An Uncertain 3D Grid Formulation for Indoor Environment Mapping

Nikos Mitsou[12], Sheraz Khan[2], Eirini Ntoutsi[34], Dirk Wollherr[2], Costas Tzafestas[1], Hans-Peter Kriegel[4]
[1]School of Electrical and Computer Engineering, National Technical University of Athens (NTUA), Greece
{nmitsou@mail,ktzaf@softlab}.ntua.gr
[2]Institute for Automatic Control Engineering, Technische Universität München (TUM), Germany
{nmitsou,sheraz.khan,dw}@tum.de
[3]Faculty of Electrical Engineering and Computer Science, Leibniz Universität Hannover, Germany
ntoutsi@kbs.uni-hannover.de
[4]Institute for Informatics, Ludwig-Maximilians Universität München (LMU), Germany
kriegel@dbs.ifi.lmu.de

***Abstract*— In this paper, the problem of creating an accurate 3D grid map by an autonomous mobile robot is addressed. The proposed algorithm focuses on two aspects of 3D mapping, an incremental and online segmentation process for surface extraction and a surface analysis based on uncertainty. The algorithm is based on an uncertain grid formulation that is used to store 3D point clouds collected by a robot with known poses. Unlike most grid based approaches that consider the distribution of the spatial attributes of the 3D points only for assigning these points into 3D cells, the proposed approach uses the uncertain 3D grid structure and propagates the uncertainty of the raw point clouds to the normal vectors and estimates the uncertainty of the plane parameters. For efficient surface extraction a state of the art stream clustering algorithm is employed. Experiments show that this grid formulation captures the actual distribution of points in the environment and leads to spatial models which contain additional information about their uncertainty.**

I. Introduction

Robotic mapping is the process of using a mobile robot for building a model of an unknown environment by collecting data through the robot sensors [1]. Mapping has immense importance in robotics, since crucial tasks such as navigation, planning and scene understanding rely heavily on the existence of accurate maps [2], [3]. The increase in computational power and the progress in sensor technology within the last decade has caused a shift in research focus from the construction of 2D maps towards building accurate 3D spatial models [4], [5]. Many approaches in this area utilize 3D grid structures to perform down-sampling of the original data and filter out noisy points. However, building meaningful 3D maps in an online fashion still remains a challenging problem. The high arrival rate and the huge amounts of collected data raise efficiency issues while the inherent errors of data collection complicate the data association.

Abstracting from point clouds towards 3D grid structures is a first step towards understanding the surroundings of the robot. Further processing has been proposed in order to extract more descriptive spatial entities such as planes and surfaces that lie in the environment. A variety of plane segmentation techniques has been proposed so far and most of them detect planes by either working with the spatial attributes of the points as in the RANSAC algorithm [6] or by extracting the normal vectors for each data point [7].

Contrary to most existing approaches, we propose a method that propagates the uncertainties of the raw data to the normal vectors and finally to the estimated plane parameters. In this way, our method can filter out cells with noisy normal vectors and also provide an estimation of the quality of each plane. Such a method is suitable for a mobile robot that navigates in an unknown environment and collects 3D point clouds through its sensors. The assumption that the robot poses are already known is followed. The proposed methods consists of three main components: (i) *point assignment to the 3D grid*, (ii) *normal vector estimation* and (iii) *surface clustering*. Once the robot receives a single scan of the environment, the first step involves the assignment of the points to the grid cells. The center of each grid cell represents a random variable based on the distribution of the points as well as their inherent uncertainty. The algorithm then tries to estimate the normal vector for the grid cells based on their nearest neighbors. The final step involves an uncertain stream clustering algorithm, which merges local clustered grid cells into one surface based on the spatial layout and the normal vectors similarity. This local spatial model, built upon a single scan, is then merged with the global representation (accumulated segments from scans over previous time instances). For the estimation of the error distribution of the raw points, we adopt a more generic model of uncertainty in which only the standard errors of the points are available. Experiments with a real robot which produces dense point clouds of indoor environments demonstrate that the algorithm can extract important spatial relations required to develop qualitative 3D maps.

The rest of the paper is organized as follows: A brief literature review is provided in Section II. The uncertain 3D grid and basic concepts are presented in Section III. Our method is presented in Section IV and experiments on both synthetic and real sensor data in Section V. Conclusions and future work are presented in Section VI.

II. Related work

The majority of the work in the domain of mobile robot mapping has focused on building spatial maps. As described in [1], almost all algorithms related to robot mapping (e.g., [8], [9]) incorporate uncertainty in the problem description thus leading to probabilistic mapping algorithms. Therefore, a proper formulation of uncertainty in the sensor measurements plays a vital role in all mapping algorithms.

