Efficient Automotive Grid Maps using a Sensor Ray based Refinement Process

Ruben Jungnickel[1], Michael Köhler[1] and Franz Korf[2]

***Abstract*— The occupancy grid mapping technique is widely used for environmental mapping of moving vehicles. Occupancy grid maps with fixed cell size have been extended using the quadtree implementation with adaptive cell size. Adaptive grid maps have proven to be more resource efficient than fixed cell size grid maps. Dynamic cell sizes introduce the necessity of a split and merge process to trigger the refinement of grid cells. This paper presents a novel ray-based refinement process in order to choose the appropriate resolution for the sensor observation. Based on measurement conflicts some approaches use an iterative refinement process until all conflicts are solved. In contrast this paper presents an non-iterative approach based on the sensor resolution. Using the measurement data efficiently we propose an algorithm, which solves the problem of partially free cells in an adaptive grid map. The proposed algorithm is compared against other widely used algorithms and methodologies.**

***Index Terms*— Evidential Grid, Occupancy Grid Map, quadtree, NDTree, sensor modelling**

I. INTRODUCTION

Accurate environmental perception is a requirement for advanced driver assistant systems (ADAS) and mandatory for highly automated driving (HAD). In order to further improve environmental perception systems it is necessary to provide accurate and comprehensive information of the static environment. As a modelling technique of the static environment, the occupancy grid framework proposed in [1] is widely used and has been applied successfully to multiple systems [2] [3]. The basic idea of occupancy grid mapping is to model the environment as a set of discrete cells containing probabilities of the presence (occupancy) or absence (freeness) of an object. The original version of occupancy grid mapping uses a fixed cell size. This leads to a conflict between a high accurate environment representation and memory (computational) efficiency. To avoid this trade-off, tree-based grids provide dynamic cell sizes. Tree data structures have been shown to reduce the memory footprint by a factor of ten or more [4]. Tree-based grids (like quadtrees) enable the user to increase (split) or decrease (merge) the resolution of a cell using its tree structure. They can be seen as an optimization technique to overcome the challenging demand of memory in a high resolution map or for mapping large outdoor environments.

[1]Ruben Jungnickel and Michael Köhler are with Ibeo Automotive Systems GmbH, System Development, 22143 Hamburg, Germany ruben.jungnickel@ibeo-as.com michael.koehler@ibeo-as.com

[2]Franz Korf is with the Department of Computer Science, Hamburg University of Applied Sciences, 20099 Hamburg, Germany franz.korf@haw-hamburg.de

A quadtree node is split by creating four equal sized sub nodes halving the size of the parent node in each dimension. The decision process of splitting and merging a cell is crucial. The techniques used for splitting and merging will control the amount of nodes and its resolution. Furthermore they control the memory and computational resources maintaining the map. This paper proposes a splitting process triggered by the measurement resolution. The measurement resolution is estimated by a inverse sensor model optimized for LIDAR (LIght Detection And Ranging) sensors. In most sensor models the free space area is bounded by unknown or occupied area. To choose the cell size of these areas correctly, a cell can be either treated as entirely free or occupied. Cells, which are measured free only can be naively treated as entirely free, where the major area of a cell is unknown [2]. A modified conflict based approach [5] tends to split free areas. In contrast the proposed algorithm splits cells at the free space border only. Figure 1 shows where a partially free cell

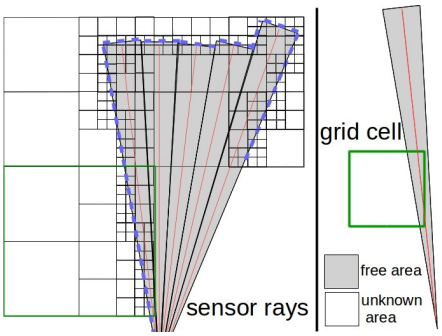


Fig. 1: green grid cell is partially free and should be split along free space border

should be split (right side). All corresponding cells along the border of the free space area needs to be refined (left side). This free space border (dashed blue line) is further described as free space polygon. A novel two step cell refinement process is proposed.

1) For the update of occupied cells the resolution of a measurement is estimated and corresponding cells are split until the requested resolution is reached.
2) Cells, which are not fully observed by a measurement needs to be further refined (figure 1). These cells are identified and split by calculating all intersecting cells along the border of a estimated free space polygon.

The two step procedure solves the problem of partially free cells by refining the crucial outline of a free space polygon (step 2) and leave other cells at coarser level.

We show on basis of a practical evaluation that our cell splitting approach is less CPU-intensive than an existing conflict-based approach. Furthermore we show its efficiency comparing the resulting map against its fixed resolution counterpart.

Requirements to a mapping process often varies depending on the use case. Two different use cases are considered:

- online mapping: The near environment of the vehicle needs to be mapped. The map is used for ADAS and only up to date measurement are of interest.
- offline mapping: The whole environment passing by the mapping vehicle should be mapped. The map building itself can be done offline. The resulting map can be used for navigation where online measurement are matched against the offline processed map.

