2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Finding the Adequate Resolution for Grid Mapping - Cell Sizes Locally Adapting On-the-Fly

Erik Einhorn and Christof Schröter and Horst-Michael Gross
Neuroinformatics and Cognitive Robotics Lab, Ilmenau University of Technology, Germany

***Abstract*— For robot mapping occupancy grid maps are the most common representation of the environment. However, most existing algorithms for creating such maps assume a fixed resolution of the grid cells. In this paper we present a novel mapping technique that chooses the resolution of each cell adaptively by merging and splitting cells depending on the measurements. The splitting of the cells is based on a statistical measure that we derive in this paper. In contrast to other approaches the adaption of the resolution is done online during the mapping process itself. Additionally, we introduce the Nd-Tree, a generalization of quadtrees and octrees that allows to subdivide any d-dimensional volume recursively with N^d children per node. Using this data structure our approach can be implemented in a very generic way and allows the creation of 2D, 3D and even higher dimensional maps using the same algorithm. Finally, we show results of our proposed method for 2D and 3D mapping using different kinds of range sensors.**

I. Introduction and Related Work

In mobile robotics map building is an important prerequisite for different navigational tasks. It provides a model of the environment that is essential for collision avoidance, path planning and localization. In the past, a variety of different techniques for representing the environment of a robot have emerged. They can be roughly classified into metric and topological maps. Topological maps [1], [2], [3] represent the robot's environment in a graph-like structure, where the nodes correspond to distinctive places, situations or landmarks. Two nodes are connected by an edge if there exists a path for traveling between the places. Topological maps are therefore sparse and compact representations that are suitable for highly efficient path planning and localization in large-scale environments.

Metric maps on the other hand are accurate and fine grained metric descriptions of the environment. Currently, occupancy grid maps [4] are probably the most widely used metric maps. They partition the robot's surrounding into regular cells, where each cell stores a probability whether the cell is occupied by an obstacle or free. Most existing occupancy grid mapping algorithms decompose the high-dimensional mapping problem into smaller one-dimensional problems by modelling the map as a Markov Random Field of order 0, where the state of each individual cell can be estimated as an independent random variable.

In the past, the maps usually have been built using sonar sensors and laser range finders that are able to measure the distance of nearby objects in a single plane only. Due to this two-dimensional characteristic, maps created using these sensors are also two-dimensional. Recently, different sensors like stereo cameras, time-of-flight cameras, Microsoft's Kinect or 3D lasers have gained more importance which are able to obtain three-dimensional information about the local surroundings. Consequently, 3D representations are necessary to take full advantage of these sensors. In previous works, multi-level surface maps [5], regular voxel representations, and octrees [6], [7] have been proposed as a 3D alternative to 2D occupancy grid maps. However, the common disadvantage of these existing approaches like "OctoMap" [6] is, that they partition the environment into regular cells with a fixed resolution. The choice of this cell resolution is crucial. Today's navigational tasks like inch-perfect navigation in narrow environments require precise maps with a high resolution. Unfortunately, an increased resolution results in a heavy rise of the memory consumption and the computational costs especially when using 3D maps. On the other hand a too coarse resolution may lead to inconsistencies and to maps with insufficient precision.

In previous works quadtrees [8] and octrees [6] are used for reducing the amount of memory while preserving the precision of the maps. In these approaches the trees are only used to compress the maps *after* they have already been built by fusing cells with similar occupancy probabilities. The mapping itself is done using a fixed resolution.

In order to overcome this limitation, we here present a novel mapping technique that chooses the resolution of each cell adaptively depending on the measurements *during* mapping. If an additional measurement becomes available that conflicts with the current map estimate, i.e. the states of the existing occupancy cells, the measurement is either an outlier or the inconsistency is caused by a too coarse resolution of the map. In the latter case the affected cells are subdivided to increase the resolution. Simultaneously, neighboring cells with similar occupancy values are merged to reduce the spatial and computational complexity.