Since the last few years, developing 3D spatial models has gained major interest in the robotics field [4], [10]–[12]. Most 3D mapping techniques operate directly on the point cloud and rely on segmentation and plane fitting techniques. In many of these works, similar approaches on surface detection and modeling have been followed. In [7], the authors use techniques such as outlier removal, re–sampling, segmentation and model fitting in order to reconstruct an indoor environment, in particular a kitchen. Their data scans refer to the whole environment, and their methods are applied over this static dataset. In [13], the authors present an optimization approach for segmenting planar regions of a grid based on normal vector uncertainty. In [14], the incremental nature of data acquisition has been considered. The authors use an incremental update of their representation by taking into account only the points that do not overlap with existing models. In [5], an incremental segmentation algorithm for 3D points has been proposed. For every point in the new scan its neighbours are found. If any of these points are already assigned to a cluster, then the new point is also assigned to it. If more than one clusters are candidates for assignment, then a merging of these clusters is performed. In [15], [16], two approaches for 3D semantic mapping of urban environments are presented. The received point cloud is segmented and planes are extracted. Their method though requires all points as input, i.e., it is static. In [17], the surface extraction is followed by classification. A set of hard-coded rules based on the position and size of the surfaces is additionally used to classify them as walls, floors, ceilings or doors.

In [18], 3D normal distribution transform has been presented which can be used for scan registration, localization and surface analysis. The approach models the grid cells using a normal distribution thus representing the surface by smooth functions. This allows generic optimization algorithms to be applied for scan registration. 3D-NDT has been used for loop closure based on histograms of surface shapes. Another application of 3D-NDT is surface analysis [18] (boulder detection in construction sites), however it only considers the surface smoothness within each cell of the grid rather than extracting planes from the sensor observations. In [19] a registration method for 3D mapping is defined based on noisy planes with covariance estimation [20]. It focuses mainly on registration of point clouds for 3D mapping whereas we assume the point clouds to be already registered. The plane and the parameter covariance is extracted directly from the point cloud, whereas our approach utilizes a grid structure (reducing computational load) and propagates uncertainties from the grid cells to the normals vectors. Although our error modeling assumptions are more simplistic compared to the above referenced work, however it allows an online and incremental segmentation of surfaces.

III. Uncertain 3D grid

In our scenario, a mobile robot autonomously navigates in an indoor environment without any prior knowledge about the environment. The robot perceives the world through its sensors by continuously scanning the environment. Each scan produces one point cloud and as the robot navigates in the environment a stream of point clouds is generated. The scans can be *overlapping*, meaning that subsequent scans can refer to the same areas in the environment. Starting from the first scan, our method efficiently detects objects that span more than one consecutive scans.

The *robot stream* consists of a sequence of registered 3D scans $\{S_1, S_2, \ldots, S_t\}$ arriving at time points $\{t_1, t_2, \ldots, t_t\}$. Each scan S_t is a point cloud, i.e., a set of 3D points, $S_t = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_i\}$. There is an inherent uncertainty associated with the location of each point $\mathbf{p}_i$, due to measuring device limitations. A common approach for handling this uncertainty is to assume knowledge of the underlying probability density function (PDF) that generates this uncertainty. However, PDFs are usually introduced as part of the modeling assumptions and roughly approximate the actual distribution. Therefore, we adopt a more flexible model of uncertainty where we assume that the standard error in the position of every 3D point is available. In our case, this error can be derived by the characteristics of the measuring device. For the kinect device, the standard errors along each axis have been analyzed in [21]. The error model takes into account the increase in the uncertainty based on the distance from the sensor. In this analysis, it has been shown that the random error in the depth measurement increases quadratically with the increase of the distance from the sensor. Furthermore, the deviation along the beam axis is always bigger than the deviation in the other two axes.

In order to incorporate the uncertainty in our method, we introduce the concept of the *uncertain point* which is represented as a tuple $(\mathbf{p}_i, \epsilon(\mathbf{p}_i))$, where $\mathbf{p}_i, \epsilon(\mathbf{p}_i) \in \mathbb{R}^3$ as in [22]. In particular, $\mathbf{p}_i = [x_i, y_i, z_i]^T$ are the values in the X, Y, Z dimensions, $\epsilon(\mathbf{p}_i)$ represents the errors in the point which consists of three random variables with zero mean and standard errors $\sigma_x, \sigma_y, \sigma_z$ for each dimension. As shown in [21], $\epsilon(\mathbf{p}_i)$ has been modeled to increase quadratically based on the distance from the sensor.

Due to the huge amount of data that is accumulated over time, it is inefficient to work upon the original raw data points. Therefore, we partition the data space into a 3D *grid* $\mathcal{U}$ consisting of *cells* $\{u_i\}$ of size ξ and we work on the grid cells afterwards instead of the original raw data. The grid cells are of the same size, however the grid itself is dynamic and expands as new data scans are accumulated from the robot stream. The number of points falling into a cell u comprises the *density* of the cell, denoted by $d(u)$. Since some cells might contain only a small number of points, we

employ a *density threshold* τ to distinguish between *dense* and *noisy* cells. For each cell we maintain a set of statistics.

