Robotica (2015) volume 33, pp. 1751–1774. © Cambridge University Press 2014
doi:10.1017/S0263574714000927

Line segment-based fast 3D plane extraction using nodding 2D laser rangefinder

Su-Yong An†*, Lae-Kyoung Lee‡ and Se-Young Oh‡

†*Institute of Industrial Technology, Samsung Heavy Industries, Daejeon, 305-380, Korea*
‡*Department of Electrical Eng., Pohang University of Science and Technology (POSTECH), Pohang, Gyungbuk 790-784, Korea*

(Accepted March 13, 2014. First published online: May 1, 2014)

SUMMARY
Three-dimensional (3D) data processing has applications in solving complex tasks such as object recognition, environment modeling, and robotic mapping and localization. Because using raw 3D data without preprocessing is very time-consuming, extraction of geometric features that describe the environment concisely is essential. In this sense, a plane can be a suitable geometric feature due to its simplicity of extraction and the abundance in indoor environments. This paper presents an online incremental plane extraction method using line segments for indoor environments. Our data collection system is based on a "nodding" laser scanner, so we exploit the incremental nature of its data acquisition in which physical rotation and 3D data processing are conducted in parallel. Line segments defined by two end points become supporting elements that comprise a plane, so a large proportion of scan points can be ignored once the line segments are extracted from each scan slice. This elimination of points reduces the algorithm complexity and computation time. Experiments with the tens of complete scan data sets which were acquired from a typical indoor environment demonstrated that our method was at least three times faster than the state-of-the-art methods.

KEYWORDS: 3D point cloud; Laser scanner; Line segment; Plane fitting.

1. Introduction

For robotic 3D mapping using point cloud data, structured features, such as cylinders, spheres, planes, or unstructured surface features can be used.[1] These normally represent floors, walls, hallways, or pillars that the robot can easily detect using a 3D sensing system. Because regular structures mainly consist of planar surfaces, planes are most useful in 3D mapping of typical indoor environments owing to their abundance.[2–6] Thus, rapid, efficient extraction of planes from a raw 3D point cloud is an essential capability for indoor robots.

To obtain dense 3D point cloud data, a 3D scanning system based on a rotating laser scanner has been widely used.[2,7–11] This system has been used for several applications, including teleoperation,[8] mobile manipulation,[12] 3D mapping and exploration,[2] and ego-motion estimation.[10] A typical system consists of a laser scanner that acquires 2D scan data and an actuator that controls the scanner's orientation. By continuous synchronized manipulation (moving and scanning), it generates accurate 3D point cloud data[13] that consist of tens of thousands to millions of scan points. One of its advantages is its long detection range in the lateral and vertical directions, because the scan angle and actuator tilt angle are both usually $>180°$. Another advantage is that we can easily control the number of acquired scan points, by adjusting the resolution of the laser and actuator.

By exploiting these advantages of the 3D scanning system, we have developed an incremental 3D plane extraction method for modeling an indoor environment. A laser scanner is mounted on a mobile robot and scans the environment incrementally in synchrony with the actuator movement. Planes are identified in three steps: (i) 2D line segments are extracted from each scan slice, then clustered according to their orientation. The line segment extraction process corresponds to voxel grid filtering

* Corresponding author. E-mail: hoppery0420@gmail.com

to reduce the number of scan points accessed,[14] (ii) the planes are extracted from the cluster of line segments by using M-estimator Sample Consensus (MSAC),[15] refined further by considering the uncertainty of each scan point, and (iii) the extracted planes are split and merged online depending on plane fitting error. The main contribution of our algorithm is embodied in the following three characteristics:

- Because scan data are acquired incrementally, the core algorithm can be run in parallel with the physical operation of the laser scanner and the actuator, so no additional running time is needed, because the entire plane extraction process is completed with the final scan step.
- Because only the end points of line segments are considered when extracting a plane, the number of scan points that are actually accessed can be greatly reduced (to $< 5\%$ of all scan points)
- The proposed method is scalable to large environments. The processing time is linearly proportional to the number of total line segments, so the size of the environment does not affect the algorithm complexity.

2. Related Work

Although various methods have been proposed for plane extraction, Random Sample Consensus (RANSAC) is the most widely used,[16] especially to find inlier points that constitute a plane. First, it selects N points randomly and fits a plane to them. If the fitting error is below a user-defined tolerance, it accepts this fit. Otherwise, RANSAC is repeated with another random sample until a suitable fit is found or a certain number of iterations has been performed.

RANSAC and iterative closest point (ICP)-based optimization techniques have been combined to extract planes.[17] A randomly-selected point and its two neighboring points are used to calculate an initial plane hypothesis, and then all inlier points are projected onto the plane. The projected and inlier points are matched using ICP, and rotation and translation ($\mathbf{R}$, $\mathbf{t}$) are found by calculating refined plane parameters. Here, the ICP method replaces iterative random sample selection and plane fitting, thus reducing the time required to estimate the plane. However, the ICP is also an iterative process and may get trapped in local minima.

A divide-and-conquer strategy has been used to extract local planes within a short time by reducing the spatial size; the local planes are merged using a region-growing technique. 3D point cloud data were first decomposed into small cubic cells having a fixed volume.[4] Then, in each cell, a best-fitting plane was extracted by RANSAC. To merge similar planes, region-growing was applied to the entire set of small planes by using normal vectors and proximity between several neighborhood planes. This approach can extract planes very quickly in one cubic cell, but decomposition of 3D space requires extra memory and the complexity of the algorithm depends on the number of the cells. Filtered normal estimation and voxel growing was used in ref. [18]; this approach searches the neighborhood in a constant time, and therefore differs from region-growing. In ref. [19], RANSAC, minimum description length, and region growing were all integrated to extract planes from noisy 3D point cloud data. Octree splitting and merging is another divide-and-conquer method in which complete data are continuously divided into subnodes.[20] At each subnode, the best-fit plane is calculated by solving a least-square problem. However, these approaches have scalability problem similar to the method in ref. [4].

For robotic mapping, planar features are used to generate simple 3D models;[21,22] planes are extracted by region growing or an expectation maximization (EM) paradigm. The results provide textured 3D indoor models but require several minutes and complete datasets. Therefore, the plane extraction algorithm should be run offline.

Randomized Hough Transform (RHT) with a novel accumulator design has been developed to extract planes from 3D point clouds.[23] However, to reduce computation time, this method needs the number of cells in an accumulator as prior information; this requirement implies that the method can be affected by the structure of the environment. Another state-of-the-art method based on region-growing method[24] exploits the neighborhood relation of pixels in range images, but the method cannot be directly applied to raw 3D point cloud.

A 3D laser scanning system that uses a nodding mechanism acquires 3D point cloud data sequentially in a specified order. Moreover, the use of line segments can reduce the number of scan points accessed to extract planes, thereby reducing the total computing time. In ref. [6], line

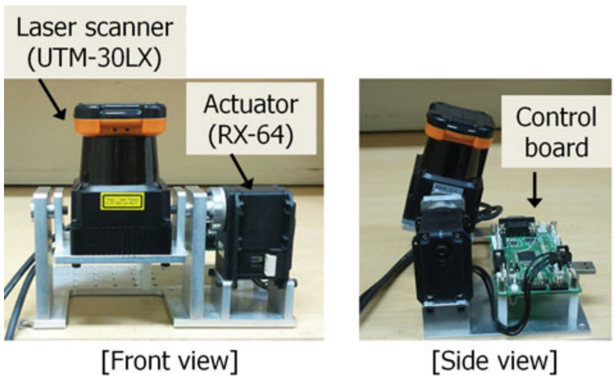

Fig. 1. System configuration.

segments were extracted using a standard Hough transform at each scan slice, then clustered online to reconstruct surfaces. After scanning is complete, objects are segmented by merging conglomerations of points, lines, and surfaces into one object. Similarly, multiple planes are segmented by sequentially extracting and clustering of line segments and are validated by using a simple image processing algorithm.[8] Finally, the ground plane is distinguished from other planes for traversability checks in teleoperation tasks. The use of line segments and their connected components representation could vastly reduce the search space for plane extraction stage.[25]

3. System

3.1. System specification

The key components of our system are a laser scanner, an actuator, and a control board (Fig. 1). Because this system has a simple structure, it can be readily applied to mobile robots that run in indoor or outdoor environments. The laser scanner is a Hokuyo URG-30LX which can detect objects up to 30 m distant. The scanning range is -45° to 225°, with angular resolution of 0.25° ; i.e., 1080 scan points when fully utilized. The scan angle can be controlled depending on the user's needs, especially when the number of resulting scan points is limited. The scan time of the laser scanner, which critically affects the system performance, is 25 ms to gather all the scan points in one planer sweep. The actuator is a Robotis Dynamixel servo.[26] It operates from 12 V to 18.5 V; its maximum torque is 5096 N $\times$ m, and minimum control angle is 0.29°. Because its rotation angle is 0° to 300°, it can gather sufficient spatial information for our purposes. The control board is equipped with a common AVR microcontroller and is connected to a PC via a USB 2.0 interface.

3.2. System control scheme

Generally a laser scanner can be rotated in three axes;[7,27] yaw, pitch, and roll. Each axis has advantages and disadvantages depending on how the system is used. We adopt pitching (i.e., tilting) scanning that gathers spatial information from bottom to top, i.e., floor to ceiling. This scanning mode has been applied in many mobile robotic applications.[2,6–10] We can control the number of resulting scan points by adjusting the angular resolution of the laser scanner and the actuator. Therefore, our system can acquire up to 1024×1080 (maximum scan steps $\times$ number of scan points per scan) $= 1{,}105{,}920$ points. We normally operate the system with a scan angle of 180°, a tilt angle of 1.16° as the unit step, and a total of 120–155 scan steps, which amounts to a tilt range of 140° –180°. This setting yields 100,000 scan points to be processed.

The control scheme operates in two steps (Fig. 2). First, the control board sends a control command to the actuator, which then moves to the designated angle. Second, the control board triggers the laser scanner to acquire scan points counterclockwise. The scanning time is 25 ms, including the time required for physical spin and data conversion. Therefore, the scanning time affects mainly the time efficiency of the proposed system. The control board continuously synchronizes the behavior of the actuator and laser scanner. Thus, the system requires at least 25 ms per control cycle, and this

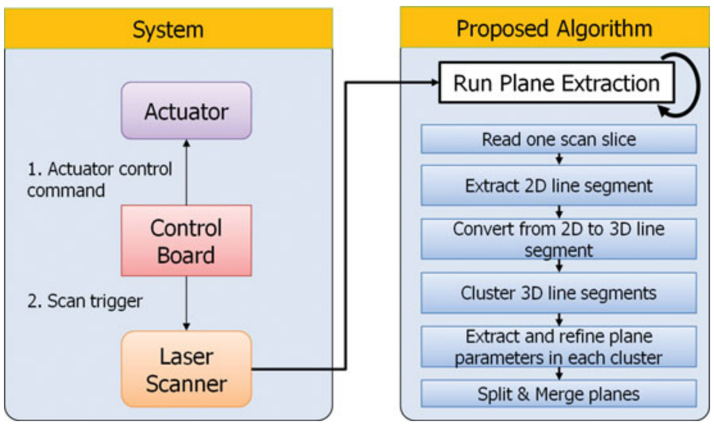

Fig. 2. Schematic of the control scheme. Physical tilting and algorithm implementation can be performed in parallel.

