efficiently determine the cells which need to be updated, a ray-casting operation is performed using a 3D variant of the Bresenham algorithm [1]. Volumes along the beam are updated as described in Sec. III-B using the following inverse

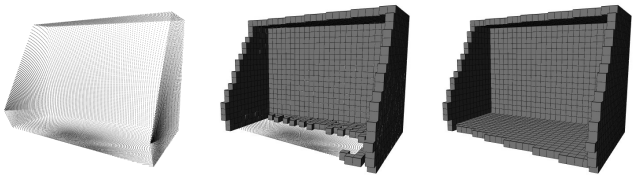

Fig. 5. A simulated noise-free 3D laser scan (left) is integrated into our 3D map structure. Sensor sweeps at shallow angles lead to undesired discretization effects (center). By updating each volume at most once, the map correctly represents the environment (right). For clarity, only occupied cells are shown.

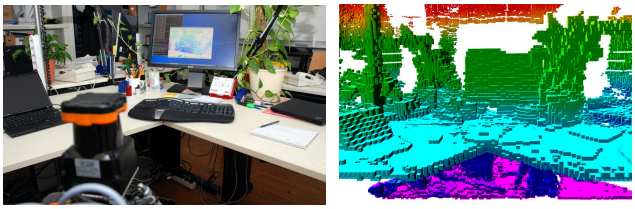

Fig. 6. A tabletop in real world (left) and visualized as 3D map (right).

sensor model:

$$\mathrm{L}(n \mid z_t) = \begin{cases} l_{\text{occ}} & \text{, if ray is reflected within volume} \\ l_{\text{free}} & \text{, if ray traversed volume} \end{cases} \quad (6)$$

The occupancy probability of all volumes is initialized to the uniform prior of $P(n) = 0.5$. Throughout our experiments, we used $logOdds$ values of $l_{\text{occ}} = 0.85$ and $l_{\text{free}} = -0.4$, corresponding to probabilities of 0.7 and 0.4 for occupied and free volumes, respectively. The clamping thresholds are set to $l_{\text{min}} = -2$ and $l_{\text{max}} = 3.5$, corresponding to the probabilities of 0.12 and 0.97. By lowering these thresholds, a stronger compression can be achieved but this obviously trades off map confidence for compactness.

Discretization effects of the ray-casting operation can lead to undesired results when mapping environments in 3D using a sweeping sensor. During a sensor sweep over flat surfaces at a shallow angle, volumes measured occupied in one 2D scan may be marked as traversed volumes in the ray-casting of following scans. This effect usually creates holes, e.g., in the floor and is illustrated using a simulated, noise-free 3D scan in Fig. 5. To overcome this problem, we treat a complete 3D scan as one measurement and update each map volume at most once. By taking care that volumes measured occupied are preserved within one 3D measurement, the described effect can be prevented.

B. Full 3D models

In this experiment, we demonstrate the ability of our approach to model real-world environments. A variety of different datasets is used.

Two indoor datasets were recorded using a Pioneer2 AT platform equipped with a SICK LMS laser range finder on a pan-tilt unit. Odometry errors were corrected using 3D scan matching. The first dataset was recorded in a small-scale indoor environment designed as a test-bed for humanoid robots (see Fig. 7). The environment features a staircase

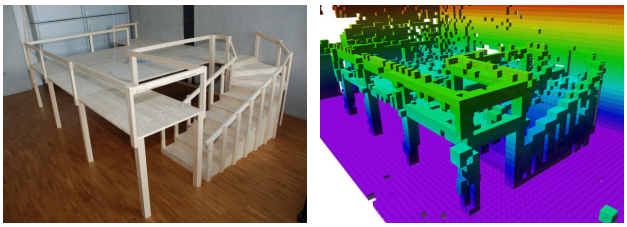

Fig. 7. A small-scale indoor environment with two floors connected by a staircase in real world (left) and visualized as 3D map (right).