Definition 1 (Cell statistics): Let u be a cell in the grid and let $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the set of uncertain points mapped to u. The cell statistics contain the following entries:

- The linear sum of the points for each dimension, $\mathbf{L} \in \mathbb{R}^3$: $\sum_{i=1}^{n} \mathbf{p}_i$.
- The square sum of the points for each dimension, $\mathbf{S} \in \mathbb{R}^3$: $\sum_{i=1}^{n} \mathbf{p}_i^2$.
- The square sum of the errors for each dimension, $\mathbf{S}_s \in \mathbb{R}^3$: $\sum_{i=1}^{n} \epsilon(\mathbf{p}_i)^2$.
- The density of u, $d(u) = n$.

When a new point is added to a cell u, the cell statistics are updated by summing up the point coordinates and errors to the corresponding entries of the cell. Based on these statistics, we compute the *virtual center* of the cell, $\mathbf{C}_u$, which is a random variable given by the current instantiation of the center and the mean of the errors associated with the points in the cell:

$$\mathbf{C}_u = \sum_{i=1}^{n} \mathbf{p}_i/n + \sum_{i=1}^{n} \epsilon(\mathbf{p}_i)/n \tag{1}$$

Note that the error is a random variable with zero mean. Therefore, $E[\epsilon(\mathbf{p}_i)] = 0$. This means that by considering the expected value of the center, the error factor is eliminated. To count for the error factor, we estimate the square of the Euclidean norm.

Lemma 3.1 (Virtual center of a cell): Let u be a cell in the grid and let $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ be the set of uncertain points mapped to u. The virtual center of u is also a random variable, denoted by $\mathbf{C}_u$. The following formula holds:

$$E[||C_u||^2] = \frac{(\sum_{i=1}^{n} \mathbf{p}_i)^2}{n^2} + \frac{(\sum_{i=1}^{n} \epsilon(\mathbf{p}_i))^2}{n^2} \tag{2}$$

Proof:

We start from Equation 1:

$$E[||\mathbf{C}_u^2||] = E[||\frac{\sum_{i=1}^{n} \mathbf{p}_i}{n} + \frac{\sum_{i=1}^{n} \epsilon(\mathbf{p}_i)}{n}||^2]$$

and evaluate the right hand side:

$$\frac{1}{n^2}[E[||\sum_{i=1}^{n} \mathbf{p}_i||^2] + E[||\sum_{i=1}^{n} \epsilon(\mathbf{p}_i)||^2] + 2E[\sum_{i=1}^{n} \mathbf{p}_i \sum_{i=1}^{n} \epsilon(\mathbf{p}_i)]]$$

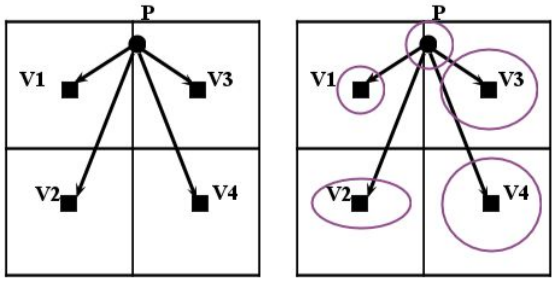
$$\frac{1}{n^2}[E[||\sum_{i=1}^{n} \mathbf{p}_i||^2] + E[||\sum_{i=1}^{n} \epsilon(\mathbf{p}_i)||^2] + 2E[\sum_{i=1}^{n} \mathbf{p}_i]E[\sum_{i=1}^{n} \epsilon(\mathbf{p}_i)]]$$

Since $E[\epsilon(\mathbf{p}_i)] = 0$, the final term cancels out and the above formula is simplified to the desired result (2). ■

The virtual center can be easily derived by the statistics of the cell. So, Equation 2 can be re-written as:

$$E[||\mathbf{C}_u||^2] = \frac{\mathbf{L}^2}{d(u)^2} + \frac{\mathbf{S}_s}{d(u)^2} \tag{3}$$

It is essential to define a mechanism through which a point is mapped to its corresponding cell in the grid. The standard approach is to perform the point assignment based on the Euclidean distance without considering the uncertainty of the point or the virtual centers of the cells as shown in Figure 1(a). In this paper, the uncertain point is mapped to the virtual centers of the grid using the expected distance. In Figure 1(b) we show the expected distance between a point and the virtual centers of four grid cells by arrows of different lengths. The uncertainties are represented by axes aligned ellipses.



(a) (b)

Fig. 1. Mapping a point to the grid without- (a) and with- uncertainty (b) (Points are represented by circles and virtual centers by squares.)

