Robotics and Autonomous Systems 61 (2013) 1641–1652

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Three-dimensional point cloud plane segmentation in both structured and unstructured environments

Junhao Xiao [a,*], Jianhua Zhang [b], Benjamin Adler [a], Houxiang Zhang [c,*], Jianwei Zhang [a]

[a] Department of Computer Science, University of Hamburg, Hamburg, Germany
[b] College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China
[c] Faculty of Maritime Technology and Operations, Ålesund University College, Ålesund, Norway

HIGHLIGHTS

- Fast 3D point cloud plane segmentation.
- Subwindow-based region growing.
- Hybrid region growing.
- Custom-built 3D laser range finder.
- Plane-based scan registration.

ARTICLE INFO

Article history:
Received 29 August 2012
Received in revised form
24 June 2013
Accepted 1 July 2013
Available online 6 July 2013

Keywords:
3D point cloud
Plane segmentation
Region growing

ABSTRACT

This paper focuses on three-dimensional (3D) point cloud plane segmentation. Two complementary strategies are proposed for different environments, i.e., a subwindow-based region growing (SBRG) algorithm for structured environments, and a hybrid region growing (HRG) algorithm for unstructured environments. The point cloud is decomposed into subwindows first, using the points' neighborhood information when they are scanned by the laser range finder (LRF). Then, the subwindows are classified as planar or nonplanar based on their shape. Afterwards, only planar subwindows are employed in the former algorithm, whereas both kinds of subwindows are used in the latter. In the growing phase, planar subwindows are investigated directly (in both algorithms), while each point in nonplanar subwindows is investigated separately (only in HRG). During region growing, plane parameters are computed incrementally when a subwindow or a point is added to the growing region. This incremental methodology makes the plane segmentation fast. The algorithms have been evaluated using real-world datasets from both structured and unstructured environments. Furthermore, they have been benchmarked against a state-of-the-art point-based region growing (PBRG) algorithm with regard to segmentation speed. According to the results, SBRG is 4 and 9 times faster than PBRG when the subwindow size is set to 3×3 and 4×4 respectively; HRG is 4 times faster than PBRG when the subwindow size is set to 4×4. Open-source code for this paper is available at https://github.com/junhaoxiao/TAMS-Planar-Surface-Based-Perception.git.

1. Introduction

Range sensors, e.g., Laser Range Finders (LRFs), Time-of-Flight (ToF) cameras, stereo vision, active vision [1] and the newly developed RGB-D style cameras 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 [2,3], simultaneous localization and mapping (SLAM) [4–7], semantic mapping [8,9] and object recognition [10,11]. Objects with planar surfaces are prevalent in both indoor and urban environments, such as floors, doors, walls, ceilings and roads. If the surfaces are extracted as polygons, they can provide a compressive representation of the point clouds; normally the data compression rate is higher than 90% [12,13]. Furthermore, planar patch has been found to be a good geometric feature for scan registration since three planes with linearly independent normals can determine the transformation between overlapping point clouds. The plane-based registration algorithm MUMC in [14] has been proven to be faster and more reliable than the classic iterative corresponding point (ICP) [15,16] algorithm and the recently proposed normal distribution transformations (NDT) algorithm [17,18]. In our previous work [19,20], a novel plane-based registration algorithm

* Corresponding authors.
E-mail addresses: xiao@informatik.uni-hamburg.de (J. Xiao), zjh@zjut.edu.cn (J. Zhang), adler@informatik.uni-hamburg.de (B. Adler), hozh@hials.no (H. Zhang), zhang@informatik.uni-hamburg.de (J. Zhang).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.robot.2013.07.001

Fig. 1. The custom-built 3D laser range finder which is installed on a Pioneer 3-AT robot. The laser range finder is connected to the on-board computer through the inside slip-ring, which enables 360° continuous pan motion without making the cables entanglement together. The UTM-30LX LRF is installed horizontally on the top bracket of D48E with its "Sensor Front" point upwards.

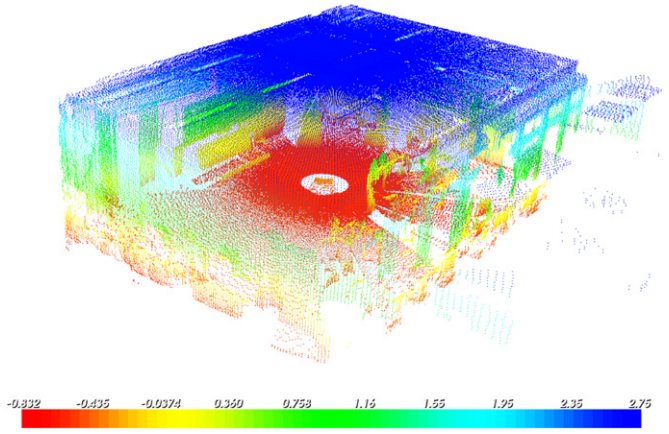


Fig. 2. A typical point cloud gathered by our customized 3D laser range finder. The data was sampled in the authors' robot laboratory. The points are colored by height, and the color map is shown under the point cloud. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

has been proposed and applied to datasets obtained with different sensors from different scenarios. From field experiments, our algorithm has been found to be fast, accurate and robust; see [19,20] for details.

Plane segmentation – which has been researched for years and is still a very hot and complex task in robotics – is the first step of plane-based mapping systems. Although there are available algorithms in the graphics community [21,22], they cannot be adopted into robotic systems directly due to relying on more exact depth information than that which robotic sensors can provide. Therefore, various algorithms on this topic [12,23–34] to deal with noisy datasets have recently been proposed by researchers from the robotics community. In this paper, two complementary plane segmentation algorithms are presented for different environments, i.e., a subwindow-based region growing algorithm for structured environments, and a hybrid region growing algorithm for unstructured environments.

A light weight Pan-Tilt Unit (PTU) and an inexpensive two-dimensional (2D) LRF were integrated for three-dimensional (3D) point cloud gathering in our research, namely the FLIR® PTU-D48E and the Hokuyo® UTM-30LX; the setup is shown in Fig. 1. PTU-D48E is a high-performance real-time positioning system which offers high precision positioning and speed control. We only make use of its pan motion which rotates the LRF to make an actuated LRF (aLRF). It has 360°-continuous pan motion due to an inside slip-ring. The finest pan resolution is 0.006° and the highest pan speed is 100°/s. The 2D LRF was designed for both indoor and outdoor environments which has a 270° field of view in its sensing plane. It is rotated by 180° on the PTU to obtain a 3D scan with a field of view 360° × 135°. In this work, the pan resolution and the laser beam resolution have been set to 0.5° and 0.25° respectively. Therefore, the resulting point clouds have 541 × 720 points. A typical point cloud gathered indoor by the sensor is illustrated in Fig. 2. Besides point clouds from the custom-built scanner, we also make use of other publicly available datasets to analyze the performance of the proposed algorithms.

The work presented in this paper is partially based on previously published results [31], with the main additions being the novel hybrid region growing algorithm and more experimental data. The paper is laid out as follows. Related work on plane segmentation come in Section 2. In Section 3, we detail our plane segmentation approaches. Then the mathematical machinery for incremental plane parameter calculation whenever a subwindow or a single point is added in the region growing is given in Section 4. Afterward, we present the experiments and results in Section 5. Finally, the paper is summarized in Section 6, which also states our conclusions, as well as future research directions.

2. Related work

The Expectation Maximization (EM) algorithm is an iterative method for finding maximum likelihood estimates of model parameters. Lakaemper and Latecki [25] employed an extended EM algorithm to fit planar patches in 3D range data. The algorithm is called Split and Merge Expectation Maximization Patch Fitting (SMEMPF); it alternated the following E-steps and M-steps until they reach a convergence. The E-step is performed for a current given set of planes (an initialization set of plane parameters is needed at the beginning); the probabilities for each point of its correspondence to all planes are calculated based on its distance to the planes. Given the probabilities computed in the E-step, the new positions of the planes are estimated in the M-step. It works on arbitrary point clouds, but is not feasible for real time operation. This is due to the iterative nature of EM, including a costly plane-point correspondence check in its core.

Hough transform is a classic feature extraction method which has been used in image processing for the detection of lines or circles. In order to use it for plane detection in 3D point cloud segmentation, Borrmann et al. [30] evaluated different variants of the Hough Transform. It was found that the main problem is the representation of the accumulator besides computational costs. To deal with this, they proposed the accumulator ball as an accumulator design. The evaluation of different Hough methods recommended the Randomized Hough Transform for dealing with plane detection in 3D point clouds. However, it still has a severe disadvantage, i.e., the processing time increases with the number of planes but not the number of points in the point cloud. As it was reported in the paper, the segmentation time using the Randomized Hough Transform would be significant larger than region growing when more than 15 planes presented in the data. Similarly, Dube and Zell [35] also proposed to use the Randomized Hough Transform for plane detection from depth images. They concentrated on the Microsoft Kinect cameras and made use of the sensor noise model to find proper parameter metrics for the Randomized Hough Transform. They evaluated the influence of local sampling and found local sampling made the result better. However, their test environment was a clean corridor with only walls, windows, roof and ground which means about 4 or even fewer planes presented in

each frame. Furthermore, the high detect rate of the walls is 82.2% which is not satisfied for detail map generation.