In the past approaches, different data structures are applied for storing the occupancy values. Usually, 2D or 3D arrays are used. Some approaches also apply quadtrees and octrees to improve the speed for operating on the data. Here, we present a data structure that is a generalization of quadtrees and octrees. It is not restricted to a certain dimensionality. Therefore, our approach can be implemented in a very generic way and allows the creation of 2D, 3D and even higher dimensional maps using the same implementation of the algorithm without any additional overhead. This allows

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2011) under grant agreement #216487

us to apply the algorithm for different range sensors that provide two- and three-dimensional data.

This paper is organized as follows. In the next section we specify the underlying generic data structure which is fundamental for our mapping approach, that is described in section III in detail. In section IV we show results that we have obtained using the presented approach for different kinds of sensors. Finally, we conclude with an outlook for future work.

II. Representing Maps using Nd-trees

When using occupancy grid maps for path planning, obstacle avoidance, localization or mapping most of the operations that are performed on these maps include intersection tests or require a fast traversal through the map. Obstacle avoidance for example requires collision tests of the robot's shape with the occupied cells of the map. Moreover, when creating the maps using range sensors like laser range finders, each sensor measurement affects the cells along a ray emerging from the sensor. Hence, all cells along that ray need to be traversed and updated.

Since many of these operations must be processed in real-time, computational efficiency is crucial here. Hence, storing geometric data like occupancy grid maps in plain arrays can be unfavorable since these data structures do not allow efficient algorithms for intersection tests. Instead, quadtrees and octrees are more suitable for this purpose.

Quadtrees (or more precisely *region quadtrees*) can be used to represent a region of the 2D space. They decompose the region by recursively subdividing it into four equal quadrants. The subdivision is represented using a tree, where each inner node has four children. The root node represents the whole region that is covered by the quadtree. The leaf nodes finally represent a small partition of the whole region and contain attributes that characterize that partition. In the case of occupancy grid maps, each leaf node contains the occupancy probability of the region covered by the node.

Octrees are the three-dimensional analog of quadtrees. Similar to quadtrees, an octree node is bisected in each dimension. Hence, each node is subdivided into $2^3 = 8$ octants.

In [9] and [10] a more generalized form of an octree is presented. It allows to divide the nodes in each dimension by an arbitrary number N rather than 2. The resulting structure is called N^3-tree, since each node is subdivided into N^3 uniform children. The special case of $N = 2$ results in a standard octree. Choosing different values for N changes the properties of the data structure and the performance of the algorithms working on it. While a small N produces deep trees that are well suited for intersection tests, a large N results in shallow trees that are more convenient for tree traversals but are less memory efficient. By adapting N a trade-off between efficiency for intersection tests and traversal efficiency is possible.

Based on the above N^3-tree we present the N^d-tree as a more general data structure in the following, that is not restricted to a certain dimensionality. Instead it can be used to represent an occupancy map $M \subset \mathbb{R}^d$ with an arbitrary dimensionality d.

The N^d-tree is a connected graph $T = (N, E)$ without cycles, consisting of nodes N and edges E with the following properties:

(1) Each node $n \in N$ represents a certain partition $\mathcal{V}(n) \subset M$ of the entire map M. For two-dimensional maps this partition is a square region. For three or more dimensions the partition in general is an axis-aligned cube-shaped volume or hyper-volume that has the same extent in each dimension. In accordance to occupancy grid maps these volumes are also called cells in the following.
(2) The volume of each inner node n is subdivided along each dimension into N intervals resulting in a total number of N^d uniform sub-volumes.
(3) Each sub-volume is associated to a child of the node. Hence, each inner node n has N^d children. Each child $c^{(j)}$ can be retrieved by the function $child(n, j)$ for $j = 0, \dots, N^d - 1$, where $(n, c^{(j)}) \in E$.
(4) The sub-volumes of the node are complete and disjoint, i.e. $\mathcal{V}(n) = \bigcup_{j=0}^{N^d-1} \mathcal{V}(child(n, j))$ and $\mathcal{V}(child(n, j)) \cap \mathcal{V}(child(n, k)) = \emptyset, j \neq k$.
(5) There's exactly one root node n_0 that has no parent.
(6) Nodes that are not further subdivided are called leaves. For each leaf n an occupancy value $occ(n)$ is assigned that specifies the probability of the volume $\mathcal{V}(n)$ covered by the node n being occupied, similar to each cell of a regular occupancy grid map.

