Intel Serv Robotics (2018) 11:1–24
https://doi.org/10.1007/s11370-017-0234-9

ORIGINAL RESEARCH PAPER

A practical 2D/3D SLAM using directional patterns of an indoor structure

Keonyong Lee[1] · **Soo-Hyun Ryu**[1] · **Changjoo Nam**[2] · **Nakju Lett Doh**[1,3]

Received: 27 August 2016 / Accepted: 3 July 2017 / Published online: 17 August 2017

Abstract This paper presents a practical two-dimensional (2D)/three-dimensional (3D) simultaneous localization and mapping (SLAM) algorithm using directional features for ordinary indoor environments; this algorithm is adaptable to various conditions, computationally inexpensive, and accurate enough to use for practical applications. The proposed algorithm uses odometry acquired from other sensors or other algorithms as the initial estimate and the directional features of indoor structures as landmarks. The directional features can only correct the rotation error of the odometry. However, we show that the greater part of the translation error of the odometry can also be corrected when the directional features are detected at almost positions accurately. In that case, there is no need to use other kinds of features to correct translation error. The directions of indoor structures have two advantages as landmarks. First, the extraction of them is not affected by obstacles. Second, the number of them is small regardless of the size of the building. Because of these advantages, the proposed SLAM algorithm shows robustness for parameters and lightweight properties. From extensive experiments with 2D/3D datasets taken from different buildings, we show the practicality of the proposed algorithm. We also demonstrate that the 2D algorithm runs in real time on a low-end smartphone.

Keywords Directional feature · Indoor environments · Kalman filters · Lightweight algorithm · Practical algorithm · Simultaneous localization and mapping (SLAM)

1 Introduction

Polaris (or polar star) is a good landmark to find a direction when someone lost his ways. There are three reasons why polaris is a good landmark. First, it is easily distinguishable among other stars. Second, there is only one landmark in a large space. Third, polaris is easily detectable because the star can be seen at everywhere. In ordinary indoor environments, there is a landmark like polaris. That is a directional pattern of an indoor structure. First, the directional pattern is also easily distinguishable. Second, there is a few number of patterns in a large building. Finally, the directional feature can be detected at almost any places in a building. In this paper, we propose a practical algorithm using the advantages of directional features for localization or environment mapping in ordinary indoor environments. The basic approach of the proposed algorithm is to use the odometry acquired from other sensor such as wheel encoder of a mobile robot or other algorithm such as registration as the initial estimate and the directional features extracted from a structure as landmarks.

The directional features extracted from the structure have distinguishable patterns, as shown in Fig. 1. These directional patterns can be used as landmarks. We term this directional landmark as *DLmark* (Direction Landmark). The DLmarks of an indoor space can be detected from linear features such as lines and planes. There are two types of directions extracted from linear features: directions of a structure and directions

Electronic supplementary material The online version of this article (doi:10.1007/s11370-017-0234-9) contains supplementary material, which is available to authorized users.