Table I. Plane extraction procedure.

```
Plane_Extraction

1:   $\Pi = \emptyset$        //initial plane
2:   $\Psi = \emptyset$        //initial line cluster
3:   while tilt step $k$ is not final step do
4:       $\Sigma^k$ = acquire_scan_points ()
5:       $\Lambda^k$ = extract_ line_ segments ($S^k$)
6:       $\Pi$ = assign_existing_planes ($\Lambda^k$, $\Pi$)
7:       $\Psi$ = line_clustering ($\Lambda^k$, $\Psi$)
8:       $\Pi$ = plane_fitting ($\Psi$, $\Pi$)        //performed every 5 scan steps
9:       $\Pi$ = split_and_merge ($\Pi$, $\Psi$)      // performed every 7 scan steps
10:      $\Pi$ = fit_remaining_lines ($\Pi$, $\Psi$)   // performed every5 scan steps
11:  endwhile
12:  return $\Pi$
```

requirement cannot be reduced any further. However, we use parallel processing to avoid the need for any additional time to run our core algorithm.

Suppose that we can process all the scan points acquired from the 0th to the $(k - 1)$ th scan step while the sensor gathers the kth set of spatial information. The system then requires no additional time to process the acquired data and finishes the entire procedure as soon as actuator tilting is completed. To realize this assumption, two questions should be considered. (i) Can the system process tens of thousands of scan points within the given time? During the first few scan steps, a small number of scan points is acquired, but the number increases linearly as the number of scan steps increases. (ii) After the full scan is completed, is any additional postprocessing required to refine the resulting planes? This depends on whether the proposed method includes refinement of interim results. Therefore, we need to greatly reduce the number of scan points that are accessed for plane extraction and also use an iterative split-and-merge scheme to process interim plane extraction results.

4. Plane Extraction from Line Segments

The first use of line segments for segmenting a 3D point cloud data[6] was a generalized version for surface detection, polygon generation, and 3D object segmentation done online and offline. However, our algorithm is specialized for plane extraction to obtain plane parameters online, and includes more sophisticated stages such as online refinement of resulting planes and an iterative split and merge step.

The proposed plane extraction method (Table I) is based on two propositions. (i) A line segment and its related parameters can be defined by only two points ($\mathbf{p}_1 = (x_1, y_1)$, $\mathbf{p}_2 = (x_2, y_2)$); the parameters of line slope-intercept equation $y = mx + b$; can be obtained as $m = (y_2 - y_1)/(x_2 - x_1)$,

$b = y_i - mx_i$. (ii) A plane $\mathbf{P}$ contains an infinite number of line segments and can be defined by only two line segments on the plane. A plane can be parameterized as

$$\hat{\mathbf{n}}\mathbf{p} - d = 0, \tag{1}$$

where $\hat{\mathbf{n}} = (n_x, n_y, n_z)$ is a unit normal vector, and d is the distance from the origin. An infinite number of points $\mathbf{p}$ on $\mathbf{P}$ satisfy Eq. (1), any two of them define a line segment on $\mathbf{P}$. Similarly, we can choose an infinite set of such paired points on the plane.

Consequently, we can draw potential planes from a finite collection of line segments that are defined by only two end points. This approach can reduce the number of scan points that are accessed to extract planes because all the intermediate scan points in the line segment can be excluded. Also, a line segment is extracted not from a 3D point cloud by exhaustive search but from 2D scan points acquired in the same scan slice, therefore the line is always extracted within a very short time (in our case, $\leq$1 ms).

Our system gathers spatial information sequentially from bottom to top. At each scan step, we first perform line extraction and then determine whether the extracted line belongs to one of the existing planes. Next, we identify the remaining line segments that do not belong to any existing plane, and cluster them according to their orientation. If the size of each cluster exceeds a threshold N_c, iterative plane fitting is performed using MSAC. By MSAC fitting, we obtain inlier line segments comprising the plane, and further refinement performs nonlinear least-squares fitting by considering the uncertainty of each scan point. Based on the fitting error, the plane is split into two or more subplanes such that the fitting error of each plane is smaller than a threshold e_f. If any two planes are coplanar, they are merged. This manipulation of the resulting planes is called split-and-merge and enables the system to refine the final results. Finally, unfitted lines in the line cluster are checked again to determine whether they can be fitted to existing planes.

Our method has two advantages. The first is that no additional time is required to run core algorithms because they are run online and in parallel with physical operation of the actuator and laser scanner. The other is that it can extract a piecewise plane from a gently curved surface owing to the presence of line segments on such a surface.

Our method has a disadvantage when a plane is far from the origin, where a dense cluster of scan points cannot be acquired. In this region, plane extraction may fail because the sparseness of the 3D points means that sufficient line segments cannot be extracted.

4.1. Line segment extraction

4.1.1. Clustering of scan points. Line extraction (Table II) begins by clustering scan points. At scan step k, we acquire a set $S^k = \{\mathbf{p}_i^k | i = 1, \ldots, N_s^k\}$ of scan points, where N_s^k is the number of scan points. Each scan point $\mathbf{p}_i^k$ is described by its range and bearing (ρ_i^k, θ_i^k), where ρ_i^k is the distance to a sensed object, and θ_i^k denotes the angle w.r.t. the x-axis in the sensor frame. Hence, the point can be converted to $\mathbf{p}_i^k = (\rho_i^k \cos\theta_i^k, \rho_i^k \sin\theta_i^k)$ in 2D Cartesian coordinates. To extract a line segment from these raw scan points, several line extraction algorithms have been proposed, such as RANSAC-based, Hough-transform-based, and EM-based methods.[28] However, all of these methods analyze a dense set of raw scan points directly without preprocessing, and therefore have higher computational costs than ours, which performs preprocessing by segmenting of the raw scan points.

As a preprocessing step, we first detect breakpoints, i.e., discontinuities in the sequence of raw measurements. Most breakpoint detection algorithms examine the distance between two consecutive points $\mathbf{p}_i, \mathbf{p}_{i-1}$ (we omit the scan step index k for brevity),

$$\|\mathbf{p}_i - \mathbf{p}_{i-1}\| < D_o, \tag{2}$$

where D_o is a predefined threshold. If Eq. (2) is satisfied, $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ are considered to be in the same cluster; otherwise, $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$ are considered breakpoints, where $\mathbf{p}_i$ is assumed to be the first element of a new cluster. Therefore, to achieve correct clustering of scan points, D_o must be selected carefully. Because D_o should depend on the scan distance ρ_{i-1}, an intuitive method of determining

Table II. Line segment extraction.

extract_ line_ segments ($\sum^k$)

```
1:  $\Lambda^k = \emptyset$
2:  for $i$=2 to $N_s^k$        //scan point clustering
3:      find breakpoint using $\|\mathbf{p}_i - \mathbf{p}_{i-1}\| < D_o$
4:      if found, create new cluster $Q^{\text{new}}$
5:      else $Q = Q \cup \{\mathbf{p}_i\}$
6:  endfor
7:  for $i$=1 to $N_q$         //splitting clusters
8:      split cluster $Q_i$ into several subclusters until the criterion is met
9:  endfor
10: for $i$=$|\Lambda^k|$ to 1        //merging clusters
11:     find best matched cluster $Q_j$ by collinearity and proximity
12:     if found
13:         $Q_{new} = Q_i \oplus Q_j$,  $L^k = L^k \cup \{Q_{new}\}$
14:     else $L^k = L^k \cup \{Q_i\}$
15: endfor
16: for $i$=$|\Lambda^k|$ to 1        //remove unavailable line segments
17:     check availability of line segment by length and # scan points
18:     if not available, $L^k = L^k - \{\lambda_i^k\}$
19: endfor
20: return $\Lambda^k$
```

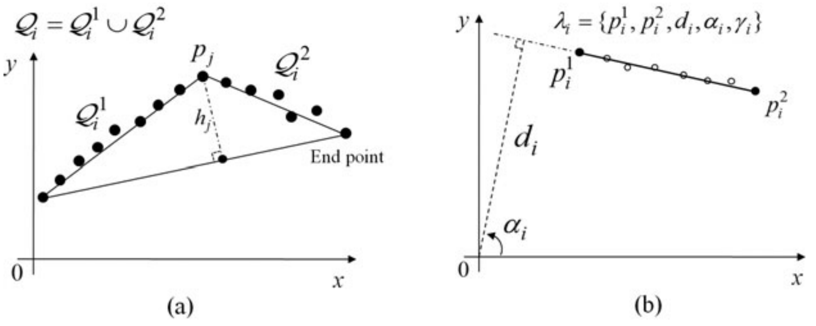

Fig. 3. (a) Splitting clusters. (b) Definition of a line segment in 2D space. White circles: intermediate points comprising the line segment; they are considered only when counting γ_i and are removed from the line segment properties.

D_o has been proposed:[29]

$$\|\mathbf{p}_i - \mathbf{p}_{i-1}\| < D_o = \rho_{i-1} \cdot \sin(\Delta\phi)/\sin(\lambda_u - \Delta\phi) + 3\sigma_d, \tag{3}$$

where $\Delta\phi$ is the sensor resolution and λ_u is chosen on the basis of user experience. The term $3\sigma_d$ is added to account for noise associated with ρ_i when $\mathbf{p}_i$ is close to the sensor frame origin. Quantitative determination of λ is an unresolved problem, but in mobile robot applications $\lambda_u = 10°$ has been reported to be satisfactory.

4.1.2. Splitting clusters using iterative end-point fitting and merging line segments. Suppose that the scan point clustering step yields a total of N_q clusters $W = \{Q_i | i = 1, \ldots, N_q\}$. These clusters should be further processed by dividing them into several line segments. To do this, we adopt iterative end-point fitting (IEPF),[30] which has been used in many vision and robotics applications.[11,29,31] This algorithm recursively splits cluster Q_i into two subsets Q_i^1, Q_i^2 until a validation criterion is satisfied (Fig. 3a). First, the initial line l_i is constructed by connecting the two end points of the cluster. Next, all scan points $\mathbf{p}_j \in Q_i$ are tested against $h_j > h_o$, where h_o is a predefined threshold. Then, Q_i is divided into two subclusters at the point having maximum h_j among those points that satisfy $h_j > h_o$. This process is iterated until no new subclusters can be created. Each resulting cluster is likely to be

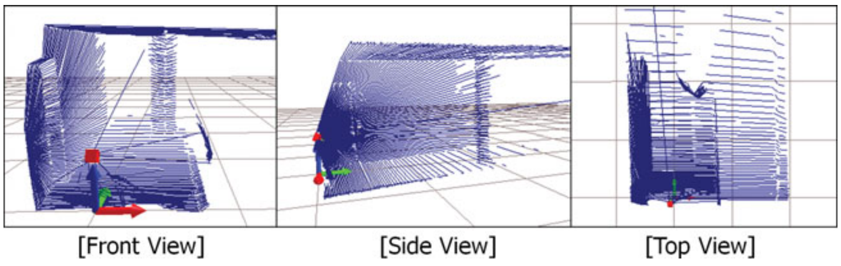

Fig. 4. Representation of 3D line segments after 3D conversion.

a potential line segment $\lambda_i = \{\mathbf{p}_i^1, \mathbf{p}_i^2, d_i, \alpha_i, n_i\}$ belonging to a certain plane, where $\mathbf{p}_i^1, \mathbf{p}_i^2$ are two end points, d_i and α_i are the line parameters, and n_i is the number of scan points that compose the line segment (Fig. 3b). All the intermediate points in the cluster are neglected thereafter and hence are never accessed again. After extracting line segments, we perform a merging process to combine short line fragments:

$$|\alpha_i - \alpha_j| < \alpha_o \wedge \left\|\mathbf{p}_i^2 - \mathbf{p}_j^1\right\| < d_o. \tag{4}$$

These conditions check for collinearity and proximity between two line segments. The individual short line segment may not be effective for the detection of salient planes, whereas connecting two or more similar line segments greatly increases the effectiveness of plane detection.

