Prevention 0f Unwanted Free-Declaration of Static Obstacles in Probability Occupancy Grids

Stefan Krause[1,*], M Scholz[2] and R Hohmann[1]

[1]DLR Institute of Flight Systems, Department of Unmanned Aircraft, German Aerospace Center, Germany
[2]DLR Institute of Transportation Systems, Department of Data Management and Knowledge Discovery, 38108 Braunschweig, Germany

E-mail: *stefan.krause@dlr.de

Abstract. Obstacle detection and avoidance are major research fields in unmanned aviation. Map based obstacle detection approaches often use discrete world representations such as probabilistic grid maps to fuse incremental environment data from different views or sensors to build a comprehensive representation. The integration of continuous measurements into a discrete representation can result in rounding errors which, in turn, leads to differences between the artificial model and real environment. The cause of these deviations is a low spatial resolution of the world representation comparison to the used sensor data. Differences between artificial representations which are used for path planning or obstacle avoidance and the real world can lead to unexpected behavior up to collisions with unmapped obstacles. This paper presents three approaches to the treatment of errors that can occur during the integration of continuous laser measurement in the discrete probabilistic grid. Further, the quality of the error prevention and the processing performance are compared with real sensor data.

1. Introduction

Obstacle detection for unmanned vehicles often uses approximated environment representations to fuse sensor data [1–3]. Obstacle maps are in general approximation of the real environment, to declare free or obstacle occupied spaces. A classification of various objects, e.g. trees or houses, is not the aim, the statement is rather: free spaces can be safely crossed and occupied areas are to be avoided in order to avoid collisions. Map-based obstacle detection approaches [2, 3] often use global discrete representation, which is also named occupancy grid [4]. A cell m of a discrete grid M describes a space that is free or partly occupied by an obstacle. The basic idea of an occupancy grid bases on the assumption of an ideal sensor. Each grid cell is described binary as free or occupied. A sensor measurement p is a distance measurement from a sensor origin O_S to an optically "impenetrable" surface. p describes the intersection between a ray and a surface. The cell containing p is described as occupied. The cells which are cut only by a ray, are set as free. The assumption of an ideal sensor, which can be trusted in any situation, cannot be maintained. Common approaches [1, 2, 5, 6] use an occupancy probability $p(\mathbf{p}_n)$ per cell, which is based on the deficiencies of the used sensor. This implementation is referred to as the occupancy probability grid.

The conversion of continuous measurements in a discrete grid M can generate contradictory statements about the occupancy of a cell m. The cause of contradictory statements is a rough grid resolution with respect the sensor resolution. m has a predefined edge length and it cannot be guaranteed that an obstacle will fill completely m. It is more probable that an obstacle occupies only a part of m and few rays end in m and another number of rays intersect m. Opposing occupancy probabilities $p(\mathbf{p}_n)$ for m

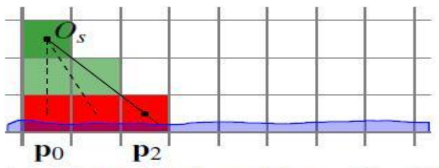
are the result. However, combinations of opposing $p(\mathbf{p}_n)$ are desirable at an occupancy grid to treat dynamic objects or sensor noise. Static obstacles are also influenced by the comparison of occupancy probability. Theconsequence is, based on the grid, angle and distance resolution as well as the trace of the measurements through the grid, areas in the grid are declared as free against the situation in the real world. Ignoring existing obstacles in M can lead to collisions between unmanned vehicles and obstacles. The hazard of ignoring static objects during the iterative conversion of continuous distance measurements in a discrete grid is followed named conditional unwanted free-declaration (CUFD).

The following section describes CUFD theoretically and by using real flight test data. Afterward, three approaches to covering CUFD are presented. Subsequently, a validation and a comparison of the methods show the quality of the prevention and their processing performance.