✉ Nakju Lett Doh
nakju@korea.ac.kr

[1] School of Electrical Engineering at Korea University, Seoul, Korea

[2] The Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, USA

[3] CEO, TeeVR Inc., Seoul, South Korea

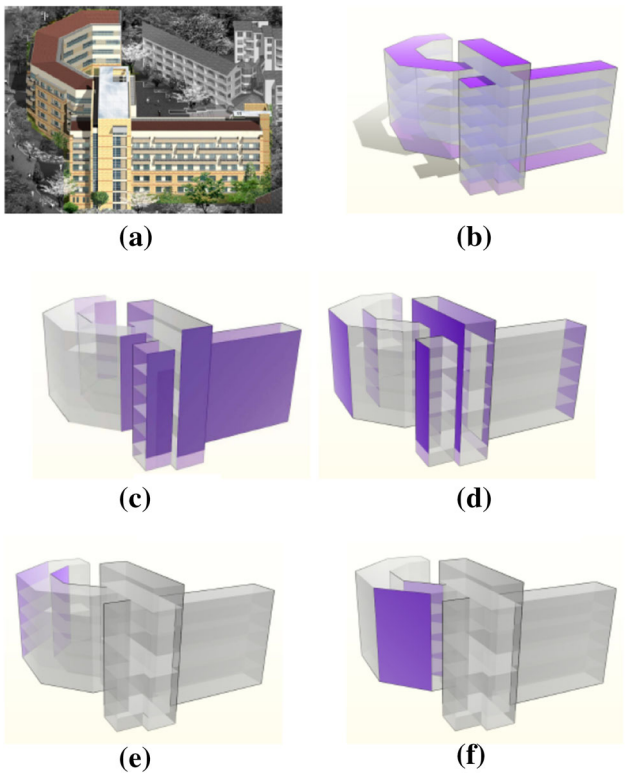


Fig. 1 Directional patterns in a building. Generally, a building is composed of a few linear structures with different directions, which are easily distinguishable. Therefore, each direction can be used as a landmark. The building in **a** has five sparse directions as shown in **b**–**f**, which are extracted from the structure of the building

of a non-structure such as interior decorations on the walls and small objects. Although the number of linear features corresponding to a structure or a non-structure depends on the environment, the total length/area of the line/plane segments from a structure is longer/larger than that from a non-structure in general. This is a characteristic of an indoor structure. We propose a DLmark detection algorithm that uses this characteristic called the *split-and-filter* algorithm. Additionally, we term the proposed SLAM algorithm as *DL-SLAM* (Direction Landmark-based SLAM).

The directional feature has been used in previous research studies. We can divide these studies into two parts. The first part involved researching a method that can detect the directional feature. In the case of vision sensor, the directional features are extracted in the form of a *vanishing point* (VP), and research studies on how to detect a VP robustly have been suggested [1–3]. On the other hand, there has been no research study yet on how to extract the directional features from LIDAR depth data. In previous research studies using LIDAR [4,5], the directional features were detected not from the result of direction extraction but from that of line extraction. The previous methods have a limitation in that the directional feature detection is dependent on the performance of the line extraction. The second part involved researching for ways to utilize the extracted directional features. There are two methods to utilize them. One is to estimate the relative rotation between adjacent poses [6–13]. This method has a limitation in that it accumulates the local rotational error. The other is to use the extracted directional features as landmarks [5,14–17]. This method can keep the global consistency of the rotation of a robot. However, the directional features are not used as the main feature, but as a subfeature. It is natural to use additional features to correct the translation error because the directional feature can only correct the rotation error. However, a large part of the global translation error is actually due to the global rotation error, and this fact has not been analyzed in previous research studies yet.

In this paper, we propose a direction extraction algorithm using LIDAR. Moreover, we show that it is possible to keep the accuracy of the global translation by using only the directional feature when the wheel-based odometry is given.

2 DLmark detection

In this section, we explain a DLmark detection algorithm, which is called the *split-and-filter*, and how the proposed method can achieve high detection rate and high accuracy without parameter tuning in various indoor environments. In the beginning, let me explain how to represent a DLmark in the following section.

2.1 Representations of a DLmark

From mathematical point of view, a directional feature is one-dimensional subspace in n-dimensional vector space. Therefore, the directional feature can be represented by an arbitrary basis in one-dimensional subspace. General choice is a unit vector. In one-dimensional subspace, there are two unit vectors, and an arbitrary unit vector can be used. Table 1 shows the characteristic of a directional feature in 2D and 3D, respectively. In 2D case, the degree of freedom (DoF) is one. Therefore, the 2D unit vector can be changed to angle parameter as follows:

$$\begin{bmatrix} a_x & a_y \end{bmatrix}^T \Longleftrightarrow \theta = \arctan\left(\frac{a_y}{a_x}\right), \quad \theta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{1}$$

In 3D case, the DoF is two. Therefore, a 3D unique direction can be represented by using two parameters as shown in

Table 1 Characteristic of directional feature in 2D & 3D

Space	2D	3D
Dimension of a directional feature	1	1
DoF of a directional feature	1	2