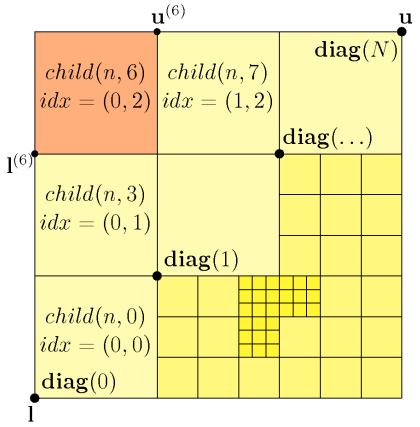

Fig. 1. Example of an N^d-tree with $N = 3$ and $d = 2$. The outer root node is divided into 3×3 child nodes, where some of these children are further subdivided. For some children their Euclidean index in the grid is indicated. Moreover, the diagonal vertices are shown that are used for computing the extents of each sub-volume when splitting the node.

A. Addressing child nodes

In the above definitions each child node $child(n, j)$ is addressed using an integral index $j \in [0, N^d - 1]$. However, in some algorithms it is convenient to use a different addressing scheme, that can be derived as follows. Splitting the volume of each node by subdividing it along each dimension into N intervals results in a regular d-dimensional grid, where each sub-volume of each child has its definite position as shown in Fig. 1. This position can be described using an index

$idx = (i_1, \ldots, i_d) \in I$ with $I = [0, N-1]^d$. Each component i_k of the index specifies the location (row, column, etc.) of the sub-volume along the k-th dimension in the parent's grid. Consequently, the j-th child node $child(n, j)$ can also be addressed by $child(n, idx)$ with $idx = (i_1, \ldots, i_d)$, where

$$j = \sum_{k=1}^{d} i_k N^{k-1}$$

Vice versa, an inverse mapping from the integral index j to the Euclidean index $idx = (i_1, \ldots, i_d)$ is given by:

$$i_k = \left\lfloor \frac{j}{N^{k-1}} \right\rfloor \bmod N$$

B. Splitting Nodes

One of the most important operations that is required by our mapping approach is the splitting of nodes. When a node is split, the volume $\mathcal{V}(n)$ covered by the node n must be subdivided in order to create its child nodes. Since the volume is cube-shaped and axis-aligned it can be described by its lower vertex $\mathbf{l} \in \mathbb{R}^d$ and upper vertex $\mathbf{u} \in \mathbb{R}^d$. For subdividing the volume the extents of the sub-volumes need to be computed. Therefore, we first compute the positions of $N+1$ vertices $\mathbf{diag}(i) \in \mathbb{R}^d$ along the diagonal passing from $\mathbf{l}$ to $\mathbf{u}$ according to:

$$\mathbf{diag}(m) = \frac{N-m}{N}\mathbf{l} + \frac{m}{N}\mathbf{u}, \quad m = 0, \ldots, N$$

As indicated in Fig. 1, using these diagonal vertices $\mathbf{diag}(m) = (diag_1(m), \ldots, diag_d(m))^\top$, the sub-volume of the j-th child node with index $idx = (i_1, \ldots, i_d)$ can be specified by its lower vertex $\mathbf{l}^{(j)} = (l_1^{(j)}, \ldots, l_d^{(j)})^\top$ and upper vertex $\mathbf{u}^{(j)} = (u_1^{(j)}, \ldots, u_d^{(j)})^\top$ as follows:

$$l_k^{(j)} = diag_k(i_k)$$
$$u_k^{(j)} = diag_k(i_k + 1), \quad k = 1, \ldots, d$$

C. Growing

When mapping the environment on-line on the robot, the map being created needs to be resized constantly in each dimension. One big advantage of representing occupancy maps using a tree as presented in this paper is that this can be done dynamically without reallocating or copying the whole memory that is used to store the map. In order to increase the area that is covered by the map, a new node needs to be created that has a coarser resolution than the root node. Its volume will be N times bigger in each dimension than the volume of the root node. The current root node will then become a child of the newly created node that itself becomes the new root node. This procedure is repeated until the map covers the desired area.