The expected distance between an uncertain point and the uncertain virtual center of a cell is defined as follows:

Lemma 3.2 (Expected distance): Let u be a cell in the grid containing the points $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and let $\mathbf{C}_u$ be its virtual center. Let $\mathbf{p}_j$ be an uncertain point, such that it has not been assigned to any cell in the grid. The expected value of the square of the distance between $\mathbf{p}_j$ and $\mathbf{C}_u$ is given by:

$$E[||\mathbf{p}_j - \mathbf{C}_u||^2] = \mathbf{p}_j^2 + \epsilon(\mathbf{p}_j)^2 + \frac{(\sum_{i=1}^{n} \mathbf{p}_i)^2}{n^2} + \frac{(\sum_{i=1}^{n} \epsilon(\mathbf{p}_i))^2}{n^2} - 2\mathbf{p}_j \frac{\sum_{i=1}^{n} \mathbf{p}_i}{n} \tag{4}$$

Proof: The expected value of the square of the distance is given by:

$$E[||\mathbf{p}_j - \mathbf{C}_u||^2] = E[||\mathbf{p}_j||^2] - 2E[\mathbf{p}_j\mathbf{C}_u] + E[||\mathbf{C}_u||^2]$$

Assuming $\mathbf{p}_j$ and $\mathbf{C}_u$ are independent, it holds: $E[\mathbf{p}_j\mathbf{C}_u] = E[\mathbf{p}_j]E[\mathbf{C}_u]$, and thus:

$$E[||\mathbf{p}_j - \mathbf{C}_u||^2] = E[||\mathbf{p}_j||^2] + E[||\mathbf{C}_u||^2] - 2E[\mathbf{p}_j]E[\mathbf{C}_u]$$

Also, since $\mathbf{p}_j$ is a random variable with expected value its current instantiation and error $\epsilon(\mathbf{p}_j)$, it holds that:

$$E[||\mathbf{p}_j||^2] = (E[\mathbf{p}_j])^2 + (E[\epsilon(\mathbf{p}_j)])^2 - 2E[\mathbf{p}_j]E[\epsilon(\mathbf{p}_j)]$$

Also, since the mean of the error is zero $E[\epsilon(\mathbf{p}_j)] = 0$:

$$E[||\mathbf{p}_j||^2] = E[\mathbf{p}_j]^2 + E[\epsilon(\mathbf{p}_j)]^2 = \mathbf{p}_j^2 + \epsilon(\mathbf{p}_j)^2 \tag{5}$$

The term $E[||\mathbf{C}_u||^2]$ can be computed as in Equation 2. Also, it holds:

$$E[\mathbf{p}_j]E[\mathbf{C}_u] = \mathbf{p}_j \frac{\sum_{i=1}^{n} \mathbf{p}_i}{n} \tag{6}$$

Combining Equations (2), (5), (6), we reach to the desired result of Equation (4). ■

The expected distance between a point and a cell can be easily derived based on the value and the error components of the point and the virtual center of the cell.

Equation 4 describes the analytical case, i.e., when the raw data points of the cell are available. In our case, the cell statistics are available so we can re-phrase Equation (4) based on the cell statistics as follows:

$$E[||\mathbf{p}_j - \mathbf{C}_u||^2] = \mathbf{p}_j^2 + \epsilon(\mathbf{p}_j)^2 + \frac{\mathbf{L}^2}{d(u)^2} + \frac{\mathbf{S}_s}{d(u)^2} - 2\mathbf{p}_j \frac{\mathbf{L}}{d(u)} \tag{7}$$

So, we can use Equation (7) to estimate the expected square distance between a point and the virtual center of a cell. A point is then assigned to the cell with the smallest expected square distance.

IV. Uncertain 3D clustering

For the robot, point clouds do not offer any direct information about the spatial structure of the environment. To allow for geometric interpretation and abstraction, the robot should be able to process this data and extract patterns of spatial information. Typically, surfaces are used for the description of an environment; walls, doors, tables consist of (or can be partitioned into component) surfaces. Thus, the patterns we are focusing on in this work are *surfaces*. However, 3D point clouds represent only a noisy sampling of surfaces that exist in the real world and the explicit information about the *orientation* and *curvature* of the surfaces is lost during the sampling process. Normal vector estimation aims at restoring this information for every sampled point by constructing a vector that is orthogonal to the tangent plane of that point. In order to do so, existing methods utilize techniques such as the least square plane fitting [23], [24].

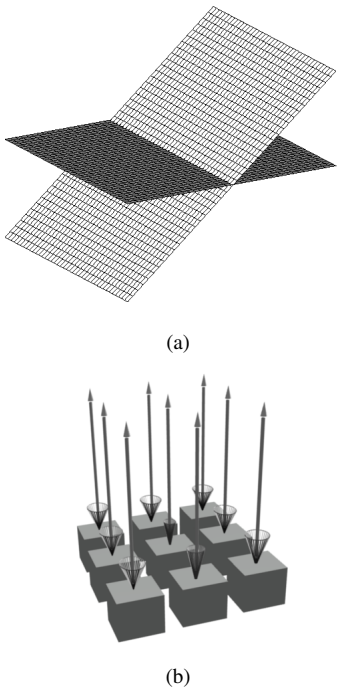
A. Normal vector estimation for grid cells

In our case, the notion of normal vector for points is extended to grid cells and also it considers the uncertainty of the points. For the normal vector computation, the cell neighborhood is first defined.