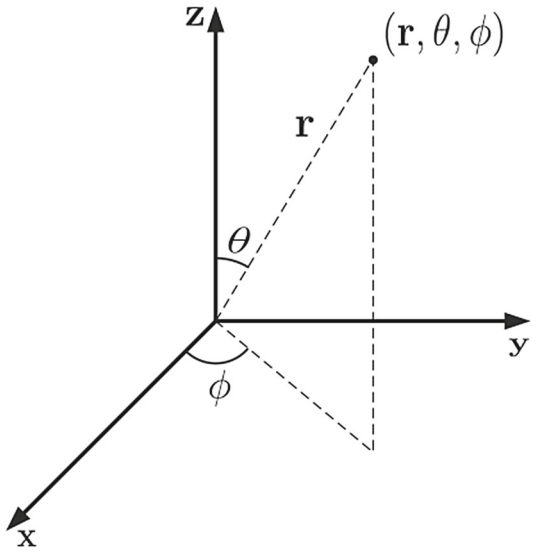


Fig. 2 A 3D unit vector can be represented by using two parameters (θ, ϕ) in spherical coordinate

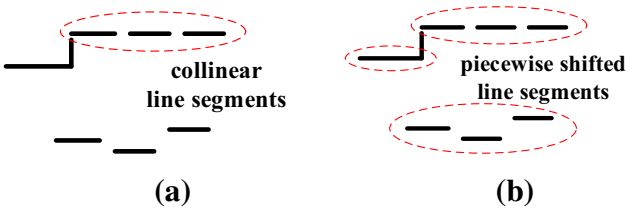


Fig. 3 **a** Collinear line segments are changed to one line segment by the merge process. **b** Piecewise shifted line segments are changed to one direction by the filter process

Fig. 2. However, there is singular case in the representation. Therefore, a 3D unit vector which does not have any singular case is used for representing a direction in 3D case even though it is a over-parameterized way.

2.2 Generalized DLmark detection algorithm for 2D/3D

The *split-and-merge* (or *divide-and-conquer*) algorithm is the most popular line/plane extraction algorithm [4,18]. The proposed algorithm replaces the merge process with a filter process. While the purpose of the merge process is to make one line/plane segment for the collinear line/plane segments, the purpose of the filter process is to make one directional feature for the piecewise shifted line/plane segments as shown in Fig. 3. It means that while the line/plane extraction considers the connectivity of the line/plane and the similarity of the direction at the same time, the direction extraction only considers the similarity of the direction. The proposed algorithm consists of three steps as shown in Fig. 4.

– (Step 1): Line/plane segments are extracted from the 2D/3D points.

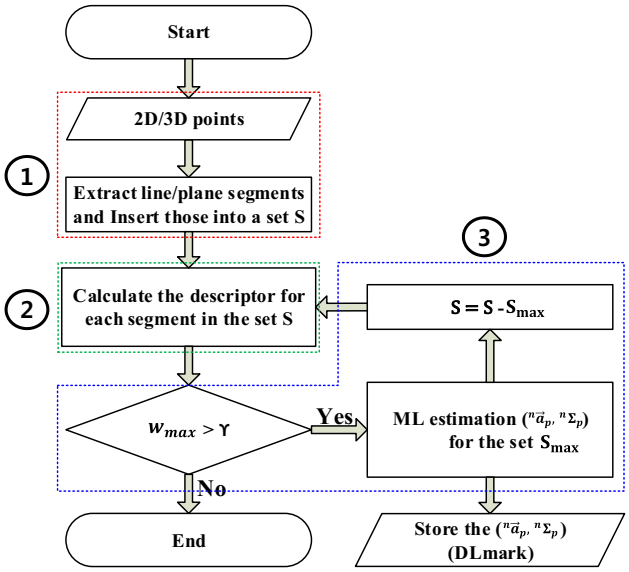


Fig. 4 Proposed DLmark detection algorithm consists of two processes: split and filter. Step 1 is the split process. Step 2 and Step 3 are the filter processes

– (Step 2): The descriptor that indicates the importance of each direction is calculated for each line/plane segment.
– (Step 3): DLmarks are extracted from the descriptor information calculated in Step 2.

Step 1 is the split process. Step 2 and Step 3 are the filter processes. The details of each step are as follows.