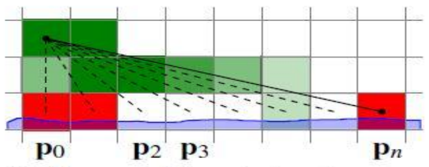
2. Problem with CUFD

A rudimentary 2D example visualizes CUFD in Figure 1. The inertial situation visualizes Figure 1a, three measurements p0 − p2 at a surface (blue) send from static sensor pose OS. The measurements are iteratively added to M. The resolution of M is dm. Figure 1b shows a subsequently incremental adding of measurements p3 up to pn and an update of M. Green cells are free spaces with an occupancy probability p(m) < 0.5. Red areas are occupied with p(m) > 0.5. White areas are unknown with p(m)= 0.5. Different intensity of colorization of freely declared cells represents different significance p(m). A p(m), which deviates from the probability of a single measurement is incrementally updated with p(pn) of subsequent measurements. Therefore p(pn) of measurements are combined with the current cell probability p(m). The example shows some darker green cells which are cut by more than one ray. These cells are probably less occupied than lighter green cells, which are only intersected by a single ray. Figure 1b shows CUFD using the example of p2 and p3. The significance for occupied and free is set identical for an incremental update to show an immediate modification. p2 is added in M in Figure 1a. The cell mp2 is dedicated as occupied. In Figure 1b is added p3, which is not within mp2 , but the ray that connects OS with p3 intersects mp2 . This situation creates competing occupancy statements for mp2 . Thus, no unique cell occupancy is possible and the resulting declaration is unknown. Several intersections enable that initially occupied cells switch into free, although they are partially occupied by obstacles. Figure 1b shows this situation for mp2 up to mpn .

The phenomenon attributes to the aliasing effect [7]. The Cause is a rough discretization of analogous measurements and representation in a grid. Figure 2a and Figure 2b show the same situation as also Figure 1.a and Figure 1b but with a grid resolution dm/2. An increased grid resolution is comparable to an increased sampling rate which suppresses the aliasing effect. Figure 2b shows that each cell is cut only by one ray and no free-declaration is made by partial occupied cells. The observation in Figure 2b is only correct taking into account the used sensor resolution and the considered distance. For dissimilar terms, a doubling of the resolution does not have to be sufficient. An increasing of the grid resolution is not unlimitedly possible because a halving of the cell size leads to a cubic increasing of cell number by a constant expansion of M. An increase can lead to computational problems on limited hardware. The example in Figure 1 is simplified. The same significance is used for occupied and free modification of a cell. Practical approaches use the bayesian-probabilistic-update in general with different significances to update the grid conservatively. Each obstacle should be added to the grid with the first measurement. Cleaning out an obstacle from the grid should require several updates. This practice requires several overlapping rays to visualize CUFD at a prior occupied cell.

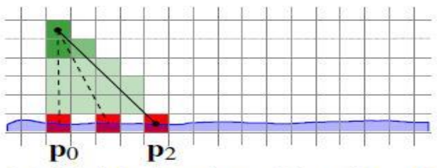


(a) Initial situation with a grid cell size d_m.

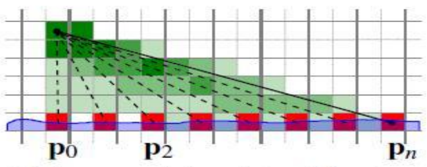


(b) Unwanted free-declaration at $\mathbf{p}_2$ up to $\mathbf{p}_n$ bases on 1a.

Figure 1: CUFD of occupied cells as Result of grid resolution d_m.



(a) Initial situation with grid cell size $d_m/2$.



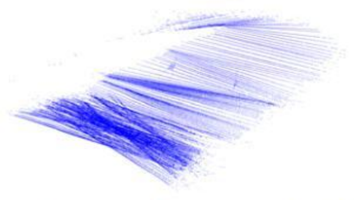
(b) Incremental update without unwanted free-declaration bases on 2a.

Figure 2: CUFD of occupied cells as Result of higher grid resolution $d_m/2$.

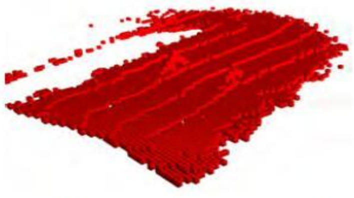
Figure 3 shows a real-environment situation. The data are generated during a flight of an unmanned helicopter through two pillars which are positioned on plane grassland, shown at Figure 3a. Figure 3b shows the used unmanned helicopter1. The data are generated with 2D Light Detection and Ranging (LiDAR) and are combined with the known pose of the helicopter to a 3D point cloud (Figure 3c). The expected result of a conversion of a recorded point cloud in Figure 3c to an occupancy grid is shown in Figure 3d. But Figure 3e shows the actual result when integrating all individual measurements sequentially into the grid. Significant parts of the grassland are not declared as occupied in the grid. For a better visualizing of the wrongly declared parts, the initial point cloud is depicted also. The incidence angles of the laser rays are almost parallel to the grassland plane (≈ 70◦). As a result, a few cells representing the grassland are intersected by several rays. A greater number of overlapping rays against ending rays lead to CUFD and without further verification with the real situation a collision of the helicopter with the grassland could be possible.