D. Ray Casting

For inserting sensor measurements into the map, the nodes need to be traversed efficiently in order to update all cells along the sensor beam. For this purpose we apply a hierarchical ray casting algorithm. The ray casting starts a the origin of the ray. Therefore, the leaf that contains the origin is searched in top-down descent. Afterwards, all neighboring siblings on the same level are traversed along the direction of the ray using a similar algorithm as described in [11]. When all children of a certain node have been traversed, the traversal continues on a coarser level. If a new node on a coarser level is entered, the algorithm checks if the node is subdivided into sub-nodes. In that case the traversal is again performed on the finer level by visiting the children of that node as shown in Fig. 2. This hierarchical ray casting ensures, that all leaf nodes along the ray are visited while the number of inner nodes that need to be entered is minimized.

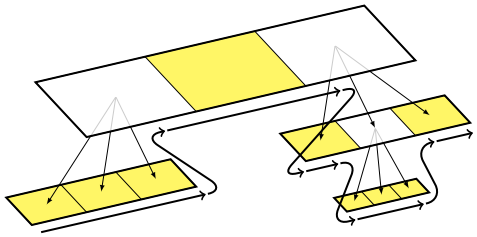

Fig. 2. Hierarchical ray casting in an N^d-tree with $N = 3, d = 1$. Leaf nodes are indicated as yellow regions, while inner nodes are shown in white. They are further subdivided and their children are drawn on a lower level. When performing a ray cast from left to right, the algorithm will traverse the tree along the indicated path in order to visit all leaf nodes in the correct order.

III. Adaptive-Resolution Mapping

As stated above, most mapping algorithms treat each cell of the occupancy map as independent random variable and estimate its occupancy probability individually. Moreover, they assume that each cell is either entirely occupied by an obstacle or completely free. Partially occupied cells are not considered. These assumptions are also valid for our mapping approach for range-based sensors that we describe in the following.

In this approach we use the N^d-tree that was described in the previous section. For mapping we usually start with an empty map that contains a single root node n_0 only. The occupancy value of the root node is set to $occ(n_0) = 0.5$ to indicate that nothing is known about the robots environment. The extent of the root node determines the minimum resolution of the map, i.e. new cells that are added to the map have an equal size or are smaller.

Each measurement that should be inserted into the map is defined by the sensor's origin $\mathbf{o}$, the direction of the measurement given by the vector $\mathbf{d}$ and the measured range r that indicates the distance of the obstacle in that direction. Using this information the end point of the measurement can be obtained by $\mathbf{e} = \mathbf{o} + r\mathbf{d}$.

When inserting a new range measurement z_t the map is grown as described in the previous section, until the origin and the end point of the measurement fit into the volume that is covered by the map. Similar to the initial root node the occupancy value of all newly created leaf nodes is initialized with 0.5. Afterwards, all cells along the sensor beam are traversed using the described ray casting algorithm in order to update their occupancy probability. Since the measurements are error-prone we apply a probabilistic map update that integrates the sensor measurements over time in order to reduce the effects of noisy measurements. The new

occupancy probability $occ(n_i) = p(n_i|z_{1:t})$ of each traversed leaf node n_i can be updated recursively from its previous value $p(n_i|z_{1:t-1})$ (which only takes previous measurements $z_{1:t-1}$ into account) using Bayes rule [4] as follows:

$$p(n_i|z_{1:t}) = 1 - \left[1 + \frac{p(n_i|z_{1:t-1})}{1-p(n_i|z_{1:t-1})} \frac{p(n_i|z_t)}{1-p(n_i|z_t)}\right]^{-1}$$

where $p(n_i|z_t)$ denotes the *inverse sensor model*. Here, we apply a one-dimensional inverse sensor model that takes the Gaussian error distribution of the measurements into account. While most researchers use an approximated sensor model, we derived our inverse model analytically. It can be described by:

$$p(n_i|z_t) = \mathrm{F}\left(\check{n_i}, s, \sigma^2\right) - \frac{1}{2}\mathrm{F}\left(\hat{n_i}, s, \sigma^2\right)$$

where $\mathrm{F}\left(x, \mu, \sigma^2\right) = \int_{-\infty}^{x} \mathcal{N}(\mu, \sigma^2)$ is the cumulative normal distribution. Moreover, $\check{n_i}$ and $\hat{n_i}$ denote the distance to the entry and exit point of the sensor beam with the volume covered by the node n_i. The variance σ^2 expresses the uncertainty of the measurement.