(Step 1)

In the split process, line/plane segments are extracted from the 2D/3D points. For 2D points, we used the IEPF (Iterative-End-Point-Fit) method [19] for the split algorithm and Pfister's method [20] for extracting the line parameters for 2D points. For 3D points, we used the method of Yeon et al. [18] to extract and represent plane segments. The extracted line/plane parameters are represented by a normal vector ${}^n\vec{\boldsymbol{a}}$, its covariance matrix ${}^n\Sigma$, and its length/area ${}^n w$. The superscript n is used to discriminate between 2D and 3D. The normal vector can be interpreted as 'orthogonal complement of $(n-1)$-dimensional hyperplane in n-dimensional vector space.' Therefore, the normal vector is one-dimensional subspace, which is represented by any bases. We used any one between two unit vectors as explained in the previous subsection. The extracted line/plane segments are put into the set S

$$\begin{aligned} S &= \{\mathrm{seg}_1, \ldots, \mathrm{seg}_K\} \\ &= \{({}^n\vec{\boldsymbol{a}}_1, {}^n\Sigma_1, {}^n w_1), \ldots, ({}^n\vec{\boldsymbol{a}}_K, {}^n\Sigma_K, {}^n w_K)\} \end{aligned} \tag{2}$$

where K is the number of extracted line/plane segments.

(Step 2)

This step extracts the descriptor, which is the sum of the lengths/areas of the line/plane segments that have a similar

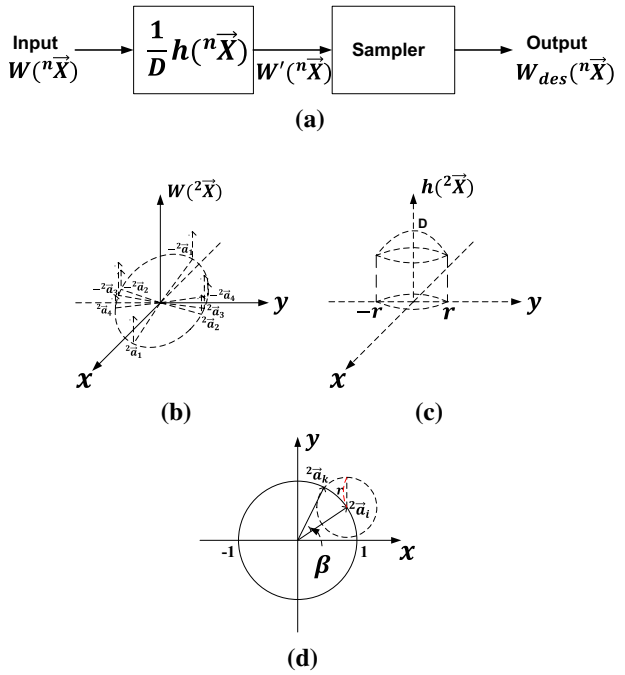


Fig. 5 **a** The procedure for calculating the descriptor of each segment. The first filter $\frac{1}{D}h(^n\vec{X})$ is a sliding window that is used to calculate the sum of lengths/areas of the line/plane segments that have similar directions. The second filter is a sampler that is used to extract discrete values from the continuous function $W'(^n\vec{X})$. **b** The input of the first filter for 2D case. **c** Truncated multivariate normal distribution function $h(^2\vec{X})$. **d** The process of going through the first filter

direction. The descriptor is calculated using two filters, as shown in Fig. 5a. First, the extracted line/plane segments can be represented by the following function as shown in Fig. 5b:

$$W(^n\vec{X}) = \sum_{i=1}^{K} {}^n w_i \delta\left\{^n\vec{X} - (\pm {}^n\vec{a}_i)\right\} \tag{3}$$

where $\|^n\vec{a}_i\| = 1, n \in \{2, 3\}$.

Second, $W(^n\vec{X})$ enters into the first filter. The impulse response $h(^n\vec{X})$ is a truncated multivariate normal distribution function that only has a value when the $\|^n\vec{X}\|$ is smaller than the value r that is decided by the parameter β as shown in Fig. 5c, d. The role of this function is to sum the lengths/areas of the line/plane segments within $\pm\beta$ like a sliding window. In other words, the parameter β means the maximum noise that is valid to be recognized as the same direction. The impulse response is written by

