2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Robust and Efficient Volumetric Occupancy Mapping with an Application to Stereo Vision

Konstantin Schauwecker[1] and Andreas Zell[1]

***Abstract*— A map of occupied and free space in a robot's environment is a common prerequisite for navigational tasks. Although the first methods for occupancy mapping relied on a 2D grid representation, 3D volumetric approaches are becoming increasingly popular. In this paper we present a new volumetric mapping approach that is based on the OctoMap method. We designed this method to be more robust against measurement errors, in particular against high temporally or spatially correlated errors usually received from a stereo vision system. For this purpose, we define a probability measure that a voxel is currently visible. An update of a voxel's occupancy probability then happens with respect to this visibility probability, allowing us to neglect measurements for voxels that are actually unobservable. Finally, we model the depth error of a stereo vision sensor, and take care of this error when performing a map update. By evaluation we show that our method produces maps with far less erroneous artifacts compared to OctoMap. Our maps also require less memory, and due to an optimized update reduction, our method is also faster than OctoMap when processing dense range measurement data.**

I. INTRODUCTION

Autonomous robots that operate in unknown or dynamic environments require a map of occupied and free space in order to facilitate navigational tasks. Traditionally, such maps were based on 2D grids, which integrate data from 2D range sensors such as laser scanners or sonars. However, 2D maps are insufficient if the robot is able to move in three dimensions, which is the case for unmanned aerial or underwater vehicles. But also for ordinary wheeled robots, a 2D map might not be sufficient. A tall robot must be aware of overhanging structures. Robots that are built on a movable base wider than the robot itself might have to drive partially below an obstacle, such as a table, in order to approach an object. Further, robots performing 3D manipulation require some understanding of the 3D environment.

When extending the 2D grid representation to 3D, we arrive at a volumetric occupancy map. If implemented naïvely, such a map can consume excessive amounts of memory. However, an efficient method has been proposed in [1], which uses compressed octrees. The authors have made their implementation, which they named *OctoMap*, publicly available and it has since been used in numerous research projects. In the initial publication of OctoMap, data from an accurate 3D laser scanner was used for evaluating the mapping performance. Unfortunately, laser scanners are not suitable for all robots, such as Micro Aerial Vehicles (MAVs), which have firm payload and power consumption constraints. Furthermore, 3D laser scanners are generally very costly, which makes them unavailable for low-cost robots. This is why we focus our research on stereo vision, which provides a low-priced alternative. Stereo vision systems can also be constructed light-weight and power-efficient, which even allows their use on MAVs [2], [3].

Data received from stereo vision is, however, much noisier than measurements from laser scanners. The stereo vision noise increases quadratically with the measured distance, and it is often spatially and temporally correlated. Stereo matching algorithms usually employ an explicit or implicit smoothness constraint that penalizes solutions with abruptly varying depth. This means that, if a stereo matching result is wrong for one image location, all neighboring locations are likely to exhibit a similar error. Furthermore, because stereo matches are determined by image similarity, similarly textured regions are likely to be repeatedly mismatched in subsequent frames. One can thus not assume that on average, measurement errors will cancel out each other.

Consequently, methods designed for processing laser range data do not necessarily perform well when applied to stereo vision. In terms of OctoMap, the correlated noise inherent in stereo measurements leads to many falsely mapped artifacts; see Sec. IV. In this paper we propose a more robust method, which is a modification of the original OctoMap approach. We show that our method is more accurate when used in conjunction with stereo vision, while also exhibiting a smaller memory footprint. At the same time, our method achieves faster processing times than OctoMap when processing dense range data. Even though we designed this method specifically for use with stereo vision, it can also be used with other range sensors that exhibit significant sensor noise. The source code of our implementation is available online[1].

II. RELATED WORK

The idea of using a regular 2D grid for mapping the occupancy of a robot environment dates back to [4]. The authors used a robot equipped with a wide-angle sonar sensor to measure the distance to close-by obstacles. Different methods have since been proposed for integrating individual range measurements into an occupancy grid. For an overview of existing methods with an in-depth evaluation of their performance, see [5].

Stereo cameras are 3D sensors. For creating 2D occupancy maps, the obtained results need to be reduced into a 2D representation, usually by a column-by-column projection of

[1]K. Schauwecker and A. Zell are with the Chair of Cognitive Systems, Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Sand 1, 72076 Tübingen, Germany

[1]See http://www.cogsys.cs.uni-tuebingen.de/software/occmapping/.

the disparity map. For a robot using 2D maps created with such a method, see [6]. A similar method has been published in [7] that creates the occupancy map in the disparity space. To improve accuracy, the authors derive a measure for the probability that an image column is actually visible. The visibility is then respected in the probability integration.