Definition 2 (Cell neighborhood):
Let u be a cell. Let d be the depth parameter, $d \geq 1$. The neighborhood of u in depth d, denoted by $N_d(u)$, consists of: *i)* all cells u' that are directly connected to u, and *ii)* all cells u'' for which there exists a path of cells $\langle u_1, u_2, \ldots, u_d \rangle$, $u_1 = u$, $u_d = u''$ such that u_{i+1} is directly connected to u_i, $1 \leq i \leq d$.

The normal vector of a cell is estimated by computing the total least square plane fitting its neighborhood. Recall that each cell is represented in terms of a virtual center. To account for the uncertainty of the virtual centers and estimate its effect on the normal vectors, we employ a *min-max approach* that computes the maximum variation in the parameters of the plane. In particular, we perturb each virtual center proportionally to its uncertainty based on the center of the plane and calculate the best fitting plane in each case. The variation in the plane parameters is a direct indicator of the uncertainty in the normal vectors. The basic idea behind this approach is graphically illustrated in Figure 2(a).

Definition 3 (Normal vector of a cell):
Let u be a cell and let $N_d(u)$ be its neighborhood in depth d. The uncertain normal vector of u, $\overrightarrow{u}$, is a random variable with an expected value given by the best plane fitting $N_d(u)$ and a deviation approximated by the min-max approach.



(a)

(b)

Fig. 2. Computing plane uncertainty with the min-max approach (a). Representing uncertainty in the normal vectors (b).

An example of an uncertain normal vector is shown in Figure 2(b). The normal vectors are shown as arrows, whereas the uncertainty of a normal vector is represented by a wire frame cone extruding from the virtual center.

B. Plane extraction

The plane extraction algorithm aims at extracting planes from the grid by merging grid cells based on their normal vector similarity and spatial vicinity. The pseudocode of the algorithm is depicted in Figure 3. It takes as input the current scan S_t at time point t, the global planes extracted up to time point $t-1$, the density threshold τ for the determination of the dense grid cells and the orientation threshold ϕ which is used to determine similar normal vectors. The output of the algorithm is the global planes at time point t.

For every new scan the points in the scan are first mapped onto the grid and cell statistics are maintained (lines 2–3). The dense cells u, with $d(u) > \tau$, are then extracted (line 4) and used to create local planes, by merging grid cells based on their spatial vicinity and normal vector similarity (line 5). The final step involves merging the local planes with the global planes (accumulated over previous time point) to form a consistent global map (lines 6–9) .

Below, we present the details of the local planes extraction and the global planes extraction steps.

Algorithm Uncertain3DClustering()
Input: Global planes Θ_{t-1},
Scan S_t,
density threshold τ,
orientation threshold ϕ,
Output: Global planes Θ_t
begin
1. while (S_t) begin
2. Map S_t onto the grid
3. Maintain cell statistics for each cell u
4. Extract the dense cells D_u w.r.t. τ;
5. θ_t = extractLocalClusters(D_u, ϕ);
6. if ($\Theta_{t-1} = null$) *//the first scan*
7. $\Theta_t = \theta_t$;
8. else
9. Θ_t = merge(Θ_{t-1},θ_t, ϕ);
10. end;
end;

Fig. 3. The pseudocode of the Uncertain3DClustering() algorithm.

1) Local planes extraction: In this step, new surfaces are created by starting with some random cell $u \in D_u$. The normal vector of the local planes l is initialized to the normal vector of u and the algorithm tries to expand the plane l based on the directly connected dense cells u' and normal vector similarity. The expansion is possible iff u' and the local plane l belong to similar surfaces, i.e. if $\overrightarrow{l}.\overrightarrow{u'} < \phi$ (dot product) where $\overrightarrow{l}$ and $\overrightarrow{u'}$ represent the uncertain normal vectors of the local plane and the candidate cell for merging, respectively. Since $\overrightarrow{l}$ and $\overrightarrow{u'}$ are uncertain normal vectors, we compute their expected distance according to Equation (7). If the addition is possible, the statistics of l are updated so as to consider the effect of u. We maintain plane statistics similarly to cell statistics (Definition 1). The procedure continues until the local plane l cannot be further expanded (due to e.g., lack of directly connected dense cells or due to the violation of the orientation distance threshold ϕ). The algorithm restarts from some other dense cell $u'' \in D_u$ that has not been visited yet and continues until all dense cells have been visited. The output is a set of local planes $\{l_1, l_2, \ldots, l_n\} \in \theta_t$ as denoted in Figure 3.