While the offline mapping plays a role for robotic navigation [6], the online mapping is much more important for ADAS and HAD. In this paper we propose a cell ageing approach, which enables the algorithm to meet the requirement for online mapping. The cell ageing is an optional processing step and can be disabled for the offline purpose.

This paper is structured as follows. Section II gives a brief overview about the state of the art occupancy grid mapping algorithm using adaptive cell resolution techniques. In section III we present an inverse sensor model, which is suitable for multilayer multiecho LIDAR. Section IV introduces our adaptive Grid Map implementation in detail. Experimental results and a practical evaluation are given in section V. Finally in section VI conclusions and future work are discussed.

II. STATE OF THE ART GRID MAP MODELLING

In this section an overview of current adaptive grid map algorithm is given. Then, the Dempster Shafer theory (DST) is introduced, which is needed for the evidential measurement model and update process used in this paper.

A. Related Work

Over the last years, several adaptive grid mapping approaches have been studied. These approaches differ on the crucial decision of the splitting and merging process. Einhorn introduces the conflict-based splitting and merging process in [2]. Conflicts of a measurement are estimated by employing histograms for hits and misses iteratively. A χ^2 test is a statistical hypotheses test, which is used to identify cells that need to be split.

Joubert [5] introduces some further improvements on the base algorithm from [2]. This paper describes that using measurements conflicts only could lead to major inaccuracies where a cell is only partially free. Joubert extends the technique proposed in [2] with an additional counter for the unknown in order to solve the problem of partially free cells. Unknown counters are incremented for cells, which are not actively measurement free or occupied despite the cells are in the measurement range.

In [3] so-called subpavings are calculated, which are sets of aligned boxes to model a sensor ray. These multiple boxes are used to update a quadtree occupancy grid map. The quadtree is refined if a subpaving box has a higher resolution than the corresponding cell of the quadtree. In his first implementation the proposed algorithm takes exponential execution time.

In [7] Schmid presents an automotive occupancy grid map that is able to map 3D environments with a dynamic level of detail. Schmid proposes multiple level of detail functions that determine the cell size for different regions of the environment application specific. He proposes a level of detail function for landmark navigation and for pre-crash systems. In contrast to this work Schmid chooses the cell size application specific. In this paper we propose a measurement specific cell size adoption.

In [8] Hornung contributes an efficient 3D mapping framework based on octrees. For each cell measured free Hornung creates a cell at the maximum measurement resolution. This will create a ray of free measured cells at small cell size. Afterwards child nodes are pruned if they have the same occupancy state. In comparison to Hornung our 2D mapping approach splits free cells only if the current scan will create a free space border on the particular cell.

In this paper the N^D-tree representation from [2] is used and evaluated. In this term a quadtree is a 2^2-tree ($N = 2, D = 2$) splitting a cell into four sub nodes. The N^D-tree offers an abstract modelling technique for further improvements or evaluations with different D (dimension) or N (number of sub nodes for each dimension). Conflict-based splitting and merging proposed in [2] and [5] are computationally challenging. All measurements are used to update free and occupied counters stored in the map leaf nodes. Nodes were split if the $/chi^2$-test indicates a conflict between the occupied and free counters. This leads to an iterative process on the measurement data until all conflicts estimated by the χ^2 test are solved. For updating the free and occupied counters the CPU-intensive ray-tracing process needs to be executed multiple times. We propose a technique where the measurement data is used to update the map at once.

B. Dempster Shafer Theory on Grid Maps

DST as evidential theory is used to update free and occupied measurements independently [9]. Furthermore DST is used to give an evidence about the correct cell resolution in the proposed merge process. The DST is a mathematical theory allowing the combinations of evidences [10]. An evidence can be estimated by multiple sources and with an own degree of belief. In DST all possible hypotheses of a system are defined by a set Ω of mutually exclusive propositions. We define for occupancy grid mapping the set as $\Omega = \{f, o\}$ where f is the proposition for a free cell and o for an occupied cell. The elements of the power set $2^\Omega = \{\emptyset, \{f\}, \{o\}, \{f, o\}\}$ can be used to represent the actual state of the system. For each X sources (like sensors) DST supports a mass function $m_X : 2^\Omega \mapsto [0, 1]$ that assigns a belief mass to any element of the power set. The belief mass function $m_X(F)$ and $m_X(O)$ are used here to define beliefs for the free and occupancy propositions ($F = \{f\}$

,$O = \{o\}$, $U = \{f, o\}$), where X can be described as the origin of the belief mass estimation.