Planar grid-based maps only provide a reduced view of the 3D world in which the robot operates. A more informative way for environment representation are 2.5D elevation maps, where the surface elevation is stored for each grid cell. Such a method has, e.g., been used in [8]. While elevation maps are often sufficient for outdoor navigation, they are incapable of mapping overhanging structures, which makes them particularly unsuitable for mapping indoor environments. This constraint can be weakened by allowing multiple surface levels as in [9]. However, the complexity of environments that can be represented with this method still remains to be limited.

A more general approach for 3D world representation are volumetric occupancy maps. An example for such a method can be found in [10], [11], where the authors create lists of occupied voxels that are stored 'on top' of each cell in a 2D grid. The methods are evaluated using data recorded with a laser scanner. In this work, we focus on volumetric obstacle maps stored in octrees, which is an efficient tree-based data structure. For a stereo vision based method that uses a data structure similar to octrees, see [12]. The authors use a complex approach that includes temporal and spatial filtering as well as 'aging' of no longer observed voxels.

The implementation of OctoMap [1], which is an octree based occupancy mapping method, has been released as open source. This has lead to its widespread use in many robotics projects, including projects relying on stereo-vision. Examples are the MAV presented in [2] or the autonomous exploration system in [13]. This mapping method casts a ray from the sensor origin to each end point of a range measurement. Observations are then generated for all voxels on these rays, and integrated using a recursive Bayes filter. A maximum and minimum probability threshold are applied, which allows for a pruning of those nodes whose children have reached saturation, resulting in a compression of the created octree. The work presented in this paper is based on the original OctoMap implementation.

III. METHOD

As mentioned previously, the correlated noise inherent in range measurements from stereo vision can promote the mapping of many erroneous artifacts. For regions that are not visible due to occlusions, all received measurements are in fact erroneous, which can lead to a random map in those areas. In [12] this problem is resolved using the mentioned spatial filtering. We however, aim at a method that solves this problem inherently. For this purpose, we estimate the probability that a given voxel is actually visible, before it is updated. In this sense, our method is similar to the one proposed in [7]. However, since [7] is a 2D method, the visibility is measured for a column of the disparity map only.

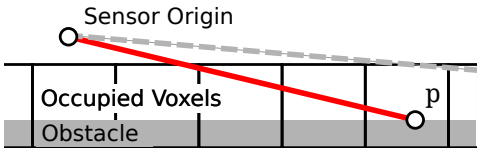


Fig. 1: Observing an obstacle at a flat angle.

The used measure is the fraction of pixels with a disparity less than a voxel's reference disparity. Hence, the visibility measure depends only on the current measurement, and does not incorporate knowledge from the created map.

Our visibility measure differs significantly from the aforementioned. First, our measure is not limited to 2D space, but can instead estimate the visibility of voxels in a 3D map. Second, our measure is derived from the created occupancy map, and not from the current range measurement. A map created from integrating multiple measurements is more accurate than only an individual measurement. In the following subsections, we describe this visibility measure, the probability integration scheme and further enhancements that we did to the original OctoMap implementation.

A. Visibility Estimation

Given a ray from the sensor origin that passes through a voxel v, the naïve model for the visibility of v would be the probability that all voxels that are on the ray and closer to the sensor origin are not occupied. This simple model, however, performs poorly when the sensor observes an obstacle at a flat angle, as shown for the solid ray in Fig. 1. Here, the ray passes through several occupied voxels before reaching the measurement end point p. The voxel containing p would hence be falsely determined as not visible.

We thus use a different method for modeling the visibility. We consider a voxel v as locally occluded if it is occluded by its direct neighbors for all possible rays that pass through the sensor origin and v. For the example given in Fig. 1, the voxel containing p is not occluded by its neighbors for the dashed ray. Let's consider a set N_v of three voxels, which border the usually three faces of v that are visible to the sensor (analogous to Fig. 2a). If all voxels in N_v are occupied, v is locally occluded, which we indicate with the event C_v. As estimate for the probability of a local occlusion $\mathrm{P}(C_v)$, we choose the smallest occupancy probability for the voxels in N_v. If $\mathrm{P}(O_v)$ denotes the probability that voxel v is occupied, we can express $\mathrm{P}(C_v)$ as follows:

$$\mathrm{P}(C_v) = \min\left\{\mathrm{P}(O_a), \mathrm{P}(O_b), \mathrm{P}(O_c)\right\},\ \{a,b,c\} \in N_v \quad (1)$$