The RANdom SAmple Consensus (RANSAC) [36] is another route for estimating parameters of a model from a dataset which may have outliers. When being applied to 3D point cloud plane segmentation, dealing with multiple models in one dataset should be considered. In [24], the cloud is decomposed into equal sized 3D cubes, and then one model is fitted to each cube using RANSAC, thus avoiding the multiple model problems. Afterwards, small planar patches are merged using the cube neighborhood information. However, the algorithm is still time consuming due to the iterative nature of RANSAC. Trevor et al. [34] utilized the RANSAC algorithm for plane segmentation in another way. In their work, the plane with the most inliers in the dataset is searched each time, then all inliers for this plane are removed from the point cloud; the RANSAC is performed again to find the next largest plane. This process terminates when no plane with a sufficient number of points could be found. However, the segmentation time has not been reported in these two papers.

The concept of organized point cloud (also known as range image or structured point cloud) should be mentioned before introducing the region growing approach. An organized point cloud resembles an organized image-like structure, where the data is split into rows and columns. Examples of such point clouds include data delivered by stereo- and ToF-cameras. The advantage of such a point cloud is that the relationship between adjacent points (like pixels in an image) is known, making nearest neighbor operations much more time-efficient. Some aLRFs can also produce organized point cloud datasets, such as ours in Fig. 1. The adjacency information is denoted as *pixel-neighborhood information* later on in this paper. Note that the nearest neighbor search is an important issue for region growing, since it has to be performed at each step of the growth.

Region growing was proposed for image segmentation based on color information between neighbor pixels. It was extended to plane segmentation in [23]. In their system, planar segments were employed for smoothing the resulted map, but not as features in the mapping phase, which means the segmentation speed was not a critical point. To embed the plane segmentation into plane-based on-line mapping systems, Poppinga et al. [27] sped it up with the following two improvements: the first step is to use pixel-neighborhood information for nearest neighbor search and the second step is an incremental version for plane parameters computation. It is further accelerated without losing any precision via an efficient plane fitting error computation in our previous work; see [31] for details. For unorganized point cloud, we have proposed a cached octree region growing algorithm in [20], which makes a compromise between time and memory. For the input point cloud, an octree is built first and the indices of nearest neighbors for each point is cached for the region growing phase. As a result, the algorithm provides an efficient plane segmentation solution for unorganized point clouds; see [20] for details.

It is noted that the growth unit is a single point in the above region growing approaches. As alternatives, the growth unit can also be a line segment or a subwindow. In Harati et al. [26], the so-called bearing angle is computed for each point as a measure of the flatness of its local area, using the pixel-neighborhood information. Based on this measure, a line-based region growing algorithm is proposed. However, since the bearing angle is the incident angle between the laser beam and edges of the scanned polygon in the selected direction, it cannot be properly calculated in cluttered environments. Georgiev et al. [32] started by extracting 2D line segments from each 2D scan slice (each row or column in an organized point cloud), where connected line segments represent candidate sets of coplanar segments. Then, a region growing algorithm is utilized to find coplanar segments and their least squares fitting (infinite) plane.

In Kaushik et al. [29], the point cloud is divided into subwindows (named patches in their paper), and plane parameters are computed for each subwindow. The resulting subwindows are clustered into large surfaces by a Breadth-First search algorithm. One drawback of this approach is that there are some subwindows whose appearance could not be approximated by a plane. Then it was extended and published in Kaushik and Xiao [12], where the planar patches and nonplanar patches are distinguished. However, the plane parameters are not updated when new data points are added; instead, the plane parameters of the selected seed patch are treated as the plane model. Therefore the resulting plane parameter – a fundamental issue for plane-based registration and SLAM – would be inaccurate. To deal with this problem, in this paper we present an incremental version for plane parameter calculation whenever a new subwindow (i.e., a patch in [12]) is added to the growing region.

3. 3D point cloud plane segmentation

The proposed plane segmentation approaches are described in detail in this section. The first part is named the subwindow-based region growing (SBRG) algorithm, since one subwindow is added to the region at each step during its growth. The second part is called the hybrid region growing (HRG) algorithm as there are two kinds of growth unit, i.e., a single point or a subwindow. The approaches are proposed for organized point clouds. For a detailed discussion on different point cloud formats, please refer to the point cloud library [37] which is under rapidly developing. The feasibility of utilizing subwindows is evaluated first since it is necessary in both algorithms, followed by detailing the two algorithms.

In this paper, the following notations are used:

x or X, $\mathbf{x}$, $\hat{\mathbf{x}}$	a scaler, a vector, an unit vector
$\mathbf{x} \cdot \mathbf{y}$	vector dot product
$\mathbf{X}$	a matrix
$\mathcal{X}$	a set

3.1. Feasibility of using subwindows in plane segmentation

Subwindow is suitable for plane segmentation if the following two assumptions are tenable. First, most subwindows which locate on planar surfaces have a planar appearance. Second, subwindows from the same physical surface have similar plane parameters. To confirm the first assumption, the point cloud is decomposed into small subwindows first, and then the subwindows are classified into two categories based on their shape appearances, namely planar or nonplanar.

To determine the appearance of a subwindow ω which contains valid points $\mathbf{p}_0, \mathbf{p}_1, \ldots \mathbf{p}_{sw}$, the scatter matrix $\mathbf{C}$ of the points is computed as in Eq. (3) (see Section 4), where $\mathbf{m}$ is the geometric center. Note that invalid points may also appear in ω, for example when there is no object in certain laser beam directions. Clearly, $\mathbf{C}$ is a positive-definite matrix. In other words, $\mathbf{C}$ has three positive eigenvalues. Given its sorted eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$, the shape of ω is decided by the below criteria:

$$\omega \in \begin{cases} \text{sparse}, & \text{if sw} < \mu \text{size}(\omega), \\ \text{planar}, & \text{if } \lambda_1 \leq \eta\lambda_2, \\ \text{nonplanar}, & \text{otherwise}, \end{cases} \tag{1}$$

where $\mu, \eta \in (0, 1)$, and size(ω) is the total number of valid and invalid points. The subwindow is marked as sparse when the number of valid points is smaller than a given threshold. A parameter tuning step is needed for η in order to yield satisfactory classification results. From the experiments, it is only related to the employed range sensor, i.e., η is a fixed value for each sensor.

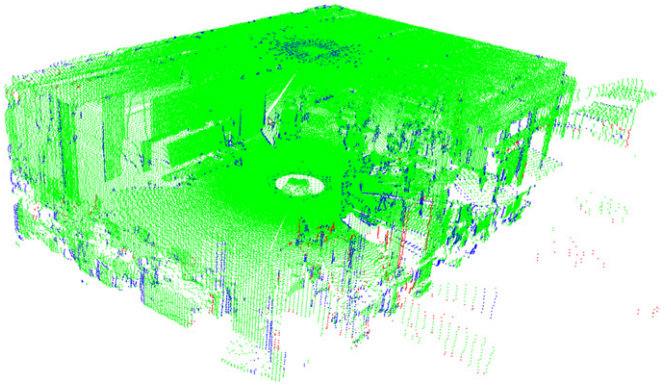


Fig. 3. The subwindow classification result of the point cloud shown in Fig. 2. The size of subwindow is 3×3. Sparse, planar and nonplanar subwindows are colored in red, green and blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

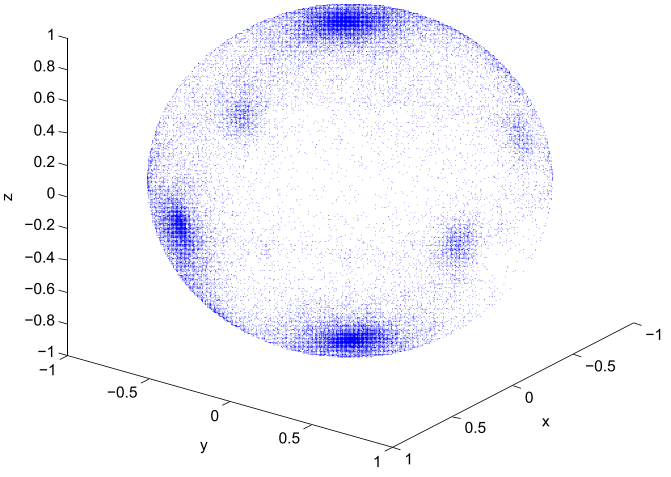


Fig. 4. Distribution of unit normals for all planar subwindows in Fig. 3. Note that it does not correspond to the view point of Fig. 3. Although normals are present almost everywhere on the sphere, it is still easy to find six dense clusters corresponding to the six big planar surfaces in the point cloud.

In this paper, the subwindow is set to be square, with size larger than 2×2, since 4 points are not adequate for shape analysis. For better understanding, the subwindow classification result of Fig. 2 is illustrated in Fig. 3. The size of the subwindow is set to 3×3; μ and η are set to 0.7 and 0.3 respectively. The point cloud library is utilized for visualization, where sparse, planar and nonplanar subwindows are colored as light gray, green and blue. It is apparent that most subwindows which were scanned from planar surfaces have planar appearance. Similar results have been found for other scans; thus the first assumption is confirmed.