(a) Onboard camera view at the pillows and grassland.

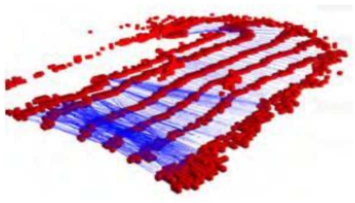
(b) Unmanned Helicopter ARTIS.



(c) Point cloud, generated with monogon LiDAR.



(d) Expected occupancy grid.



(e) Actually generated occupancy grid.

Figure 3: Example situation for CUFD.

The grid resolution should be based on the Nyquist-Shannon-sampling theorem to prevent the aliasing effect and CUFD. The grid resolution should be at least twice the sensor resolution at the coordinates at which a sensor measurement is made on an object surface. Such high grid resolution is not always predictable, especially onboard real-time approaches. The use of occupancy grids with a resolution that is less than the sensor resolution requires the development of approaches which reduce the effects of the aliasing effect. This means that the aliasing effect is not impeded, but it is less visible, as seen in Figure 3e.

3. Approaches

The current section presents three approaches to covering CUFD. The first approach is based on common literature and conducts as a comparison object. The second method works with an increase of the grid resolution based on a specific grid data structure. The last approach solves CUFD by considering grid cells not as independent spaces but rather by evaluating the cells based on their positions along the measuring ray.

3.1. Reduction of CUFD with large point clouds

Scherer [3] and Wurm [8] suggest as a solution of CUFD not to add single measurements to a grid but rather to add a pre-combined set of measurements Ci. This combination limits the mutual influence of neighboring measurements. Hornung [9] presents a FIFO buffering of several LiDAR scans St in a point cloud Ct−n:t=$\bigcup_{t-1:t}^{t} S_t$with n = const. For each measurement in Ct−n:t a ray and the respectively intersected cells in the grid are determined. The determined cells are uniquely stored in two separate sets (free, occupied). Subsequently, the elements of the sets are modified in the grid. Based on the assumption, the declaration significances of a one-time free- and occupancy-declaration are not the same. This procedure prevents a free-declaration of uniquely occupied cells for the measurements in Ci.

3.2. Covering CUFD with an octree

The use of a grid data structure that allows data to be extracted for subsequent computations in a reduced resolution from the grid as the resolution in which the data were added to the grid can also solve CUFD. The octree [10] data structure enables this function. If a measurement ray is added to an octree, leaves at the tree height h are declared with the respective occupancy probability. Updates of the inner tree nodes from the direct parent node to the root node follow the declaration of a leaf. In the current case, the update $\hat{p}(n)$ of an internal node is performed by $\hat{p}(n) = max_i P(n_i)$ on the basis of the maximum $p(n_i)$ of the child nodes. This procedure suppresses the free-declaration of nodes and the CUFD with each reduction of the tree depth d. Measurements are added to the tree at a tree depth $d_h = h - 1$. The measurements are obtained from the tree at level $d_{h-\lambda}$ with $\lambda \geq 1$. This approach works with an increase of the grid resolution with the drawback of a cubic growing cell number by halved resolution. Consequences are a higher memory consumption and higher computing requirements. In order to limit this drawback, only the additions of measurements in M at a high resolution ensure. Follow-up processes, such as the obstacle avoidance, extract the map at a lower resolution and therefore, no increasing memory and computing requirements are expected for these processes.

3.3. Conditional free-declaration of occupied cell

CUFD can also be prevented by the fact that the occupancy of a grid cell is not interpreted as an independent space description but rather is set in relation to its neighbors along a measurement ray. If the incidence angle of a ray Rp leading to a measurement p is relatively small, CUFD accrues among spatially close measurements along Rp between the first intersected occupied cell and the cell which contains the end of Rp. This condition occurs in two cases. Firstly, the gradient of the hit surface is locally almost parallel to the grid axes. The ray Rp runs from the first contact with an occupied cell in the interior of occupied cells. No free cells are cut. This situation represents the measurement p1 in Figure 4. Secondly, the gradient of the hit surface is locally non-parallel to the grid axes. This case eventuates the aliasing effect, Rp intersects recurrently occupied and free cells. Measurement p0 visualizes this case in Figure 4.