Now, let's consider a ray R with a set of voxels $v_i \in R$. For a voxel v_i the visibility depends on the events C_{v_i} that this voxel is locally occluded and $V_{v_{i-1}}$ that the previous voxel on the ray, which is closer to the sensor origin, is visible. Given the law of total probability, we can compute the probability $\mathrm{P}(V_{v_i})$ that voxel v_i is visible as follows:

$$\begin{aligned}\mathrm{P}(V_{v_i}) = {} & \mathrm{P}(V_{v_i}|C_{v_i}, V_{v_{i-1}})\,\mathrm{P}(C_{v_i})\,\mathrm{P}(V_{v_{i-1}}) \\ & + \mathrm{P}(V_{v_i}|\neg C_{v_i}, V_{v_{i-1}})\,\mathrm{P}(\neg C_{v_i})\,\mathrm{P}(V_{v_{i-1}}) \\ & + \mathrm{P}(V_{v_i}|C_{v_i}, \neg V_{v_{i-1}})\,\mathrm{P}(C_{v_i})\,\mathrm{P}(\neg V_{v_{i-1}}) \\ & + \mathrm{P}(V_{v_i}|\neg C_{v_i}, \neg V_{v_{i-1}})\,\mathrm{P}(\neg C_{v_i})\,\mathrm{P}(\neg V_{v_{i-1}})\end{aligned} \quad (2)$$

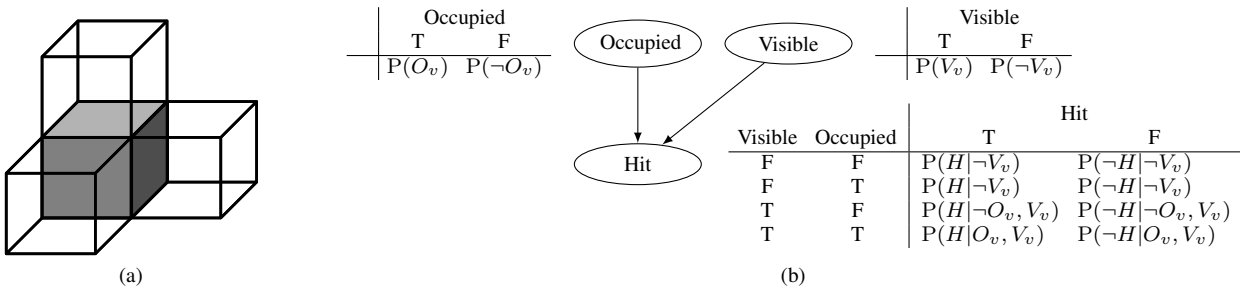


Fig. 2: (a) Visible faces of a voxel with neighboring voxels and (b) Bayesian network of new probability integration scheme.

We define the probability of voxel v_i being visible if voxel v_{i-1} is not visible to be 0. This means that the probability $\mathrm{P}(V_{v_i})$ will never be greater than $\mathrm{P}(V_{v_{i-1}})$. In this case, Eq. 2 can be simplified as follows:

$$\begin{aligned}\mathrm{P}(V_{v_i}) = &\mathrm{P}(V_{v_{i-1}})[\mathrm{P}(V_{v_i}|C_{v_i}, V_{v_{i-1}})\,\mathrm{P}(C_{v_i}) \\ &+ \mathrm{P}(V_{v_i}|\neg C_{v_i}, V_{v_{i-1}})\,\mathrm{P}(\neg C_{v_i})]\end{aligned} \tag{3}$$

OctoMap performs a clamping of the occupancy probabilities to $[p_{min}, p_{max}]$, which means that we need to consider clamped probabilities when applying Eq. 3. We hence introduce a new threshold $q_{max} \leq p_{max}$. Probabilities $\mathrm{P}(V_v) \geq q_{max}$ are clamped to 1, which also prevents the visibility from dropping too quickly. At the same time, we introduce a lower threshold $q_{min} > 0$, and stop the processing of a ray once $\mathrm{P}(V_v) < q_{min}$.

B. Occupancy Probability Integration

To respect the visibility, we need a new probability integration scheme that incorporates the probability $\mathrm{P}(V_v)$ of voxel v being visible. The method we chose can be expressed as a Bayesian network, as shown in Fig. 2b. This network contains three random variables for the events that a voxel is *visible*, that the received measurement was a *hit* (an occupied measurement), and that the voxel is actually *occupied*.

