Fast Plane Detection for SLAM from Noisy Range Images in Both Structured and Unstructured Environments

Junhao Xiao, Jianhua Zhang and Jianwei Zhang
Department of Computer Science
University of Hamburg
D-22527, Hamburg, Germany
{*xiao, jzhang, zhang*}*@informatik.uni-hamburg.de*

Houxiang Zhang and Hans Petter Hildre
Department of Technology and Nautical Science
Aalesund University College
N-6025, Aalesund, Norway
{*hozh,hh*}*@hials.no*

***Abstract*—This paper focuses on fast plane detection in noisy range images. First, two improvements to the state-of-the-art region growing algorithm are presented to make it faster without losing precision for unstructured environments. One is to add the seed selection procedure based on local shape information to avoid blind growth. The other is to simplify the plane fitting mean square error computation complex. Second, a novel algorithm called grid-based region growing is presented for structured environments. The point cloud is divided into small patches based on neighborhood information when it is viewed as a range image. The small patch is called grid. Then the grids are classified into different categories according to their local appearance, including sparse, planar, spherical and linear. Finally, the planar grids are clustered into big patches by region growing. The plane parameters are incrementally computed whenever a new grid is added. The resulting planes can be used for 3D plane simultaneous localization and mapping (SLAM). Experimental results show promising plane detecting speed for both structured and unstructured environments.**

***Index Terms*—Plane detection, SLAM, 3D feature extraction**

I. INTRODUCTION

Range sensors, e.g., laser range finders, time-of-flight cameras and stereo vision are becoming more and more popular in the application of mobile robotic systems. Noisy range images from such sensors can be used for various kinds of tasks, such as navigation [1], simultaneous localization and mapping (SLAM) [2], [3], and recognition [4]. If there are objects with planar surfaces in the scenario, such as floors, doors, walls, ceilings, tables and chairs, the surfaces can be extracted as polygons which provide a compressive representation of the point cloud. The planar patch has been found to be a good feature for SLAM. Moreover, 3D Plane SLAM [5], [6] has been proved to be faster and more reliable than the classic iterative corresponding point (ICP) [7] algorithm and the recently proposed normal distribution transformations (NDT) algorithm [8].

Plane detection in range images is a complex task which has attracted increasing attention from both the computer graphics and robotics community in recent years. The computer graphics algorithms rely on relatively exact range data while the range images from robotic sensors are too noise-prone. Therefore, methodologies from computer graphics are not suitable for robotics [9]. As a result, researchers from the robotics community have proposed algorithms on this topic [9]–[12].

Hähnel *et al.* [9] proposed the region growing algorithm in order to learn a smooth model of indoor/outdoor environments. The algorithm is time consuming. Following this route, Poppinga *et al.* presented two optimizations which made it faster [12]. It will be accelerated further here without losing any precision. First, the seed selection procedure is introduced to avoid blind region growth. Second, an efficient method for computing the plane fitting mean square error (MSE) is presented. The modified algorithm is called point-based region growing algorithm.

The work of Weingarten *et al.* [10] and Kaushik *et al.* [13] is closely related to our novel algorithm. In [10], the point cloud was discretized into small partitions. However, the neighborhood information in the range image has not been employed. Instead, the octree data structure was used to construct small cubes. For each cube, RANSAC [14] is employed to find coplanar points. Then the least squares method is used to fit an optimal plane to the coplanar points. After that, the small patches are merged into big ones. However, the parameters for the resulted planes have not been provided. The algorithm is time consuming since RANSAC is used. In [13], the point cloud was also divided into small grids, and plane parameters are computed for each grid. The grids are clustered into big planes by the so called Breadth-First-Search algorithm. One drawback is that there are some grids whose appearance can not be approximated by a plane. Furthermore, as in [10], the parameters of the resulted planes are not given which is a very fundamental issue for plane SLAM. To deal with this problem, an incremental version is proposed to compute the plane parameters whenever a new grid is added to the growing region. The algorithm is called grid-based region growing.

The remainder of this paper is organized as follows: The point-based region growing algorithm and the grid-based region growing algorithm will be presented in Section II and Section III respectively. Experimental results follow in Section IV. The conclusion and future work will be drawn in the last section.