$$h(^n\vec{X}) = \begin{cases} \frac{f(^n\vec{X})}{\int_{\|^n\vec{X}\|\le r} f(^n\vec{X})\, d^n\vec{X}} & \|^n\vec{X}\| \le r \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$f(^n\vec{X}) = \frac{\exp\left\{-\frac{1}{2}{}^n\vec{X}^T \cdot^n \mathbf{C}^{-1} \cdot^n \vec{X}\right\}}{(2\pi)^{\frac{n}{2}} |^n\mathbf{C}|^{\frac{1}{2}}}$$
$$,^n\mathbf{C} = \begin{bmatrix} \sigma_h^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_h^2 \end{bmatrix} \tag{5}$$

where $r = \sqrt{2 - 2\cos\beta}$, and $f(^n\vec{X})$ is a probability density function (PDF) of a multivariate normal distribution. The standard deviation σ_h determines the shape of the sliding window $h(^n\vec{X})$. If σ_h is small, the weights for the $^n\vec{X}$ in the vicinity of the mean vector are larger than that in the outside. If σ_h is large, the weights have similar values. In this paper, we set the σ_h to infinity. It means that the same weights are applied to the $^n\vec{X}$ within $\|^n\vec{X}\| \le r$. The value D means the maximum weight, which is decided by σ_h, and we multiply $h(^n\vec{X})$ by $\frac{1}{D}$ to make the maximum weight of the filter equal to one. Note that the process going through the first filter is represented by the convolution form $W'(^n\vec{X}) = \frac{1}{D}(W * h)(^n\vec{X})$.

Third, we insert it into the sampler filter, which is represented by $\sum_{i=1}^{K}\delta(^n\vec{X} - {}^n\vec{a}_i)$. Through the second filter, the descriptors corresponding to the extracted line/plane segments are extracted. The output of the second filter is as follows:

$$W_{des}(^n\vec{X}) = \frac{1}{D}\sum_{i=1}^{K}\left\{(W * h)(^n\vec{a}_i)\cdot\delta(^n\vec{X} - {}^n\vec{a}_i)\right\} \tag{6}$$

(Step 3)

This step extracts the DLmarks for the line/plane segments that are used for making the largest descriptor in the following way. First, we find the maximum descriptor value w_{max} using the following equation:

$$w_{max} = max\left(W_{des}(^n\vec{X})\right) \tag{7}$$

If w_{max} is smaller than the predetermined parameter γ (m/m^2), it is concluded that all the K segments are extracted from a non-structure. Then, the algorithm is terminated. If w_{max} is larger than γ, it is concluded that there is a direction extracted from the structure. Note that the parameter γ means the minimum value of the descriptor corresponding to the direction that is extracted from the structure. Then, index z for the w_{max} value is extracted using the following equation:

$$z = \text{argmax}_i(W_{des}(^n\vec{a}_i)), \quad i \subseteq \{1, \ldots, K\} \tag{8}$$

Then, the set S_{max} corresponding to w_{max} is constructed as follows:

$$S_{max} = \left\{ \text{seg}_i | \theta_{zi} \leq \beta, i \subseteq \{1, 2, \ldots, K\} \right\} \tag{9}$$

This set includes directions whose angle difference from θ_z is smaller than β(rad). That is, this set contains the line/plane segments that are used to make w_{max}. Here, θ_{ij}, which represents the angle difference between two directions, is defined as follows:

$$\theta_{ij} = {}^n\vec{\boldsymbol{a}}_i \ominus^n \vec{\boldsymbol{a}}_j = \arccos\left(|{}^n\vec{\boldsymbol{a}}_i \cdot^n \vec{\boldsymbol{a}}_j|\right), \quad \theta_{ij} \in [0°, 90°] \tag{10}$$

Next, ${}^n\vec{\boldsymbol{a}}_p$ and ${}^n\Sigma_p$ are calculated from the set S_{max} and these are stored as a DLmark. We used the maximum likelihood (ML) method to find ${}^n\vec{\boldsymbol{a}}_p$ as follows:

$${}^n\vec{\boldsymbol{a}}_p = \text{argmax}_{{}^n\vec{\boldsymbol{a}}_p} \left\{ P({}^n\vec{\boldsymbol{a}}_1, \ldots, {}^n\vec{\boldsymbol{a}}_Q, {}^n w_1, \ldots, {}^n w_Q | {}^n\vec{\boldsymbol{a}}_p) \right\} \tag{11}$$