For the case of voxel v not being visible, the probability for receiving a hit $\mathrm{P}(H|\neg V_v)$ or a miss $\mathrm{P}(\neg H|\neg V_v)$ does not depend on the voxel's occupancy, but only on the a priori probability of measuring a hit or a miss for v. Hence, no matter what measurement we receive, we will gain no information about the occupancy of v. For the cases where the voxel is visible, the hit probabilities depend on a predefined sensor model. If a voxel is occupied and visible, the probability of a hit is $\mathrm{P}(H|O_v, V_v)$. Similarly, the probability of a hit is set to $\mathrm{P}(H|\neg O_v, V_v)$ if the voxel is visible but not occupied. Both, $\mathrm{P}(H|O_v, V_v)$ and $\mathrm{P}(H|\neg O_v, V_v)$, are assumed to be constant and configured according to the sensor properties.

We can solve the Bayesian network from Fig. 2b for the occupancy probability. Given the previous occupancy probability $\mathrm{P}(O_v)$, we can calculate the new probability $\mathrm{P}(O_v|M)$ after a measurement M, which is either a hit H or a miss $\neg H$. This can be done using Bayes theorem:

$$\mathrm{P}(O_v|M) = \frac{\mathrm{P}(M|O_v)\,\mathrm{P}(O_v)}{\mathrm{P}(M)} \tag{4}$$

The probabilities $\mathrm{P}(M|O_v)$ and $\mathrm{P}(M)$ can be calculated using the law of total probability:

$$\mathrm{P}(M|O_v) = \mathrm{P}(M|\neg V_v)\,\mathrm{P}(\neg V_v) + \mathrm{P}(M|O_v, V_v)\,\mathrm{P}(V_v) \tag{5}$$

$$\begin{aligned}\mathrm{P}(M) = &\mathrm{P}(M|\neg V_v)\,\mathrm{P}(\neg V_v) + \mathrm{P}(M|O_v, V_v)\,\mathrm{P}(O_v)\,\mathrm{P}(V_v) \\ &+ \mathrm{P}(M|\neg O_v, V_v)\,\mathrm{P}(\neg O_v)\,\mathrm{P}(V_v)\end{aligned} \tag{6}$$

This new update equation is considerably more complex than the one used in OctoMap. Further, while OctoMap's update equation can be expressed using log-odds, which allows for a simplification to a simple sum, this is not possible for Eq. 4. This makes our approach more computationally expensive. However, as we will see later, the probability integration is not particularly performance critical.

C. Sensor Depth Error Modeling

For a stereo camera, the range measurement error increases quadratically with the measured depth, which is also the case for other camera based depth sensors such as the Microsoft Kinect. Hence, only integrating a hit for the voxel containing the measurement end point is particularly incorrect for distant points. Therefore, we estimate the average depth error, which we assume to be normally distributed. Given the depth measurement z, the standard deviation of the depth measurement σ_z can be computed as follows:

$$\sigma_z = \lambda \cdot z^2, \quad \lambda = \frac{\sigma_d}{b \cdot f} \tag{7}$$

Here, λ is a constant factor that depends on the sensor properties. For a stereo camera, it can be computed from the disparity standard deviation σ_d, baseline b and focal length f. When using sensors with non-quadratically increasing depth error, Eq. 7 should be replaced.

With σ_z we know the accuracy for the current range measurement at depth z. Given a point $q = (x_q, y_q, z_q)$ on the current ray R, the cumulative normal distribution function $F_z(z_q)$ with standard deviation σ_z provides us with the probability that q is already inside the obstacle at depth z. We hence reduce each voxel $v \in R$ to a point and compute the probability $P(I_v)$ that this point is inside the obstacle.

The original ray casting scheme is then modified such that we continue to traverse voxels in ray direction after we reached p. In case where a voxel is far from p, we can keep the update equation from Eq. 4 with $M = \neg H$. For voxels within the proximity of p, however, the correct occupancy probability should be between $\mathrm{P}(O_v|\neg H)$ and $\mathrm{P}(O_v|H)$. We