Unexpected short line segments can be extracted on an uneven surface; we eliminate these by removing segments that are less than a threshold $L_{\min}$ long, or that have too few scan points:

$$\left\|\mathbf{p}_i^1 - \mathbf{p}_i^2\right\| < L_{\min} \vee n_i < N_e. \tag{5}$$

4.1.3. 3D Conversion of 2D line segments. At scan step k, we have N_l^k potential line segments in 2D space. Because a line segment is defined by two end points, we need to convert only these points to 3D space (Fig. 4). The conversion equations are:

$$\begin{aligned} x &= \rho\cos\theta, \\ y &= \rho\sin\theta\cdot\cos\left(\varphi - \frac{\pi}{2}\right) + d_l\cos\varphi, \\ z &= \rho\sin\theta\cdot\sin\left(\varphi - \frac{\pi}{2}\right) + d_l\sin\varphi, \end{aligned} \tag{6}$$

where ρ and θ are the range and bearing measurements, respectively, of the laser scanner, φ is the pitch angle, and d_l is the distance between the pitch axis and the laser scanner. Then, a line segment is defined in 3D space again:

$$\lambda_i^k = \left\{\mathbf{u}_i^k,\ \mathbf{v}_i^k,\ \boldsymbol{\delta}_i^k\right\}, \tag{7}$$

where $\mathbf{u}_i^k$, $\mathbf{v}_i^k$ are the two end points, and $\boldsymbol{\delta}_i^k$ is a unit vector that indicates the direction of the line segment, which is defined by $(\mathbf{u}_i^k - \mathbf{v}_i^k)/\|\mathbf{u}_i^k - \mathbf{v}_i^k\|$.

4.1.4. Assigning extracted line segments to existing planes. At the end of the line extraction stage, each resulting 3D line segment is assigned to one of the existing planes, if possible (Table III). The association rules for determining whether line segment λ_i^k belongs to a plane $\Pi_j : \ \hat{\mathbf{n}}_j\mathbf{p} - d_j = 0$ are based on proximity and orthogonality constraints: the minimum distance between the plane and the two end points of the line segment should be shorter than a threshold and the direction vector of the line segment and the normal vector of the plane should be orthogonal. That is, the inner product of

Table III. Assigning line segments to existing planes.

```
assign_existing_planes ($\Lambda^k$, $\Pi$)

1: for $i$=$|\Lambda^k|$ to 1
2:     find the best matched plane $\Pi_j$ using proximity and orthogonality
3:     if found, $\Pi_j = \Pi_j \cup \{\lambda_i^k\}$,  $L^k = L^k - \{\lambda_i^k\}$
4: endfor
5: return $\Pi$
```

Table IV. Line segment clustering.

```
line_clustering ($\Lambda^k$, $\Psi$)

1: for $i$=$|\Lambda^k|$ to 1
2:     find the best matched cluster $X_j$ using directional vector
3:     if found,
4:         $C_j = C_j \cup \{\lambda_i^k\}$
5:         $\bar{\boldsymbol{\delta}}'_m = \tau\vec{\boldsymbol{\delta}}_j^k + (1-\tau)\bar{\boldsymbol{\delta}}_m$        //update average directional vector
6:     else create a new cluster, $C^{new} = C^{new} \cup \{\lambda_i^k\}$,  $Y = Y \cup \{C^{new}\}$
7:     $L^k = L^k - \{\lambda_i^k\}$
8: endfor
9: return $\Psi$
```

the two vectors should be nearly zero:

$$\begin{aligned} &D_{ij} = \min\left(\left|\hat{\mathbf{n}}_j\mathbf{u}_i^k - d\right| / \|\hat{\mathbf{n}}_j\|^2, \left|\hat{\mathbf{n}}_j\mathbf{v}_i^k - d\right| / \|\hat{\mathbf{n}}_j\|^2\right) < D_m \\ &\left|\hat{\mathbf{n}}_j \cdot \boldsymbol{\delta}_i^k\right| < \varepsilon_o \end{aligned}, \tag{8}$$

where D_m and ε_o represent the degrees of proximity and orthogonality, respectively. Therefore, the ith line segment is part of the jth plane if D_{ij} has a minimum value over all j. This step reduces unnecessary clustering of line segments by first assigning them to existing planes.

4.2. Clustering

The extracted line segments are clustered incrementally according to their direction vectors (Table IV). We have a set of line segment clusters $Y = \cup_{i=1}^{N_c} C_i = \{N_i, \lambda_i^{1:N_i}, \bar{\boldsymbol{\delta}}_i\}$, where N_c is the total number of clusters constructed so far, N_i is the number of line segments contained in the ith cluster, and $\bar{\boldsymbol{\delta}}_i$ is the average direction vector of the ith cluster. The only criterion used to associate a line segment λ_j^k and a cluster C_i is the inner product of two direction vectors, $\boldsymbol{\delta}_j^k$ and $\bar{\boldsymbol{\delta}}_i$:

$$\left|\boldsymbol{\delta}_j^k \cdot \bar{\boldsymbol{\delta}}_i\right| > \delta_c, \tag{9}$$

where δ_c is a predefined threshold. The line segment λ_j^k then belongs to the cluster C$_m$, that has the maximum inner product with λ_j^k. Then, $\bar{\boldsymbol{\delta}}_i$ can be recalculated by a simple moving average (SMA):

$$\bar{\boldsymbol{\delta}}'_m = \left(N_m\bar{\boldsymbol{\delta}}_m + \boldsymbol{\delta}_j^k\right)/(N_m + 1). \tag{10}$$

Although SMA usually works, it may not if the direction vector changes with increasing scan step, even though the line segments lie in the same plane (Fig. 5a). To solve this problem, we have two alternatives, a linearly weighted moving average (LWMA) and an exponentially weighted moving average (EWMA). Both apply higher weights to recently added line segments than to those added earlier. In LWMA, the weights decrease linearly:

$$\bar{\boldsymbol{\delta}}'_m = \left((N_m+1)\boldsymbol{\delta}_j^k + N_m\boldsymbol{\delta}_m^{N_m} + \cdots + 2\boldsymbol{\delta}_m^2 + \boldsymbol{\delta}_m^1\right) / \left((N_m+1) + N_m + \cdots + 2 + 1\right). \tag{11}$$

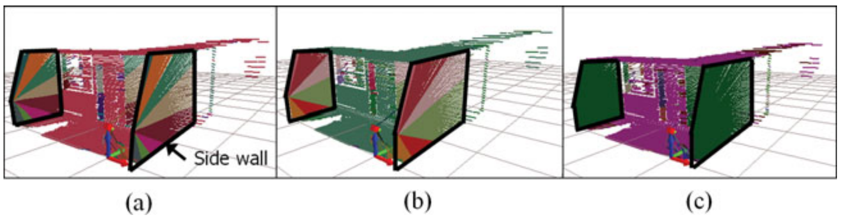

Fig. 5. Line segment clustering results. Region enclosed by thick black lines corresponds to a side wall; each cluster has its own color. (a) Clustering by SMA (eight clusters). (b) Clustering by LWMA (six clusters). (c) Clustering by EWMA (one cluster).

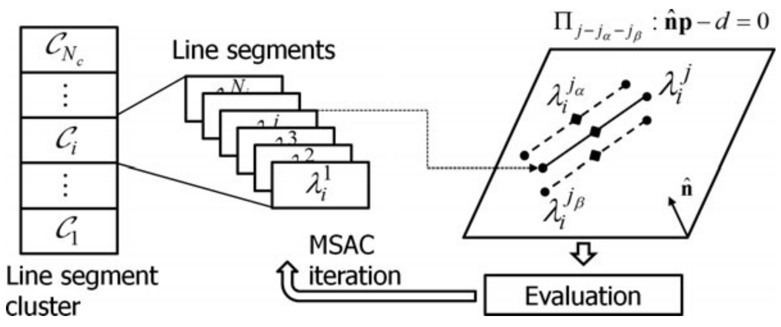

Fig. 6. Plane fitting with three line segments using RANSAC. Six end points from three line segments compose a plane, which is then evaluated by calculating the fitting error. The number of RANSAC iterations is linearly proportional to the number of line segments.

Table V. Plane fitting.

```
plane_fitting ($\Psi$, $\Pi$)

 1:  for $i=|\Psi|$ to 1
 2:    do
 3:      generate $N_p$ plane hypotheses from $C_i$ using MSAC
 4:      choose best fitting plane $\Pi_j$ which has lowest cost
 5:      if $\Pi_j$ is available, refine the plane parameter by nonlinear optimization
 6:        $P = P \cup \{\Pi_j\}$
 7:        plane extraction flag=TRUE
 8:      else plane extraction flag=FALSE
 9:    while plane extraction flag=TRUE
10:  endfor
11:  return $\Pi$
```

This process can reduce the number of clusters, but several may still exist in the same plane (Fig. 5b). In EWMA, the weights decrease exponentially:

$$\bar{\boldsymbol{\delta}}'_m = \tau \vec{\boldsymbol{\delta}}^k_j + (1-\tau)\bar{\boldsymbol{\delta}}_m, \tag{12}$$

where $0 \leq \tau \leq 1$ represents the degree of weigh decay. EWMA yields the desired result (Fig. 5c). If the line segment does not belong to an existing cluster, we use it as the first member of a new cluster.

4.3. Plane fitting

Plane extraction is divided into an initial plane fitting that uses MSAC and a refinement that uses nonlinear least squares (Table V). In the first step, for each cluster, a plane is segmented by randomly selecting three line segments in the local region (Fig. 6). This local selection of line segments is used to consider the local distribution of planes; i.e., several planes can exist in one cluster. After extracting potential planes, the plane parameter can be refined by considering the uncertainty of each end point that comprises the plane. This process uses iterative nonlinear optimization to further reduce the plane fitting error.

4.3.1. Initial fitting using MSAC. The previous stages yielded a set of line segment clusters $Y = \cup_{i=1}^{N_c} C_i = \{N_i,\ \lambda_i^{1:N_i},\ \bar{\boldsymbol{\delta}}_i\}$. Planes are extracted on a per-cluster basis, so line segments contained in different clusters are not chosen simultaneously. This constraint significantly reduces the computational cost by reducing the candidate search space (Fig. 6). First, we select the *j*th line segment λ_i^j in the *i*th cluster; to construct the candidate plane, we also select the two line segments $\lambda_i^{j_\alpha}$, $\lambda_i^{j_\beta}$ that are closest and second-closest to λ_i^j. In this case, closeness is defined as the inverse of the distance between the midpoints of the two line segments. The six end points of the three selected line segments then construct a candidate plane $\Pi_{j-j_\alpha-j_\beta}:\ \hat{\mathbf{n}}\mathbf{p} - d = 0$, where the plane parameters $\hat{\mathbf{n}}$ amd d are determined by solving the following homogeneous linear system:

$$\begin{bmatrix} \hat{\mathbf{n}}\mathbf{u}_i^j - d \\ \hat{\mathbf{n}}\mathbf{v}_i^j - d \\ \vdots \\ \hat{\mathbf{n}}\mathbf{u}_i^{j_\beta} - d \\ \hat{\mathbf{n}}\mathbf{v}_i^{j_\beta} - d \end{bmatrix} = \begin{bmatrix} u_{xi}^j & u_{yi}^j & u_{zi}^j & -1 \\ v_{xi}^j & v_{yi}^j & v_{zi}^j & -1 \\ \vdots & \vdots & \vdots & \vdots \\ u_{xi}^{j_\beta} & u_{yi}^{j_\beta} & u_{zi}^{j_\beta} & -1 \\ v_{xi}^{j_\beta} & v_{yi}^{j_\beta} & v_{zi}^{j_\beta} & -1 \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \\ d \end{bmatrix} = \mathbf{A}\cdot\mathbf{x} = \mathbf{0} \qquad (13)$$

$$\text{subject to } \|\mathbf{x}_{1:3}\|^2 = 1,$$

where the constraint is added to reject the trivial solution $\mathbf{x} = \mathbf{0}$. This system is overconstrained, so instead of an exact solution, it yields an approximate one that minimizes the sum of squared errors over all variables under white Gaussian noise added to each end point. The numerically-best way to solve Eq. (13) is to use singular value decomposition on matrix $\mathbf{A}$; this process factors the matrix into a diagonal matrix $\mathbf{D}$ and two orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ (i.e., $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$). Then, the least-squares solution is given by the last column of $\mathbf{V}$, which corresponds to the smallest diagonal entry of $\mathbf{D}$.