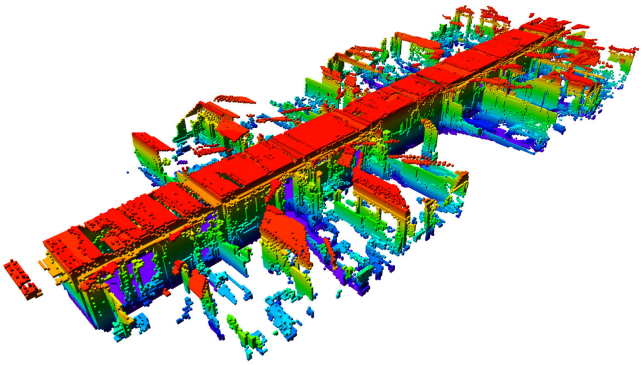

Fig. 8. 3D map of the corridor of building 079 on the Freiburg campus, as seen from the top. The structure of the adjacent rooms has been partially observed through the glass doors (size of the scene: $43.8\,\text{m} \times 18.2\,\text{m} \times 3.3\,\text{m}$).

and two different levels. The data set consists of eleven 3D measurements recorded at different poses. Considerable interpolation noise of the laser scanner at sharp edges exists in the individual scans. The second dataset was recorded in a corridor of building 079 at the Freiburg campus. The robot traversed the corridor three times and the resulting dataset consists of 66 scans.

A further indoor data set was recorded using a Hokuyo 30LX laser range finder on a pan-tilt unit (see Fig. 6). Here, the environment consists of a tabletop with several objects, which represents a typical environment for a manipulation task.

A fairly large outdoor dataset was recorded at the computer science campus in Freiburg[1]. It consists of 81 dense 3D scans covering an area of $292\,\text{m} \times 167\,\text{m}$.

In addition, we use laser range data of the *New College* data set [18]. This data was recorded in a large-scale outdoor environment with two laser scanners sweeping to the left and right side of the robot as it advances. For this dataset, an optimized estimate of the robot's trajectory generated by visual odometry was used [17].

A visualization of the resulting models can be seen in Fig. 6, 7, 8, and 9. Note that the free space is modeled but not shown in the figures.

C. Memory Consumption

In this experiment, we evaluate the memory consumption of our approach. Several datasets are processed at various

[1]Courtesy of B. Steder and R. Kümmerle, available at `http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/`