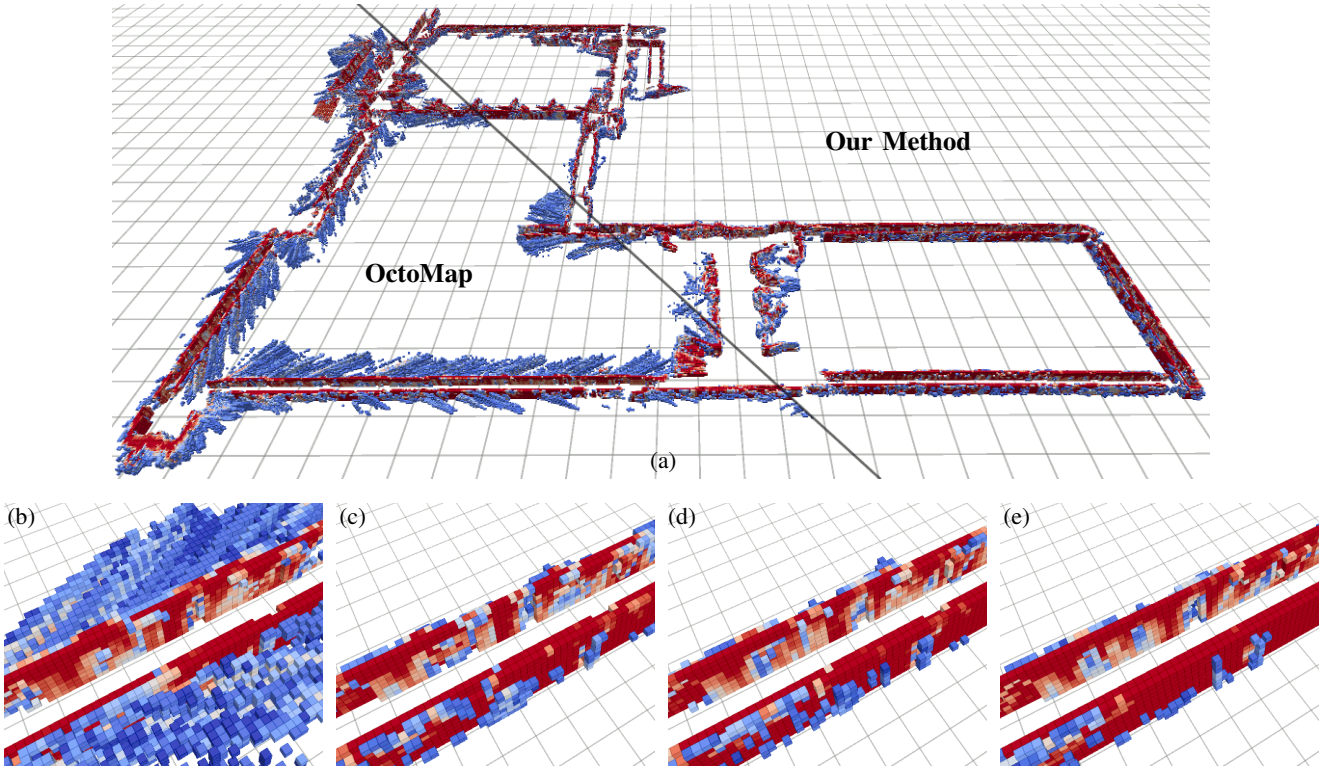


Fig. 3: (a) Full map created with our method and OctoMap for half resolution ELAS, and corridor mapped with (b-c) full, (d) half and (e) quarter resolution ELAS with (b) OctoMap and (c-e) our method.

hence perform a linear interpolation between both, using the probability $P(I_v)$ as interpolation factor:

$$\mathrm{P}(O_v|M) = \mathrm{P}(I_v)\,\mathrm{P}(O_v|H) + \mathrm{P}(\neg I_v)\,\mathrm{P}(O_v|\neg H) \quad (8)$$

We stop the ray casting once $\mathrm{P}(I_v)$ is close to 1. To make sure that at least one full hit is integrated, we continue the ray casting for one further voxel with $\mathrm{P}(O_v|M) = \mathrm{P}(O_v|H)$

D. Performance Considerations and Optimizations

Although our method is more complex than OctoMap, the performance impact remains limited. This is due to the fact that OctoMap spends most of its time on ray casting and on reducing the resulting updates to only one update per voxel. The actual probability integration, for which we now use the more complex formula from Eq. 4, only has a relatively small performance impact. The visibility calculation from Sec. III-A can be performed after the updates have been reduced, which saves much computation time.

With the depth error modeling from Sec. III-C we also extended the ray casting. However, most voxels are still processed as before. Only for voxels close to the measurement end point p, we determine the probability $\mathrm{P}(I_v)$ that they are inside the detected obstacle. We pre-compute $\mathrm{P}(I_v)$ for a discretized set of depth values and distances to p.