II. Point-based Region Growing Plane Detection

The region growing algorithm for detecting planes in noisy point clouds was proposed in [9] and was modified to a incremental version in [12]. One further step is made here to accelerate the speed without losing precision. Part A details the algorithm procedure and part B formulates the efficient MSE computation.

A. Point-based Region Growing Algorithm

Our improved algorithm proceeds as follows: one point p_s is selected from the point cloud data (PCD), and its qualification as a new seed is examined (Algorithm 1, line 4 – 5). The point p_s will be regarded as a new seed if it meets the following two criteria: First, except its investigated neighbors in the range image, there are six or more points within distance δ. Second, the local appearance of p_s is planar. Except the threshold δ, the procedure is the same as the shape classification step in the grid-based algorithm (This will be detailed in Section III-A). If p_s is a new seed, it together with its neighbors within distance δ will be marked as a new growing region GR (Algorithm 1, line 6). GR is extended by considering its neighbors within distance δ. Suppose the considering point is p_c, it will be added to GR if: First, the distance from p_c to the optimal plane of $GR \cup p_c$ is less than γ; Second, the mean square error (MSE) of the points in $GR \cup p_c$ to the optimal plane is less than ϵ. The growth will continue until no more points can be added to GR (Algorithm 1, line 7 – 12). Afterwards, if it contains more than θ points, it will be assigned to be a new plane and added to the regions set R. Otherwise, it is added to the uncertain area R' (Algorithm 1, line 14 – 18).

Algorithm 1 Point-based Region Identification

Input: PCD: point cloud data
Output: R: regions constructed by coplanar grids

1: $R \leftarrow \emptyset$
2: $R' \leftarrow \emptyset$
3: **while** $(PCD \setminus (R \cup R') \neq \emptyset)$ **do**
4: $select\ point\ p_s\ in\ PCD \setminus (R \cup R')$
5: **if** $isNewSeed(p_s) == true$ **then**
6: $GR = initializeSeed(p_s)$
7: **while** $new\ point\ can\ be\ found$ **do**
8: $select\ a\ neighbor\ p_c\ with\ dis(GR, p_c) < \delta$
9: **if** $(MSE(GR \cup p_c)) < \epsilon$ && $dis(plane(GR \cup p_c), p_c) < \gamma)$ **then**
10: $GR \leftarrow GR \cup p_c$
11: **end if**
12: **end while**
13: **end if**
14: **if** $size(GR) \geq \theta$ **then**
15: $R \leftarrow R \cup GR$
16: **else**
17: $R' \leftarrow R \cup GR$
18: **end if**
19: **end while**

The algorithm ends when every point has been assigned to R or R'. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are preset thresholds. One parameter tuning step is needed for one specific range sensor.

B. Efficient MSE Computation

The background of fitting the optimal plane to a point set is explained below, it is also used to find local plane parameters for the grid-based region growing algorithm. Suppose the coplanar points are $p_i = (x_i, y_i, z_i)^{\mathrm{T}}, i = 1, ...K$. In order to find the optimal plane, based on the idea of least square fitting, the sum of orthogonal distances to the plane should be minimized.

The mass center m of the point set is defined as $m = (\sum_{i=1}^{K} p_i)/K$. A matrix C which is similar to the covariance of the point positions can be computed as $C = \sum_{i=1}^{K}(p_i - m)(p_i - m)^{\mathrm{T}}$. Obviously, C is the product of the position covariance and the number of points. Hessian plane parameters is used in this paper. Suppose that the equation of the plane is $\vec{n} \cdot p = d$, where p is an arbitrary point on the plane, $\vec{n}$ is the normal direction and d is the orthogonal distance from the plane to the origin. Now the goal is to minimize $\sum_{i=1}^{K}(\vec{n} \cdot p_i - d)^2$. For consistency, all the normal vectors have been normalized and their direction is from the origin to the plane, which means $d > 0$.