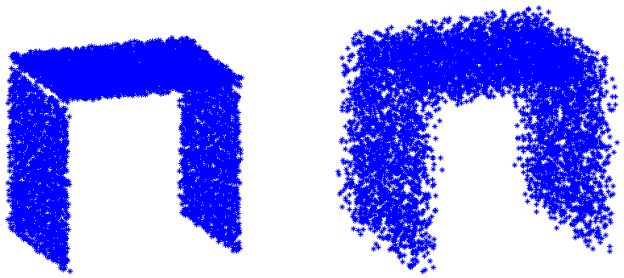
2) Global cluster extraction: The so far built global planes Θ_{t-1} should be updated according to the local planes θ_t discovered on the current scan S_t. The update or merging is done on the basis of i) the vicinity between the local and the global planes, and ii) their orientation similarity. Intuitively, a local plane is considered as a *continuation* of a global plane if they are close to each other in the grid and also if their corresponding surface orientations are similar. A local plane might be *absorbed* by a single global plane or by more than one global planes resulting in *plane merge*. If neither of the two cases hold, the local plane might comprise the start of a *new* global plane.

V. Experiments

In this section, we experimentally evaluate our approach on both synthetic and real data. On synthetic data, we evaluate the notion of virtual centers and examine how the uncertainty of the points effects the point distribution in the cells and the normal vectors. On the real data, collected by a mobile robot navigating in an unknown indoor environment, we compare our method to the RANSAC algorithm [6] in terms of cluster formation and normal vector estimation.

A. Results on synthetic data

Our synthetic dataset consists of three tangent surfaces that form a table structure as shown in Figure 4. The purpose of the experiment is to evaluate how accurately the grid captures the real distribution of the points. Different levels of noise are added to the 3D points and the distances between the points and cell centers are calculated for both cases (fixed grid centers and virtual centers).



(a) $Noise = 0.1$ (b) $Noise = 0.2$

Fig. 4. Our synthetic dataset under different levels of uncertainty/error in the points.

In Figure 5, the average distances between the actual data points and the virtual centers of the uncertain grid (blue color) as well as with the fixed grid cell centers (red color) are shown. We can see that the uncertain grid cells approach represents the actual distributions of the points in a more accurate manner than the fixed centers approach. For both approaches, the quality of the grid approximation drops with increasing uncertainty.

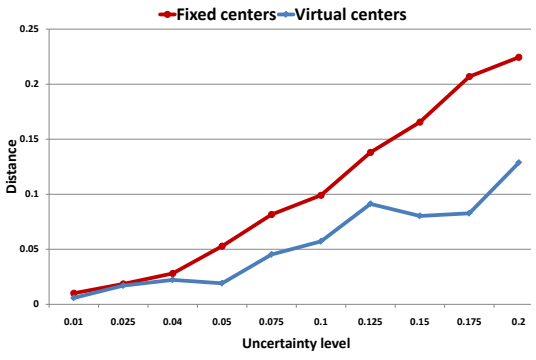


Fig. 5. Distances between actual data and cell centers with respect to point uncertainty/error.

For two different levels of uncertainty in the dataset (first case 0.1, second case 0.2), the average standard devations for all normal vectors were calculated. In the first case, the

deviation was measured to be equal to 3.82° while in the second case it was equal to 7.34°. The fact that the ratio between the uncertainty on the points and on the normal vectors for both levels of uncertainty is very close is an indication that our *min-max* approach presented in Section IV is correct.

B. Results on real data

We evaluate our approach in an actual scenario where a mobile robot navigated in an indoor environment for more than 200 s with an average speed of 0.3 meters per second. The experimental environment consisted of several objects (e.g. walls, doors, tables and chairs). Scans were collected through a kinect device mounted on top of the robot. The typical point cloud size from the kinect device is 300,000. A size of 8cm was selected for the grid cell size and the experiment was conducted on a 3 GHz AMD Phenom® II X4 with 8GB memory, running Ubuntu 12.04 and ROS Fuerte.

1) Time efficiency: Figure 7(a) shows the time required to process each incoming scan and generate an up-to-date map of the environment. Figure 7(b) shows the number of planes extracted over time. Depending on the size of the received point cloud, each step can take from 0.07 up to 0.72 seconds and the average time required per scan is 0.33 seconds.

In Figure 6, a detailed time analysis for a period of 45 seconds is presented where the processing time for each step of the algorithm is presented. As shown in the above mentioned figure, the most time consuming step is the assignment of points to the corresponding 3D cells. This is expected since for every new point, the distance between the point and different virtual centers has to be calculated. Another interesting point in this figure is that the total time for the whole algorithm varies a lot. This happens due to the field of view of the sensor which leads to different sizes of point clouds.

Fig. 6. Time analysis for the algorithm and each module separately

2) Comparison with RANSAC: In Figure 9, the planes found by our algorithm and a typical offline RANSAC implementation are presented. In the RANSAC implementation, the point clouds were accumulated over time and planes were extracted when all point clouds had been received. In order to evaluate the quality of our results, we compare the normal vectors of our algorithm with those produced by RANSAC. From the visual inspection of the image, it is clear that both algorithms are able to extract the majority of the planes of the environment. However, our algorithm was unable to detect parts of the door in one specific region (dark red plane) which is present in the results from RANSAC. The main reason for this is the high uncertainty in the normal vectors in those regions.