III. INVERSE SENSOR MODEL

Map cell belief masses need to be updated with respects to sensor measurement and its uncertainties. In this section an inverse sensor model for 2D grids that combines map cell belief masses and physical properties of a LIDAR sensor is proposed. We employ classified LIDAR data to identify echoes caused by ground (gnd), underdrivable objects (und), noise (nos) or from a valid target(val). The classification utilizes the vertical measurement resolution and is provided by the LIDAR. Based on classified LIDAR data a ray-based measurement model is developed, which provides belief masses for a free and occupied measurement in a so-called distance border (Db). Distance borders are used

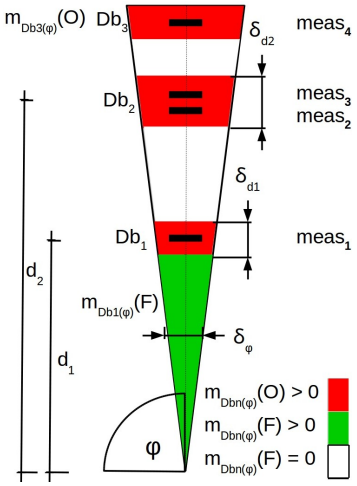


Fig. 2: Four measurements $meas_n$ create three distance borders ($Db_1 \dots Db_3$) for a single ray (topview)

to abstract the underlying sensor technology and can be described as target area hit by any sensor measurement. Please note that distance borders shape a bounded sensor model similar to bounded error models contributed in [3] by Langerwisch. Langerwisch pointed out that probabilistic error models could suffer on a overconfidence at the ray center. He compared bounded error sensor model against a original grid map with probabilistic sensor model. In contrast we propose a bounded evidential sensor model using classified LIDAR measurements.

Field	Description
ϕ	horizontal measurement angle
d_n	measured distance
δ_{dn}	distance uncertainty
δ_ϕ	angular uncertainty
$clas_n$	set of classification results
$m_{Db_n}(O)$	belief mass Db_n is occupied
$m_{Db_n}(F)$	belief mass area to Db_n is free

TABLE I: attributes of a distance border Db_n

A. Analyse Sensor Rays

As we are using multilayer LIDAR scanners the sensor model is built depending on individual rays splitting measurement in horizontal bins. One ray $S(\phi) := \{meas_1, meas_2, \dots, meas_i\}$ encompasses up to three scan layers (see figure 3). Due to the capability of the sensor to measure multiple distances (multi echo) a ray can contain $i = 9$ distance measurements ($meas_i$) (for three echoes per layer). Please note that the measurements $meas_2$ and $meas_3$ are near to each other and create one distance border Db_2 with a higher probability of occupancy. For the analysis of a single ray $S(\phi)$ we show three generic distance borders (Db_n) depicted in figure 2. For each measurement in the same ray $S(\phi)$ a new distance border is created. The distance borders attributes are shown in table I. The angular uncertainty δ_ϕ is mainly caused by the horizontal ray divergence (for our sensor $0.12°$). The distance uncertainty (δ_{dn}) depends on the sensor temperature and changes over time [3]. For each distance border a belief mass function calculates occupied belief mass $m_{Db_n}(O)$ and $m_{Db_n}(F)$ for the belief of freeness (see table II). The belief masses are stored in its distance border.

Figure 3 shows a right hand view of a three layer

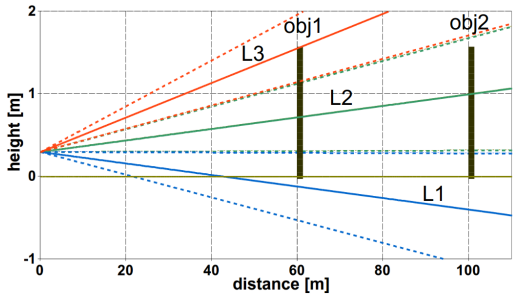


Fig. 3: vertical scan pattern of multilayer LIDAR on 1.6m solid object at 60m and at 100m distance (side view)

($L1, L2, L3$) LIDAR measurement. Consider the two solid objects ($obj1$, $obj2$). $Obj1$ is measurable with all depicted scan layers where $obj2$ in contrast is only measurable with $L1$ and $L2$. Therefore measurements can be classified to be possibly underdrivable und using the measured distance and its scan layer. An object at a distance $\geq 100m$, which is measured only with $L3$ could be underdrivable.

B. Estimation of Belief Masses for Measurements

In the proposed sensor model, distance borders are used to calculate belief masses. For the belief mass calculation several measurement dependent considerations should be made. A distance border could include measurements that are: ground (gnd), underdrivable (und), noise (nos) or valid (val). To incorporate all these kinds of measurement and create a belief mass for each Db_n of a ray $S(\phi)$ we use the equations of table II. Where the $measS(Db_n)$ function return the number of measurements used to create the distance border. The occupancy and free belief masses from table II are staged in three different mass levels $minOcc, minFree$ lower level, $occ, free$ mid level and $maxOcc, maxFree$ for the highest belief masses level. Measurements that were likely to be overdrivable (gnd) are regarded as free space. The area between the sensor and the gnd measured can be

$m_{Db_n}(O) =$	$\begin{cases} 0 & \text{if } Db_n.clas \cap nos = Db_n.clas \\ \alpha(\phi)minOcc & \text{if } Db_n.clas \cap und = Db_n.clas \\ \alpha(\phi)maxOcc & \text{if } measS(Db_n) > 1 \\ \alpha(\phi)occ & \text{else} \end{cases}$		$m_{Db_n}(F) =$	$\begin{cases} \alpha(\phi)maxFree & \text{if } Db_n.clas \cap gnd = Db_n.clas \\ 0 & \text{if } n > 1 \\ \alpha(\phi)minFree & \text{if } Db_1.clas \cap und = Db_1.clas \\ \alpha(\phi)free & \text{else} \end{cases}$	

TABLE II: measurement belief masses

seen as truly free. If there is a ray without any echo, we create a distance border at the visibility limit of the sensor using $m_{Db_n}(F) = \alpha(\phi) * free$. Where $\alpha(\phi) \mapsto [0, 1]$ is a weighting function to incorporate the angle specific range for a certain LIDAR.