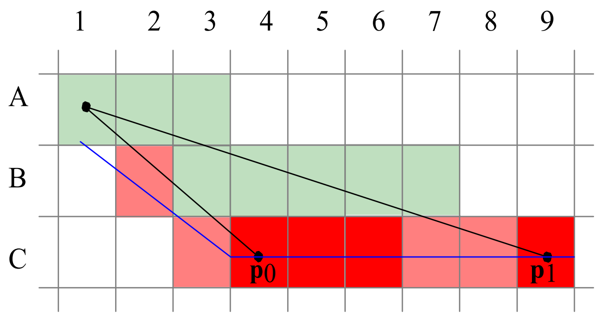


Figure 4: Ray path within the occupancy grid, which leads to CUFD.

$$S = \{q_0; q_1; q_2; q_3; q_4\}$$
$$\Sigma = \{occupied(p > p_{uk}); free(p \leq p_{uk})\}$$
$$\delta = \left\{ \begin{array}{ll} \sigma_0 & = \{p_m \leq p_{uk} \;\&\; i \leq th_{free}\}; \\ \sigma_1 & = \{p_m \leq p_{uk} \;\&\; i > th_{free}\}; \\ \sigma_2 & = \{p_m > p_{uk} \;\&\; j \leq th_{occ}\}; \\ \sigma_3 & = \{p_m > p_{uk} \;\&\; j > th_{occ}\}; \\ \sigma_4 & = \left\{p_m > p_{uk} \;\&\; m = \left\lfloor \frac{\mathbf{p}_i - \mathbf{t}_{mi}}{d_m} + 0.5 \right\rfloor \right\} \end{array} \right\}$$
$$F = \{q_3; q_4\}$$

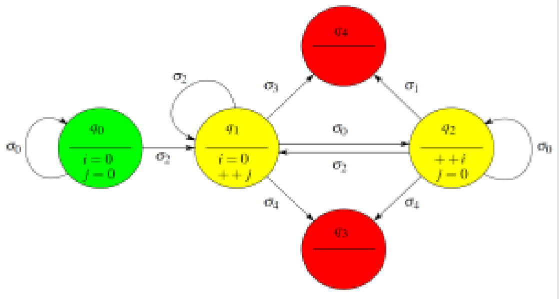


Figure 5: A finite-state machine for Conditional free-declaration along a ray Rp.

CUFD can be prevented in these cases by allowing a ray to intersect a number thocc occupied and thfree free cells in the neighborhood of p without reducing p(Z) of the intersected cells. This behavior requires that the cells in the neighborhood around p are not independent spatial descriptions but rather in relation to each other along Rp. The referencing of cells along Rp is carried out with a finite-state machine A = (S, Σ, δ , F, q0) runs in addition to the inverse sensor model for each cell. The parameters are: S set of states, δ state-transition functions, Σ input alphabet, F set of final states and q0 initial state. The behavior and the transitions of A are shown in Figure 5. The ray Rp starts at the sensor origin mOs and A is initialized at state q0. A remains at q0 to Rp intersects a cell mk with (p > puk). At q0 two counters i = 0 and j = 1 are initialized which control the thresholds thfree and thocc. thocc describes the maximum number of occupied cells in series along a ray, which could be intersected by Rp without modifying their occupancy probability. The second threshold thfree defines the maximum number of free cells between two occupied cells which are intersected by Rp without modifying their occupancy probability. If i and j are greater than the thresholds the current “the conditional free-declaration” approach is aborted for the current ray and is returned to the original approach. All measurements along Rp are updated independently of the relations between the cells with the original bayesian-probabilistic-update. A switches from q0 to q1 if Rp arrives the first occupied cell mocc,k, k = 1. The current occupancy probability of mocc,1 is not modified at q1. If further occupied cells mocc,k follow on mocc,1, A stays in q1 under the condition j ≤ thocc. If the condition j ≤ thocc is torn, A switches to the final state q4. The arrival of q4 prevents the conditional declaration for the current ray. If mocc,1 if followed by the end of the ray mp, A jumps in the final state q3. The arrival of q3 enables the conditional free-declaration. All cells intersected by Rp are not modified, except for mp which is updated by the bayesian-probabilistic-update. Follows on mocc,k a free cell mf ree,1, A switches from q1 to q2. This branch of A treats CUFD at an aliasing-situation. Follows on q2 further free cells mfree,k, A stays at q2 under the restriction i ≤ thfree. If the restriction breaks, A switches to q4 and it follows an update of all intersected cells. Is mfree,k; followed by an occupied cell with maintaining i ≤ thfree, A goes back to q1 and the branch to treat the aliasing effect is reinitialized. The reinitializing enables to deal with longer aliasing sections. If mfree,k if followed by the end of the treated ray mp, A is transmitted in the final state q3 and allows an execution of the conditional free-declaration. The ability to handle dynamic obstacles remains with this implementation.