The result of ML method is the same as the weighted mean using a length/area as a weight. However, the unit vector is not in a vector space but a rotational space [21]. Therefore, a new mean calculation should be used. The samples of a unit vector are selected by using general unscented transform method and are normalized to 1. The distance metric to compute the mean vector for the set S_{max} is as follows:

$$\theta_{ij} = \arccos\left({}^n\vec{\boldsymbol{a}}_i \cdot^n \vec{\boldsymbol{a}}_j\right) \tag{12}$$

Note that Eq. (12) does not include the symbol of the absolute value for $\left({}^n\vec{\boldsymbol{a}}_i \cdot^n \vec{\boldsymbol{a}}_j\right)$ in contrast to Eq. (10) because the unit vectors in the set S_{max} are organized to head a similar direction in advance. In order to compute a closed form approximation of the mean, the distance metric is linearized. For linearization, the following taylor series equation is used.

$$\cos(\theta) = 1 - \frac{1}{2}\theta^2 + o(\theta^4) \tag{13}$$

$$\theta^2 = 2(1 - \cos(\theta) + o(\theta^4)) \tag{14}$$

$$\simeq 2(1 - \cos(\theta)) \tag{15}$$

$$(\theta_{ij})^2 \simeq 2(1 - \cos(\theta_{ij})) \tag{16}$$

$$= 2(1 - {}^n\vec{\boldsymbol{a}}_i \cdot^n \vec{\boldsymbol{a}}_j) \tag{17}$$

The calculation of mean is occurred nearby $\theta = 0$. Therefore, above linearization uses the nominal point $\theta = 0$. The mean unit vector is the following unit vector ${}^n\vec{\boldsymbol{a}}_p$ that minimizes the quadratic cost function,

$$E^2 = \sum_{i=1}^{Q} ({}^n\vec{\boldsymbol{a}}_p \ominus^n \vec{\boldsymbol{a}}_i)^2 \tag{18}$$

where Q is the number of elements in S_{max}. Using the approximation method of (17), the unit vector mean ${}^n\vec{\boldsymbol{a}}_p$ is determined as

$${}^n\vec{\boldsymbol{a}}_p = \arg\min_{\|{}^n\vec{\boldsymbol{a}}_p\|=1} \sum_{i=1}^{Q} {}^n w_i \, ({}^n\vec{\boldsymbol{a}}_p \ominus^n \vec{\boldsymbol{a}}_i)^2 \tag{19}$$

$$= \arg\min_{\|{}^n\vec{\boldsymbol{a}}_p\|=1} \sum_{i=1}^{Q} {}^n w_i \cdot 2(1 - {}^n\vec{\boldsymbol{a}}_p \cdot^n \vec{\boldsymbol{a}}_i) \tag{20}$$

$$= \arg\max_{\|{}^n\vec{\boldsymbol{a}}_p\|=1} \sum_{i=1}^{Q} {}^n w_i \, ({}^n\vec{\boldsymbol{a}}_p \cdot^n \vec{\boldsymbol{a}}_i) \tag{21}$$

$$= \arg\max_{\|{}^n\vec{\boldsymbol{a}}_p\|=1} {}^n\vec{\boldsymbol{a}}_p \sum_{i=1}^{Q} {}^n w_i \, {}^n\vec{\boldsymbol{a}}_i \tag{22}$$

$$= \frac{\sum_{i=1}^{Q} {}^n w_i \, {}^n\vec{\boldsymbol{a}}_i}{\| \sum_{i=1}^{Q} {}^n w_i \, {}^n\vec{\boldsymbol{a}}_i \|} \tag{23}$$

Here, it is assumed that all of the directions in the set S_{max} are uncorrelated Gaussian random variables. Therefore, the ${}^n\Sigma_p$ is calculated as follows:

$${}^n\Sigma_p = \frac{\sum_{i=1}^{Q} {}^n w_i^2 \, {}^n\vec{\boldsymbol{a}}_i}{\| \sum_{i=1}^{Q} {}^n w_i \, {}^n\vec{\boldsymbol{a}}_i \|^2} \tag{24}$$

In this process, the length/area of each line/plane is used as a weight. After this process, the used line/plane segments are discarded through $S = S - S_{max}$. Then, Step 2 and Step 3 are repeated for the updated set S. This iteration is terminated when there is no DLmark in the set S.