In order to reach the minimal, two valuable results are reached after some linear algebra. First, the mass center is located on the optimal plane. In other words, $\vec{n} \cdot m = d$. Second, the normal direction is the eigenvector of matrix C corresponding to its minimal eigenvalue. After plane fitting, MSE of the point set to the optimal plane can be computed as:

$$MSE = \frac{1}{K}\sum_{i=1}^{K}(\vec{n} \cdot p_i - d)^2 \tag{1}$$

One main contribution of [12] is an incremental version of computing the MSE and C whenever a new point is added. However, the relationship between MSE and C has not been realized while C has been tracked for plane parameters. The mathematical derivation of the relationship is drawn below, (1) can be rewritten as

$$MSE = \frac{1}{K}\sum_{i=1}^{K}(\vec{n} \cdot p_i - \vec{n} \cdot m)^2 \tag{2}$$

Equation (2) gives $MSE = (\vec{n}^{\mathrm{T}} C \vec{n})/K$. Note that $\vec{n}$ is the eigenvector of C corresponding to the minimal eigenvalue. This yields a nice result:

$$MSE = \frac{1}{K}\lambda_{min}(C) \tag{3}$$

where $\lambda_{min}(C)$ stands for the minimal eigenvalue of matrix C. Compared to [12], our algorithm not only save half the memory by tracking less variables but also reduce the computation cost. The algorithm evaluation will be presented in Section IV after proposing the grid-based algorithm.

III. Grid-based Region growing

From the application point of view, the advantage of the point-based region growing algorithm is that it can be used in both structured and unstructured environments. Planar detection in structured environments could be faster with our proposed grid-based region growing. This idea is motivated by two observations: First, a structured environments is mainly constructed by big planes; second, the point cloud is closer to uniform sampling than in a unstructured environment. Therefore, if the point cloud is divided into small grids, most of them should be in a planar shape. Then optimal planes can be fitted to grids with planar appearance. After plane fitting, the small planar patches from the same physical object surface should have similar parameters. Only the planar grids are considered for plane detection. Like our point-based region growing algorithm, the seed is selected first, then it is extended by considering its neighbors. Compared to [13], our algorithm computes the plane parameters for the growing region whenever a new grid is added. This is advantageous for plane SLAM which cannot be performed without plane parameters. The algorithm will be presented in detail in the second part of this section.

A. Feasibility of Grid-based Region Growing

The grid-based algorithm is suitable for structured environments if the following two assumptions are confirmed. First, most of the grids which locate on plane surfaces have a planar appearance. Second, the grids from the same physical surface have similar plane parameters. To confirm the first assumption, the grids are classified into different categories. Using the same notations as Section II, a matrix C is computed for each grid. Given its sorted eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ from a grid g. The appearance of g can be decided by the following criteria:

---

if $size(g) < \mu$ **then**
  $g \subset sparse$
else if $\lambda_2 \leq \alpha\lambda_3$ **then**
  $g \subset linear$
else if $\lambda_1 \leq \beta\ \lambda_2$ **then**
  $g \subset planar$
else
  $g \subset spherical$
end if

The grid size is fixed to 3×3. g will be marked as a sparse grid if it contains less than μ valid points. μ is set to seven since less points will probably cause a problem to the shape classification criteria. Taking six valid points as example, there may be one row or one column which is invalid. Regardless of the orientation, only two such configurations exist as shown in Fig.1. The right one has degenerated to 2×3 and needs a new set of shape parameters. Testing whether a grid with six or less points is degenerated will increase the computational complexity. As a result, only grids with more than six points are picked up. α and β are thresholds which belong to $(0, 1)$, a parameter tuning step is needed in order to give good classification results. For better understanding, an example is illustrated in Fig. 2. It was sampled by a panning 2D laser range finder and the data is stored in a 2D array as a range image. For details on the data gathering step, the reader is referred to [13]. In Fig. 2, $\alpha = 0.006$, $\beta = 0.4$.

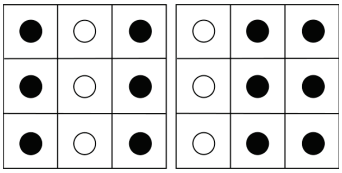
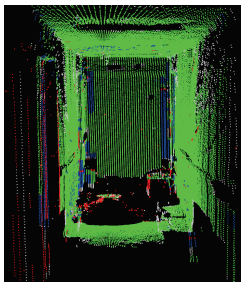
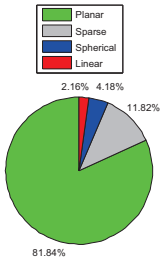


Fig. 1. Two possible configurations when there is one invalid line and six valid points in a 3×3 grid, regardless of the orientation.