A. Splitting

When using the mapping algorithm described so far, all created leaf nodes will have the same fixed extent that was defined by the initial root node as described before. However, it was already mentioned that a fixed resolution has several disadvantages. Therefore, we add a mechanism that splits nodes to increase the resolution when this is appropriate. For each node we additionally store a histogram that contains the number of *hits* and *misses*. Hits are those measurements that end in the volume associated to the node. They indicate that the volume is occupied by an obstacle. Misses are the measurements that pass through the volume and end in a cell somewhere behind. They indicate that the volume is free.

If we assume a volume that is entirely occupied, a noise-free sensor would obtain exactly n hits and 0 misses out of n measurements. Vice versa, it would yield exactly n misses and 0 hits, if the volume was completely free. However, if the volume is partially occupied, some measurements will hit the obstacle while others will miss it and hence, both the number of hits and misses will be non-zero. Consequently, the number of hits and misses can be used to determine if a certain node violates the assumption that its volume is either entirely occupied by an obstacle or completely free. In this case the node needs a further refinement and must be subdivided.

However, in practice the sensor measurements are error-prone. Due to erroneous measurements, a real sensor usually will yield misses even if the volume is entirely occupied. In the following the error rate for those false negative detections is denoted by the probability $p(miss|occ)$. Additionally, the sensor may obtain hits even if the volume is completely free. The error rate for these false positive detections is expressed by the probability $p(hit|free)$. Both probabilities can be measured for a given sensor. Using these probabilities we can now compute the expected number of hits and misses for a certain number of n measurements. For an entirely occupied volume one would expect the number of hits and misses to be as follows:

$$\begin{aligned} E(hits|occ) &= n \cdot [1 - p(miss|occ)] \\ E(misses|occ) &= n \cdot p(miss|occ) \end{aligned}$$

The corresponding discrete distribution is shown in Fig. 3a as a histogram. Vice versa, the number of hits and misses for an entirely free cell is expected to be:

$$\begin{aligned} E(hits|occ) &= n \cdot p(hit|free) \\ E(misses|occ) &= n \cdot [1 - p(hit|free)] \end{aligned}$$

The corresponding histogram is indicated in Fig. 3b.

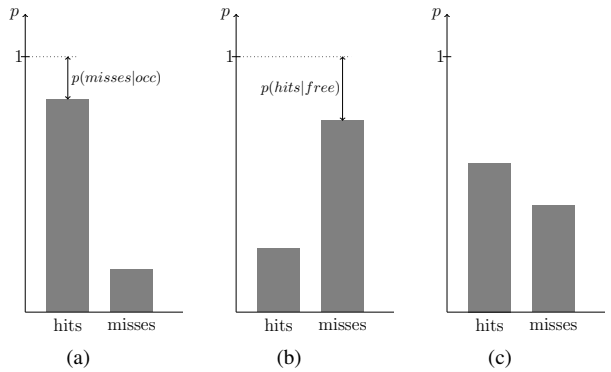

Fig. 3. Histograms of the expected distributions of hits and misses. (a) Expected distribution for an entirely occupied cell that is measured with a noisy sensor. (b) Expected histogram of hits and misses for a completely free cell. (c) Histogram of actual measured hits and misses for a node that most likely is partially occupied and needs to be split, since the histogram differs significantly from both histograms shown on the left.

If the counted hits and misses of a node differ considerably from the two expected distributions above, the discrepancy cannot be explained by erroneous measurements since these errors are already modeled in the expected distributions. Instead it must be caused by a violation of the assumption that the volume is entirely occupied or free. For testing if the measured distribution differs significantly from the expected ones, we apply the Chi-square test. Therefore we compute the values of the test-statistics as follows:

$$\chi^2_{occ} = \frac{[hits - E(hits|occ)]^2}{E(hits|occ)} + \frac{[misses - E(misses|occ)]^2}{E(misses|occ)}$$

$$\chi^2_{free} = \frac{[hits - E(hits|free)]^2}{E(hits|free)} + \frac{[misses - E(misses|free)]^2}{E(misses|free)}$$

Finally, a node needs to be split if $\min(\chi^2_{occ}, \chi^2_{free}) > \chi^2_{1,0.005}$, i.e. if the measured distribution differs significantly from both expected distributions with a statistical significance of $\alpha = 0.995$. The occupancy value of the newly created child nodes is set to 0.5.