Now we deal with the second assumption; it means that the normal of a subwindow is a good estimation of the surface. Hähnel et al. pointed out that local surface normals from a planar surface in the real world are almost uniformly distributed in [23]. However, two orthogonal 2D laser scanners on a mobile robot were used for data collection in their robotic system. A horizontal laser was employed to perform 2D SLAM to localize the robot. At the same time, a vertical upward pointing laser scanned the 3D structure of the environment. Therefore, both localization error and measurement noise exist in their 3D point clouds. The noise level should be higher than that of a point cloud sampled with the so-called stop–scan–go (also known as stop-and-scan) style by an aLRF, which only contains measurement noise. Considering the planar subwindows of Fig. 3, their unit normals have been visualized in Fig. 4. Although some random normals are still present, several dense clusters are apparent according to the planar surfaces in Fig. 3; thus the second assumption has been verified. Therefore, it can be concluded that subwindow can be used in plane segmentation.

Algorithm 1: Subwindow-based region growing

Input: Ψ: an organized point cloud
Output: $\mathcal{R}$: planar segments, $\mathcal{R}'$: uncertain points

1 $\mathcal{R} \leftarrow \emptyset, \mathcal{R}' \leftarrow \emptyset, \mathcal{G} \leftarrow \emptyset, \mathcal{Q} \leftarrow \emptyset, \Omega \leftarrow \emptyset$;
2 Ω=planarSubwindows(Ψ) ;
3 **while** $\Omega \setminus (\mathcal{R} \cup \mathcal{R}') \neq \emptyset$ **do**
4 select ω with minimum e in $\Omega \setminus (\mathcal{R} \cup \mathcal{R}')$;
5 $\mathcal{G} \leftarrow \omega, \mathcal{Q} \leftarrow$NN($\omega$) ;
6 **while** $\mathcal{Q} \neq \emptyset$ **do**
7 $\omega_c = \mathcal{Q}$.pop() ;
8 **if** $|\hat{\mathbf{n}}_G \cdot (\mathbf{m}_G - \mathbf{m}_{\omega_c})| < \gamma$ && $\hat{\mathbf{n}}_G \cdot \hat{\mathbf{n}}_{\omega_c} > \delta$ && mse($\mathcal{G} \cup \omega_c$) $< \epsilon$ **then**
9 $\mathcal{G} \leftarrow \mathcal{G} \cup \omega_c$;
10 $\mathcal{Q} \leftarrow \mathcal{Q} \cup$ NN(ω_c) ;
11 **end**
12 **end**
13 **if** size($\mathcal{G}$) $\geq \theta$ **then**
14 $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{G}$;
15 **else**
16 $\mathcal{R}' \leftarrow \mathcal{R}' \cup \mathcal{G}$;
17 **end**
18 **end**

3.2. Subwindow-based region growing (SBRG)

The proposed SBRG algorithm proceeds as follows. For an input organized point cloud Ψ, it is decomposed into subwindows first using the image-like structure. The subwindows are classified as planar and nonplanar based on the method presented in Section 3.1, and then only the planar subwindows will be kept for plane segmentation (Algorithm 1, line 2). At the same time, local plane parameters for each planar subwindow are computed as well as the mean square error (MSE); MSE is denoted by e in this paper. Afterwards, the subwindow ω with the minimum MSE among all unidentified subwindows is chosen as a new seed (Algorithm 1, line 4). A growing region $\mathcal{G}$ is initialized by ω, and its unidentified neighbors are put into a First-In–Last-Out (FILO) queue $\mathcal{Q}$ which keeps $\mathcal{G}$'s nearest neighbors (Algorithm 1, line 5). Then, $\mathcal{G}$ is extended by investigating its neighbors in $\mathcal{Q}$. Suppose that ω_c is the neighbor subwindow being considered; it is assigned to $\mathcal{G}$ iff it meets the following criteria (Algorithm 1, line 8–11).

1. The dot product between the normal vectors of ω_c and $\mathcal{G}$ is greater than δ. Actually $\arccos(\hat{\mathbf{n}}_G \cdot \hat{\mathbf{n}}_{\omega_c})$ is the angle between $\mathcal{G}$ and ω_c, so this criterion is to ensure that the investigated subwindow has a similar surface normal direction to $\mathcal{G}$.
2. To avoid adding a subwindow which is parallel but not coplanar to $\mathcal{G}$, the distance from the mass center of ω_c to the optimal plane of $\mathcal{G}$ should be less than γ.
3. To guarantee an acceptable flatness of the resulting segment, the plane fitting error e of $\mathcal{G} \cup \omega_c$ should be less than ϵ.

This process terminates when no more neighbors can be added to $\mathcal{G}$, i.e., when $\mathcal{Q}$ is empty (Algorithm 1, line 6–12). Since our goal is to extract large planes in the scene, only a $\mathcal{G}$ with more than θ subwindows is regarded as a planar segment and added to the plane set $\mathcal{R}$; otherwise it is added to the uncertain region set $\mathcal{R}'$ (Algorithm 1, line 13–17). The algorithm ends when every planar subwindow is assigned either to $\mathcal{R}$ or $\mathcal{R}'$. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are pre-set thresholds which need to be tuned.

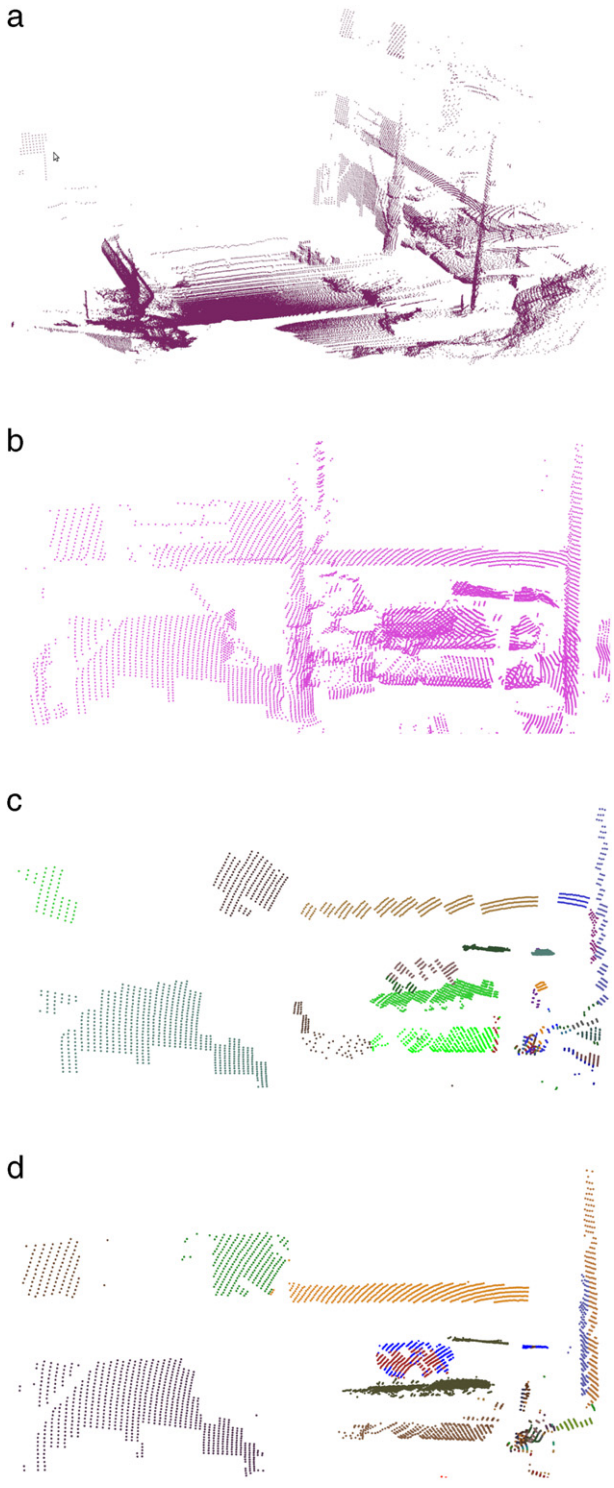


Fig. 5. A segmentation quality comparison between the Subwindow-based and hybrid region growing when applied to unstructured environments. (a) One scan from the on-line dataset Collapsed Car Parking Lot. (b) A close-up view of one part in (a) which has abundant planar surfaces. (c) and (d) Close-up views of the plane segmentation results.

3.3. Hybrid region growing (HRG)

The SBRG algorithm has good performance in structured environments. However, the result is unsatisfactory when applied to unstructured environments—see Fig. 5 for an example. In such cases, the segmentation result from pure subwindow-based region growing is not satisfied since the surface edges may be classified as nonplanar or sparse. This also happens when there are trees in front of a wall, as part of the wall will be occluded by the leaves and branches, making the corresponding subwindow nonplanar. The HRG algorithm is proposed to cope with this problem by considering all subwindows in the region growing process.