Once the hypothesis $\Pi_{j-j_\alpha-j_\beta}$ is generated, we determine its cost as follows:[15]

$$\upsilon = \sum_{i=1}^{N_i} \mu\left(e_i^2\right)$$
$$\mu\left(e_i^2\right) = \begin{cases} e_i^2 & e_i^2 < T^2 \\ T^2 & e_i^2 > T^2 \end{cases}, \qquad (14)$$

where T is the threshold for inliers and $e_i = |\hat{\mathbf{n}}\mathbf{p}_i - d|/\|\hat{\mathbf{n}}\|$, which is simply the distance between the point $\mathbf{p}_i$ and the plane. We generate N_p plane hypotheses, score each hypothesis, and finally choose the best-fitting plane that has the lowest cost. This process is performed iteratively until no new possible plane can be extracted; the algorithm then moves to the next cluster. This initial plane fitting assumes that all the relevant points have equal uncertainty; however, the uncertainty model of actual 3D points may be different according to their position; i.e., the end points of a line segment do not all have the same Gaussian uncertainty. Therefore, we can refine the plane parameters by considering the uncertainty of each scan point to improve the fit (Fig. 7).

4.3.2. Uncertainty modeling of scan points. A raw measurement of a single scan point $\mathbf{p}$ is represented by $(\rho,\ \theta,\ \varphi)$ and converted to 3D Cartesian coordinates by using Eq. (6). Thus, the noise added to

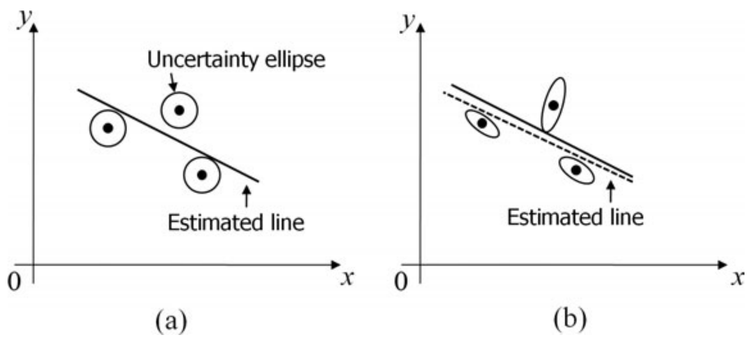

Fig. 7. Examples of line fitting using isotropic and nonisotropic uncertainty. (a) When uncertainty is isotropic, line parameters can be calculated by minimizing the average distance between points and lines. (b) When uncertainty is nonisotropic, appropriate uncertainty modeling of each point is used to update the estimated line (dotted line) toward minimizing the average Mahalanobis distance between points and lines. Similarly, a plane fitting can be improved by considering each point's uncertainty.

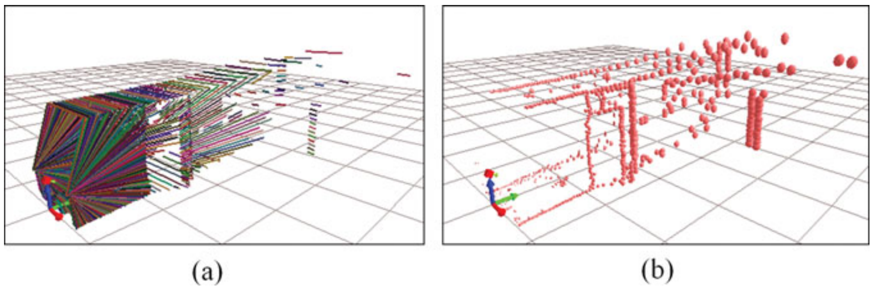

Fig. 8. (a) Line segment extraction in corridor. Each 3D line segment is represented by a different color. (b) Representation of uncertainty of line segments' end points in (a).

the raw measurement $(\rho,\ \theta,\ \varphi)$ is propagated to 3D space according to an error propagation rule[5]

$$\mathbf{C}_p = \mathbf{J}\mathbf{C}^r_{\mathbf{p}}\mathbf{J}^{\mathrm{T}},$$

$$\mathbf{C}^r_{\mathbf{p}} = \begin{bmatrix} \rho^2\sigma^2_\rho & 0 & 0 \\ 0 & \sigma^2_\theta & 0 \\ 0 & 0 & \sigma^2_\varphi \end{bmatrix}, \tag{15}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \rho} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \varphi} \\ \frac{\partial y}{\partial \rho} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \varphi} \\ \frac{\partial z}{\partial \rho} & \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\rho\sin\theta & 0 \\ \sin\theta\sin\varphi & \rho\cos\theta\sin\varphi & \rho\sin\theta\cos\varphi - d_l\sin\varphi \\ \sin\theta\cos\varphi & -\rho\cos\theta\cos\varphi & \rho\sin\theta\sin\varphi + d_l\cos\varphi \end{bmatrix},$$

where $\mathbf{C}^r_{\mathbf{p}}$ is the covariance matrix of the raw measurements. The uncertainty of r is modeled as proportional to the squared distance, and those of q and j are modeled as constant. Because our algorithm uses line segments, we need not calculate the uncertainty of all scan points, but only that of the two end points of line segments (Fig. 8). This greatly reduces the computation time.

4.3.3. Derivation of refined plane fitting. Suppose that we have a target plane $\Pi:\ \hat{\mathbf{n}}\mathbf{p} - d = 0$ and N fitting points $\mathbf{p}_{1:N}$, where each point has a covariance matrix $\mathbf{C}_{\mathbf{p}_i}$. Our goal is to minimize the error function, which represents the average squared Mahalanobis distance

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{p}_i - \mathbf{p}_i^{\Pi_\perp}\right)^{\mathrm{T}}\mathbf{C}^{-1}_{\mathbf{p}_i}\left(\mathbf{p}_i - \mathbf{p}_i^{\Pi_\perp}\right), \tag{16}$$

where $\mathbf{p}_i^{\Pi\perp}$ is the perpendicular foot of $\mathbf{p}_i$ to the plane Π. Equation (16) can be further manipulated so that it is represented by only plane parameters and fitting points (Appendix A),

$$E(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\left[\left((\hat{\mathbf{n}}\mathbf{p}_i - d)\hat{\mathbf{n}}\right)/\|\hat{\mathbf{n}}\|^2\right]^{\mathrm{T}}\mathbf{C}_{\mathbf{p}_i}^{-1}\left[\left((\hat{\mathbf{n}}\mathbf{p}_i - d)\hat{\mathbf{n}}\right)/\|\hat{\mathbf{n}}\|^2\right] = \frac{1}{N}\sum_{i=1}^{N}\bar{\mathbf{n}}_i^{\mathrm{T}}\mathbf{C}_{\mathbf{p}_i}^{-1}\bar{\mathbf{n}}_i$$
$$= \frac{1}{N}\sum_{i=1}^{N}(H_i(\mathbf{x}))^2,\ H_i(\mathbf{x}) = \left(\mathbf{C}_{\mathbf{p}_i}^{-1}\right)^{\frac{1}{2}}\bar{\mathbf{n}}_i, \tag{17}$$

where $\mathbf{x} = [\hat{\mathbf{n}}^{\mathrm{T}}\ d]^{\mathrm{T}}$ is the plane parameter vector. If we assume that $\mathbf{C}_{\mathbf{p}_i}$ is isotropic, where the diagonal term is w_i (which is normally determined as the inverse of the trace of $\mathbf{C}_{\mathbf{p}_i}$) and the off-diagonal is zero, the solution becomes exactly the same as that of the weighted least-square fitting.[5] If $w_i = 1$, the solution becomes the same as that of Eq. (13).

Because the error function (17) is nonlinear in terms of the plane parameters, we use a nonlinear least-square method to iteratively optimize the solution. Here, we can approximate $\mathrm{H}_i(\mathbf{x})$ using the first term of the Taylor expansion

$$H_i(\mathbf{x}+\delta\mathbf{x}) = H_i(\mathbf{x}) + \delta\hat{\mathbf{n}}\frac{\partial H_i}{\partial\hat{\mathbf{n}}}(\mathbf{x}) + \delta d\frac{\partial H_i}{\partial d}(\mathbf{x}) + O\left(|\delta\mathbf{x}|^2\right) \approx H_i(\mathbf{x}) + \nabla H_i(\mathbf{x})\cdot\delta\mathbf{x}, \tag{18}$$

where $\nabla H_i(\mathbf{x}) = [\frac{\partial H_i}{\partial\hat{\mathbf{n}}}\ \frac{\partial H_i}{\partial d}]^T$ is the gradient of H_i calculated at $\mathbf{x}$. $O(|\delta\mathbf{x}|^2)$ is a higher-order term that can be neglected. By an iterative process, the error function $E(\mathbf{x}+\delta\mathbf{x})$ can be minimized with respect to $\delta\mathbf{x}$ given a value $\mathbf{x}$:

$$E(\mathbf{x}+\delta\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{n}(H_i(\mathbf{x}+\delta\mathbf{x}))^2 \approx \frac{1}{N}\sum_{i=1}^{n}(H_i(\mathbf{x}) + \nabla H_i(\mathbf{x})\cdot\delta\mathbf{x})^2 = \frac{1}{N}|\mathbf{J}\delta\mathbf{x} - \boldsymbol{\Theta}|^2, \tag{19}$$

where

$$\mathbf{J} = \left.\begin{pmatrix} \frac{\partial H_i(\mathbf{x}')}{\partial\hat{\mathbf{n}}} & \frac{\partial H_i(\mathbf{x}')}{\partial d} \\ \vdots & \vdots \\ \frac{\partial H_N(\mathbf{x}')}{\partial\hat{\mathbf{n}}} & \frac{\partial H_N(\mathbf{x}')}{\partial d}\end{pmatrix}\right|_{\mathbf{x}'=\mathbf{x}},\quad \boldsymbol{\Theta} = -\left.\begin{pmatrix} H_i(\mathbf{x}') \\ \vdots \\ H_N(\mathbf{x}')\end{pmatrix}\right|_{\mathbf{x}'=\mathbf{x}}$$
$$\frac{\partial H_i(\mathbf{x}')}{\partial\hat{\mathbf{n}}} = \left(\mathbf{C}_{\mathbf{p}_i}^{-1}\right)^{\frac{1}{2}}\left[\left((\mathbf{p}_i\hat{\mathbf{n}}^{\mathrm{T}} + (\hat{\mathbf{n}}\mathbf{p}_i - d)\mathbf{I}\right)\|\hat{\mathbf{n}}\|^2 - 2(\hat{\mathbf{n}}\mathbf{p}_i - d)\hat{\mathbf{n}}\hat{\mathbf{n}}^{\mathrm{T}}\right)/\|\hat{\mathbf{n}}\|^4\right]^{\mathrm{T}},$$
$$\frac{\partial H_i(\mathbf{x}')}{\partial d} = -\left(\mathbf{C}_{\mathbf{p}_i}^{-1}\right)^{\frac{1}{2}}.$$

Therefore, we can minimize Eq. (17) by setting $\mathbf{J}\delta\mathbf{x} = \boldsymbol{\Theta}$, yielding :

$$\delta\mathbf{x} = (\mathbf{J}^{\mathrm{T}}\mathbf{J})^{-1}\mathbf{J}^{\mathrm{T}}\boldsymbol{\Theta} = \mathbf{J}^{*}\boldsymbol{\Theta}, \tag{20}$$

where $\mathbf{J}^{*}$ indicates a pseudoinverse of $\mathbf{J}$. By evaluating Eq. (19) using an iterative gradient descent method (Table VI), we obtain the final plane parameters. The iterative gradient descent method can find a global minimum if an initial guess $\mathbf{x}_0$ is provided within a reasonable region. In our case, this is met by Eq. (13).