(a) Grid classification result.



(b) Category percentages.

Fig. 2. Shape classification result of the grids and the corresponding percentage of each shape. For better visualization, pseudo-color is used for shape classification.

The Point Cloud Library (PCL) [17] is employed for visualization. In Fig. 2(a), different colors are assigned to the four classes: green for planar, blue for spherical, red for linear and light gray for sparse grids. The percentage of each category has also been given in Fig. 2(b). Similar results have been found for other scans, thus the first assumption is confirmed. The second assumption means that the local normal of a grid is a good estimation of the surface. In [9], the authors pointed out that the local surface normals from a planar surface in the real world are almost uniformly distributed. However, two orthogonal 2D laser scanners on a mobile robot were used for data collection. The horizontal one was employed to perform 2D SLAM to locate the robot. At the same time the upward pointing laser was scanning the 3D structure of the environment. Therefore, both localization error and measurement noisy exist in their 3D point cloud. The noise level should be higher than point cloud sampled by a rotating/panning laser range finder, which only contains measurement noise.

Considering the planar grids of Fig. 2(a), their unit normals can be seen in Fig. 3. Although some random normals are still present, it is apparent that it has five dense clusters according to the five plane directions in Fig. 2(a). This gives a positive answer to the second question. As a result, grid-based region growing is feasible for structured environments.

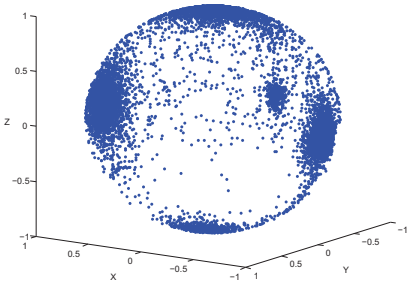


Fig. 3. Distribution of the unit normals for all planar grids in Fig. 2.

B. Grid-based Region Growing

The next step is to get big planes from separated planar grids. Given the planar grids set G from a point cloud data, the grid g which has the smallest MSE of all the unidentified grids is selected. This is the initial growing region GR (Algorithm 2, Line 4 – 5), which will be extended by investigating its neighbors. Suppose that g_c is the neighbor grid being considered. g_c is assigned to GR if it meets the following criteria (Algorithm 2, Line 8 – 9):

1) The dot product between the normal vectors of g_c and GR is greater than a predefined threshold η. Actually $\arccos(\vec{n}_{GR} \cdot \vec{n}_{g_c})$ is the angle between GR and g_c, so this criterion is to ensure that the investigated grid has similar direction to GR.
2) To avoid adding grid parallel but not coplanar to GR, the distance from the mass center of g_c to the optimal plane of GR is calculated. It should be less than γ.
3) The MSE to the optimal plane fitted to $GR \cup g_c$ should be less than ϵ.

Algorithm 2 Grid-based Region Identification

Input: G: planar grids with plane parameters
Output: R: regions constructed by coplanar grids

1: $R \leftarrow \emptyset$
2: $R' \leftarrow \emptyset$
3: **while** $(G \setminus (R \cup R') \neq \emptyset)$ **do**
4: *select g with minimal MSE in $G \setminus (R \cup R')$*
5: $GR \leftarrow g$
6: **while** *new neighbor of GR can be found* **do**
7: *select a neighbor g_c*
8: **if** $(\vec{n}_{GR} \cdot \vec{n}_{g_c} > \eta \;\&\&\; |\vec{n}_{GR} \cdot (m_{GR} - m_{g_c})| < \gamma \;\&\&\; MSE(GR \cup g_c) < \epsilon)$ **then**
9: $GR \leftarrow GR \cup g_c$
10: **end if**
11: **end while**
12: **if** $size(GR) \geq \theta$ **then**
13: $R \leftarrow R \cup GR$
14: **else**
15: $R' \leftarrow R \cup GR$
16: **end if**
17: **end while**

This process will be ended when no more neighbors can be added (Algorithm 2, Line 6 – 11). Since our goal is to extract big planes in the scene, only a GR with more than θ grids is regarded as a plane and added to the plane set R, otherwise it will be treated as uncertain region R' (Algorithm 2, Line 12 – 16). The algorithm ends when every grid is assigned to R or R'.