4. Validation

The functionality of the three presented approaches is validated by a comparison against an expected nominal condition. The used data are generated with the LiDAR "Hokuyo UTM-30LX" carried by an unmanned helicopter. The expected nominal condition presents Figure 3d. The validation ensues in two steps. Firstly, several exemplary parameter sets are presented to get an idea which sets best prevent the CUFD. The used data contain three obstacles, two pillars and a ground plane. The pillars are clearly visible in Figure 3d as well as in Figure 3e. Otherwise, the ground shows a lot of holes base on CUFD. Therefore the aim is to identify sets which generate a closed ground plane. This validation is qualitative. The presented approaches are used in online mapping application for an Unmanned Aircraft (UA). Therefore the second validation focuses on run-time, complexity and memory consumption. This rating uses the best parameter set per approach of the first validation to get the processing performance for the best CUFD prevention.

4.1. Parameter set for the representation of a closed ground

The origin for parameter identification bases on an occupancy grid with a cell dimension $d_m = 0:5$m. The grid bases on an octree structure with a tree height h = 10. The result of pasting the example measurements into the grid shows Figure 6 which is a top view of the scene in Figure 3.

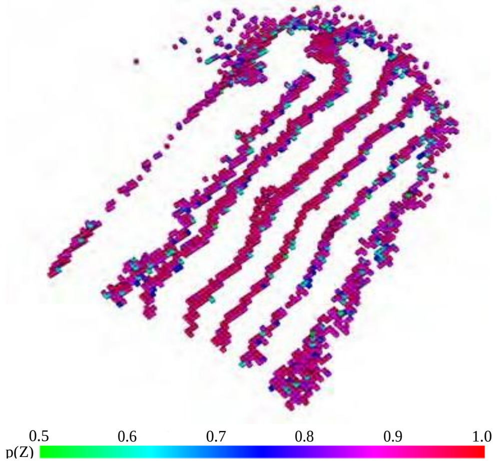


Figure 6: Original situation, without correction of CUFD. [11]

Table 1: Parameter sets to prevent CUFD.

approach	**3.1:** combine j scans S_i to a cloud $C = \bigcup_{i=0}^{j} S_i$	**3.2:** add at tree depth d_{in}, withdrawal at d_{out}, with $d_{out} < d_{in}$	**3.3:** conditional occupancy-declaration based on the thresholds th_{free} and th_{occ}
set 1	$j = 20$	$d_{in} = 12$ $d_{out} = 11$	$th_{free} = 1$ $th_{occ} = 1$
set 2	$j = 40$	$d_{in} = 12$ $d_{out} = 10$	$th_{free} = 4$ $th_{occ} = 1$
set 3	$j = 60$		$th_{free} = 1$ $th_{occ} = 4$
set 4	$j = 80$		$th_{free} = 2$ $th_{occ} = 8$

The coloring represents the occupancy probability p(Z) of each cell. The measurements of the used LiDAR are added in the grid with p(Z) = 0.8. Deviating probabilities hint to a usage of the bayesian-probabilistic-update. The figure shows only occupied cells with $0.5 \leq p(Z) \leq 1.0$. The grid shows that several parts of the ground plane are not visible and many cells have a $p(Z) \leq 0.8$. This observation indicates a contrary occupancy interpretation of some cells. A major cause is next to sensor noise the CUFD. The approaches are intended to close these holes and covers CUFD. To enables this aim, a choice of parameter sets for each approach is shown in table 1.

Results to Approach 3.1 Figure 7 presents the results of approach 3.1, for each subfigure j scans are combined to point cloud Ci and integrated into the grid. The increase from j = 20 in Figure 7a to j = 80 in Figure 7d reduces CUFD and increases the number of cells with p(Z) > 0.8. But even Figure 7d shows holes and therefore a complete prevention of CUFD cannot be presented.

Results to Approach 3.2 Figure 8 shows the use of this approach by adding measurements at tree depth d = 11 or d = 12 and extracting at d = 10. Figure 8a shows an increased input depth about one level which reduces CUFD. An increase about a further level of the input depth covers CUFD (Figure 8b) completely.