4.4. Splitting and merging planes

So far, we have a set of planes $P = \cup_{i=1}^{N_\Pi}\Pi_i = \{\hat{\mathbf{n}}_i, d_i, \lambda_i^{1:N_i}\}$, and they can be refined by being split and merged again (Table VII). First, planes that have average fitting error E_{Π_i} greater than the 2σ-boundary (i.e., $E_{\Pi_i} > 2^2$; Eq. (16)) are split into two or more planes until $E_{\Pi_i} < 4$. Actually, E_{Π_i} of the plane cannot be > 4 at the moment of construction because we reject such hypotheses; however, we later add line segments to the plane gradually, so E_{Π_i} is likely to be >4 at some point.

Table VI. Nonlinear optimization of plane parameters.

```
Nonlinear optimization (J, $\Theta$, e$_{min}$, i$_{max}$)

1: $\mathbf{x}_0$ ←Choose an initial solution
2: while $E(\mathbf{x}_i) > e_{min}$ and $i<i_{max}$
3:     $\delta\mathbf{x}_i = \mathbf{J}^*(\mathbf{x}_i)\mathbf{\Theta}(\mathbf{x}_i)$
4:     $\mathbf{x}_i \leftarrow \mathbf{x}_i = \mathbf{x}_i + \delta\mathbf{x}_i$
5:     $E(\mathbf{x}_{i+1}) = E(\mathbf{x}_i + \delta\mathbf{x}_i) = |\mathbf{J}(\mathbf{x}_i)\delta\mathbf{x}_i - \mathbf{\Theta}(\mathbf{x}_i)|^2$
6:     $i = i+1$
7: endwhile
8: return x
```

Table VII. Split-and-merge process.

```
split_and_merge ($\Pi$, $\Psi$)

1:  for $i$=|$\Pi$| to 1        //split plane
2:    if $E_{\Pi_i}$ is beyond the $2\sigma$-boundary
3:      $R_i = \{\forall\lambda_j \in \Pi_i\}$
4:      $P_i^s$=plane_fitting ($R_i$, Ø)
5:      $L_i^r = \{\forall\lambda_j \notin \Pi_i^s\}$
6:      $\Psi$ = line_clustering ($L_i^r$, $\Psi$)//reclustering of unfitted line segments
7:      $P = \Pi \cup \Pi_i^s - \{\Pi_i\}$
8:  endfor
9:  for $i$=|$\Pi$| to 1        //merge plane
10:   find best matched plane $\Pi_j$ by collaterality and pairwise proximity
11:     if found
12:       $\Pi_i^{new} = \Pi_i \oplus \Pi_j,\ \ \Pi = \Pi \cup \left\{\Pi_i^{new}\right\} - \left\{\Pi_i \cup \Pi_j\right\}$
13: endfor
14: return $\Pi$
```

The plane split is implemented by iterative refitting of line segments, so it is a repetition of the procedure in Section 4.3; the only difference is that the line segments are selected from those in Π_i, rather than from those in the cluster. Once the plane is split into smaller planes, the remaining unfitted line segments originally contained in Π_i are reclustered using the clustering step in Section 4.2.

After the splitting step, plane merge is conducted over two similar planes, Π_i and Π_j, where the similarity is defined in terms of collaterality and pairwise proximity. The collaterality is verified by the inner product of the two planes' normal vectors,

$$\left|\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j\right| > \nu_o, \tag{21}$$

where ν_o is a predefined threshold. The pairwise proximity is based on the distance between the two planes; however, it cannot be defined in practice if they are not exactly parallel. Therefore, in this case, we arbitrarily define the distance as

$$\delta_{\Pi_{ij}} = \max\left[\max\left(\left|\hat{\mathbf{n}}_i \mathbf{p}_{\lambda_j^k} - d_i\right| / \|\hat{\mathbf{n}}_i\|\right),\ \max\left(\left|\hat{\mathbf{n}}_j \mathbf{p}_{\lambda_i^l} - d_j\right| / \left\|\hat{\mathbf{n}}_j\right\|\right)\right],\ \ \forall k, l\,, \tag{22}$$

where $\mathbf{p}_{\lambda_j^k}$ is the end point of the kth line segment contained in Π_j. Thus, the plane pair $\Pi_i - \Pi_j$ that has a minimum distance is then merged.

5. Experiments

Our scanning system was mounted on a mobile platform remotely operated by manual control (Fig. 9). We constructed the INDOOR dataset. The system configuration was set according to the environmental characteristics (Table VIII). We assume that an indoor environment includes many planes; hence, we set the actuator tilt resolution to one quarter of its maximum. We wrote the proposed

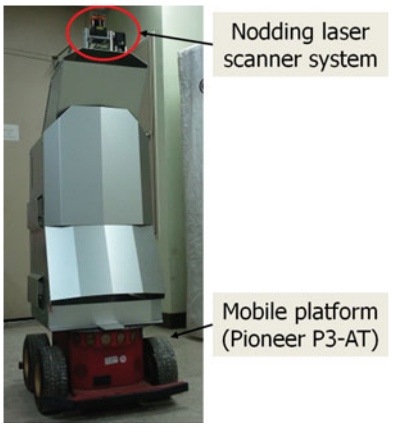

Fig. 9. Nodding laser scanner system mounted on our mobile platform.

Table VIII. Specifications of INDOOR data set.

	INDOOR
Actuator tilt angle	$160°$
Tilt resolution	$1.16°$
Laser scan angle	$0\sim180°$
Scan resolution	$0.25°$
# Scan points per scan	98,777
Complete scanning time (ms)	3.4
# Scans	70
Memory used (MB)	31

algorithm using Microsoft visual C++ 2008 and ran it on three computer systems: (A) desktop with 2.66-GHz clock speed (quad core) and 3 GB RAM, (B) laptop with 2.2-GHz (dual core) clock speed and 2 GB RAM, and (C) laptop with 2.53-GHz clock speed (single core) and 3 GB RAM.

5.1. *Application to indoor environments*

5.1.1. Representation of extracted planes. We collected the INDOOR dataset in the LG research building at POSTECH (Appendix B). This building is a typical building with planar floors, ceilings, and walls. Among 66 scans we selected four scans, acquired at different positions, based on their representativeness of indoor structure. In a typical corridor environment (Figs. 10–12), four main planes were detected: floor, ceiling, and side walls. Near doors, we also extracted minor planes consisting of very short line segments. Each environment included an unfitted line cluster that includes line segments that could represent a plane, but this plane was not supported by a sufficient number of line segments. If these very minor plane candidates must still be captured, we can adjust the minimum number of line segments for supporting a plane, but this adjustment may result in erroneous plane extraction. The user settings determine whether or not only very salient planes are extracted. In Fig. 12, two separate planes appear on the right; these seem to indicate erroneous plane segmentation. In fact, the surface described is one that curves gently, so our algorithm automatically divided it into two planar surfaces because a 2D curve can be represented by piecewise line segments and so a continuous surface can be represented by piecewise planes. A plane extracted from such a curved surface shows a rather large fitting error compared to that of a planar surface, but this error is still within the $2s$-boundary, which is a reasonable error threshold. In Fig. 13, the workspace is approximately 20 m $\times$ 25 m, and we could even extract planes that were far from the laser scanner.

5.1.2. Data access ratio. To measure the degree of data reduction, we defined a data access ratio

$$\mathrm{DAR}(\%) = \frac{\#\,\text{accessed scan points}}{\#\,\text{total scan points}} \times 100 = \frac{2 \times \#\,\text{line segments}}{\#\,\text{total scan points}} \times 100. \qquad (23)$$

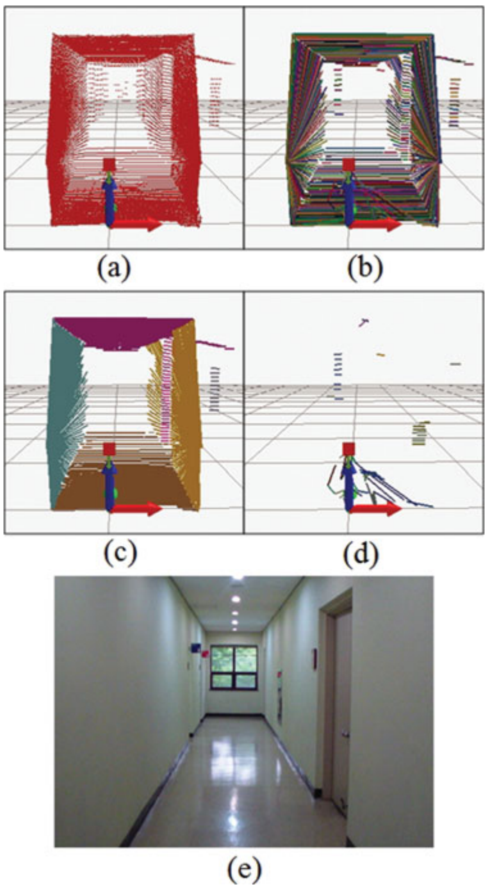

Fig. 10. (a) Raw 3D point cloud. (b) Extracted line segments. (c) Plane extraction result. (d) Unfitted line segments. Number of line segments in (b) equals the sum of those in (c) and (d). Different planes are represented by different colors; grid is 2 m × 2 m. (e) Real environment.

Table IX. Statistics for INDOOR dataset.

	Scan			
Measure	5	13	39	63
# Total lines	652	920	1,000	1,941
# Fitted lines	589	847	910	1,652
# Unfitted lines	63	73	90	289
# S/M[a] process	42	56	21	0
# Planes	7	10	12	37
Avg. fitting error	0.864	0.685	0.543	0.387
DAR (%)	1.32	1.86	2.02	3.93
T_{proc}(A)[b] (ms)	171.6	202.3	240.7	378.2
T_{proc}(B) (ms)	280.6	340.2	380.45	621.2
T_{proc}(C) (ms)	735.0	932.8	995.0	1,629.8

[a]S/M: Split-and-merge; [b]T_{proc}(A): Processing time on system A.

Huge numbers of scan points are acquired during full 3D scanning, but the algorithm discards most of them to reduce the processing time. In our algorithm, only a small fraction of scan points is repeatedly accessed to fit the plane. In all experiments, DAR $< 5\%$ for the INDOOR dataset (Table IX). We also measured the processing time on the three computer systems described above. The algorithm running time never exceeds the actuator moving time or the data acquisition time (i.e., 3.4 s); thus, the proposed method is suitable for an online real-time system. The processing time was

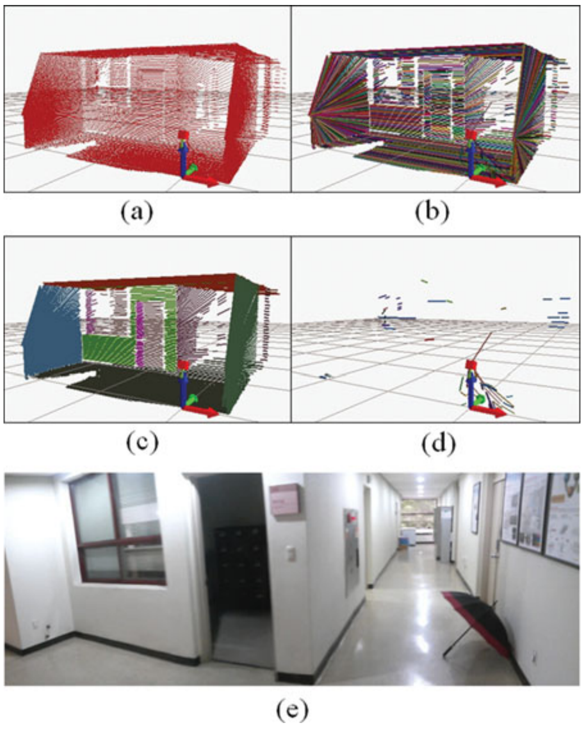

Fig. 11. Descriptions are the same as in Fig. 10.