Algorithm 2: Hybrid region growing

Input: Ψ: an organized point cloud
Output: $\mathcal{R}$: planar segments, $\mathcal{R}'$: uncertain points

1 $\mathcal{R} \leftarrow \emptyset, \mathcal{R}' \leftarrow \emptyset, \mathcal{G} \leftarrow \emptyset, \mathcal{Q} \leftarrow \emptyset, \Omega \leftarrow \emptyset, \Theta \leftarrow \emptyset$;
2 (Ω, Θ)=subwindows(Ψ) ;
3 **while** $\Omega \setminus (\mathcal{R} \cup \mathcal{R}') \neq \emptyset$ **do**
4 select ω with minimum e in $\Omega \setminus (\mathcal{R} \cup \mathcal{R}')$;
5 $\mathcal{G} \leftarrow \omega, \mathcal{Q} \leftarrow$NN($\omega$) ;
6 **while** $\mathcal{Q} \neq \emptyset$ **do**
7 $\omega = \mathcal{Q}$.pop() ;
8 **if** isPlanar(ω)==true **then**
9 **if** mse($\mathcal{G} \cup \omega_c$) $< \epsilon$ && $\hat{\mathbf{n}}_G \cdot \hat{\mathbf{n}}_{\omega_c} >$ δ && $|\hat{\mathbf{n}}_G \cdot (\mathbf{m}_G - \mathbf{m}_{\omega_c})| < \gamma$ **then**
10 $\mathcal{G} \leftarrow \mathcal{G} \cup \omega_c$;
11 $\mathcal{Q} \leftarrow \mathcal{Q} \cup$NN($\omega_c$) ;
12 **end**
13 **else**
14 **for** each point $\mathbf{p}_c$ in ω **do**
15 **if** mse($\mathcal{G} \cup \mathbf{p}_c$) $< \epsilon$ && $\hat{\mathbf{n}}_G \cdot (\mathbf{m}_G - \mathbf{p}_c) < \gamma$ **then**
16 $\mathcal{G} \leftarrow \mathcal{G} \cup \mathbf{p}_c$;
17 $\mathcal{Q} \leftarrow \mathcal{Q} \cup$NN($\omega_c$) ;
18 **end**
19 **end**
20 **end**
21 **end**
22 **if** size($\mathcal{G}$) $\geq \theta$ **then**
23 $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{G}$;
24 **else**
25 $\mathcal{R}' \leftarrow \mathcal{R}' \cup \mathcal{G}$;
26 **end**
27 **end**

The procedure used in the HRG algorithm is quite similar to that of the SBRG algorithm. For a point cloud Ψ, it is decomposed into subwindows first and the subwindows are classified into planar, nonplanar, and sparse based on the method presented in Section 3.1; the planar subwindows are put into a list Ω and the other subwindows are put into another list Θ (Algorithm 2, line 2). When there are still unidentified planar subwindows, the subwindow ω with the minimum MSE among all unidentified planar subwindows is chosen as a new seed (Algorithm 2, line 4). A growing region $\mathcal{G}$ is initialized by ω, and its unidentified neighbors are put into a FILO queue $\mathcal{Q}$ (Algorithm 2, line 5). Then $\mathcal{G}$ is extended by investigating its neighbors. If the considering subwindow is planar, the criteria for determining whether to add it into $\mathcal{G}$ are the same as those in SBRG (Algorithm 2, line 8–13). Otherwise, each point in the subwindow will be investigated separately, and a point $\mathbf{p}_c$ will be added to $\mathcal{G}$ iff it passes the following tests (Algorithm 2, line 15–18).

1. The distance from the point to the optimal plane fitted to $\mathcal{G}$ is smaller than γ, making sure that p_c is a coplanar point of $\mathcal{G}$.
2. The plane fitting error of $\mathcal{G} \cup \mathbf{p}_c$ should be less than ϵ; this ensures the flatness of the segment acceptable.

This process will continue until no new neighbor of $\mathcal{G}$ can be found. Afterward, if $\mathcal{G}$ has more points than a threshold θ, it will be viewed as a planar segment; otherwise, it will be marked as uncertain area (Algorithm 2, line 22–26). The algorithm ends when every point has been assigned to $\mathcal{R}$ or $\mathcal{R}'$. The aforementioned parameters $(\delta, \gamma, \epsilon, \theta)$ are pre-set thresholds. One parameter tuning step is needed for a specific range sensor.

4. Incremental plane parameter calculation

As depicted in both algorithms, a plane should be fitted to the growing region whenever a new subwindow or a single point is added; the time cost of it is critical for a plane-based mapping system. In order to make it fast, the consequent incremental version is proposed. Different forms of plane equations exist in the literature. A comparison of them can be found in [38]. The Hessian form equation is chosen to represent planes in this work, for the reason that it can be obtained straightforward when the surface normal vector and an arbitrary point on the plane are known. It is described as

$$\hat{\mathbf{n}} \cdot \mathbf{p} = d, \tag{2}$$

where $\hat{\mathbf{n}}$ is the unit normal vector of the plane, $\mathbf{p}$ is an arbitrary point on the plane and d is the distance from the origin to the plane. To make the equation of each plane unique, we choose $\hat{\mathbf{n}}$ in the direction which makes $d > 0$.

In order to determine the shape appearance of a subwindow ω, the scatter matrix of the valid points in it is computed as

$$\mathbf{C} = \frac{1}{K}\sum_{i=1}^{K}(\mathbf{p}_i - \mathbf{m})(\mathbf{p}_i - \mathbf{m})^{\mathrm{T}} \tag{3}$$

where K is the number of valid points and

$$\mathbf{m} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{p}_i \tag{4}$$

is the geometric center. Then the eigenvalues of $\mathbf{C}$ are computed through eigenvalue decomposition and the subwindow classification is carried out as in Section 3.1.

If a subwindow is classified as planar, a plane will be fitted to it using the least squares. According to the least square fitting, the geometric center $\mathbf{m}$ is on the optimal plane, and the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$ is the plane normal. As a result, d can easily be computed as

$$d = \hat{\mathbf{n}} \cdot \mathbf{m}. \tag{5}$$

When shape classification is finished, the following quantities are tracked for each planar subwindow besides $\mathbf{C}$, $\mathbf{m}$, $\hat{\mathbf{n}}$ and d:

$$\begin{cases} e = \dfrac{1}{K}\lambda_{\min}(\mathbf{C}) \\ \mathbf{J} = \displaystyle\sum_{i=1}^{K}\mathbf{p}_i\mathbf{p}_i^{\mathrm{T}} \end{cases} \tag{6}$$

where e is the plane fitting error, $\mathbf{J}$ is the second order moment about the origin, and $\lambda_{\min}(\mathbf{C})$ stands for the minimum eigenvalue of $\mathbf{C}$.

The above quantities are also tracked for the growing region; they are used to derive the new plane parameters together with that of the subwindow to be added. At the beginning of the region growing, i.e., when a new seed is selected, the quantities of the growing region are simply assigned to those of the seed. Then the quantities of the growing region is updated incrementally which is explained below. Suppose that there are K_G points in the current growing region; the subwindow ω has passed the coplanar tests and is going to be added, which has K_ω points. In Eq. (7), $\mathcal{G}$ and ω are used as subscripts to denote the growing region and the investigated subwindow respectively, while there is no subscript for the combined region. Note that $\mathcal{G}$ may contain just one subwindow, i.e., when only the seed subwindow is added. Eq. (7) is developed to calculate the plane parameters of the combined region using the above tracked quantities. Obviously, it can also be used for computing the plane parameters when merging two coplanar segments:

$$\begin{cases} \mathbf{s} = \mathbf{m}_G K_G + \mathbf{m}_\omega K_\omega \\ \mathbf{m} = \mathbf{s}/(K_G + K_\omega) \\ \mathbf{J} = \mathbf{J}_G + \mathbf{J}_\omega \\ \mathbf{C} = \dfrac{1}{K_G + K_\omega}\displaystyle\sum_{i=1}^{K_G+K_\omega}(\mathbf{p}_i - \mathbf{m})(\mathbf{p}_i - \mathbf{m})^{\mathrm{T}} \\ \hat{\mathbf{n}} = \hat{\mathbf{e}}_{\min}(\mathbf{C}) \\ d = \hat{\mathbf{n}} \cdot \mathbf{m} \\ e = \dfrac{\lambda_{\min}(\mathbf{C})}{K_G + K_\omega} \end{cases} \tag{7}$$

In Eq. (7), $\hat{\mathbf{e}}_{\min}(\mathbf{C})$ stands for the normalized eigenvector corresponding to $\lambda_{\min}(\mathbf{C})$. All the other equations need constant time except the computation of $\mathbf{C}$. The time complexity to find the eigenvalues of an $n \times n$ matrix is $O(n^3)$, so the eigenvalue decomposition of $\mathbf{C}$ is the most time consuming part when K_G is small at the starting stage of a growing region. Then K_G increases as the region grows, and the calculation of $\mathbf{C}$ becomes the most time consuming part when there are many points ($K_G > n^3$, $n = 3$) in the growing region. In order to make the algorithm fast, we calculate $\mathbf{C}$ using other tracked quantities. After some algebra, it can be simplified as

$$\mathbf{C} = \mathbf{J} - \mathbf{s}\mathbf{m}^{\mathrm{T}}. \tag{8}$$

Eqs. (7) and (8) yield an incremental version for computing the plane parameters in the SBRG algorithm. However it is not sufficient for the HRG algorithm, because the plane parameters should also be updated when adding a single point. Assume that the point $\mathbf{p}$ has passed the coplanar tests and is going to be added. The new plane parameters will be calculated as

$$\begin{cases} \mathbf{s} = \mathbf{m}_G K_G + \mathbf{p} \\ \mathbf{m} = \mathbf{s}/(K_G + 1) \\ \mathbf{J} = \mathbf{J}_G + \mathbf{p}\mathbf{p}^{\mathrm{T}} \\ \mathbf{C} = \mathbf{J} - \mathbf{s}\mathbf{m}^{\mathrm{T}} \\ \hat{\mathbf{n}} = \hat{\mathbf{e}}_{\min}(\mathbf{C}) \\ d = \hat{\mathbf{n}} \cdot \mathbf{m} \\ e = \dfrac{\lambda_{\min}(\mathbf{C})}{K_G + 1}. \end{cases} \tag{9}$$

It can be seen that Eq. (9) is a special case of Eq. (7), i.e., when $K_\omega = 1$. To conclude this section, the plane parameters are computed incrementally when a subwindow or a single point is added to the segment, which makes SBRG and HRG fast.