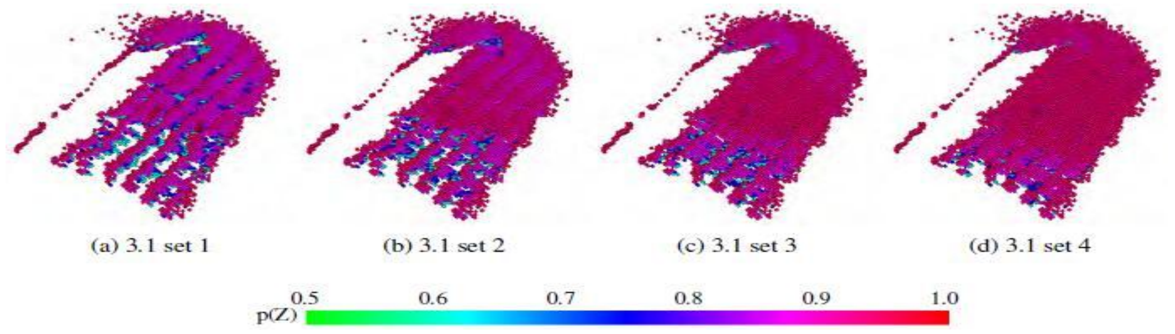


Figure 7. Parameter comparison for Approach 3.1.

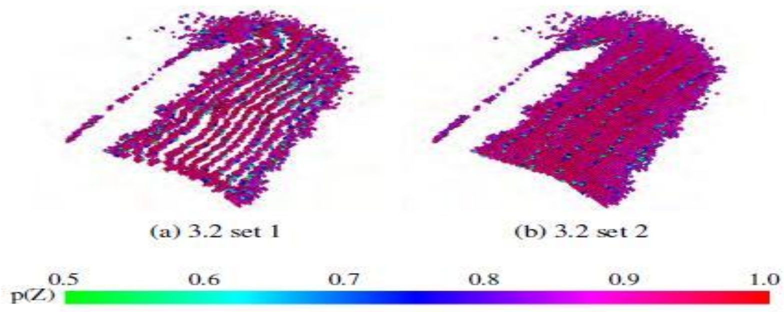


Figure 8. Parameter comparison for Approach 3.2.

Results to Approach 3.3 Figure 9 shows the results of approach 3.3. Figure 9a and Figure 9b show marginal changes against Figure 6. An increase of th_{occ} in 3.3(3) leads to a reduction of CUFD. Further adaptions of the thresholds lead to a complete prevention of CUFD and fewer cells with $p(Z) \leq 0.8$.

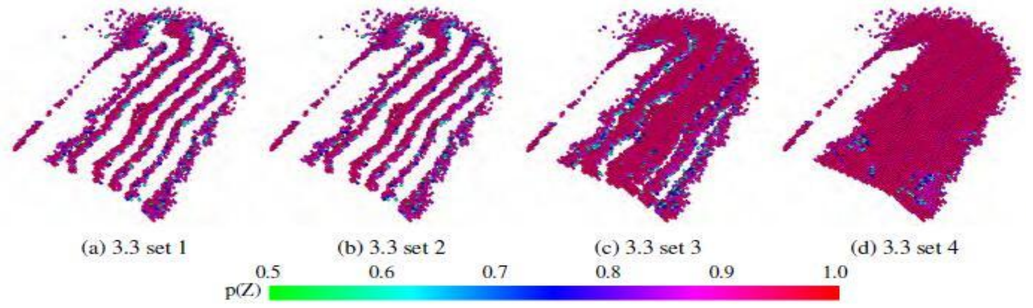
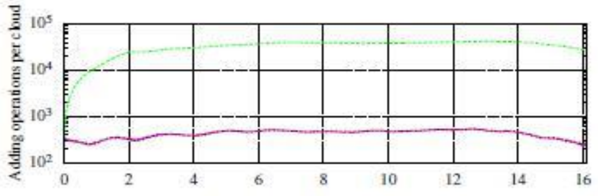


Figure 9. Parameter comparison for Approach 3.3

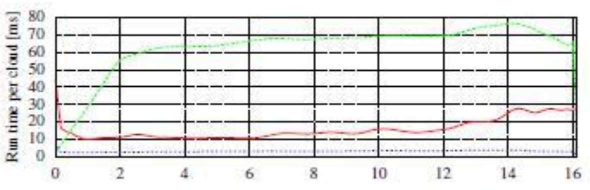
The results of the comparison show that all approaches are useable to more or less prevent CUFD at the ground plane for the used example. In a visual comparison, the parameter sets 3.1(4), 3.2(2) and 3.3(4) show the best results. These sets are used in the next section to analyze the performances between the three approaches.