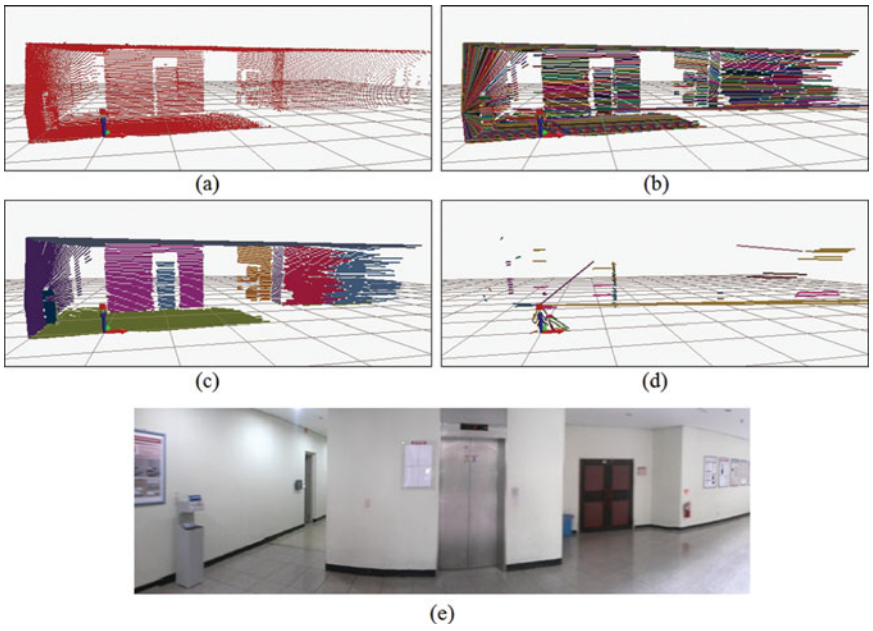

Fig. 12. Descriptions are the same as in Fig. 10.

linearly proportional to the number of line segments irrespective of the test computer used (Fig. 14); the only difference was in the slope, which is related to the system's parallel processing power. This linear characteristic will be further discussed in Section 5.2.2.

5.1.3. Stepwise analysis. We also examined four statistics on a step-by-step basis. The number of fitted lines showed a continual increase owing to sequential line acquisition (Fig. 15a); the stair-shaped graph occurred due to the periodical plane extraction process (i.e., one per five scan steps). The number of unfitted lines showed peaks every five steps due to repetition of data acquisition and plane extraction (Fig. 15b): the rising edge corresponds to continuous data acquisition, whereas the

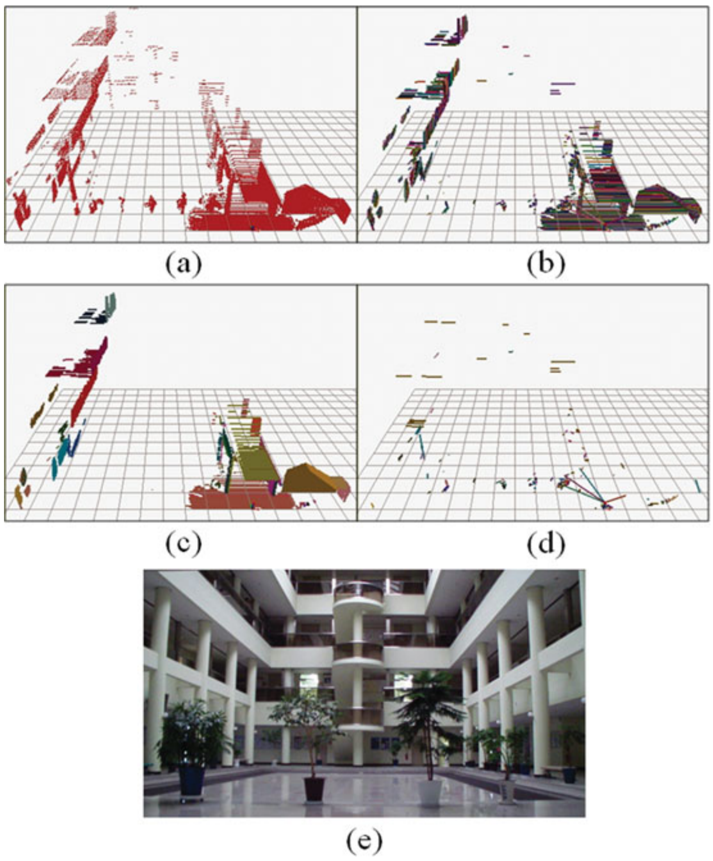

Fig. 13. Descriptions are the same as in Fig. 10.

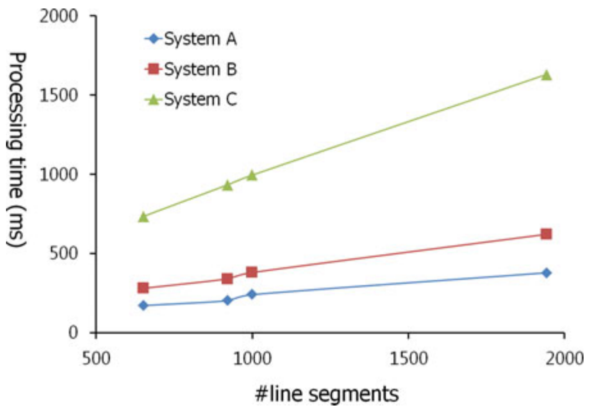

Fig. 14. Processing time on computer systems A, B, C (described in the text) was linearly proportional to the number of line segments. Slopes and intercepts decrease as the parallel processing power increases.

falling edge indicates abrupt execution of plane extraction. The number of planes extracted increased overall, but decreased occasionally as a result of plane merging (Fig. 15c). The average fitting error eventually decreased to within the 2σ-boundary although it temporarily increased to 3.2, because the split-and-merge process was triggered (Fig. 15d). As a result, planes having a fitting error >2 were split and then reorganized into small planes with lower fitting errors.

5.2. *Time and complexity analysis*

5.2.1. Time analysis. To measure the total processing time, several sets of scan data were acquired from the same position but under different data acquisition conditions (Table X). All of the following

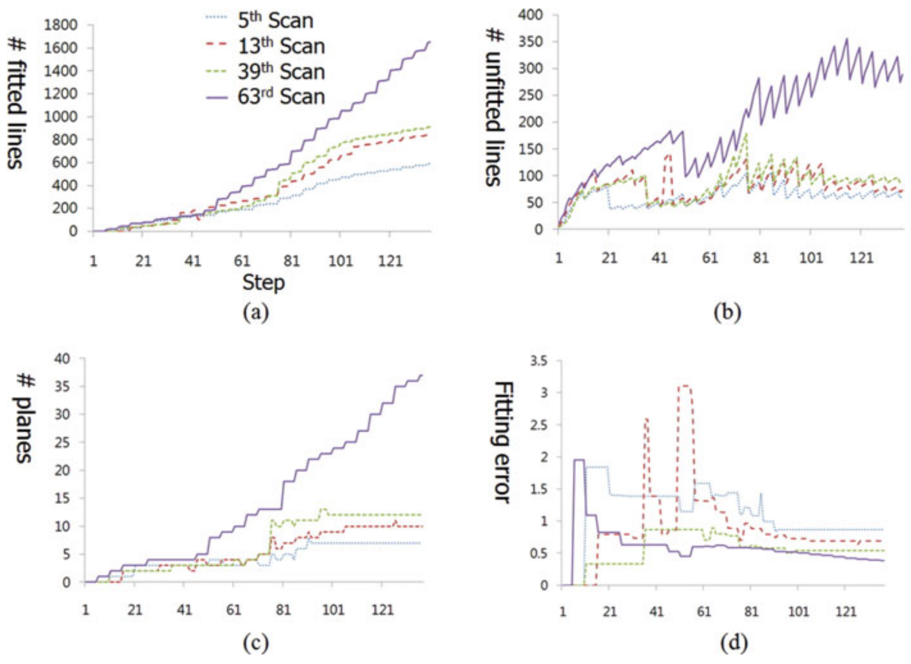

Fig. 15. (a) Number of fitted line segments versus tilt step. (b) Number of unfitted line segments. (c) Number of planes extracted. (d) Fitting error (average Mahalanobis distance between points and plane).

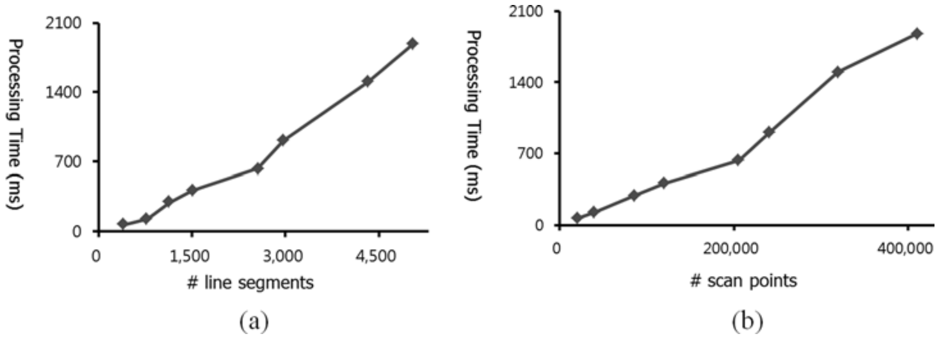

Fig. 16. Total processing time versus number of (a) line segments and (b) scan points.

Table X. Data acquisition conditions.

	1	2	3	4	5	6	7	8
Tilt angle	130	130	140	160	150	160	130	150
Tilt resolution	2.32	1.16	1.16	1.16	0.58	0.58	0.29	0.29
Scan angle	0∼180	0∼180	0∼180	−20∼200	−10∼190	−20∼200	0∼180	−10∼190
Scan resolution	0.5	0.5	0.25	0.25	0.25	0.25	0.25	0.25
# scan points	20,216	40,071	86,520	120,697	205,857	241,394	320,214	410,913
# Total lines	397	779	1,139	1,508	2,558	2,972	4,320	5,045
DAR (%)	3.93	3.89	2.63	2.50	2.49	2.46	2.69	2.46
Processing time (ms)	68.9	127.4	293.9	411.7	630.1	914.6	1502.2	1883.6

experiments were performed on computer A, and we considered only the pure algorithm running time excluding the data acquisition time. The processing time increased linearly with the number L of line segments; therefore our method has complexity of $O(L)$ (Fig. 16a). Once a plane is extracted from a line cluster, we rarely need to access the constituent lines again except during the split-and-merge process; therefore, we must process only newly-acquired lines. The processing time was also linearly proportional to the number of scan points (Fig. 16b), but this is not always true; this is easily validated

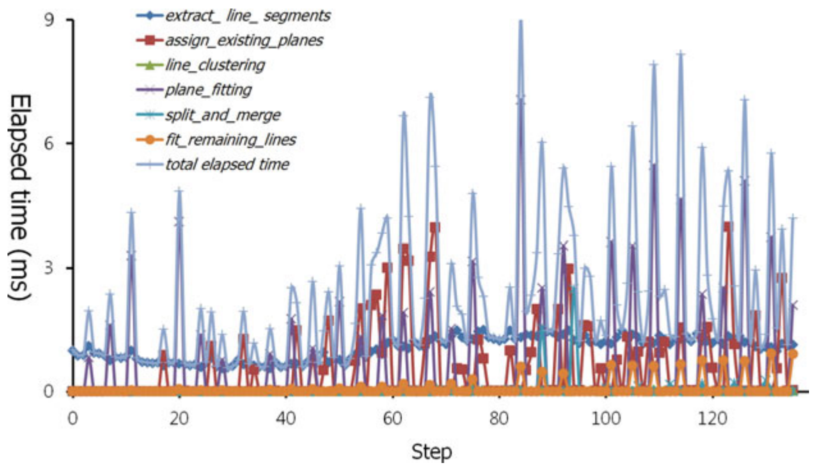

Fig. 17. Elapsed time of each phase versus scan step.