4.1. Computational complexity analysis

Suppose an organized point cloud from structured environment contains n points, which is to be segmented using the SBRG algorithm. The subwindow size is set to k, which means there are $m = \lfloor n/k \rfloor$ subwindows. Shape classification needs constant time for each subwindow with a time complexity $O(m)$. In addition, the neighbor search executes with a time complexity at most $O(m \log m)$, for all subwindows belonging to one segment. Computing the plane parameter when a new subwindow is added to the region also needs constant time with time complexity of $O(m)$. To sum up, the overall time complexity of the SBRG algorithm is $O(m \log m)$. For each subwindow and segment, at most seven variables are tracked which yields the memory complexity $O(m)$.

There are two growth units in HRG, namely a single point or a subwindow. As a result, the time complexity of HRG is between the point-based algorithm and the subwindow-based algorithm. Its time complexity is equal to SBRG when there are no nonplanar/sparse subwindows and is equal to point-based region growing

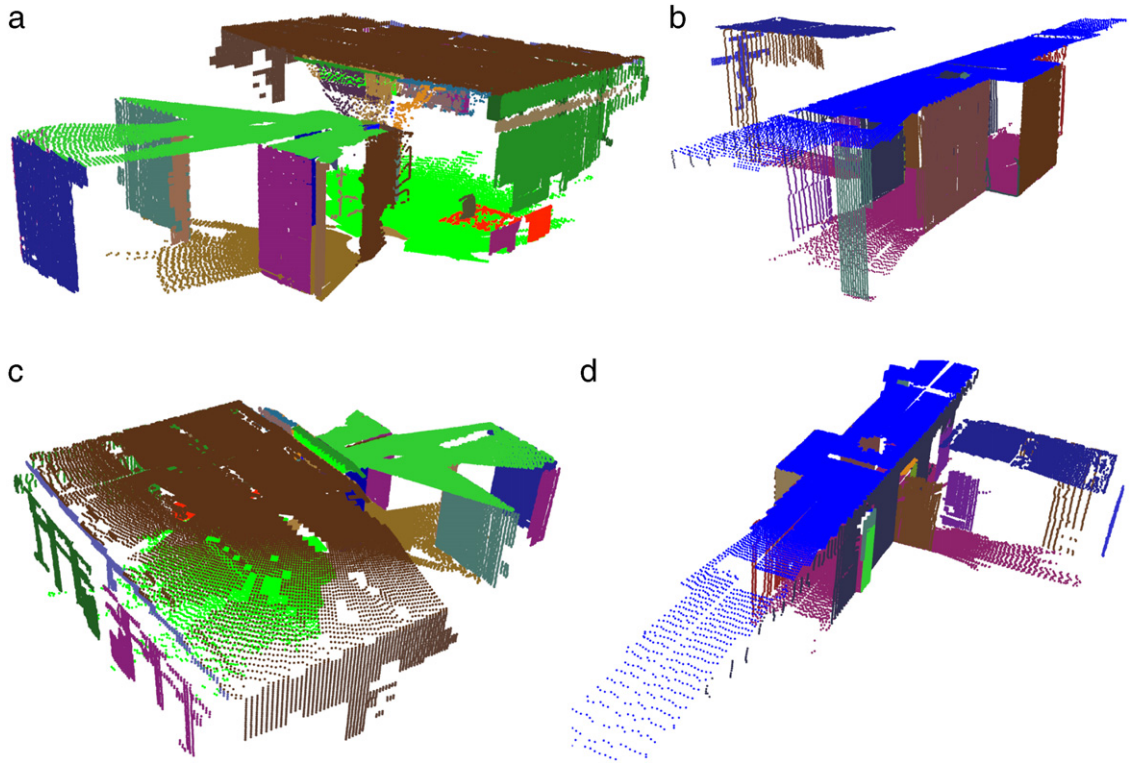


Fig. 6. Subwindow-based region growing plane segmentation for indoor structured environment, (a) and (c) are two different viewpoints from the segmentation result of one point cloud, while (b) and (d) are two different viewpoints for another point cloud.

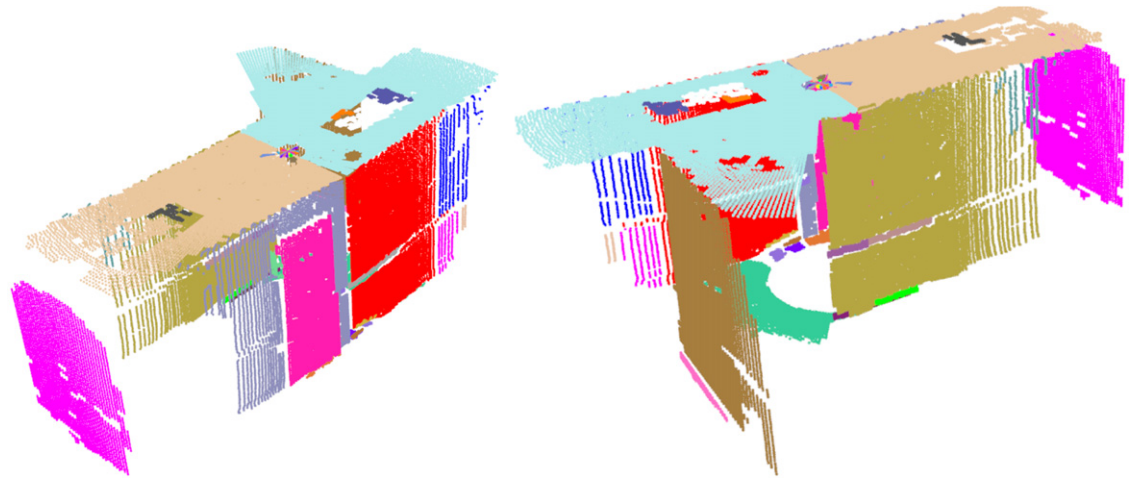


Fig. 7. One typical segmentation result using the subwindow-based region growing algorithm in the second indoor dataset. Two different viewpoints have been given.

when there are no planar subwindows in the given point cloud. For a point cloud with n points, the time complexity of SBRG is $O(\frac{n}{k}\log\frac{n}{k})$, where k is the subwindow size. The time complexity of point-based region growing is $O(n\log n)$ as reported in [27]. Consequently, the time complexity of HRG is in the range $[O(\frac{n}{k}\log\frac{n}{k}), O(n\log n)]$; it increases with the clutter level of the environment and vice versa.

5. Experiments and results

Both the proposed algorithms have been implemented in C++; the code has been published online and can be accessed from the following URL: https://github.com/junhaoxiao/TAMS-Planar-Surface-Based-Perception.git. All experiments were carried out on a standard desktop computer under Ubuntu 12.04. The efficiency of linear algebra is crucial in the approaches, especially for calculating the eigenvalues and eigenvectors of a square matrix, as it has to be performed whenever a point or a subwindow is investigated during the region growing phase. Therefore Eigen [39], a C++ template library for linear algebra, has been employed. The point cloud library has been utilized for point cloud reading and writing, as well as 3D visualization. For figures illustrating segmentation results, the segments have been colored randomly; therefore, one color may be patched to multiple segments. Due to space limitation, we cannot present all the segmentation results. Instead, some typical results have been selected for explanation.