4.2. Performance analysis

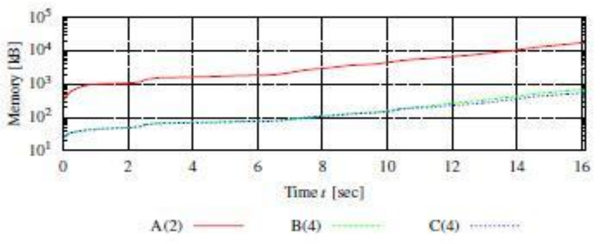
The presented approaches are used in an online application by UA and therefore have to be computable on limited hardware (Proc.: i5-4210, RAM: 4GB) onboard a vehicle. A comparison of the parameter sets 3.1(4), 3.2(2) and 3.3(4) should show which approach is best suited to run on limited hardware. Therefore the runtime per added point cloud, the number of iterations in the octree per point cloud and memory requirements of the grid are collected. Figure 10c shows the results. The regarded values are logged over recording time and bases on the shown real LiDAR data. A comparison between the approaches for the number of insertions in the octree per point cloud is presenting in Figure 10a. Approach 3.1(4) deviates from 3.2(2) or 3.3(4). The deviation shows a magnitude of about eighty times the results of 3.2(2) per step. This observation bases on the combination of j = 80 scans to a point cloud at 3.1(4) which is repeatedly added. The number of insertions at 3.1(4) also leads to an increased runtime per added point cloud into the octree, shown in Figure 10b. The differences are less because single measurements of several scans are pre-combined before they are added to the tree. The combination of measurements reduces the deviations between the approaches, but the great number of insertions at 3.1(4) cannot be balanced. The number of insertions is approximately equal between 3.2(2) and 3.3(4). This observation bases on the procedure, that both approaches add measurements directly into the octree without further pre-processing.



(a) A number of insertions in the octree per point cloud.



(b) Adding runtime per point cloud.



(c) Memory requirements of the octree.

Figure 10. Performance comparison of 3.1(4), 3.2(2) and 3.3(4). [11]

The runtime per added point cloud shows deviations between 3.2(2) and 3.3(4). These differences base on the size of the respective octree and on the number of constitutive operations like the number of single cell operations for each added ray. Some operations in an octree such as traversing ($\Theta(n)$) have an asymptotic behavior and depend on the number of nodes n at the tree. An increased tree height h leads to an increased number of nodes (cells), $n_{h=12} = N^2 * n_{h=10}$, whereby N = 8 is the number of children per node. This property leads to increased processing effort and runtime for trees with a taller height. An example is traversing trough inner nodes after updating a cell declaration. A further cause of increased runtime is a higher cell resolution at 3.2(2). This leads to a situation that fewer measurements are spatially located in the same cell. The modification of a cell often bases on a single measurement per cell. An insertion of a point cloud into a higher resolved tree results in a major number of occupied cells. Additionally, based on the reduced cell size, a ray intersects more cells in a higher resolved tree and therefore more free cells. Both points lead to increased number of touched cells and more subsequent operations which lead to an increased runtime at 3.2(2).

The memory requirements of a cell are constant and independent of its depth in the tree. An increased tree height leads to an increased number of inner nodes. Further, each added measurement touches more cells in a higher resolved tree than in a lower resoled tree. Both points are connected in 3.2(3) and lead to higher memory requirements against the other presented approaches, as seen in Fig 10c. Approach 3.1(4) has higher memory requirements compared to 3.3(4), which cannot be explained by a different tree height. At the end (16.15 sec) of the scenario, the number of the occupied cells is n_{occ} = 4880 for 3.1(4) and n_{occ} = 4895 for 3.3(4). This deviation cannot be the cause of different memory requirements. The cause is the number of free cells n_{free} = 20957 for 3.1(4) and n_{free} = 14590 for 3.3(4), at the end of the logged test. The difference between the numbers of the free cells is attributable to properties of the added point clouds. Each cloud has a single sensor position per cloud for all included measurements. For the transformation of single measurements from sensor coordinates into global world coordinates and combination into a point cloud, a compensation of the sensor motions is necessary. This procedure requires that for each measurement the correct global sensor position is used at the moment of recording, especially if a system like a rotating LiDAR is used. After this procedure, each point cloud often gets only one single sensor position for a roughly global assignment. Commonly the sensor position of the first or the last added measurement is used. This characteristic is also used in the implementation described above. An insertion of a point cloud into a grid requires a conversion of contained measurements into rays, which based on measurements and the overall sensor position of the cloud. Therefore, the generated rays are not identical with the actually measured rays. This adaption of a ray leads to the situation that measurements theirself stay at the same position and the number of occupied cells does not change. But trace and length of a ray change and lead to an adjusted number of free cells and changed cell positions compared to the original rays. A rudimentary example of this description is depicted in Figure 11. The continuous black rays connect real sensor positions with the corresponding measurements.