Based on an approach proposed in [9] dynamic measurements (caused by moving objects) are detected and pruned. The dynamic detection is based on the arising conflict between belief masses generated from a distance border m_{Db} and belief masses from a maps leaf node m_L.

IV. ADAPTIVE OCCUPANCY GRID MAP

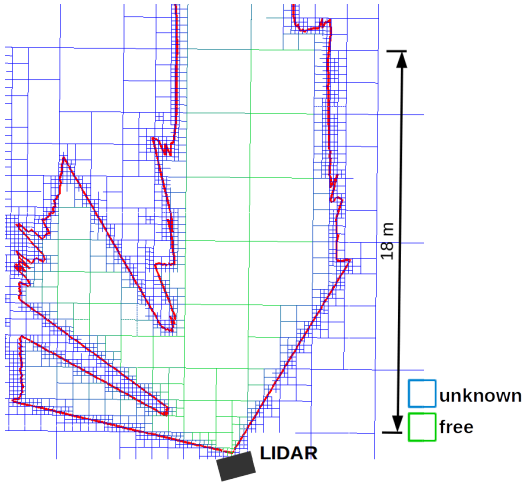


Fig. 4: The free space polygon (red) changes the cell size on its outline if appropriate.

Using the techniques for ray-based LIDAR modelling presented above, an evidential occupancy grid map with adaptive cell sizes can be updated. First we introduce an adaptive cell structure, which stores its resolution independently from cell size. Then, we present a free space refinement process, which first needs to build a free space polygon using the distance borders and then split cells intersecting the free space polygon edges. Neighbouring cells, where the belief mass state is equal are merged. Finally an ageing process for cells is presented, which enables continuous online operation.

A. Cell Structure and Modelling

As already described above we use a node modelling technique proposed by [2] named N^D-tree. Each node has N^D sub nodes (for 2D modelling N^2 sub nodes). While the cell size is controlled by any measured distance border we have to make sure that a cell is only split if a higher resolution is requested. So we equip our leaf nodes with a resolution attribute $node.res$ that is independent from the current cell size $node.size$. This enables us to store a measured cell resolution and only split cells if a measurement provides a higher resolution. A parent node node, where sub nodes are merged receives its resolution from the first containing leaf node.

The cell size is limited by the global maximum cell resolution $MaxR$. At initialization the root node size needs to be set. But the tree is able to grow in ascending and descending direction as described in [2]. So we are able to minimize the tree depth of our adaptive map tree. To enable a fast neighbour finding (needed for traversal algorithms) each node is equipped with its parent node index. This enables us via the function $node.neighbourNode()$ to find a requested neighbouring node at the same or higher level.

Algorithm 1: $split\ (node, distance\ border)$

```
Data: node n, distance border Db_k
if [res(Db_k) > n.res or MaxR > n.size] and n is leaf then
    update(n, Db_k);
    return;
else
    createSubNodes(n);
end if
N_sub = calcIntersectNode(n, Db_k);
for sub : N_sub do
    split(sub, Db_k);
end for
```

B. Splitting and Merging of Cells

The grid map refinement process is divided into two steps. First, if any cell is intersected by a distance border Db_n with $m_{Db_n}(O) > 0$, it will be split according to the ray resolution of the distance border ($res(Db_k)$). Algorithm 1 shows the split process scheme. The split function acts recursively for every intersected sub node (sub). The algorithm will terminate if the requested ray resolution ($res(Db_k)$) or the maximum cell resolution $MaxR$ is reached. All sub nodes which are intersected by a distance border ($calcIntersectNode$) are split and updated with belief masses from the distance

Algorithm 2: $merge\ (node)$

```
Data: node n
if n is no leaf then
    N_sub = getSubNodes(n);
    data = getFirstLeafData(N_sub);
    if allNodesEqual(N_sub, data) then
        n.dat = data;
        n.res = data.res;
        deleteAllSubNodes(n);
    else
        for sub : N_sub do
            merge(sub);
        end for
    end if
end if
```

border. The cell update process (called from algorithm 1 at line 2) is described in section IV-D. After the appropriate cells have been updated with new measurements, the merge process in algorithm 2 is triggered. The merge algorithm can be called using the current root node as parameter. It will merge all equal leaf nodes within the node n. For the merge operation an arbitrary belief mass of a leaf within node n needs to be determined ($getFirstLeafData$). This belief mass ($data$) is used to check all leafs nodes within n for equality. The merge process is triggered in descending direction and corresponds to a depth first search (DFS). Node equality is determined by using algorithm 3 traversing sub nodes in descending direction and checking condition τ from equation 1. A node is merged if free and occupied belief