In order to measure how well our method compares to RANSAC, we associate the planes of the two sets and calculate the average normal vector difference between them. The average normal vector distance for the detected 15 planes in Figure 9 was found to be 5.96° which indicates that our approach can extract accurate planes in an online fashion.

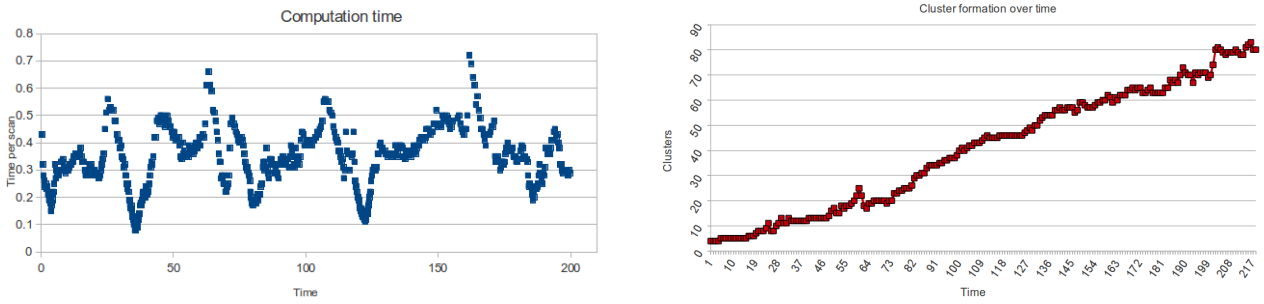
3) Normal vector uncertainty: In order to demonstrate that our algorithm can model the uncertainty of the normal vector for each grid cell accurately, we present the 3D grid used for the RANSAC comparison showing the normal vector uncertainty based on color in Figure 8. The dark colored cells denote normal vectors with very low uncertainty whereas the bright red colored cells correspond to normal vectors with higher uncertainty. It can be seen that cells which lie on the edges of planes or which are at higher distance from the robot such as the segment on the top right have higher uncertainty.

4) 3D map: The 3D map constructed by the robot is presented in Figure 10. Different colors indicate different surfaces. From the visual inspection of this map, it is clear that the robot captured all essential properties of the environment. Most parts of the ceiling and the floor belong to the same clusters. However, certain parts of the same plane are not always connected and are represented in different colors since the algorithm utilizes the spatial vicinity as a criterion for merging. In some corners and areas that the robot turned, less number of clusters were detected since those areas remained outside of the field of view of the sensor.

VI. Conclusions and Outlook

In this paper, we present an uncertain 3D grid concept for the problem of mapping. The proposed structure takes into account the inherent uncertainty of the sensor points for assigning the points to the grid and furthermore, it introduces the concept of uncertain 3D cells for modeling uncertainty at the cell level. The uncertainty of the grid cells is propagated to the surfaces, which are also modeled as uncertain entities. Our method maintains the grid online and thus all the necessary measures can be derived online. We couple the grid with a fast stream clustering algorithm and we are able to derive and maintain in an online fashion the surface clusters as new scans are accumulated over time. Our experiments show that the uncertain 3D grid formulation is able to efficiently detect planes and model uncertain areas in the environment.

As future work, different directions are planned to be investigated. Firstly, experiments and comparisons need to be done with other techniques. Secondly, an evaluation of our method in outdoor environments where sensor noise is usually higher and the environment is less structured is planned. Finally, a different approach that eliminates the



(a) Time per scan

(b) Number of clusters over time

Fig. 7. Cluster Formation & Time analysis

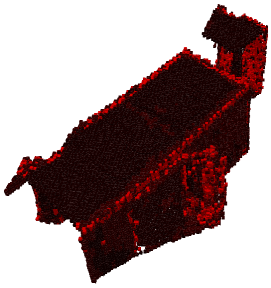


Fig. 8. Uncertainty of normal vectors. Red color means higher uncertainty

grid structure and represents the environment through virtual centers that are allowed to freely move around following the distribution of the data will be examined.

REFERENCES

[1] S. Thrun, "Robotic mapping: a survey," in *Exploring artificial intelligence in the new millennium*, G. Lakemeyer and B. Nebel, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1–35.

[2] ——, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[3] ——, "Learning occupancy grid maps with forward sensor models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, 2003.

[4] J. Weingarten and R. Siegwart, "Ekf-based 3d slam for structured environment reconstruction," in *IEEE International Conference on Intelligent Robots and Systems*, 2005, pp. 3834–3839.

[5] K. Klasing, D. Wollherr, and M. Buss, "Realtime segmentation of range data using continuous nearest neighbors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communication of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[7] R. B. Rusu, Z. C. Marton, N. Blodow, A. Holzbach, and M. Beetz, "Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments," in *IEEE International Conference on Robotics and Systems (IROS)*, 2009.