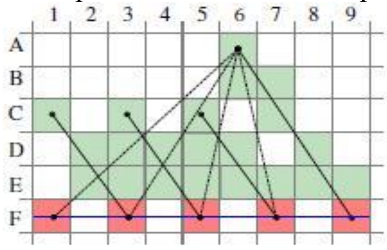


Figure 11. Adding single measurement rays against adding a cloud with one origin.

Cells which contain a measurement are declared occupied and colored red. All intersected cells with no measurements are colored green and declared as free. The overall point cloud contains the same example measurements but uses only one sensor position, in this example the position at cell (A6). Based on this position will be defined occupied and free cells with dashed rays. The declaration of occupied cells shows no difference between adding measurements continuously and adding the overall point cloud at once. But the declaration of free cells show some differences when looking at B5, C4, C6 and D7 which are intersected only by dashed rays. These intersections declare cells as free without real measurement support for this classification. This behavior is critical and can be the cause of collisions. The example describes a combination of single measurements to a point cloud. Approach 3.1(4) combines several point clouds to a major point cloud. This procedure leads to a much heavier influence of this phenomenon. Therefore the described behavior is assumed to be a cause for the difference between the memory requirement of 3.1(4) and 3.3(4). The qualitative and quantitative comparison between the approaches 3.1, 3.2 and 3.3 and especially the parameter sets 3.1(4), 3.2(2) and 3.3(4) show that 3.3 generates the best results to prevent CUFD during mapping LiDAR data for the presented flight scenario. This statement bases on less runtime, memory requirements and correct visualization of the mapped environment.

5. Conclusion

This paper deals with the phenomena of the conditional unwanted free-declaration (CUFD) which can occur during insertion of continuous distance measurements into a discrete probability occupancy grid. The cause of CUFD is a low spatial resolution of an occupancy grid compared to continuous distance measurements which are added into the grid and the resulting possibility that several measurement rays intersect a single grid cell. If this cell is only partly occupied by an obstacle, opposing occupancy statements (free/occupied) are generated. Contrary occupancy statements are generally desirable at probability occupancy grids to handle sensor noise or dynamic obstacles. But CUFD can also accrue at static objects and leads to a free-declaration of partly occupied cells. Free-declaration of occupied spaces in an environment representation which is used for unmanned obstacle avoidance is a danger because it can result in a collision with a non-mapped obstacle.

The aim is to reduce the differences between a mapping representation and the real world, to prevent possible collisions. Therefore, three approaches are presented to suppress CUFD. These approaches are validated qualitatively and quantitatively to compare the prevention of CUFD and the ability to process on limit hardware.

As result an approach is preferred which interprets cells along a measurement ray not as independent spaces rather the occupancy probability of a cell is set in relation to its neighbors. This procedure allows handling CUFD with less processing performance compared to other approaches.

References

[1] F. Andert and L. Goormann. "Combined grid and feature-based occupancy map building in large outdoor environments". In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. 2007, pp. 2065–2070. ISBN: 978-1-4244-0912-9.

[2] S. Hrabar and G. Sukhatme. "Vision-based navigation through urban canyons". In: Journal of *Field Robotics* 26.5 (2009), pp. 431–452.

[3] S. Scherer. "Low-Altitude Operation of Unmanned Rotorcraft". PhD thesis. Pittsburgh and PA: The Robotics Institute, Carnegie Mellon University, 2011.

[4] H. P. Moravec and A. Elfes. "High resolution maps from wide angle sonar". In: *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. Vol. 2. 1985, pp. 116–121.

[5] A. Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989), pp. 46–57.

[6] U. Raschke and J. Borenstein. "A comparison of grid-type map-building techniques by index of performance". In:*Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. 1990, pp. 1828–1832. ISBN: 0-8186-9061-5.

[7] E. Angel. *Interactive computer graphics: A top-down approach using OpenGL@*. 4th ed., internat. ed., [Nachdr.] Boston and Mass: Pearson/ Addison-Wesley, 2006. ISBN: 032131252X.

[8] K. M. Wurm et al. "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems". In: *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation* (2010).

[9] A. Hornung et al. "OctoMap: an efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[10] D. Meagher. *Geometric modeling using octree encoding*. Vol. IPL-TR-81-005. Reports / Rensselaer Polytechnic Inst Troy NY Image Processing LAB.

[11] S. Krause. "Sichtfelderweiterung eines Laserscanners mittels einer Kamera fuer unbemannte Luftfahrzeuge". PhD thesis. Braunschweig: Technischen Universitt Carolo-Wilhelmina zu Braunschweig, 2016.