One drawback of the applied statistical test is that it will also yield significant differences to the expected histograms, if the obtained sensor measurements contain less erroneous measurements than expected. If for example all measurements of an entirely occupied cell coincidentally are

hits while at least some measurements are expected to be misses, the resulting histogram may differ significantly from the expected ones. We handle these cases where the sensor measurements are more accurate than expected separately and skip the statistical tests to prevent an unnecessary subdivision of the cell.

B. Merging

Beside splitting, we also merge nodes to reduce the number of leaf nodes and to achieve some kind of lossless compression similar to [6]. The child nodes $c^{(j)}$ of a node n can be merged, if all occupancy values $occ(c^{(j)})$ are either very close to 0 or very close to 1. In the first case all child nodes are assumed to be free with a very high probability and in the second case all child nodes are most likely occupied. In both cases a significant change in the probability of any child is unlikely. Hence, the region covered by the children can also be represented by their parent node without any information loss. This is achieved by pruning the child nodes. The parent node n will then become a leaf and its occupancy value is set to the mean of its former children: $occ(n) = \frac{1}{N^d} \sum occ(c^{(j)})$. This merging process is done after each measurement update cycle for all updated nodes that fulfill the above requirements.

Using the proposed method for splitting and merging nodes adaptively, the cell resolution of the created map will be adjusted adequately. Regions that are homogeneously covered by obstacles or that are entirely free are represented by coarse cells, while more fragmented regions are mapped using fine-grained cells in a higher resolution. The maximum resolution usually is restricted by the sensor noise only. The subdivision of cells will stop automatically if no higher precision can be achieved by further subdividing a cell due to the sensor's noise. Additionally, the resolution can be limited by specifying a maximum level of subdivision.

IV. Results

We have tested the proposed algorithm for adaptive resolution mapping using three different sensors on our mobile robot platform. Fig. 4 shows 2D maps that were created using a 2D laser range finder. The maps cover an area of 26 m $\times$ 16 m. In Fig. 4a the leaf nodes of the created N^d-tree map are indicated. The width of the shown cells varies from 1.6 m to 0.05 m. Hence, the effective resolution of the map is 0.05 m. This high precision takes full advantage of the accuracy of the laser range finder. Fig. 4b shows the same N^d-tree map, where the leaf nodes are colored according to their occupancy value. Occupied cells are shown in white while free cells are shown in black. Gray indicates an occupancy value of 0.5. For those cells it is unknown whether they are free or occupied since they have not been observed by the laser range finder.

For comparison, a normal occupancy grid map with a fixed resolution of 0.1 m is shown in Fig. 4c. It was created using the same sensor data. When comparing Fig. 4b and Fig. 4c, it becomes apparent that the N^d-tree has a higher level of detail due to its high resolution of up to 0.05 m. Moreover, some fine-grained structures are not even visible