Finally, we use an optimized method for the update reduction. OctoMap uses a hash table in which the updates generated by ray casting are inserted. Even though a hash table provides a constant lookup time, the time required for a single lookup still has a high performance impact. We hence first perform a lookup against the previously processed ray, for which we store the update of each voxel. We then compare the voxels with the same index in the previous and current ray. If both voxels are equal, we can use the stored voxel update rather than performing a slower hash table lookup. In our experiments, the great majority of ray voxels could be processed without relying on the hash table.

IV. EVALUATION

For evaluating the proposed method, we chose the dataset *Bicocca_2009-02-25b* from the rawseeds project [14]. This dataset provides a 29 min indoor recording from a mobile robot, driving for about 774 m. The robot is equipped with various sensors, including laser scanners and a trinocular stereo system with VGA resolution. Several solutions for the robot's trajectory in this dataset have been published. We use the solution provided for the graph-based laser SLAM method in [15]. We performed a cubic spline interpolation of the pose estimates in this solution in order to overcome the low update rate of this method.

The chosen stereo matching method for this evaluation is ELAS [16], for which an efficient open source implementation is available. We selected this method for its fast processing rate and accurate matching results. An example disparity map created for a scene from our evaluation dataset is shown in Fig. 4. Here, red and blue hues represent large and small disparities, and black represents regions with no disparity estimate. For time critical applications, ELAS can produce half-resolution disparity maps, while preserving the full disparity resolution. We extended ELAS such that it can also produce a quarter resolution disparity map at an even

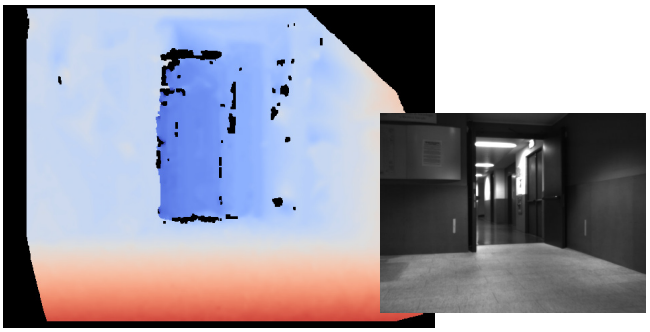


Fig. 4: Disparity map created by ELAS.

Our Method		OctoMap	
$\mathrm{P}(H\|V_v, \neg O_v)$	$= 0.43$	$\mathrm{P}(O_v\|\neg H)$	$= 0.45$
$\mathrm{P}(H\|V_v, O_v)$	$= 0.55$	$\mathrm{P}(O_v\|H)$	$= 0.55$
$\mathrm{P}(H\|\neg V_v)$	$= 0.05$		
$\mathrm{P}(Vv_i\|Cv_i, Vv_{i-1})$	$= 0.20$		
$\mathrm{P}(Vv_i\|\neg Cv_i, Vv_{i-1})$	$= 1.00$		
$\{q_{min}, q_{max}\}$	$= \{0.1, 0.7\}$		

TABLE I: Parameters selected for our method and OctoMap.

faster processing rate. We have included all three resolution options in our evaluation.

A. Map Quality Analysis

We processed the left and right camera images of the approx. 26.000 trinocular stereo frames in the evaluation dataset. With the stereo matching results, we created a volumetric map with OctoMap 1.6.0 and our method. Since both methods use a different probability integration scheme, each method requires its own set of parameters, which are shown in Tab. I. This unfortunately means that the presented results are only valid for the specific parameterizations. However, we attempted to find a good set of parameters for each method, which should provide representative results.

Fig. 3a contains the resulting maps when using the half-resolution version of ELAS, with red hues indicating high occupancy probabilities and blue indicating probabilities close to 0.5. We used a voxel size of 0.2 m and cut-off all voxels below and above a minimum and maximum height, effectively removing the floor and ceiling. One can clearly see that OctoMap produces many erroneous artifacts, which are not visible in our approach. A close-up view on a corridor in this dataset is shown in Fig. 3b and 3c for OctoMap and our method with full resolution ELAS. Even though the erroneous artifacts are mostly removed in our results, the wall is still densely mapped. Particularly when mapping neighboring rooms or intersections, the produced artifacts can lead to a disruption of previously correct map areas. For comparison, Fig. 3d and 3e contain the maps for the same corridor with half and quarter resolution ELAS and our mapping method. In both cases, the corridor is still densely mapped despite the smaller image resolutions.