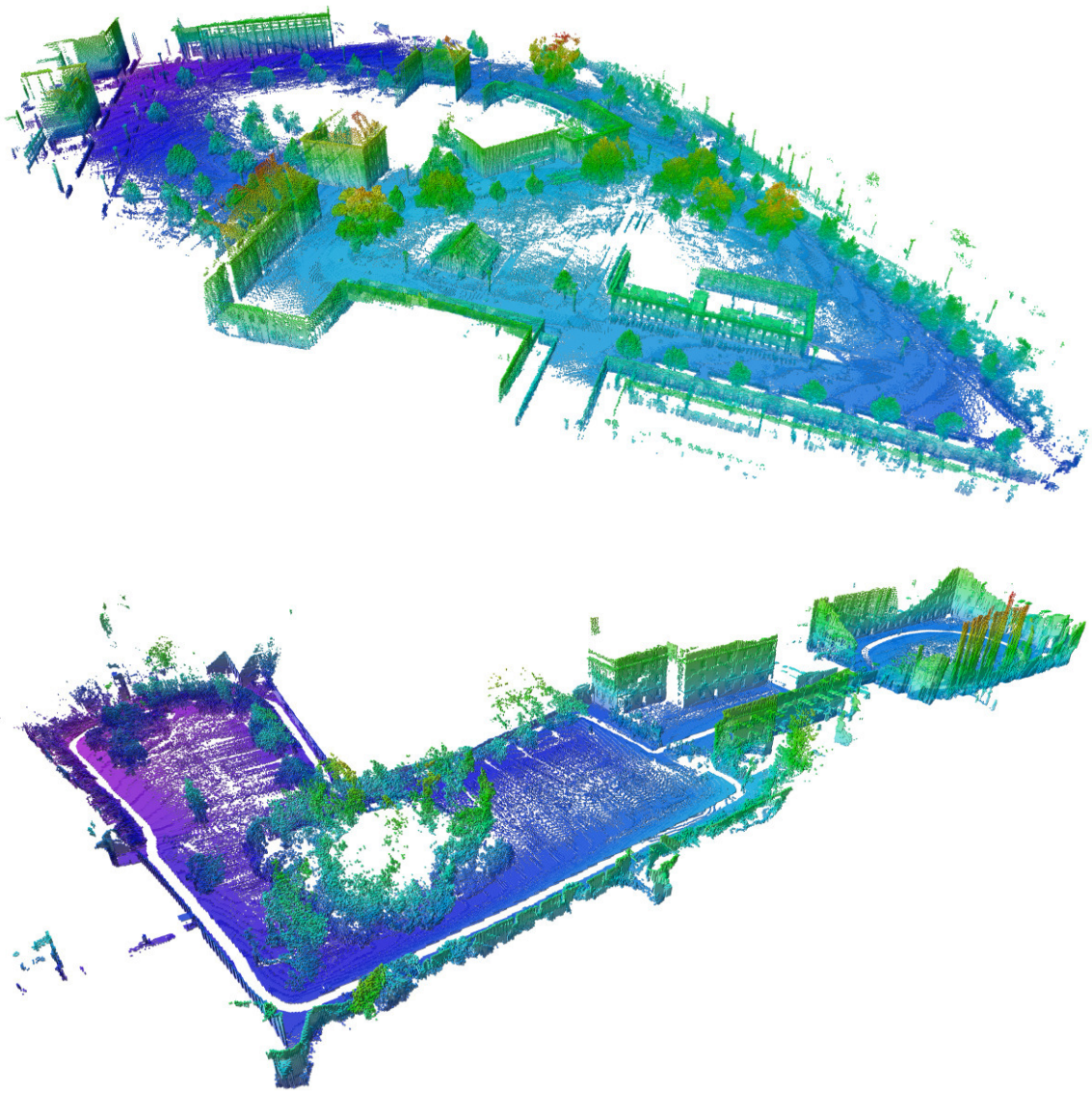

Fig. 9. Resulting octree maps of two outdoor environments at 0.2 m resolution. For clarity, only occupied volumes are shown with height visualized by a color (gray scale) coding. Top: *Freiburg campus* (size of the scene: 292 m × 167 m × 28 m), bottom: *New College* (size of the scene: 250 m × 161 m × 33 m).

tree resolutions. We analyze the memory usage of our representation with and without performing lossless compression. For comparison, we also give the amount of memory that would be required by a minimal full 3D grid which is initialized linearly in memory. Each map is furthermore written to disk using the full probabilisitc model and the binary format described in Sec. III-E, and the resulting file size is given.

The memory usage for exemplary resolutions are displayed in Table I. It can be seen that high compression ratios can be achieved especially in large outdoor environments. Here, pruning will merge considerable amounts of free space volumes. On the other hand, the map structure is also able to model fine-graded indoor environments with moderate memory requirements. In very confined spaces, an optimally aligned 3D grid may take less memory than an uncompressed mapping octree. However, this effect is diminished as soon as compression techniques are used.

For the 079 corridor dataset we also analyze the evolution of memory consumption during mapping (Fig. 10, left). The robot explored new areas up to scan number 22 and from scan number 39 to 44. In the remaining time, previously mapped areas were revisited where memory usage remained nearly constant. A slight increase can still be noticed which is due to new information gathered by scanning from different viewpoints.

As expected, memory usage increases exponentially with the tree resolution. Figure 10 (right) illustrates this using the Freiburg outdoor dataset. Please note that a logarithmic scaling is used in the plot.