Algorithm 3: $allNodesEqual\ (N_n, data)$

```
   input : node set N_n, belief masses data
   output: returnVal=true if all leaf nodes in N_n are equal
1  returnVal = true;
2  for n : N_n do
3      if n is leaf then
4          if not τ(data, n.dat) then
5              returnVal = false;
6              return returnVal;
7          end if
8      else
9          N_sub = getSubNodes(n);
10         returnVal = allNodesEqual(N_sub, data);
11     end if
12 end for
13 return returnVal;
```

masses for all sub nodes nodes are similar. Equation 1 shows the required conditions for a node merge. Note that the merge condition is mainly based on the unknown belief mass ($m_{n_n}(U)$). The cell resolution should remain high, if there is any doubt of the cells state (this is controlled by κ). The threshold variable λ controls the information loss caused by a cell merge ($(\lambda = 0) \implies$ no loss). The current node n_i has to be compared against a neighbouring node n_k.

$$\tau(n_i, n_k) = \begin{cases} true & \text{if } m_{n_i}(U) = 1 \wedge m_{n_k}(U) = 1 \\ false & \text{if } m_{n_i}(U) > \kappa \vee m_{n_k}(U) > \kappa \\ true & \text{if } \Xi(n_i, n_k, O) \wedge \Xi(n_i, n_k, U) \end{cases} \quad (1)$$

$$\Xi(n_i, n_k, A) = \begin{cases} true & \text{if } |m_{n_i}(A) - m_{n_k}(A)| \leq \lambda \\ false & \text{else} \end{cases}$$

C. Free Space Refinement Process

Using the distance border introduced in section III-A a free space polygon is built taking into account distance borders with a free belief mass $m(F) > 0$ only. Please note that distance borders with occupied belief mass $m(O) > 0$ are already used to split nodes with algorithm 1. The free space polygon F_p can be defined by equation 2 where $dist(Db)$ returns the measured distance of a particular distance border and $ray(Db)$ returns a set of all distance borders measured in the same ray. D_B is the set containing all distance borders of the current measurement cycle.

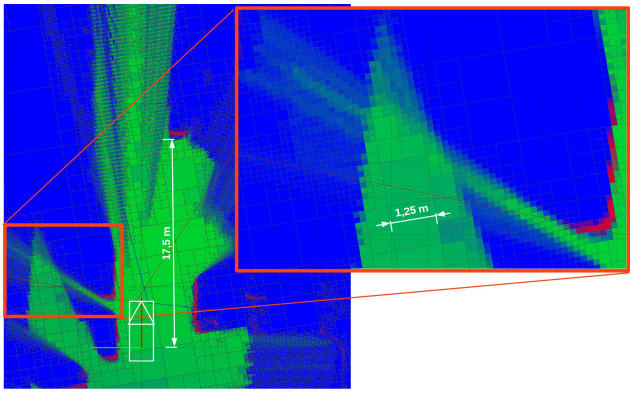
$$F_p = \{v | v \in D_B \wedge m_v(F) > 0 \wedge \forall k \in ray(v) : dist(k) \geq dist(v)\} \quad (2)$$

Employing the free space polygon, partially measured cells can be identified and split. Each cell which intersects the polygon edge needs to be split until a certain ray resolution is reached. The ray resolution is calculated according to the distance border uncertainties. To identify the particular cell we use a modified voxel traversal algorithm presented in [11]. At algorithm 4 the start S_P and end points E_P represent a polygon edge of the free space polygon. The fast voxel

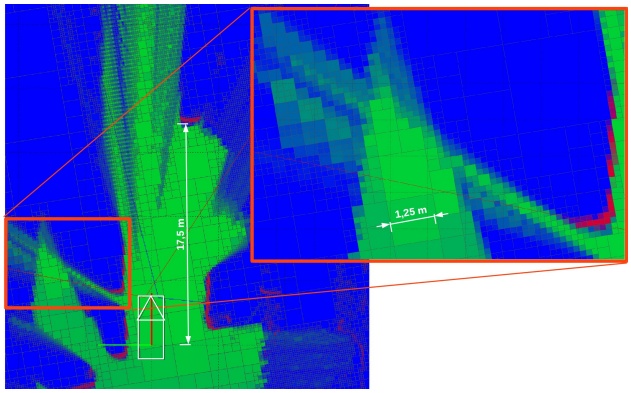
Algorithm 4: traversal split algorithm 2D

```
   Data: start S_P, end E_P, root node m
1  node = m.search(S_P);
2  node = node.getParent();
3  boundingBox= calcBoundingBox(S_P,E_P);
4  u⃗ = S_P;
5  t = 0;
6  while boundingBox contain u⃗ do
7      //initialize: tMaxX, tMaxY, tDeltaX, tDeltaY, stepX,
       stepY according to [11]
8      [X, Y] = node.getIndx(u⃗);
9      nextNode = node.get(X, Y)
10     while nextNode is leaf ∧ X, Y is in node do
11         split(nextNode);
12         if tMaxX < tMaxY then
13             tMaxX = tMaxX + tDeltaX;
14             X = X + stepX;
15             t = tMaxX;
16         else
17             tMaxY = tMaxY + tDeltaY;
18             Y = Y + stepY;
19             t = tMaxY;
20         end if
21         nextNode = node.getChild(X, Y);
22     end while
23     u⃗ = u⃗ + t v⃗;
24     if nextNode is leaf then
25         nextNode = nextNode.search(u⃗);
26     else if X, Y is not in node then
27         [X, Y] = node.neighbourIndx(X, Y);
28         node = node.neighbourNode(X, Y);
29     end if
30     node = node.getParent();
31 end while
```