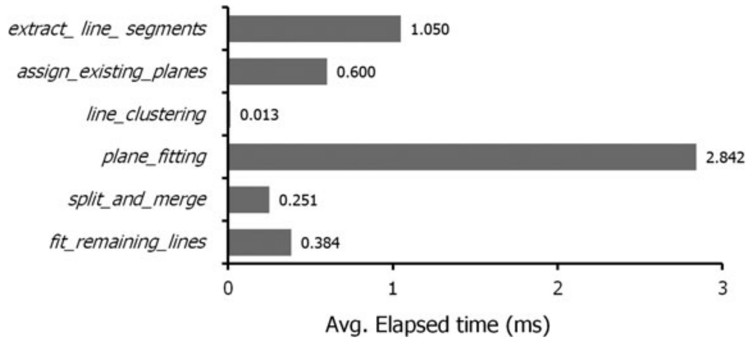

Fig. 18. Average elapsed time of each phase.

by inspecting Fig. 14, in which the sets of scan data all have the same number of points (98,777) but different numbers of line segments. Usually the number of line segments depends on how many planes exist in the scanned environment.

We also measured the elapsed time of each phase versus the scan steps using the 63rd set of scan data (Fig. 17). The *plane_fitting* and *assign_existing_planes* steps took most of the time. The *plane_fitting* process is iterative, so it naturally requires more time than do other phases; *assign_existing_planes* also took considerable time because it contains *plane_fitting* as an internal process (*plane_fitting* is triggered whenever 10 new line segments are added to the existing plane). The *extract_line_segment* process defines the line segment properties (i.e., covariance matrix calculation of each end point and direction vector), which requires the inverse of the covariance matrix. If we consider only pure line extraction, the time is reduced to approximately 0.5 ms. Next, we measured the average elapsed time of each phase (Fig. 18). *Plane_fitting*, *split_and_merge*, and *fit_remaining_lines* are occasional phases, so the average elapsed time was calculated only when these phases were performed. The maximum total elapsed time never exceeded 10 ms, which is less than the scanning time (25 ms in typical settings), so our system could run without time delay.

5.2.2. Complexity analysis. Our algorithm is incremental and based on line segments. Thus, the complexity of each phase can be represented by the number L_k of line segments at step k (Table XI). The only exception is that the complexity of *extract_line_segments*, which uses the split-and-merge scheme in conjunction with IEPF, was represented by the number of scan points n_k, because this phase is based on raw 3D points. After this phase, all the complexities are represented by L_k, C_k, P_k, $c_{k,l}$, and $p_{k,l}$, where C_k and P_k are the numbers of line clusters and extracted planes up to step k, and $c_{k,l}$ and $p_{k,l}$ are the numbers of line segments in certain line clusters and planes. We can further approximate the complexity of each phase by assuming that an arbitrary limited environment has a finite set of planes, which is true for indoor environments. Therefore, for all k, the following

Table XI. Analysis of complexity of the proposed method.

	Complexity	Approximation
extract_line_segments	$O(n_k \log n_k)$	$O(\bar{n} \log \bar{n})$
assign_existing_planes	$O(L_k P_k)$	$O(L_k \bar{P})$
line_clustering	$O(L_k C_k)$	$O(L_k \bar{C})$
plane_fitting	$O(c_{k,l} C_k)$	$O(\bar{c}\bar{C})$
split_and_merge	$O(p_{k,l} P_k + (P_k)^2)$	$O(\bar{p}\bar{P} + \bar{P}^2)$
fit_remaining_lines	$O(c_{k,l} C_k P_k)$	$O(\bar{c}\bar{C}\bar{P})$
Total	N/A	$O(\bar{L}_T)$

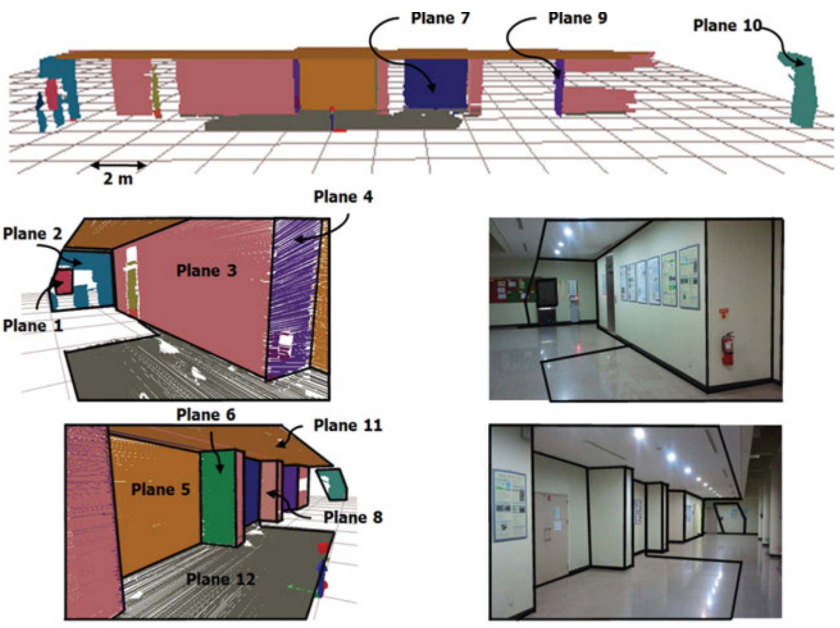

Fig. 19. Planes used for accuracy test.

inequalities hold: $C_k \leq \bar{C}$, $P_k \leq \bar{P}$, $c_{k,l} \leq \bar{c}$, $p_{k,l} \leq \bar{p}$. The number of scan points n_k at step k is also bounded by $\bar{n}$. The total complexity can be calculated by summing all the complexities, resulting in $O(U \sum_{k=1}^{N} L_k)$, where U denotes a constant term resulting from the summing of all unnecessary upper bounds, and N is the total number of tilt steps. By the same assumption, we can set the upper bound of L_k as $\bar{L}$; thus, the total complexity reduces to $O(UN\bar{L}) \cong O(N\bar{L}) \cong O(\bar{L}_T)$, where $\bar{L}_T$ is the total number of line segments. This linear complexity is demonstrated in Figs. 14 and 16a, which show a linear increase in processing time with increasing $\bar{L}_T$.

5.3. *Comparison test*

To evaluate our approach, we performed comparison tests with three methods. The first two are based on Progressive Sample Consensus (PROSAC) which is a RANSAC-like algorithm but is faster than RANSAC. The method PROSAC_Only begins with voxel grid filtering to reduce the number of fitting points followed by iteratively extracting the best-supported planes without normal vector information. In contrast, PROSAC_Normal, exploits normal vector information to cluster point clouds first, then extracts planes at each cluster. These two methods were implemented with the Point Cloud Library.[14] The other state-of-the-art method is RHT-based plane extraction which uses a novel accumulator design,[23] where it was compared with another state-of-the-art region growing method.[24]

We compared the accuracy and computation time of the four methods. To test accuracy, we measured ground truth parameters ($\mathbf{n}$, d) of 12 selected planes (Fig. 19, Table XII). The errors of

Table XII. Ground truth and estimated parameters of each plane.

	Ground truth		Proposed	
Plane	**n**	*d*	**n**	*d*
1	$-1, 0, 0$	12.6	$-1.000, -0.0017, -0.0029$	12.601
2	$-1, 0, 0$	12.28	$-0.9989, -0.0442, -0.0062$	12.273
3	$0, 1, 0$	2.58	$0.0059, 0.9994, 0.03337$	2.561
4	$-1, 0, 0$	2.11	$-0.9997, 0.0143, -0.0148$	2.120
5	$0, 1, 0$	3.69	$0.0040, 0.9994, 0.0327$	3.700
6	$1, 0, 0$	1.83	$0.9997, 0.0248, -0.0033$	1.899
7	$0, 1, 0$	3.37	$0.0006, 0.9988, 0.0477$	3.399
8	$1, 0, 0$	6.34	$0.9993, 0.0366, 0.0069$	6.444
9	$1, 0, 0$	10.79	$0.9997, 0.0235, 0.0025$	10.842
10	$1, 0, 0$	21.28	$1.0000, 0.0042, -0.0020$	21.297
11	$0, 0, 1$	2	$0.0009, -0.0354, 0.9993$	1.992
12	$0, 0, -1$	0.9	$0.0016, 0.0443, -0.9989$	0.860

Table XIII. Comparative results.

	PROSAC_Only		PROSAC_Normal		RHT		Proposed	
	e^n	e^d	e^n	e^d	e^n	e^d	e^n	e^d
Avg.	0.017986	0.018682	0.015565	0.021703	0.019381	0.03152	0.02176	0.032875
Std.	0.013198	0.022496	0.012123	0.023857	0.009114	0.033766	0.011078	0.029487
Max	0.059338	0.094771	0.062828	0.089116	0.039752	0.109042	0.04879	0.12192
Min	0.001414	0.000227	0	0.002799	0.001414	0.000532	0	0.00116

plane parameters are defined as:

$$e_{ij}^{\mathbf{n}} = cos^{-1}\left(\frac{\mathbf{n}_i^g \cdot \mathbf{n}_{ij}}{\left\|\mathbf{n}_i^g\right\| \cdot \left\|\mathbf{n}_{ij}\right\|}\right), \quad e_{ij}^d = \left|d_i^g - d_{ij}\right|, \tag{24}$$

where $\mathbf{n}_i^g$ is the ground truth normal vector and d_i^g is the distance of the ith plane in the jth experiment. PROSAC_Normal was the most accurate because normal vector information of all fitting points is used to extract the planes; however, due to calculation of normal vectors, its computation time was longer than the proposed and PROSAC_Only method (Fig. 20, Table XIII). Although RHT showed faster computation than the proposed method on 10 of the 66 tests, it required several seconds to complete the analysis in several cases. This implies that speed of RHT can be affected by the structures of the environments. Overall, the errors of plane parameters were slightly larger than those of PROSAC_Normal; however, this error increase can be seen as a trade-off between accuracy and speed. The average computation time of the proposed method was 391 ms, so it was at least three times faster than the other methods.