As mentioned in Sections 3.2 and 3.3, parameter tuning is necessary for both algorithms. Experimental tuning was performed offline in this work. Obviously, the parameters depend on the measurement model of the 3D scanner. Actually the parameters can

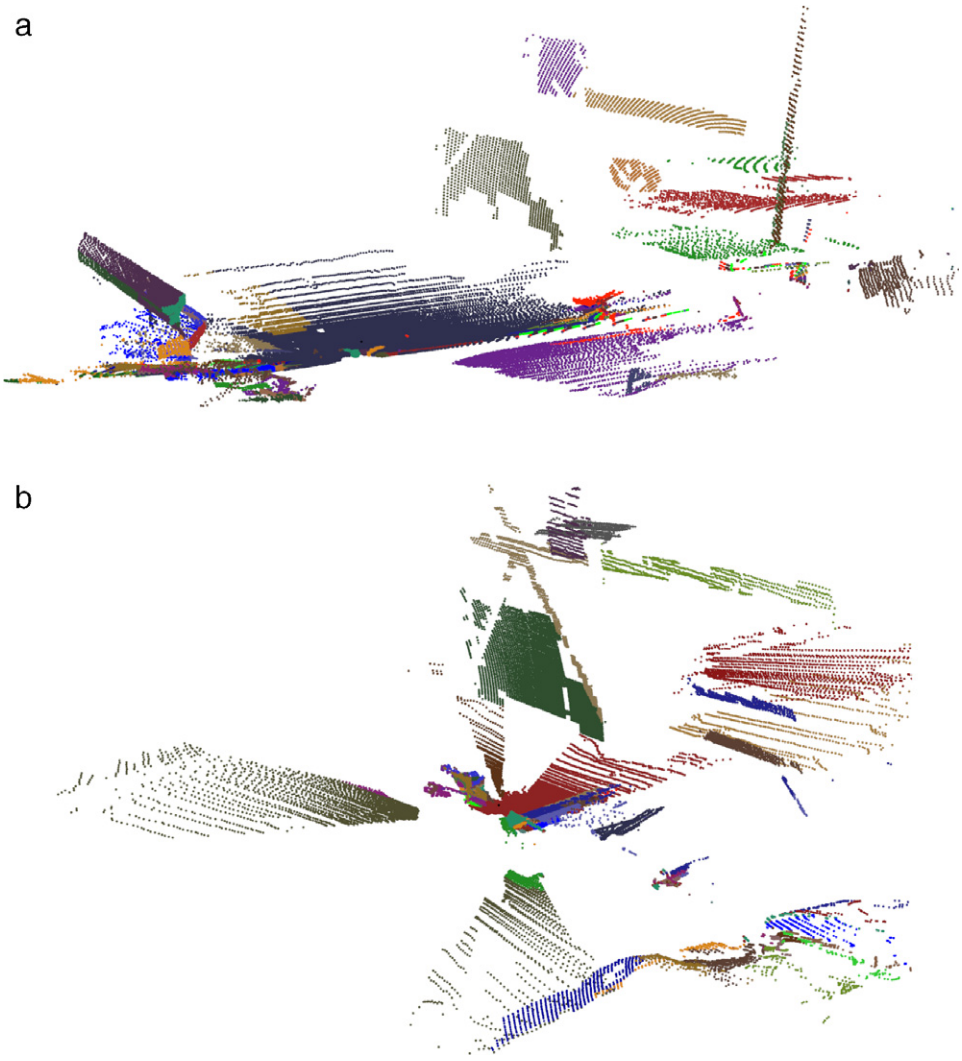


Fig. 8. Two typical segmentation results using the HRG algorithm for the Collapsed Car Parking Lot dataset.

be intuitively selected—the thresholds can be chosen by manually identifying the planar surfaces in a given scan.

Furthermore, Kaustubh Pathak has kindly provided us the access to their code used in [27]. Their algorithm serves as a baseline for comparison and is denoted as PBRG (Point-Based Region Growing). It should be noted that plane parameter uncertainties have also been computed during the region growing procedure in PBRG, which has not been considered in the proposed algorithms. Therefore, the comparison is not fully impartial; however, the uncertainty computation does not induce much time complexity compared to the coplanar points detection; see [27] for details.

5.1. Structured environment datasets

Two indoor datasets have been employed for evaluating the proposed SBRG algorithm. The first dataset is named TAMS Office. It was gathered using our customized 3D scanner at different positions on a floor that has a kitchen, a robot laboratory and several offices, including 40 point clouds. One typical scan in the laboratory is illustrated in Fig. 2. The dataset has been made publicly available at http://tams.informatik.uni-hamburg.de/research/datasets/index.php. The second dataset is from [29]; it was obtained by spinning a Hokuyo® URG-04LX which is designed for indoor use only. It is denoted as Indoor Hokuyo in this paper.

Two segmentation results from the first dataset using the SBRG algorithm are depicted in Fig. 6, and one from the second dataset is depicted in Fig. 7. Two different viewpoints have been given for each result.

5.2. Unstructured environment datasets

Two datasets have been employed to evaluate the proposed HRG algorithm. Both of them are publicly available. The first dataset, Collapsed Car Parking Lot, was gathered in an unstructured environment during NIST Response Robot Evaluation Exercise 2008 at Disaster City, Texas. The dataset is accessed at http://www.robotics.jacobs-university.de/datasets/RAW/RREE08/crashedCarPark/. The scans were gathered by a track robot equipped with an aLRF; the aLRF is based on a SICK S 300 which has a FoV of 270° of 541 beams. The sensor is pitched from −90° to +90° at a spacing of 0.5°, which leads to an organized point cloud of $541 \times 361 = 195{,}301$ points per sample. See [7] for details.

The second dataset, Barcelona Robot Lab, covers 10,000 square meters of the UPC Nord Campus in Barcelona, including 400 dense 3D point clouds. There are about 380,000 points in each scan. Since only unorganized point clouds are provided in this dataset, the points have been ordered using their spherical coordinates; see [20]. The original dataset is available at http://www.iri.upc.edu/research/webprojects/pau/datasets/BRL/php/dataset_data_access.php; in addition, the organized point clouds for it can be downloaded at http://tams.informatik.uni-hamburg.de/

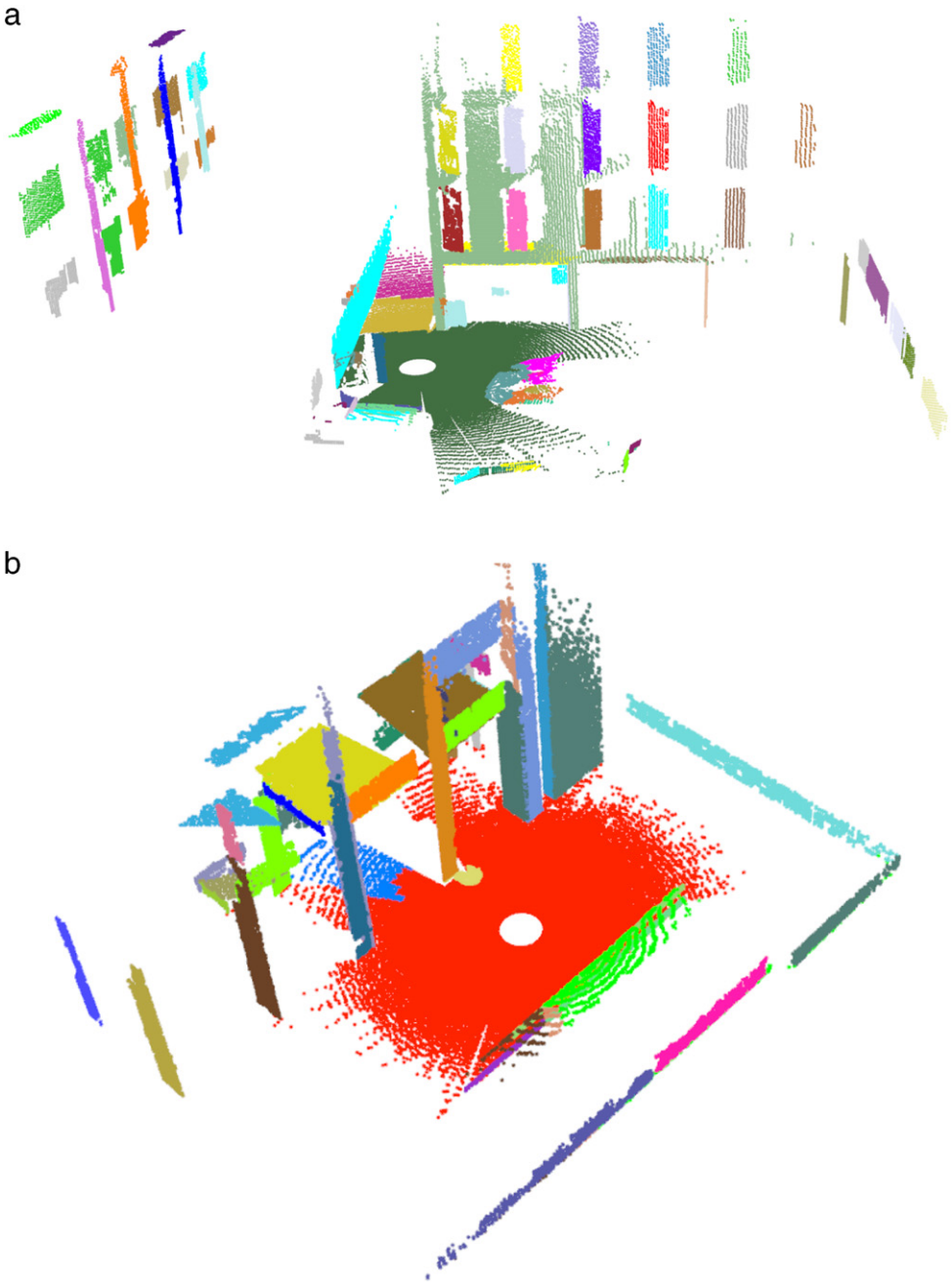


Fig. 9. Two typical segmentation results using the HRG algorithm for the Barcelona Robot Lab dataset.

research/datasets/index.php. Typical segmentation results have been drawn in Figs. 8 and 9.