traversal algorithm needs to be extended for non equidistant squares. The original algorithm is developed for equidistant voxel grids and is located in lines 10 to 22. So we treat the sub nodes $nextNode$ of a particular node $node$ as a single voxel grid. We place a second loop around the sub node traversal to ensure that we stay within the node's boundary and that we keep the traversal on the same level. The level of current node traversed ($node$) is always located one level higher than a leaf node. The inner loop is skipped if either the particular node $nextNode$ is no leaf node or the requested index $[X, Y]$ is not in the current node $node$. At line 23 of algorithm 4 the ray equation $\overrightarrow{u} + t\overrightarrow{v}$ is used to find the next point on the ray. Figure 4 show the result of algorithm 4. The free space polygon is shown in red, the grid cells are shown in blue. The grid status is shown after integrating

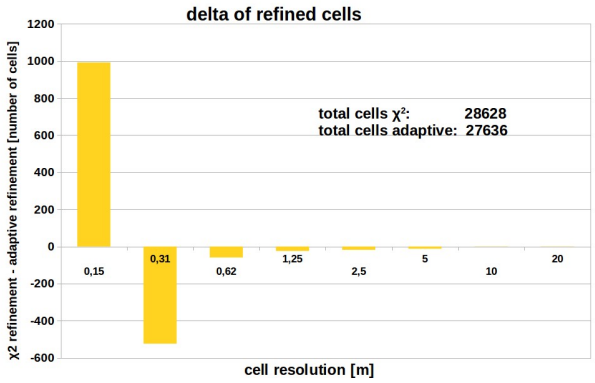


(a) χ^2 based refinement (uses 28628 nodes)



(b) ray-based refinement (uses 27636 nodes)

(c) video view as reference for an urban scenario



(d) histogram of the cell resolution for the current frame

Fig. 5: Comparison of evaluated refinement algorithm. Figure 5a shows an adaptive map build using the statistical χ^2-test. The resulting map in 5a shows a finer resolution in free space area between two parked cars. The proposed adaptive algorithm in 5b choose a coarser resolution which results in a higher free space probability. A Video reference is given in 5c. A comparison of amount of refined cells and its resolution is shown in 5d.

only a single measurement, so the occupancy (red) and free states ($green$) remain with low masses.

D. Evidential Cell Update and Cell Ageing

The next step employs the measurement belief masses in order to update particular map cells. As a last step, belief masses of a map cell need to be decremented if the environment may have changed. So we introduce a cell ageing process here. The cell ageing can be applied optional if the memory footprint of the map matters. For offline map building the ageing process is unnecessary for reasonable map sizes.

1) Evidential Cell Update: For updating belief masses of the map the Dempster's rule for combination (see equation 3) is used to combine the belief mass of a leaf node m_L with a belief mass of a distance border m_{Db}.

$$m_{L\oplus Db}(A) = \frac{1}{\eta} \sum_{B\cap C=A\neq\emptyset} m_L(B)\cdot m_{Db}(C) \tag{3}$$

$$\eta = 1 - m_L(O)m_{Db}(F) - m_L(F) * m_{Db}(O)$$

$$m_L(O) = m_{L\oplus Db_n,\phi,k}(O) \tag{4}$$

$$m_L(F) = m_{L\oplus Db_n,\phi,k}(F) \tag{5}$$

$$m_L(U) = m_{L\oplus Db_n,\phi,k}(U) \tag{6}$$

The set $A \subseteq \Omega$ contains any state of interest. B and C are arbitrary subsets of Ω, creating A by their intersection. Where η can be regarded as the agreement of the belief of the masses. Equations 4 to 6 update the concrete belief masses of a cell.

To update occupied cells, we have to find leaf cells overlapping a distance border. For the update of free cells, a ray tracing or traversal algorithm has to be utilized. We use the same traversal algorithm 4 for the free space update as we already used for the cell splitting. The start point S_P and end point E_P of the traversal have to be replaced by the sensor origin and the particular distance border. If multiple measurement corresponds to the same cell the cell is updated multiple times.

$$m_{M,k}(O) = \frac{m_{M,k}(O)}{\gamma} \tag{7}$$

$$m_{M,k}(F) = \frac{m_{M,k}(F)}{\gamma} \tag{8}$$

$$\gamma = m_{M,k}(O) + m_{M,k}(F) + m_t$$

$$m_t = m_{M,k}(\Omega) + \kappa$$

Arising conflicts from these updates can be used to identify dynamic objects, which is presented in [9].

An overview of all processing steps is given in figure 5 showing the grid map processing in an urban scenario.

2) Cell Ageing: A cell ageing process ensures that the map corresponds to the actual measured environment. In adaptive grid maps the ageing process additionally conserves computational and memory resources and enable the mapping to be processed online. The ageing equations 7 and 8 are adapted from [12], where m_t is a temporary mass variable used for increasing a cell's unknown mass using κ.