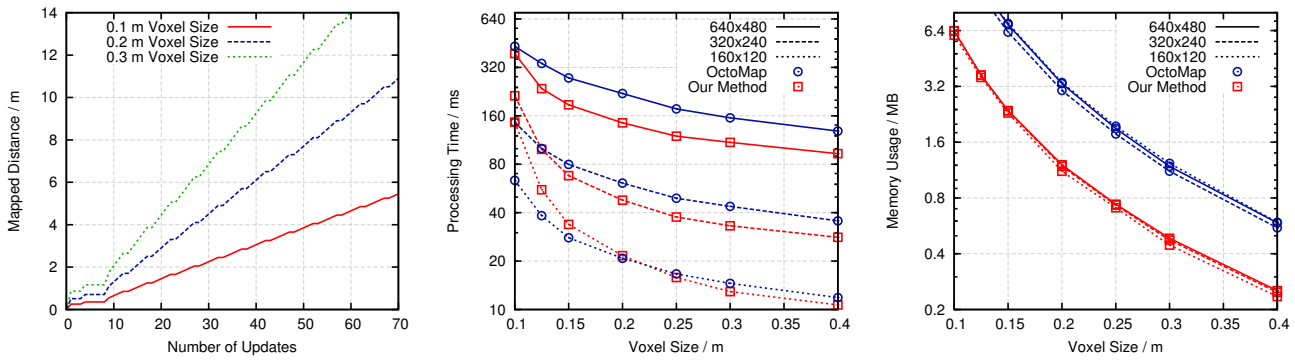
The mapping behavior of our method differs significantly from OctoMap. When facing in a direction that has previously not been observed, OctoMap immediately expands its map to the maximum visible distance. Our method, however, gradually increases the mapped distance after each sensor update. We have analyzed this behavior in Fig. 5a for the parameters from Tab. I and different voxel sizes. We repeatedly processed one image from our evaluation dataset that shows a long corridor, and measured the distance to the farthest voxel with an occupancy probability $\mathrm{P}(O_v) > 0.5$. With a voxel size of 0.2 m, 65 updates are required to reach a distance of approx. 10 m, which represents a time span of 4.3 s in our test dataset. This time span strongly depends on the voxel size, q_{max} and $P(V_{v_i}|V_{v_{i-1}}, C)$. Hence, by adjusting these parameters, one can chose a compromise between the speed of the map expansion and map quality.

B. Performance Evaluation

In addition to the map quality assessment, we analyzed the runtime performance on a PC with an Intel i5 dual core CPU with 3.3 GHz. We processed a one-minute section of the evaluation dataset with various voxel sizes and the three different resolutions of ELAS. Figure 5b shows the average processing times excluding stereo matching in logarithmic scale. Except for small voxel sizes and the quarter resolution version of ELAS, our method provides significantly lower processing times than OctoMap. The largest performance difference was observed for full-resolution ELAS with 0.2 m voxel size, where our method required only 66% of the computation time used by OctoMap. This result might seem surprising, given that our method is more complex, but it can be explained with the optimized update reduction from Sec. III-D. This method works well if neighboring rays traverse mostly the same voxels, which is not the case for small sensor resolutions or voxel sizes.

Our method with half-resolution ELAS and 0.2 m voxel size, for which we received an average processing time of 48 ms, should be fast enough to process our dataset in real time, i.e. 15 frames per second. However, we also need to account for the processing time required for stereo matching. For the full, half and quarter resolution versions of ELAS, we require an average of 122 ms, 48 ms and 24 ms. Hence, for half-resolution ELAS, real-time processing of our dataset is possible if stereo matching and occupancy mapping are run in parallel on both CPU cores.

Since OctoMap tends to map many erroneous artifacts with our stereo matching results, it also requires more memory. The memory consumptions, which we measured for the previously examined one minute test run, are shown in a logarithmic scale in Fig. 5c. On average over all test runs, our method required only 37% of the memory used by OctoMap. We observed the largest difference for quarter-resolution ELAS with 0.1 m voxel size, where our method required only 32% of the memory used by OctoMap. For the map of the full data set created with half-resolution ELAS, our method requires 24 MB, which is 37% of the memory used by OctoMap. Figure 5c also reveals that the resolution of ELAS only has a small impact on the memory consumption. The reason for the poor performance of OctoMap in this case is that the erroneous artifacts are highly unstructured, which impedes the octree compression.



(a) (b) (c)

Fig. 5: (a) Growth of mapped distance, and (b) processing times and (c) memory consumption for our method and OctoMap.