6. Conclusions

In this paper, we presented an online incremental method of extracting planes from 3D point clouds acquired sequentially by a nodding laser scanner. To fully exploit the data acquisition characteristics, our algorithm uses line segments as supporting elements of a plane. Line segments were extracted sequentially from every scan slice, then clustered according to their 3D orientation. Plane extraction was performed in each cluster using nonlinear least-square fitting; MSAC provided the initial plane parameters to reduce the number of fitting iterations. Because the line segments were defined by only two end points, we could reduce the number of scan points accessed to fit the plane to just twice the number of line segments, and hence reduce the plane extraction time. During data acquisition, planes were periodically split and merged, yielding iterative refinement of the plane parameters. Experiments were conducted using a dataset acquired from a typical indoor environment; average

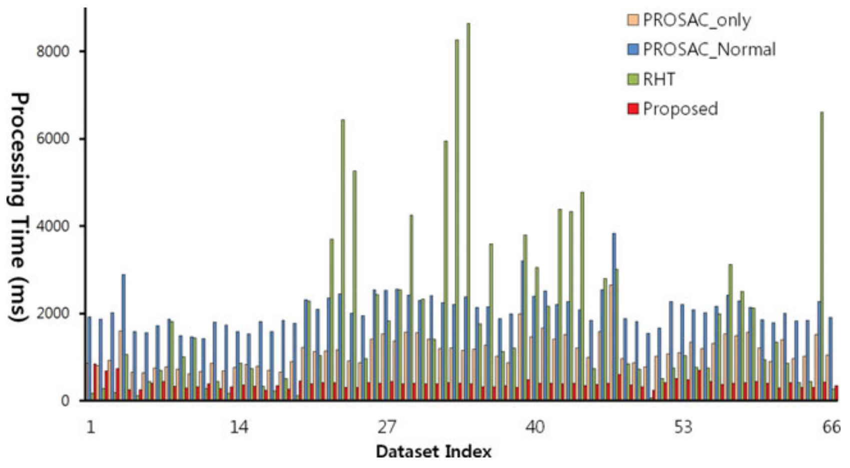

Fig. 20. The comparison of time required to extract planes using three methods and the proposed method.

time required to process ∼100,000 scan points was <400 ms. The complexity of our algorithm was linear to the total number of line segments. The potential advantages of our algorithm, in addition to the short processing time, are (i) its scalability to the size of the environment and (ii) ability to model a gently curved surface as piecewise planes. First, complexity analysis proves that our algorithm does not depend on the environment size. Second, a curve can be modeled as piecewise line segments during line extraction, and likewise a surface can also be modeled as piecewise planes consisting of line segments. By using the proposed method with simple plane matching, we could also construct a planar map of an entire indoor environment in which planes are the dominant boundaries (Appendix B).

Acknowledgements
This work was supported in part by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the “IT Consilience Creative Program” (NIPA-2013-H0203-13-1001) supervised by the NIPA (National IT Industry Promotion Agency) and the MKE, Korea, under the “National Robotics Research Center” support program supervised by the NIPA (NIPA-2012-H1502-13-1001).

References

1. G. Taylor and L. Kleeman, “Robust Range Data Segmentation Using Geometric Primitives for Robotic Applications,” *Proceedings of the IASTED International Conference on Signal and Image Processing* (2003) pp. 467–472.
2. J. W. Weingarten and R. Siegwart, “3D SLAM using Planar Segments,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2006) pp. 3062–3067.
3. G. M. Hedge and C. Ye, “Extraction of Planar Features from SwissRanger SR-3000 Range Images by a Clustering Method Using Normalized Cuts,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2009) pp. 4034–4039.
4. J. W. Weingarten, G. Gruener and R. Siegwart, “A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Recognition,” In *Proceedings of the International Conference on Advanced Robotics (ICAR)* (2003).
5. J. W. Weingarten, G. Gruener and R. Siegwart, “Probabilistic Plane Fitting in 3D and an Application to Robotic Mapping,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2004) pp. 927–932.
6. H. Surmann, K. Lingemann, A. Nuchter and J. Hertzberg, “A 3D Laser Range Finder for Autonomous Mobile Robots,” *Proceedings of the 32nd International Symposia. on Robotics (ISR)* (2001) pp. 153–158.
7. O. Wulf and B. Wagner, “Fast 3D Scanning Methods for Laser Measurement Systems,” *Proceedings of the International Conference on Control Systems and Computer Science* (2003) pp. 312–317.

8. A. Roennau, G. Liebel, T. Schamm, T. Kersher and R. Dillmann, “Robust 3D Scan Segmentation for Teleoperation Tasks in Areas Contaminated by Radiation,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2010) pp. 2419–2424.
9. R. B. Rusu, N. Blodow, Z. C. Marton and M. Beetz, “Close-Range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2009) pp. 1–6.
10. D. Viejo and M. Cazorla, “3D Plane-Based Egomotion for SLAM on Semi-Structured Environment,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2007) pp. 2761–2766.
11. S.-Y. An, L.-K. Lee and S.-Y. Oh, “Fast Incremental 3D Plane Extraction from a Collection of 2D Line Segments for 3D Mapping,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2012) pp. 4530–4537.
12. R. B. Rusu, I. A. Sucan, B. Gerkey, S. Chitta, M. Beetz and L. E. Kavraki, “Real-Time Perception-Guided Motion Planning for a Personal Robot,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2009) pp. 4245–4252.
13. C. Ye and J. Borenstein, “Characterization of a 2D Laser Scanner for Mobile Robot Obstacle Negotiation,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2002) pp. 2512–2518.
14. R. B. Rusu and S. Cousins, “3D is Here: Point Cloud Library (PCL),” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (2011) pp. 1–4.
15. P. H. S. Torr and A. Zisserman, “MLESAC: A new robust estimator with application to estimating image geometry,” *Comput. Vis. Image Underst.* **78**(1), 138–156 (2000).
16. A. F. Martin and C. B. Robert, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM* **24**(6), 381–395 (1981).
17. A. Nuchter and J. Hertzberg, “Towards semantic maps for mobile robots,” *Robot. Auton. Syst.* **56**(11), 915–926 (2008).
18. J.-E. Deschaud, “A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing,” *Presented at the International Symposia on 3D Data Processing, Visualization, and Transmission (3DPVT)* (2010).
19. M. Y. Yang and W. Forstner, *Plane Detection in Point Cloud Data*,Technical Report, TR-IGG-P-2010-01 (Department of Photogrammetry, Institute of Geodesy and Geoinformation, University of Bonn, 2010).
20. Y.-H. Tseng and M. Wang, “Automatic Plane Extraction from LIDAR Data Based on Octree Splitting and Merging Segmentation,” *Proceedings of the IEEE International Geoscience and Remote Sensing Symposia* (2005) pp. 3281–3284.
21. D. Hahnel, W. Burgard and S. Thrun, “Learning Compact 3D Models of Indoor Environments with a Mobile Robot,” *Proceedings of the Fourth European Workshop on Advanced Mobile Robots (EUROBOT)* (2001) pp. 91–98.
22. Y. Liu, R. Emery, D. Chakrabarti, W. Burgard and S. Thrun, “Using EM to Learn 3D Models of Indoor Environments with Mobile Robots,” *Proceedings of the IEEE International Conference on Machine Learning (ICML)* (2001) pp. 329–336.
23. D. Borrmann, J. Elseberg, K. Lingemann and A. Nuchter, “The 3D Hough transform for plane detection in point cloud: A review and a new accumulator design,” *3D Res. Express* **2**(2), 1–13 (2011).
24. J. Poppinga, N. Vaskevicius, A. Birk and K. Pathak, “Fast Plane Detection and Polygonalization in Noisy 3D Range Images,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2008) pp. 3378–3383.
25. K. Georgiev, R. T. Creed and R. Lakaemper, “Fast Plane Extraction in 3D Range Data Based on Line Segments,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2011) pp. 3808–3815.
26. Robotis company, [Online] Available http://www.robotis.com/eng.
27. A. Desai and D. Huber, “Objective Evaluation of Scanning Ladar Configurations for Mobile Robots,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2009) pp. 2182–2189.
28. V. Nguyen, S. Gachter, A. Martinelli, N. Tomatis and R. Siegwart, “A comparison of line extraction algorithms using 2D range data for indoor mobile robotics,” *Auton. Robot.* **23**(2), 97–111 (2007).
29. G. A. Borges and M.-J. Aldon, “Line extraction in 2D range images for mobile robotics,” *J. Intell. Robot. Syst.* **40**(3), 267–297 (2004).
30. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (John Wiley & Sons, New York, 1973).
31. S.-Y. An, J.-G. Kang, L.-K. Lee and S.-Y Oh, “SLAM with Salient Line Feature Extraction in Indoor Environments,” *Proceedings of the 11th International Conference on Control, Automation, Robotics, and Vision* (2010) pp. 410–416.

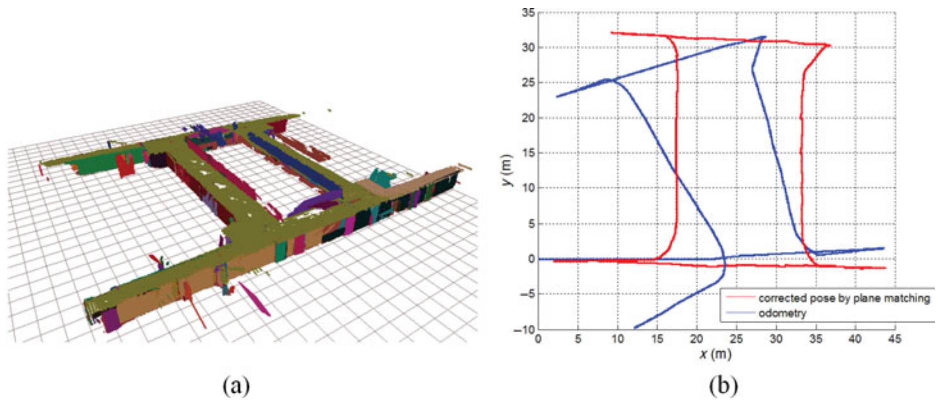

Fig. 21. (a) Constructed 3D map. (b) Trajectory of raw odometry and corrected robot pose by plane matching.

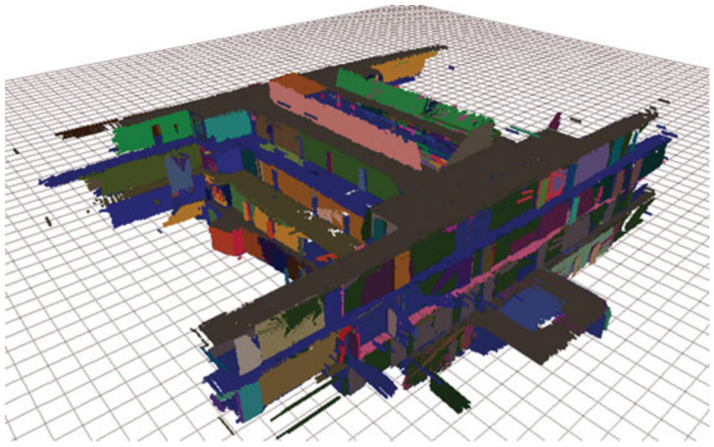

Fig. 22. Integrated 3D map of LG research center in POSTECH.

Appendix A

Perpendicular foot of a point to the plane

Denote the perpendicular foot of p_i to the plane $\Pi : \hat{\mathbf{n}} p - d = 0$ by $p_i^{\Pi_\perp}$. A vector formed by $p_i - p_i^{\Pi_\perp}$ is parallel with the normal vector $\hat{\mathbf{n}}$ of Π, and $p_i^{\Pi_\perp}$ is on the plane, so

$$i)\, p_i - p_i^{\Pi_\perp} = k\hat{\mathbf{n}}, \tag{A.1}$$

$$ii)\, \hat{\mathbf{n}} p_i^{\Pi_\perp} - d = 0. \tag{A.2}$$

(A.1) implies $p_i^{\Pi_\perp} = p_i - k\hat{\mathbf{n}}$; we substitute appropriately in (A.2). Some manipulation yields $k = (\hat{\mathbf{n}} p_i - d)/\|\hat{\mathbf{n}}\|^2$; substituting appropriately into (A.1) yields $p_i - p_i^{\Pi_\perp} = (\hat{\mathbf{n}} p_i - d)\,\hat{\mathbf{n}}/\|\hat{\mathbf{n}}\|^2$.

Appendix B

The LG research center is a four-story building composed of many orthogonal planes and some gently curved surfaces. To map the whole building, we first extract planes at every floor using the proposed method and a "stop-scan-go" procedure.[2] Whenever the robot stops and scans the environment, the extracted planes are matched incrementally with the already-built map. The matching of planes is based on the plane parameters and raw odometry (Fig. 21). In this way, we can construct the map of each floor separately, and then manually merge all the four maps (Fig. 22). The constructed map is not exactly the same as the real environment, but we can roughly model indoor building without using complicated SLAM algorithms.