C. Mathematical Machinery for Parameter Computing

As mentioned, plane parameters for the growing region are incrementally computed whenever a new grid is added. When shape classification is finished, the following quantities have been tracked for each planar grid: the valid point number K, the matrix C similar to position covariance, and the mass center m. The items in (4) are also computed after classification.

$$\begin{cases} Q &= \sum_{i=1}^{K} p_i p_i^{\mathrm{T}} \\ \vec{n} &= eigvec_{min}(C) \\ d &= \vec{n} \cdot m \\ MSE &= \lambda_{min}(C) \end{cases} \tag{4}$$

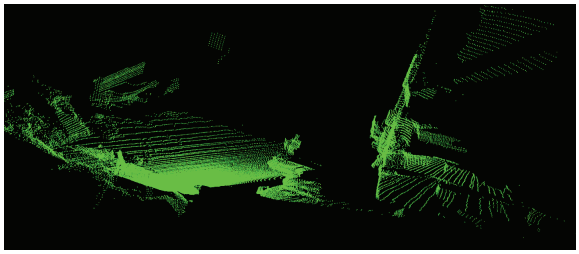
where Q is the product of K and the second order moment about the origin, $eigvec_{min}(C)$ stands for the eigenvector according to the minimal eigenvalue of C. Suppose the growing region is GR, there are K_{GR} points in it. The grid to be added is NG, and there are K_{NG} points in it. In (5), GR and NG are used as subscripts to denote the growing region and the grid to be added respectively. There is no subscript for the combined region. Note that GR may just contain one grid, i.e., at the beginning phase of a new region.

$$\begin{cases} S &= m_{GR}K_{GR} + m_{NG}K_{NG} \\ m &= S/(K_{GR} + K_{NG}) \\ Q &= Q_{GR} + Q_{NG} \\ C &= \sum_{i=1}^{K_{GR}+K_{NG}} (p_i - m)(p_i - m)^{\mathrm{T}} \\ \vec{n} &= eigvec_{min}(C) \\ d &= \vec{n} \cdot m \\ MSE &= \lambda_{min}(C)/(K_{GR} + K_{NG}) \end{cases} \tag{5}$$

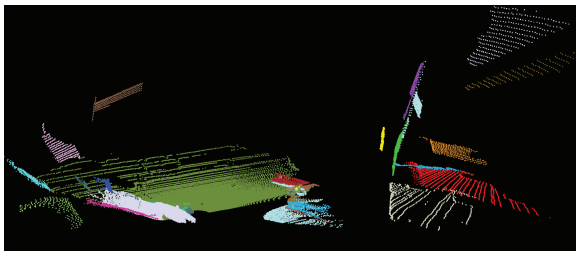
After some algebra, matrix C in (5) can be computed as follows:

$$C = Q - Sm^{\mathrm{T}} \tag{6}$$

Equations (5) and (6) yield a fast incremental algorithm. Suppose each point cloud contains n points, and the grid size is k, which means there are $m = \lfloor n/k \rfloor$ grids. For each grid, shape parameter computation and classification need constant time, which means the time complexity is $O(m)$. In addition, the neighbor search executes with a time complexity of $O(m \log m)$, for all the grids belonging to one region. Computing the plane parameter when a new grid is added also



(a) Point cloud data before plane detection.



(b) Plane detection result.

Fig. 4. Point cloud and the correspondence planar patches resulted by the point-based region growing algorithm, the detected planes have been colored randomly. The point cloud data is from [6].

needs constant time with time complexity of $O(m)$. To sum up, the overall time complexity of the algorithm is $O(m \log m)$. For each grid and region, at most seven variables are tracked which yields the memory complexity $O(m)$.

IV. Experiments and Results

Currently our panning indoor/outdoor laser rang finder platform is under construction, which includes an indoor/outdoor pan-tilt-unit (PTU) and an indoor/outdoor 2D laser scanner. The PTU can perform 360-continuous pan thanks to an integrated slip ring. Since it is still under construction, other data-sets were used in this paper. Both algorithms were implemented in C++ and run on an Intel Core 2 Duo 2.53GHz, 2GB RAM under Ubuntu 10.04. For both algorithms, the efficiency of linear algebra is crucial. This is especially so for calculating the eigenvalues and eigenvectors of a matrix, for it has to be performed whenever one point (grid) is investigated. Therefore the Eigen library [18] which is a C++ template library for linear algebra was employed.