V. CONCLUSIONS

In this work we presented a new method for volumetric occupancy mapping. Our method is based on the well known OctoMap, which we extended with a new probability integration scheme that incorporates the probability that a voxel is actually visible. If we receive a measurement for a not visible voxel, this measurement must be erroneous and the occupancy probability for this voxel should not be updated. For estimating the visibility probability of a voxel, we traverse the measurement ray that passes through this voxel. For each voxel on the ray, we determine the probability that this voxel is locally occluded, by examining the occupancy probabilities of its neighbors. A visibility estimate is then computed by considering the probability that the voxel is locally occluded, and the probability that the previous voxel on the ray is visible. Further, we model the sensor depth error and consider it when updating the occupancy probability.

We designed this method specifically for use with stereo vision. Compared to laser scanners, the data received from a stereo vision system contains a much higher measurement noise, which tends to be correlated. In our evaluation we have shown that this situation can lead to a high number of erroneous map artifacts, when using the original OctoMap method. With our mapping approach, we are able to effectively remove those artifacts, which provides us a clear map of the observed area. Because our method does not exhibit such artifacts, the generated maps are also considerably smaller. In our evaluation, our method required as little as 32% of the memory consumed by OctoMap.

Although our method is more complex than OctoMap, we received a lower processing time for most test runs. This can mainly be credited to an optimization of the update reduction performed by OctoMap. In principle, this optimization could also be ported to OctoMap. However, a performance benefit will only be observed for dense measurements, as provided by a dense stereo algorithm. Even though we designed our method specifically for use with stereo vision, it can also be applied to other range sensors. When the range measurements exhibit a high measurement noise, we expect our method to deliver more robust results.

REFERENCES

[1] 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 2010 Workshop on Best Practice in 3D Percept. and Model. for Mobile Manipulation*, 2010.

[2] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*, 2012, pp. 4557–4564.

[3] K. Schauwecker and A. Zell, "On-Board Dual-Stereo-Vision for Autonomous Quadrotor Navigation," in *Int. Conf. on Unmanned Aircraft Syst. (ICUAS)*. IEEE, 2013, pp. 332–341.

[4] H. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar," in *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, vol. 2, 1985, pp. 116–121.

[5] T. Collins, J. Collins, and C. Ryan, "ccupancy Grid Mapping: An Empirical Evaluation," in *IEEE Mediterranean Conf. on Control & Autom. (MED)*, 2007, pp. 1–6.

[6] D. Murray and J. J. Little, "Using Real-Time Stereo Vision for Mobile Robot Navigation," *Autonom. Robots*, vol. 8, no. 2, pp. 161–171, 2000.

[7] M. Perrollaz, J.-D. Yoder, A. Nègre, A. Spalanzani, and C. Laugier, "A Visibility-Based Approach for Occupancy Grid Computation in Disparity Space," *IEEE Trans. on Intell. Trans. Syst.*, vol. 13, no. 3, pp. 1383–1393, 2012.

[8] R. Hadsell, J. A. Bagnell, D. Huber, and M. Hebert, "Accurate Rough Terrain Estimation with Space-Carving Kernels," in *Robot.: Sci. and Syst. (RSS)*, 2009.

[9] R. Triebel, P. Pfaff, and W. Burgard, "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*, 2006, pp. 2276–2282.

[10] J. Ryde and H. Hu, "3D Mapping with Multi-Resolution Occupied Voxel Lists," *Autonom. Robots*, vol. 28, no. 2, pp. 169–185, 2010.

[11] I. Dryanovski, W. Morris, and J. Xiao, "Multi-Volume Occupancy Grids: An Efficient Probabilistic 3D Mapping Model for Micro Aerial Vehicles," in *IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*, 2010, pp. 1553–1559.

[12] M. Bajracharya, J. Ma, A. Howard, and L. Matthies, "Real-Time 3D Stereo Mapping in Complex Dynamic Environments," in *Int. Conf. on Robot. and Autom.-Semantic Mapping, Percept., and Explor. (SPME) Workshop*, 2012.

[13] R. Shade and P. Newman, "Choosing Where to Go: Complete 3D Exploration with Stereo," in *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, 2011, pp. 2806–2811.

[14] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, G. Sorrenti, and J. D. Tardos, "RAWSEEDS: Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets," in *ROS200 Workshop on Benchmarks in Robot. Research*, 2006.

[15] G. Grisetti, D. L. Rizzini, C. Stachniss, E. Olson, and W. Burgard, "Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning," in *IEEE Int. Conf. on Robot. and Autom. (ICRA)*, 2008, pp. 1880–1885.

[16] A. Geiger, M. Roser, and R. Urtasun, "Efficient Large-Scale Stereo Matching," in *Asian Conf. on Comput. Vis. (ACCV)*. Springer, 2011, pp. 25–38.