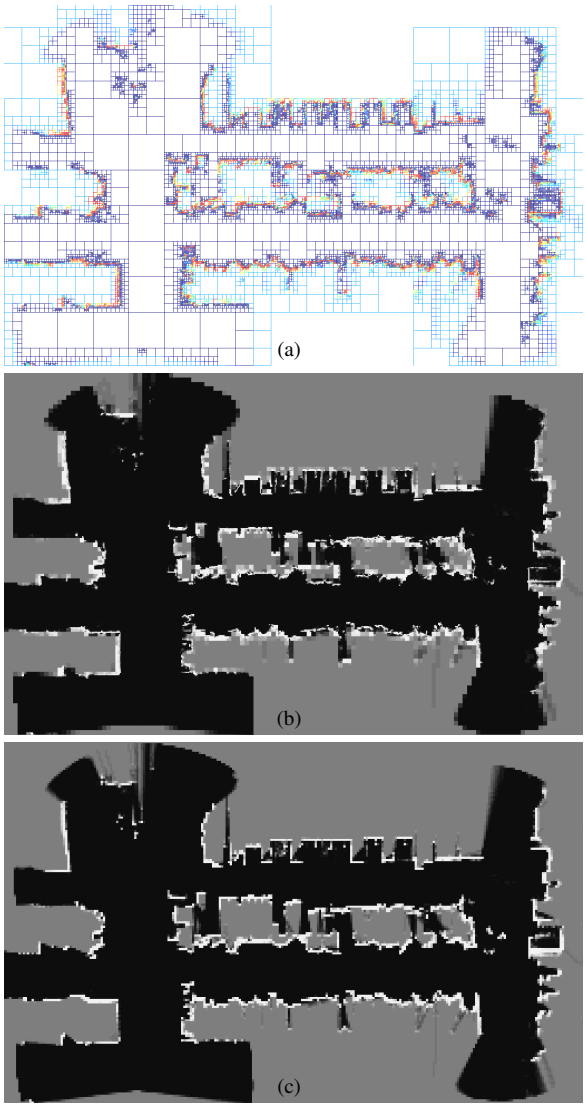

Fig. 4. (a) Leaf nodes of an N^d-tree map with $N = 2, d = 2$ (b) Same N^d-tree map where the leaf nodes are colored according to their occupancy value. (white=1, black=0) (c) Grid map with a fixed resolution of 0.1 m

in the fixed-resolution grid map. Additionally, the fixed-resolution grid map consists of 42,120 cells, while the N^d-tree contains 21,857 nodes, where only 16,390 nodes are leaf nodes. Hence, using our approach a reduction of approx. 50% in the required amount of memory can be achieved for the shown map while doubling the effective resolution.

In Fig. 5 and 6 two 3D maps are shown that were created using the same adaptive mapping approach. Due to the generic characteristic of the N^d-tree we were able to use the same implementation as for creating the 2D maps above. The three-dimensional measurements for the map in Fig. 5 were obtained using a Kinect depth camera. The measurements for Fig. 6 were generated using a feature-based shape-from-motion approach [12] that is able to obtain a reconstruction of the local surroundings using a monocular camera. Each reconstructed feature is used as a range measurement and inserted into the map. In both 3D maps only those cells that were estimated as occupied are shown using different colors, where the color codes the height of each cell.

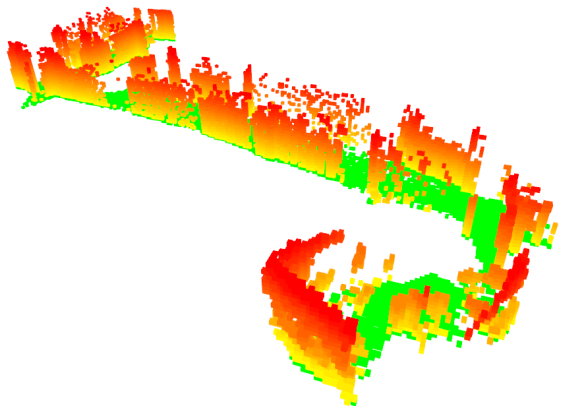

Fig. 5. 3D map based on an N^d-tree with $N = 2, d = 3$ with a resolution of up to 0.05 m. The 3D data was obtained using a Kinect depth camera.

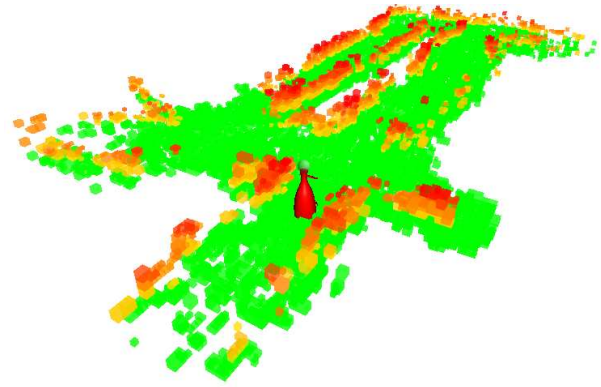

Fig. 6. 3D map with a resolution of up to 0.10 m. The data was obtained using a monoclar approach for scene reconstruction [12]