A. Unstructured Environment

The on-line data-set "Collapsed Car Parking Lot" [6] was used as an unstructured environment. There are 195301 points in each scan. In order to evaluate the performance of our point-based algorithm, it was tested on all the 26 scans. One typical result and its corresponding point cloud are illustrated in Fig. 4. The resulted planes have been randomly colored, so one color could have been patched to more than one plane. The grid-based region growing algorithm was also applied to this data-set, the relatively big planes could be extracted while some of the relatively small ones were lost in the result. The reason is that grids close to the surface edges will be probably classified as sparse or spherical when decomposing the range image.

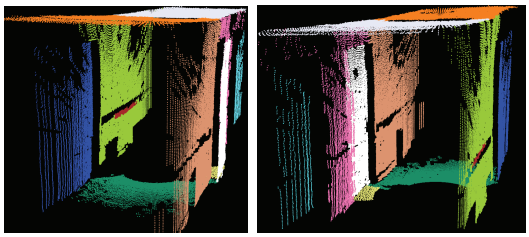


Fig. 5. Two different view angles of one typical plane detection result using the grid-base region growing algorithm, the data is from [13].

B. Structured Environment

The data-set from [13] was used as a structured environment. There are 149577 points in each scan. Both the point-based and grid-based algorithm were applied to this data-set. Except speed, the only difference is that under the point-based algorithm, there are some points which belong to more than one patches, i.e. they lie on the joint area between surfaces. However, this has no effect on the resulting plane parameters. From the speed point of view, the grid-based algorithm is much better, it is four times faster than the point-based algorithm. One typical result from the grid-based algorithm with two different view angles is illustrated in Fig. 5, the colors were also randomly selected for the planes.

C. Discussion

A box plot of execution times can be seen in Fig. 6. The mean time of the point-based algorithm is 0.7666 seconds when applied to the unstructured data-set, and 0.6911 seconds when applied to the structured data-set. They are close due to the similar valid point number in each point cloud. It implies the algorithm computation complexity to be linear. The grid-based algorithm is four times faster with a mean time of 0.1555s second when applied to the structured environment. The exact processing time depending on the number of regions extracted has been drawn in Fig. 7. Note the linear relation between the processing time and the number of detected regions. The slope of the point-based algorithm is different when applied to the two data-sets. It is caused by a different average point number in one region.

Both algorithms work well on a structured environment while the grid-based algorithm makes a faster execution. Using the grid-based algorithm, the total execution time is below 170 milliseconds (see Fig. 6). It demonstrates that a 6 Hz update rate can be reached even when dealing with point clouds including 1.5×10^5 points. It is quite promising for the SLAM system which requires planar features in structured environments, thus has attracted application areas such as home service robots and industrial robots.

For unstructured environments, the point-based region growing algorithm is preferred since some information will be

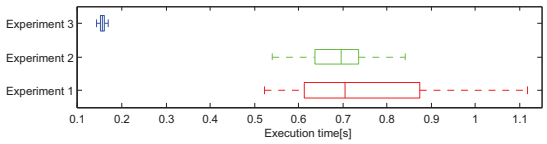


Fig. 6. Total plane detection time span of the three experiments, shown using box whisker plots. Whiskers encompass the full range of time, box bounds show upper and lower quartiles. Experiment 1: point-based region growing algorithm applied to unstructured data-set; Experiment 2: point-based region growing algorithm applied to structured data-set; Experiment 3: grid-based region growing algorithm applied to structured data-set.