Map files generated using the bitstream encoding are comparably small. For example, the visualization of the Freiburg outdoor dataset given in Fig. 9 uses 816 kB as a PNG file while the full 3D model including free and

TABLE I

MEMORY CONSUMPTION OF VARIOUS 3D DATASETS

Map dataset	**Mapped area [m^3]**	**Resolution [m]**	**Memory consumption [MB]**			**File size [MB]**	
			Full grid	No compression	Lossless compression	All data	Binary
Small scale indoor	$3.5 \times 5.2 \times 1.7$	0.05	1.03	1.91	1.38	0.54	0.02
FR-079 corridor	$43.8 \times 18.2 \times 3.3$	0.05	80.54	73.64	41.70	15.80	0.67
		0.1	10.42	10.90	7.25	2.71	0.14
Freiburg outdoor	$292 \times 167 \times 28$	0.20	654.42	188.09	130.39	49.75	2.00
		0.80	10.96	4.56	4.13	1.53	0.08
New College (Epoch C)	$250 \times 161 \times 33$	0.20	637.48	91.43	50.70	18.71	0.99
		0.80	10.21	2.35	1.81	0.64	0.05

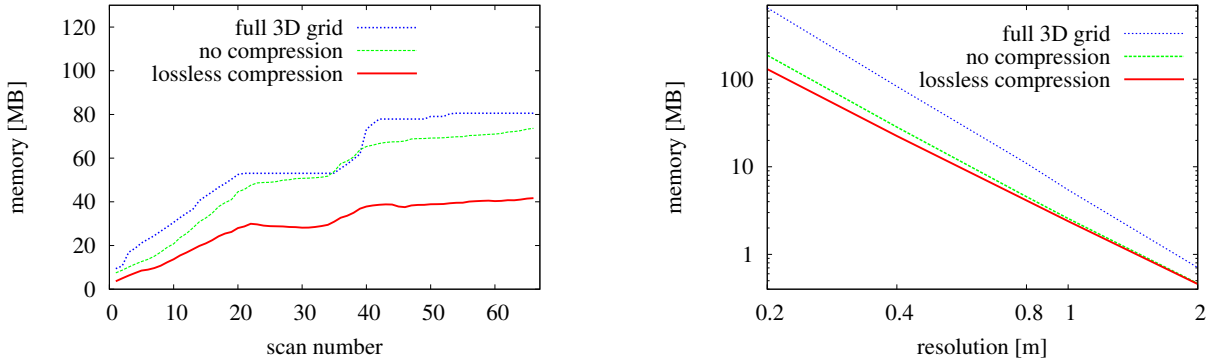

Fig. 10. Left: Evolution of the memory usage while mapping the FR-079 corridor dataset (resolution 0.05 m). Right: Effect of resolution on memory usage of the Freiburg outdoor dataset. Note that a logarithmic scaling is used.

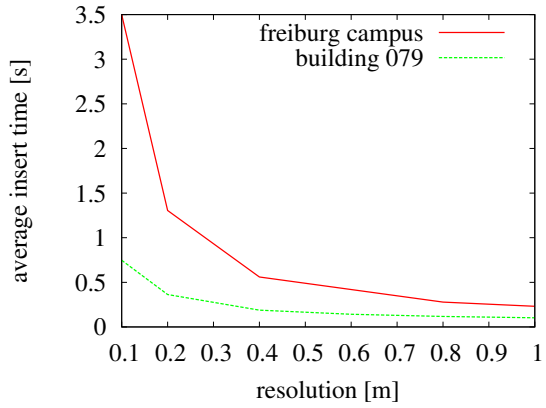

Fig. 11. Average time to insert 100,000 data points from the Freiburg campus dataset (red) and the FR-079 indoor dataset (green).

unknown areas requires 2 MB.

Table I shows the file sizes of the binary maximum likelihood map (denoted with "Binary") and the full probabilistic model ("All data"). Note that map files can be compressed even further by using standard file compression methods.

D. Runtimes

In this experiment, we analyze the time required to integrate range data using the proposed method. This time depends on the map resolution and the length of the beams that are integrated. We process the FR-079 indoor dataset with a maximum range of 10 m and the Freiburg campus dataset. Maps are computed at several resolutions. The average insert times for 100,000 beams on a standard CPU (Intel E8600, 3.3 GHz) are given in Fig. 11.