5.3. Discussion

As shown in Figs. 5, 7, and 6, SBRG has good performance in structured indoor environments while producing unsatisfactory segmentation results in unstructured environments. HRG can deal with the unstructured environments which can be seen from Figs. 8 and 9. It has also been applied to the indoor datasets and is proven to have no advantage when compared to the SBRG algorithm in such environments.

To analyze their speeds, the algorithms are compared to the PBRG algorithm. We do not compare the algorithms to the Hough Transform method [30], because each point cloud in our experiment contains much more than 15 planar surfaces, and Borrmann et al. already reported that the Hough Transform is much slower than PBRG in such datasets. We do not compare our algorithms to that of [29] either, as the plane parameters are not updated during the patch clustering procedure in their algorithm; see Section 2.

For comparison, PBRG, SBRG and HRG have been applied to the indoor environments, while PBRG and HRG have been applied to the unstructured environments. The datasets Indoor Hokuyo and Barcelona Robot Lab are chosen for benchmarking the speed since they contain more point clouds than the other two datasets and are therefore better for statistical analysis. The segmentation time for each point cloud with different algorithms is depicted in Fig. 10 and Fig. 11. In Fig. 10, the subwindow sizes of 3×3 and 4×4 are used for both SBRG and HRG, and the point cloud size corresponds to the number of valid points in each point cloud. Since most subwindows have a planar appearance, SBRG and HRG have approximately the same speed when using equal subwindow sizes. The average

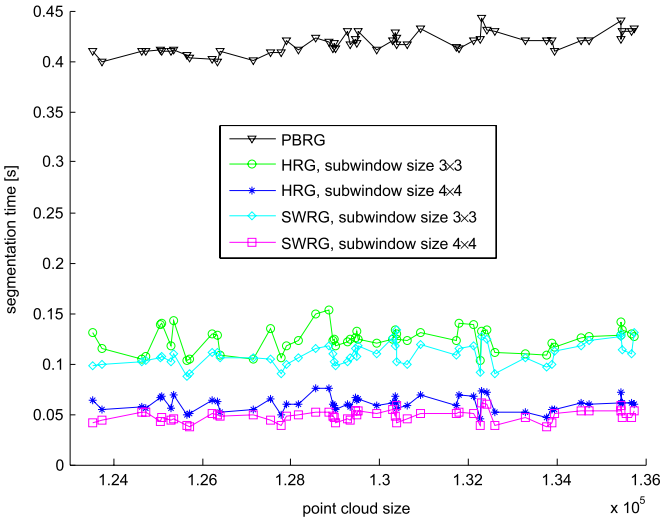


Fig. 10. Segmentation speed benchmarking result for a structured environment; all point clouds in the Indoor Hokuyo dataset have been employed.

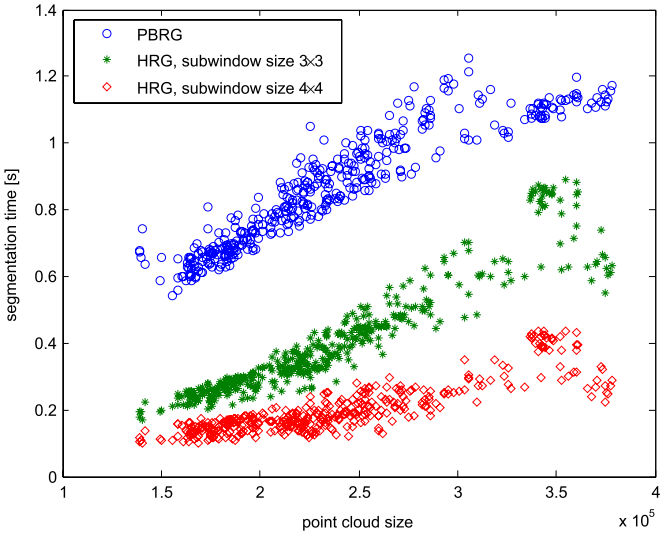


Fig. 11. Segmentation speed benchmarking result for an unstructured environment; all point clouds in the Barcelona Robot Lab dataset have been employed.

segmentation time has been illustrated in Table 1; it can be seen that SBRG is about 4 times faster than PBRG when the subwindow size is set to 3×3 and 9 times faster for subwindow size 4×4. For a point cloud with about 135,000 points, SBRG needs only 0.1 s for plane segmentation. For aLRFs, the plane segmentation time using SBRG is much shorter than obtaining a point cloud which is usually tens of seconds.

In Fig. 11, subwindow sizes 3×3 and 4×4 were utilized in the HRG algorithm. Even larger subwindow sizes have also been tested; however, they made the narrow planar surfaces unidentified. Note the approximately linear relation between the processing time and the point cloud sizes. The same as Fig. 10, the point cloud size corresponds to the number of valid points. The linearity of HRG is worse than PBRG since two kinds of growth units are used in it, and the time complexity depends on the clutter level of the environment which is analyzed in Section 4.1. When the subwindow size is increased, the segmentation speed is faster. For a point cloud with as much as 380,000 points, HRG needs less than 0.4 s for plane segmentation when the subwindow size is set to 4×4. Again, the plane segmentation time is much smaller than the data gathering time which means real-time data processing is promised. The average plane segmentation time is illustrated in

Table 1
Average plane segmentation time for the Indoor Hokuyo dataset. The time for SBRG and HRG is specified to each underlying subwindow size.

Algorithm	PBRG	SBRG		HRG	
		3×3	4×4	3×3	4×4
Time (s)	0.418	0.109	0.048	0.124	0.061

Table 2
Average plane segmentation time for the Barcelona Robot Lab dataset. The time for HRG is specified to each underlying subwindow size.

Algorithm	PBRG	HRG	
		3×3	4×4
Time (s)	0.864	0.408	0.198

Table 2. It shows that HRG is 4 times faster compared to PBRG when using the subwindow size of 4×4.

In general, the speed of both algorithms is faster when compared to the PBRG algorithm. The results are promising for plane-based mapping systems in both structured or unstructured environments and thus have attracted application areas such as home service robots and field robots. For indoor structured environments, the SBRG algorithm is suggested while the HRG algorithm is recommended for unstructured environments.

However, the algorithms have their limitations. First, they can only be employed when dealing with organized point clouds due to their need of the image-like structure for generating subwindows. Second, an optional subwindow size should be tuned for the working environment. If the size is too small, the plane parameter estimated for each subwindow is disturbed by the sensor noise, and the segmentation time is longer compared to a larger subwindow size. If the size is too large, many subwindows which locate on the surface boundaries are classified as nonplanar, making surface details missing in the SBRG algorithm. Furthermore, the long narrow planar surfaces will be lost in the segmentation result for both algorithms. From the experiments, we suggest to use subwindow size not smaller than 3×3 and not larger than 10×10. Third, the algorithms are limited to segmenting planar segments, while higher order surfaces are common and should be considered for more general environments.

6. Conclusion and future work

We presented two plane segmentation approaches in this paper, one for structured environments and the other for unstructured environments. The main idea is to use a relatively larger growth unit in the region growing procedure, i.e., using subwindows. It was found that the pure subwindow-based region growing algorithm is suitable for structured environments and has unsatisfactory performance when applied to unstructured environments. In order to deal with this problem, we proposed to use both subwindow and single point as the alternative growth unit in the hybrid region growing algorithm.

Both algorithms were evaluated using real world datasets, which gives promising results. For structured environments, the subwindow-based region growing algorithm can extract the planar surfaces from a point cloud with about 135,000 points within 0.1 s. For unstructured environments, the hybrid region growing algorithm needs less than 0.4 second for segmenting a point cloud with as many as 380,000 points. From the experiments, the algorithms are about 4 times faster than the point-based region growing when a proper size is set.

After segmentation, each point cloud can be represented as a set of planar segments. The area of each segment can be calculated using the range-image-based method proposed in our previous publication [20]. Additionally, the resulting area-attributed can

be used for determining corresponding segments between overlapping scans; based on the correspondences, the scans can be registered into a common coordinate system, the reader is referred to [20] for details of the registration approach.

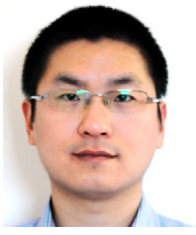
In the future, we will focus on learning other attributes of the resulted segments, such as plane parameter uncertainties and 2D outlines. Furthermore, since our custom-built 3D scanner can provide intensity information for each point, we will also try to find intensity features inspired by current image processing techniques such as gray-scale histograms. These attributes can help to improve the robustness of our registration algorithm proposed in [20]. An even further step of our research could be embedding the plane-based registration approach into a pose-graph SLAM system.

Acknowledgments

Junhao Xiao was funded by the China Scholarship Council (CSC), which is gratefully acknowledged. We would also like to thank Prof. Kaustubh Pathak from the Jacobs Robotics group at Jacobs University Bremen for providing us an access to their code (PBRG).