Fig.7 shows a comparison of our adaptive mapping approach with a mapping algorithm that uses a fixed resolution (e.g. "OctoMap" [6]). We compared the number of cells in the final maps shown in Fig. 4 and Fig. 5. For the 2D map the number of cells reduced to $16,390$ which is only 13% of the cells needed when using a fixed resolution approach. For the 3D map the adaptive resolution approach requires about one third of the cells compared to a fixed resolution algorithm for representing the same content. The decreased number of cells results in less memory consumption and also in a significant reduction of the computational costs during mapping, since less cells need to be updated when inserting new sensor measurements. This is shown in the lower part of the table, where the total number of cell updates is compared. Our adaptive approach needs to update only 30%-50% of the cells compared to a fixed resolution approach leading to a performance gain of the factor 2-3.

		2D map (Fig. 4)	3D map (Fig. 5)
Cells	**adaptive**	16,390	93,815
	fixed	120,346	252,582
	ratio	13.6%	37.1 %
Updates	**adaptive**	$10 \cdot 10^6$	$71 \cdot 10^6$
	fixed	$33 \cdot 10^6$	$141 \cdot 10^6$
	ratio	33%	49%

Fig. 7. The total number of cells and cell updates for the 2D and 3D maps shown in Fig. 4 and Fig. 5. The number of cells and updates needed by the proposed adaptive approach is compared with a fixed resolution algorithm. Lower values correspond to less memory consumption and less computational costs.

V. Conclusion

In this paper we have presented a data structure and algorithms for creating occupancy maps with an adaptive resolution. During mapping the appropriate resolution is chosen dynamically by splitting cells of the map when the number of conflicting measurements exceeds the expected number of erroneous measurements due to sensor noise. The maps created using the proposed approach provide a high accuracy and are suitable for high precision navigation. Additionally, the number of cells that are necessary to represent the maps can be decreased. This results in a reduction of the memory consumption and the computational costs. We have shown that the same algorithm can be used to create two-dimensional maps using a laser range finder as well three-dimensional maps using depth cameras and monocular scene reconstruction.

Beside the statistical Chi-square test for detecting if a cell needs to be further refined, different measures could be applied. At the moment we are working on a probabilistic measure that computes the probability whether a cell is partially occupied. Combined with an information theoretic merging procedure, that merges those cells where only a small loss of information is expected, we try to further improve the presented approach.

References

[1] M. Liu, D. Scaramuzza, C. Pradalier, R. Siegwart, and Q. Chen, "Scene Recognition with Omnidirectional Vision for Topological Map using Lightweight Adaptive Descriptors," in *IROS*, 2009.

[2] H. Jacky Chang, C. S. George Lee, Y. Charlie Hu, and Yung-Hsiang Lu, "Multi-Robot SLAM with Topological/Metric Maps," in *IROS*, 2007, pp. 1467–1472.

[3] S. Thrun, "Learning Metric-Topological Maps for Indoor Mobile Robot Navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[4] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[5] R. Triebel, P. Pfaff, and W. Burgard, "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," in *IROS*, 2006.

[6] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems," in *ICRA, Workshop on 3D Perception and Modeling*, 2010.

[7] P. Payeur, P. Hébert, D. Laurendeau, and C. M. Gosselin, "Probabilistic Octree Modeling of a 3-D Dynamic Environment," in *ICRA*, vol. 2, 1997, pp. 1289–1296.

[8] G. K. Kraetzschmar, G. Pagès Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," in *Proc. of the 5th Symp. on Intelligent Autonomous Vehicles*, 2004.

[9] S. Lefebvre, S. Hornus, and F. Neyret, *GPU Gems 2*. Addison-Wesley Professional, 2005, ch. Octree Textures on the GPU, pp. 595–613.

[10] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann, "GigaVoxels : Ray-Guided Streaming for Efficient and Detailed Voxel Rendering," in *Proc. of the Symp. on Interactive 3D Graphics and Games*, 2009, pp. 15–22.

[11] J. Amanatides and A. Woo, "A Fast Voxel Traversal Algorithm for Ray Tracing," in *Proc. of Eurographics*, 1987, pp. 3–10.

[12] E. Einhorn and H.-M. Schröter, Gross, "Monocular Scene Reconstruction for Reliable Obstacle Detection and Robot Navigation," in *Proc. ECMR*, 2009.