In our experiments, single 3D scans usually consist of about 90,000 non-maxrange measurements and the time to acquire the data using a SICK LMS on a pan-tilt unit is about 6 s. Typically, such a scan can be integrated into the map in less than one second. Even with long measurement beams and high map resolutions, updating the map will not take longer than a few seconds.

V. CONCLUSION

In this paper, we presented an approach for the 3D modeling of environments that is relevant for several robotic tasks including robot mapping, navigation, and mobile manipulation. It builds upon a tree-based map structure which facilitates multi-resolution map queries and leads to a compact memory representation. Using probabilistic occupancy estimation, our approach is able to represent full 3D models including free and unknown areas. The proposed approach uses a bounded per-volume confidence. This allows for an lossless compression scheme which substantially reduces memory usage. We evaluated our approach using both real-world and simulated data sets. The results demonstrate that our approach is able to model the environment in an accurate way and at the same time minimizes memory requirements.

We implemented the described system and made the implementation available as an open source C++ library to facilitate future developments in the context of 3D mapping.

REFERENCES

[1] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," in *Proceedings of Eurographics*, Amsterdam, The Netherlands, August 1987.

[2] D. Cole and P. Newman, "Using laser range data for 3D SLAM in outdoor environments," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006, pp. 1556–1563.

[3] N. Fairfield, G. Kantor, and D. Wettergreen, "Real-time SLAM with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, 2007.

[4] J. Fournier, B. Ricard, and D. Laurendeau, "Mapping and exploration of complex environments using persistent 3D model," in *Computer and Robot Vision, 2007. Fourth Canadian Conf. on*, 2007, pp. 403–410.

[5] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3D perception and environment map generation for humanoid robot navigation," *Int. J. Rob. Res.*, vol. 27, no. 10, pp. 1117–1134, 2008.

[6] R. Hadsell, J. A. Bagnell, and M. Hebert, "Accurate rough terrain estimation with space-carving kernels," in *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[7] M. Hebert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, vol. 2, May 1989, pp. 997–1002.

[8] G. Kraetzschmar, G. Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," in *Proc. of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, M. I. Ribeiro and S. J. Victor, Eds., Lisbon, Portugal, July 2004.

[9] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, 1982.

[10] H. Moravec, "Robot spatial perception by stereoscopic vision and 3D evidence grids," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-34, September 1996.

[11] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, St. Louis, MO, USA, 1985, pp. 116–121.

[12] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM—3D mapping outdoor environments: Research articles," *J. Field Robot.*, vol. 24, no. 8-9, pp. 699–722, 2007.

[13] K. Pathak, A. Birk, J. Poppinga, and S. Schwertfeger, "3D forward sensor modeling and application to occupancy grid based sensor fusion," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007, pp. 2059–2064.

[14] P. Payeur, P. Hebert, D. Laurendeau, and C. Gosselin, "Probabilistic octree modeling of a 3-d dynamic environment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1997.

[15] Y. Roth-Tabak and R. Jain, "Building an environment model using depth information," *Computer*, vol. 22, no. 6, pp. 85–90, Jun 1989.

[16] J. Ryde and H. Hu, "3D mapping with multi-resolution occupied voxel lists," *Autonomous Robots*, pp. 1–17, 2010.

[17] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[18] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *International Journal for Robotics Research (IJRR)*, vol. 28, no. 5, pp. 595–599, May 2009.

[19] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[20] J. Wilhelms and A. Van Gelder, "Octrees for faster isosurface generation," *ACM Trans. Graph.*, vol. 11, no. 3, pp. 201–227, 1992.

[21] M. Yguel, O. Aycard, and C. Laugier, "Update policy of dense maps: Efficient algorithms and sparse representation," in *Field and Service Robotics, Results of the Int. Conf., FSR 2007*, vol. 42, 2007, pp. 23–33.

[22] M. Yguel, C. T. M. Keat, C. Braillon, C. Laugier, and O. Aycard, "Dense mapping for range sensors: Efficient algorithms and sparse representations," in *Proceedings of Robotics: Science and Systems*, June 2007.