V. EVALUATION

The proposed algorithms are experimentally evaluated in a challenging urban scenario (for details see table III). Highway and rural scenarios differ in the mapped area which increases the memory resources only. We compare the proposed ray-based refinement approach against our own implementation of the statistical approach proposed in [5]. In order to further analyse adaptive map techniques we measure the runtime of the map update process. The map update

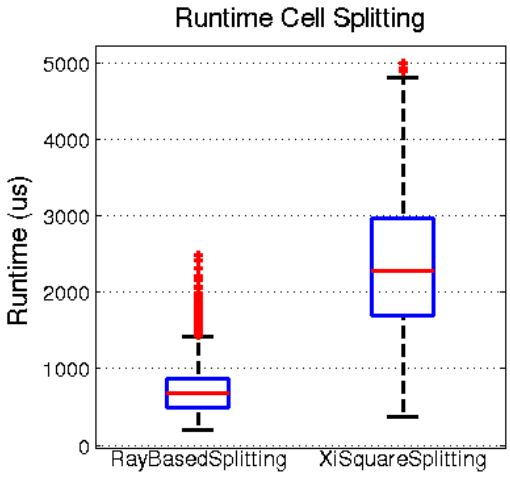


Fig. 6: box plot shows runtime comparison between χ^2-splitting and the proposed ray-based splitting

process also compares the adaptive resolution approaches against a well known static map. The static map is an own implementation using a 2D Array with fixed cell resolution. Furthermore we compare the different all three approaches in its resulting cell resolution and memory consumption. Two Ibeo LUX sensors are used for the evaluation. Scans were fused sequentially.

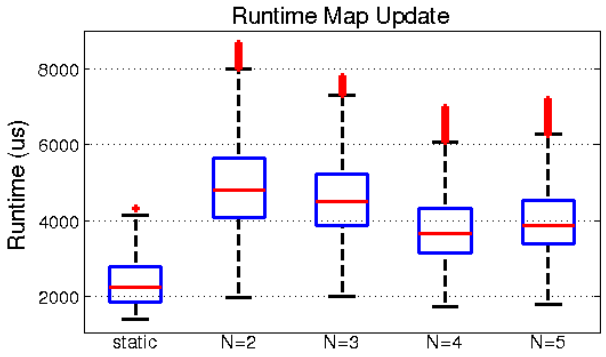


Fig. 7: box plot shows runtime comparison of map update with adaptive cell sizes and static grid resolution

scenario	travelled distance(m)	scan points per frame	scan points total	AVG cov. area(m^2)
urban	3639	1100	12020184	212.53

TABLE III: evaluation scenario

A. Adaptive Refinement

Applying the algorithms from section IV we compare two different adaptive refinement approaches using the scenario from table III. We compare our ray-based refinement against the statistical approach presented in [5]. The algorithm given in [5] introduces an unknown counter to solve the problem of partially free cells. They are incremented for each cell, which is not reached by a sensor ray because an obstacle has been hit before. From the computational point of view the algorithm proposed in [5] needs to update all cells in the possible sensor range. The algorithm we propose needs to update all cells on the way from the sensor to the measurement.

The different refinement approaches are compared in figure 5 comparing the final cell resolution. The zoomed region in figure 5a and 5b is observed in a short timespan only. For the proposed adaptive approach the cell resolution results at coarser level. This is caused through the splitting algorithm which only splits a cell which intersects the free space polygon. Using the statistical split process all cells generating a conflict were split. The cells in figure 5a were split because the whole area generates a conflict between free and unknown space. Applying statistical χ^2 algorithm a ray measuring free will split all intersecting unknown cells. Observing the same cells for a longer timespan both algorithm will produce the same cell resolution. Figure 5d shows the difference produced by both algorithms in the number of cell for a particular resolution. The χ^2 based splitting produces 992 more cells at 15 cm resolution (maximum resolution $MaxR$) than the proposed approach. The adaptive refinement process described in section IV-C has an average runtime of 0.701 ms, making it 3.4 times faster than the statistical χ^2-based splitting (figure 6). This is caused by the CPU-intensive iterative split process. The evaluation has been made on an Intel i7 running at 2.5 Ghz.

B. Map Update

In static and adaptive map representations the update process remains basically the same. Free measurements need to be integrated through a ray tracing algorithm and for each occupied measurement the corresponding cell needs to be found. Figure 7 shows the update process runtime of a $static$ and $adaptive$ implementation using a maximum cell resolution of 20 cm. Please note that adaptive map from figure 7 is updated using the proposed ray based approach. The map update performance with the χ^2 based approach differs only marginally from our approach and is not shown in figure 7 for sake of clarity. The amount of child nodes N^2 is varied for the adaptive grid map. Please note that the

amount of child nodes has only a marginal effect on the update process runtime. In our implementation of the 4^2-tree the update process is in average 0.33 slower than our static grid map. That is caused by the expensive ray tracing process on adaptive grids. The amount of touched cells using ray tracing for an adaptive update is smaller than for static map updates. Unfortunately ray tracing on non equal sized grids using a non optimized algorithm [11] is computational heavy and leads to a worse update performance.