The realization of the *split-and-filter* is explained in Algorithm 1. Line 1 indicates the Step 1, and lines 3–11 using two *for-loop*s indicate the Step 2, which means the procedure of passing two filters. Finally, lines 12–17 indicate the Step 3.

2.3 Discussion

In a cluttered environment, there are many distractions such as static and dynamic objects. These obstacles cause to extract wrong DLmarks and hinder to detect the structure's information from the sensor. To overcome these difficulties, we used the property of indoor structures: the amount of information from a structure is larger than that from a non-structure, even in cluttered environments and the length/area of line/plane segments is the means to detect the amount of information. The *split-and-filter* uses this property by using the convolution method with a sliding window, which sums the lengths/areas of line/plane segments that have a similar direction. Therefore, the proposed feature detection algorithm can distinguish DLmarks from non-DLmarks with the high accuracy and high detection rate, even in various environments.

Figure 6a shows a crowded environment, and Fig. 6b illustrates the 2D points obtained in such an environment. Figure 6c shows $W(\theta)$, which represents the extracted line

Algorithm 1: Split-and-Filter Algorithm

Input: 2D/3D points
Output: DLmarks $({}^{n}\vec{\boldsymbol{a}}, {}^{n}\Sigma)$

1 $\mathbf{S} \leftarrow$ Split(2D/3D points);
2 **while** *($\mathbf{S}$ is not empty)* **do**
3 **for** $i \leftarrow (1$ ***to*** $m)$ *(m: # of segments in $\mathbf{S}$)* **do**
4 **for** $j \leftarrow (1$ ***to*** $m)$ **do**
5 $\theta_{ij} \leftarrow {}^{n}\vec{\boldsymbol{a}}_i \ominus {}^{n}\vec{\boldsymbol{a}}_j$;
6 **if** $\theta_{ij} < \beta$ **then**
7 group(i) $\leftarrow$ [group(i), j] ;
8 $W_{des}({}^{n}\vec{\boldsymbol{a}}_i) \leftarrow W_{des}({}^{n}\vec{\boldsymbol{a}}_i) + \frac{1}{D}h({}^{n}\vec{\boldsymbol{a}}_j - {}^{n}\vec{\boldsymbol{a}}_i) \cdot W({}^{n}\vec{\boldsymbol{a}}_j)$;
9 **end**
10 **end**
11 **end**
12 **if** $(w_{max} > \gamma)$ **then**
13 $[{}^{n}\vec{\boldsymbol{a}}_p, {}^{n}\Sigma_p] \leftarrow$ ML Estimator($\mathbf{S}_{max}$) ;
14 $\mathbf{S} \leftarrow \mathbf{S} - \mathbf{S}_{max}$;
15 **else**
16 $\mathbf{S} \leftarrow$ empty ;
17 **end**
18 **end**

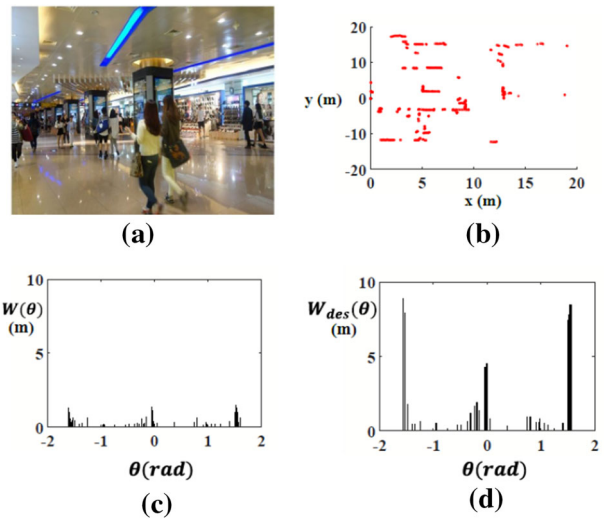


Fig. 6 Two-dimensional points acquired in a crowded environment such as **a** are untidy as in **b**. The impulse function **c** for the extracted line segments is changed to the function **d** for the descriptors through the proposed filter process

segments; it was determined that there were various angles, and the values of y axis were not distinctly different. On the other hand, the values of y axis after passing through the filter process were different distinctly as shown in Fig. 6d. This example clearly shows the usefulness of the filter process.