[8] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[9] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *International Joint Conference on Artificial Intelligence*, vol. 18. LAWRENCE ERLBAUM ASSOCIATES LTD, 2003, pp. 1151–1156.

[10] 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.

[11] S. Khan, A. Dometios, C. Verginis, C. Tzafestas, D. Wollherr, and M. Buss, "Rmap: a rectangular cuboid approximation framework for 3d environment mapping," *Autonomous Robots*, vol. 37, no. 3, pp. 261–277, 2014.

[12] S. Khan, D. Wollherr, and M. Buss, "Adaptive rectangular cuboids for 3d mapping," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2132–2139.

[13] S. Khan, N. Mitsou, D. Wollherr, and C. Tzafestas, "An optimization approach for 3d environment mapping using normal vector uncertainty," in *12th International Conference on Control Automation Robotics Vision (ICARCV)*, 2012, pp. 841–846.

[14] Z. C. Marton, R. B. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy datasets," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[15] D. Wolf, A. Howard, and G. Sukhatme, "Towards geometric 3d mapping of outdoor environments using mobile robots," in *IEEE International Conference on Robotics and Systems (IROS)*, 2005.

[16] R. Valencia, E. Teniente, E. Trulls, and J. Andrade-Cetto, "3d mapping for urban service robots," in *IEEE International Conference on Robotics and Systems (IROS)*, 2009.

[17] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, 2008.

[18] M. Magnusson, "The three-dimensional normal-distributions transform — an efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro University, 2009, Örebro Studies in Technology 36.

[19] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-d mapping," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

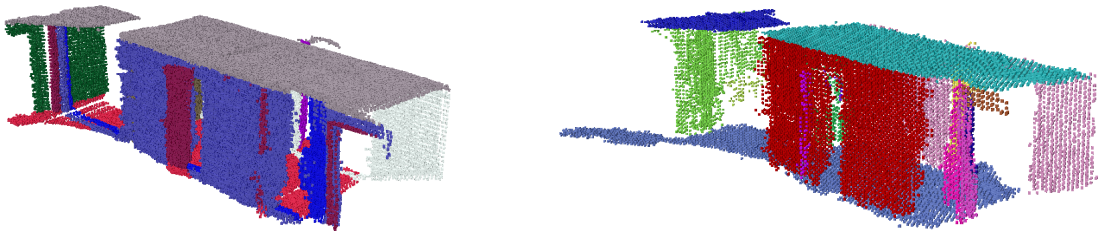
[20] K. Pathak, N. Vaskevicius, and A. Birk, "Uncertainty analysis for optimum plane extraction from noisy 3d range-sensor point-clouds," *Intelligent Service Robotics*, vol. 3, no. 1, pp. 37–48, 2010.

[21] K. Khoshelham and S. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[22] C. C. Aggarwal and P. S. Yu, "A framework for clustering uncertain data streams," in *IEEE International Conference on Data Engineering (ICDE)*, 2008, pp. 150–159.

[23] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

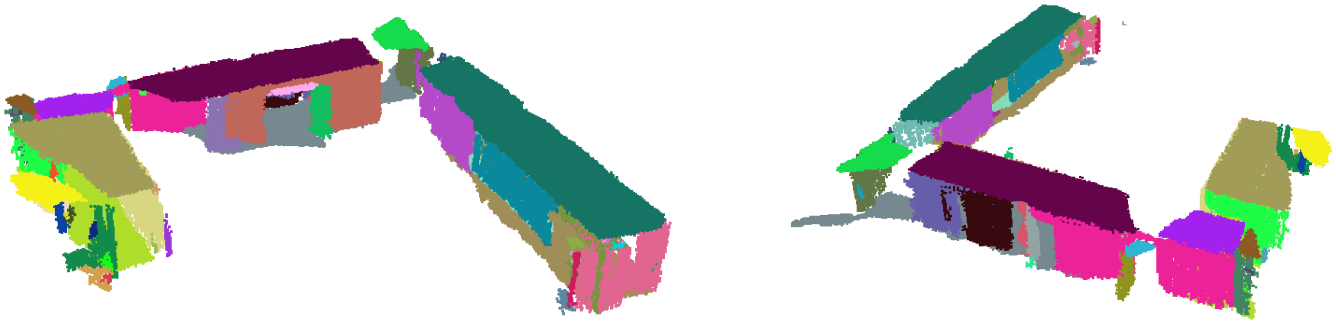
[24] K. Pathak, N. Vaskevicius, and A. Birk, "Revisiting uncertainty analysis for optimum planes extracted from 3d range sensor point-clouds," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1631–1636.



(a) Planes extracted by the RANSAC algorithm

(b) Planes extracted by our algorithm

Fig. 9. Extracted planes by the two methods



(a)

(b)

Fig. 10. The 3D map of the environment under two different perspectives