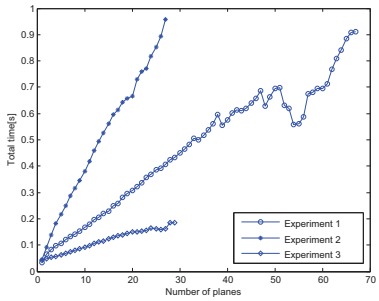


Fig. 7. Processing time depends on the number of extracted regions of the three experiments. Experiment i ($i \in \{1, 2, 3\}$) is same as that in Fig. 6.

lost using the grid-based method. It is faster compared to the state-of-the-art algorithm [12]. It is not solid to claim that our algorithm is three times faster than theirs, since the our computing-hardware is different from theirs. There are two factors which make it faster: First, our derivation saves computation cost for computing the plane fitting MSE. Second, only qualified seeds seeds are grown (Algorithm 1, line 5) thus blind growths of points not on a planar surface are avoided. Note that blind growth can be quite expensive, for example, when the starting point is located in the dense part but not on a surface of the point cloud.

V. Conclusion and Future Work

An improved point-based region growing algorithm for unstructured environments and a novel grid-based region growing algorithm for structured environments have been presented. For both algorithms, the neighborhood information is used when the point cloud is viewed as a range image. For the point-based algorithm, qualified seed points are detected firstly using local shape information and extend them to big patches. The local shape information is also employed in our novel grid-based algorithm, where the range image is broken down to small grids with fixed size. The grids in which the points are approximately coplanar will be used in the later steps. Similar to the point-based algorithm, a seed grid is searched first and then extended using our incremental version for computing the optimal plane parameters.

In the future, the grid-based algorithm can be extended in order to be employed under unstructured environments. It will make sense for real time 6D SLAM in unstructured environment.

Acknowledgment

Many thanks to Kaushik *et al.* [13] for kindly providing us the point cloud data they acquired in structured environment.

References

[1] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying", in *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, pp. 2878–2883, 2009.

[2] J. Weingarten and R. Siegwart, "EKF-based 3D SLAM for structured environment reconstruction", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canada, pp. 2–6, 2005.

[3] P. Kohlhepp, P. Pozzo, M. Walther, and R. Dillmann, "Sequential 3D SLAM for mobile action planning", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, pp. 722–729, 2004.

[4] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D point cloud based object maps for household environments", *Robotics and Autonomous Systems*, Special Issue on Semantic Knowledge, vol. 56, no. 11, pp. 927–941, 2008.

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

[6] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online 3D SLAM by registration of large planar surface segments and closed form pose-graph relaxation", *Journal of Field Robotics (JFR)*, Special Issue on 3D Mapping, vol.27, no. 1, pp.52–84, 2010.

[7] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM for 3D mapping outdoor environments", *Journal of Field Robotics (JFR)*, Special Issue on Quantitative Performance Evaluation of Robotic and Intelligent Systems, vol. 24, no. 8/9, pp. 699–722, 2007.

[8] M. Magnusson, T. Duckett and A. J. Lilienthal, "Scan registration for autonomous mining vehicles using 3D-NDT", *Journal of Field Robotics (JFR)*, Special Issue on Mining Robotics, vol. 24, no. 10, pp. 803–827, 2007.

[9] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3D models of indoor and outdoor environments with a mobile robot", *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[10] J. Weingarten, G. Gruener, and R. Siegwart, "A fast and robust 3d feature extraction algorithm for structured environment reconstruction", in *International Conference on Advanced Robotics (ICAR)*, Coimbra, Portugal, pp. 390–397, 2003.

[11] G. M. Hegde and C. Ye, "Extraction of planar features from swissranger sr-3000 range images by a clustering method using normalized cuts", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Louis, USA, pp. 4034–4039, 2009.

[12] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, pp. 3378–3383, 2008.

[13] R. Kaushik, J. Xiao, S. Joseph and W. Morris, "Fast planar clustering and polygon extraction from noisy range images acquired in indoor environments", in *IEEE International Conference on Mechatronics and Automation (ICMA)*, Xi'an, China, pp. 483–488, 2010.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] M. Magnusson, H. Andreasson, A. Nüchter and A. Lilienthal, "Automatic appearance-based loop detection from 3D laser data using the normal distributions transform", *Journal of Field Robotics*, vol 26, no. 11-12, pp. 892–914, 2009.

[16] M. Bosse and R. Zlot, "Continuous 3D scan-matching with a 2D spinning laser", in *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, pp. 4312–4319, 2009.

[17] R. B. Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)", in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[18] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, homepage: http://eigen.tuxfamily.org, 2010.