root node area (m^2)	branch nodes	leaf nodes	sub nodes (1st level)	tree depth	memory (MB)
static mapping					
3276.8	0	268435456	268435456	1	1073.74
6553.6	0	1073741824	1073741824	1	4294.96
ray based refinement mapping					
3276.8	2048672	6146017	4	14	98.34
11809.8	1390306	11122449	9	10	150.15
3276.8	1512740	22691104	16	7	290.45
15625	1544293	37063035	25	7	463.29

TABLE IV: memory consumption static and adaptive grid with minimum cell size 20 cm (scenario from table III)

However the real strength of a tree-based grid is the memory consumption. Table IV shows the number of branch/leaf nodes, which are allocated in static and adaptive representations. Please note that the static map implementation allocates 4 Bytes per node where the tree based map implementation allocates 12 Bytes per node. The memory consumption can only be compared taking the mapped area from table IV into account. Static mapping cannot compete with adaptive mapping approaches in terms of memory. The memory consumption of the adaptive grid map is 10.9 times smaller than the static grid map observing $3276.8m$ field of view. The amount of sub nodes N^2 increase the memory consumption significantly. In relation to a quadtree (2^2) a 5^2-tree, consumes 4.7 times more memory in our test scenario. This is caused by the expensive subdivision in 5^2-trees. When a node is split 25 child nodes are allocated but probably not all of them can be used later.

C. Implementation Details

Both grid map representations (static and adaptive) are implemented using C++ templates. For the N^D-tree implementation the parameters N and D can be chosen freely at compile time. Realizing ray tracing on static grid maps a Bresenham algorithm is used for near range measurements, for far range measurements a triangle rasterization algorithm is used. For the motion compensation of the map vehicle odometry is used for the online map. For the offline approach any improved ego motion can be used (e.g. Graph SLAM).

VI. CONCLUSION AND FUTURE WORK

In this paper an adaptive grid map is proposed, which is refined using LIDAR sensor rays. The sensor rays create so-called distance borders. These distance borders are employed to build a free space polygon splitting partially free cells. Furthermore we presented an algorithm, which applies to online and offline mapping requirements using a cell ageing approach. The proposed free space refinement process is faster than the statistical split test from [5]. The resulting resolution of cells laying in regions which were observed in a short timespan only turn into coarser level faster using the proposed algorithm. That cause a slightly lower memory consumption. However, an implementation of the ray-based refinement approach requires more knowledge about the sensor.

We have shown that the proposed adaptive algorithm is memory efficient but performs worse in terms of CPU-runtime comparing it to the static grid map counterpart. Though the expensive tracing of rays could be replaced with location code-based traversal approaches presented in [13]. We will explore further ways to accelerate updates of adaptive grid maps including parallelization. Future work we will investigate in sensor modelling and mapping 3D data in an optimized adaptive grid map.

VII. ACKNOWLEDGEMENTS

This work is funded by the Federal Ministry of Education and Research of Germany (BMBF) within the RECBAR project.

REFERENCES

[1] A. Elfes, "Occupancy grids: a probabilistic framework for robot perception and navigation," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1989.

[2] E. Einhorn, C. Schroter, and H. Gross, "Finding the adequate resolution for grid mapping - cell sizes locally adapting on-the-fly," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 1843–1848.

[3] M. Langerwisch and B. Wagner, "Building variable resolution occupancy maps assuming unknown but bounded sensor errors," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 4687–4693.

[4] G. K. Kraetzschmar, G. P. Gassull, K. Uhl, G. Pages, and G. K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," in *Eds.), Proceedings of the 5th IFAC/EURON*, 2004.

[5] D. Joubert, W. Brink, and B. Herbst, "Pose uncertainty in occupancy grids through monte carlo integration," in *Advanced Robotics (ICAR), 2013 16th International Conference on*, Nov 2013, pp. 1–6.

[6] H. Carrillo, P. Dames, V. Kumar, and J. Castellanos, "Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 487–494.

[7] M. Schmid, M. Maehlisch, J. Dickmann, and H.-J. Wuensche, "Dynamic level of detail 3d occupancy grids for automotive use," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, June 2010, pp. 269–274.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9321-0

[9] J. Moras, V. Cherfaoui, and P. Bonnifait, "Moving objects detection by conflict analysis in evidential grids," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 2011, pp. 1122–1127.

[10] G. Shafer, *A Mathematical Theory of Evidence*. Princeton: Princeton University Press, 1976.

[11] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," in *In Eurographics 87*, 1987, pp. 3–10.

[12] J. Effertz, "Autonome fahrzeugführung in urbaner umgebung durch kombination objekt- und kartenbasierter umfeldmodelle," Ph.D. dissertation, Technischen Universität Carolo-Wilhelmina zu Braunschweig, Februar 2009.

[13] S. F. Frisken and R. N. Perry, "Simple and efficient traversal methods for quadtrees and octrees," *Journal of Graphics Tools*, vol. 7, p. 2002, 2002.