References

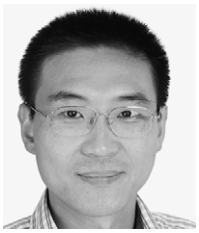
[1] S. Chen, Y. Li, N.M. Kwok, Active vision in robotic systems: a survey of recent developments, The International Journal of Robotics Research 30 (11) (2011) 1343–1377.
[2] R. Manduchi, A. Castano, A. Talukder, L. Matthies, Obstacle detection and terrain classification for autonomous off-road navigation, Autonomous Robots 18 (2005) 81–102.
[3] S. Kagami, R. Hanai, N. Hatao, M. Inaba, Outdoor 3D map generation based on planar feature for autonomous vehicle navigation in urban environment, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 1526–1531.
[4] P. Kohlhepp, P. Pozzo, M. Walther, R. Dillmann, Sequential 3D-slam for mobile action planning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004, pp. 722–729.
[5] J. Weingarten, R. Siegwart, Ekf-based 3D slam for structured environment reconstruction, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, 2005, pp. 3834–3839.
[6] R. Sun, S. Ma, B. Li, M. Wang, Y. Wang, A simultaneous localization and mapping algorithm in complex environments: SLASEM, Advanced Robotics 25 (6–7) (2011) 941–962.
[7] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, J. Poppinga, Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation, Journal of Field Robotics 27 (1) (2010) 52–84.
[8] A. Nüchter, J. Hertzberg, Towards semantic maps for mobile robots, Robotics and Autonomous Systems 56 (11) (2008) 915–926.
[9] D.F. Wolf, G.S. Sukhatme, Semantic mapping using mobile robots, IEEE Transactions on Robotics 24 (2) (2008) 245–258.
[10] R.B. Rusu, Z.C. Marton, N. Blodow, M. Dolha, M. Beetz, Towards 3D point cloud based object maps for household environments, Robotics and Autonomous Systems 56 (11) (2008) 927–941.
[11] B. Steder, G. Grisetti, W. Burgard, Robust place recognition for 3D range data based on point features, in: International Conference on Robotics and Automation, Anchorage, Alaska, USA, 2010, pp. 1400–1405.
[12] R. Kaushik, J. Xiao, Accelerated patch-based planar clustering of noisy range images in indoor environments for robot mapping, Robotics and Autonomous Systems 60 (4) (2012) 584–598.
[13] N. Vaskevicius, A. Birk, K. Pathak, S. Schwertfeger, Efficient representation in 3D environment modeling for planetary robotic exploration, Advanced Robotics 24 (8–9) (2010) 1169–1197.
[14] K. Pathak, A. Birk, N. Vaskevicius, J. Poppinga, Fast registration based on noisy planes with unknown correspondences for 3-D mapping, IEEE Transactions on Robotics 26 (3) (2010) 424–441.
[15] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, International Journal of Computer Vision 13 (2) (1994) 119–152.
[16] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM–3D mapping outdoor environments, Journal of Field Robotics 24 (8–9) (2007) 699–722.
[17] P. Biber, W. Strasser, The normal distributions transform: a new approach to laser scan matching, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, USA, 2003, pp. 2743–2748.
[18] M. Magnusson, A. Lilienthal, T. Duckett, Scan registration for autonomous mining vehicles using 3D-NDT, Journal of Field Robotics 24 (10) (2007) 803–827.
[19] J. Xiao, B. Adler, H. Zhang, 3D point cloud registration based on planar surfaces, in: 2012 IEEE International Conference on Multisensor Fusion and Information Integration, Hamburg, Germany, 2012, pp. 40–45.
[20] J. Xiao, B. Adler, J. Zhang, H. Zhang, Planar segment based three-dimensional point cloud registration in outdoor environments, Journal of Field Robotics (2013) 1–31.
[21] C.L. Bajaj, F. Bernardini, G. Xu, Automatic reconstruction of surfaces and scalar fields from 3D scans, in: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1995, pp. 109–118.
[22] N. Amenta, M. Bern, M. Kamvysselis, A new Voronoi-based surface reconstruction algorithm, in: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, 1998, pp. 415–421.
[23] D. Hähnel, W. Burgard, S. Thrun, Learning compact 3D models of indoor and outdoor environments with a mobile robot, Robotics and Autonomous Systems 44 (1) (2003) 15–27.
[24] J. Weingarten, G. Gruener, R. Siegwart, A fast and robust 3D feature extraction algorithm for structured environment reconstruction, in: International Conference on Advanced Robotics, Coimbra, Portugal, 2003, pp. 390–397.
[25] R. Lakaemper, L. Jan Latecki, Using extended em to segment planar structures in 3D, in: Proceedings of the 18th International Conference on Pattern Recognition, Washington, DC, USA, 2006, pp. 1077–1082.
[26] A. Harati, S. Gächter, R. Siegwart, Fast range image segmentation for indoor 3D-SLAM, in: IFAC Symposium onIntelligent Autonomous Vehicles, Toulouse, France, 2007, pp. 475–480.
[27] J. Poppinga, N. Vaskevicius, A. Birk, K. Pathak, Fast plane detection and polygonalization in noisy 3D range images, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 2008, pp. 3378–3383.
[28] G.-P. Hegde, 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, St. Louis, USA, 2009, pp. 4034–4039.
[29] R. Kaushik, J. Xiao, S. Joseph, W. Morris, Fast planar clustering and polygon extraction from noisy range images acquired in indoor environments, in: International Conference on Mechatronics and Automation, Xi'an, China, 2010, pp. 483–488.
[30] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, The 3D hough transform for plane detection in point clouds: a review and a new accumulator design, 3D Research 2 (2011) 32:1–32:13.
[31] J. Xiao, J.H. Zhang, H. Zhang, J.W. Zhang, H.P. Hildre, Fast plane detection for slam from noisy range images in both structured and unstructured environments, in: International Conference on Mechatronics and Automation, Beijing, China, 2011, pp. 1768–1773.
[32] K. Georgiev, R.T. Creed, R. Lakaemper, Fast plane extraction in 3D range data based on line segments, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, USA, 2011, pp. 3808–3815.
[33] F. Mufti, R. Mahony, J. Heinzmann, Robust estimation of planar surfaces using spatio-temporal ransac for applications in autonomous vehicle navigation, Robotics and Autonomous Systems 60 (1) (2012) 16–28.
[34] A. Trevor, J. Rogers, H. Christensen, Planar surface slam with 3D and 2D sensors, in: IEEE International Conference on Robotics and Automation, St. Paul, MN, USA, 2012, pp. 3041–3048.
[35] D. Dube, A. Zell, Real-time plane extraction from depth images with the randomized hough transform, in: 2011 IEEE International Conference on Computer Vision Workshops, Barcelona, Spain, 2011, pp. 1084–1091.
[36] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (1981) 381–395.
[37] R.B. Rusu, S. Cousins, 3D is here: point cloud library (PCL), in: IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 1–4.
[38] J. Weingarten, Feature-based 3D SLAM, Ph.D. Thesis, EPFL, 2006.
[39] G. Guennebaud, B. Jacob, et al. Eigen v3, 2010. http://eigen.tuxfamily.org.



Junhao Xiao (M'12) is a Ph.D. student in the Institute of Technical Aspects of Multimodal Systems (TAMS), Department Informatics, University of Hamburg. He received his bachelor degree in Automation from the National University of Defense Technology (NUDT), 2007. He got a scholarship from the China Scholarship Council (CSC) and joined TAMS in Sep. 2009. His research interests include mobile robotics, sensor fusion and especially 3D robotic mapping.

Jianhua Zhang (M'11) received the M.Sc. degree at Zhejiang University of Technology in 2009 and the Ph.D. degree at the University of Hamburg in 2012. He joined Zhejiang University of Technology, China, in December 2012. His research interests include category discovery, object detection, image segmentation, medical image analysis.

Benjamin Adler is a scientific assistant at the Institute of TAMS, Department Informatics, University of Hamburg. He received his diploma degree in Computer Science at the University of Hamburg in 2008 and is currently working on his Ph.D. Thesis. His research interests include mobile robotics, GNSS systems and multi-sensor fusion.



Houxiang Zhang (M'04–SM'12) received Ph.D. degree in Mechanical and Electronic Engineering from Beijing University of Aeronautics and Astronautics, China, in 2003. From 2004, he worked as Postdoctoral Fellow at TAMS, Department of Informatics, University of Hamburg, Germany. Then he joined the Faculty of Maritime Technology and Operations, Ålesund University College, Norway in April 2011 where he is a full Professor on Robotics and Cybernetics. The focus of his research lies on mobile robotics, especially on climbing robots and urban search and rescue robots, modular robotics, and nonlinear control algorithms.

Jianwei Zhang (M'92) received both his Bachelor of Engineering (1986, with distinction) and Master of Engineering (1989) from the Department of Computer Science of Tsinghua University, Beijing, China, and his Ph.D. (1994) at the Institute of Real-Time Computer Systems and Robotics, Department of Computer Science, University of Karlsruhe, Germany. Dr. Jianwei Zhang is professor and head of TAMS, Department of Informatics, University of Hamburg, Germany.