3 Data association

In general, the data association method in SLAM is based on a probability model (e.g., a Gaussian assumption) [22] for accuracy. However, the accuracy is valid only when the probability model is correct. Since the probability model changes depending on the environment, the performance of probabilistic data association (PDA) depends on the environment. To be robust to the change in the environments in terms of the parameters, we did not use PDA but DDA (deterministic data association), which only uses deterministic values. The probabilistic distance, such as the Mahalanobis distance, of PDA is calculated using the uncertainty information of each feature. On the other hand, DDA uses a metric distance such as the Euclidean distance. For DLmarks, the metric distance is just the angle difference. Algorithm 2 illustrates the entire procedure of DDA. In line 3, the distance between a measurement ${}^{n,r}\vec{\boldsymbol{a}}_{m,i}$ and an existing landmark ${}^{n,r}\vec{\boldsymbol{a}}_{f,j}$ is calculated using (10). The superscript r means that the information is represented with respect to the robot reference frame, and the subscripts m and f mean a measurement and a feature, respectively. Note that when the angle is used for 2D case, the distance is calculated using (10). Then, the following procedure from line 4 to line 10 is the same as the general data association. The parameter used in DDA is α, which represents the maximum error of a matched pair. If θ_{ij} is smaller than α, it is decided that the measurement is matched to the existing landmark. However, if θ_{ij} is larger than 2α, it is decided that the measurement is a new landmark. We set the margin α between two choices to minimize the false positives.

Algorithm 2: Deterministic Data Association

Input: measurements ${}^{n,r}\vec{\boldsymbol{a}}_{m,i}$, $i \in [1, s]$,
stored landamarks ${}^{n}\vec{\boldsymbol{a}}_{f,j}$, $j \in [1, t]$
Output: (matched pairs) or (new landmark id)

1 **for** $i \leftarrow (1$ ***to*** $s)$ **do**
2 **for** $j \leftarrow (1$ ***to*** $t)$ **do**
3 $\theta_{ij} \leftarrow {}^{n,r}\vec{\boldsymbol{a}}_{m,i} \ominus {}^{n,r}\vec{\boldsymbol{a}}_{f,j}$;
4 **if** $\theta_{ij} < \frac{\alpha}{2}$ **then**
5 ${}^{n,r}\vec{\boldsymbol{a}}_{m,i}$ is matched to ${}^{n}\vec{\boldsymbol{a}}_{f,j}$;
6 **else if** $\theta_{ij} > \alpha$ **then**
7 ${}^{n,r}\vec{\boldsymbol{a}}_{m,i}$ is a new landmark;
8 **else**
9 ${}^{n,r}\vec{\boldsymbol{a}}_{m,i}$ is nothing;
10 **end**
11 **end**
12 **end**

4 DL-SLAM

4.1 2D DL-SLAM equations

This section explains the entire procedure of 2D DL-SLAM, which is composed of the initial estimate and a Kalman filter.

4.1.1 Initial estimate

A robot pose is expressed as follows:

$$\mathbf{x}_r = \begin{bmatrix} x_r & y_r & \phi_r \end{bmatrix}^T \tag{25}$$

$$= \begin{bmatrix} \mathbf{x}_{tr} & \phi_r \end{bmatrix}^T \tag{26}$$

Here, subscripts r and tr denote robot state and translational part, respectively. A transformation between the adjacent steps is acquired directly from the odometry as follows:

$${}^{k-1}\hat{\mathbf{x}}_{r,k} = \begin{bmatrix} {}^{k-1}\hat{x}_k & {}^{k-1}\hat{y}_k & {}^{k-1}\hat{\phi}_k \end{bmatrix}^T \tag{27}$$

where the superscript '$k-1$' and the subscript 'k' denote the 'previous step' and the 'current step,' respectively. This transformation indicates a local translation and a local rotation. The transition equations of the robot pose are as follows:

$$\hat{\mathbf{x}}_{tra,k} = \hat{\mathbf{x}}_{tra,k-1} + \Delta\hat{\mathbf{x}}_{tra,k-1} \tag{28}$$

$$= \hat{\mathbf{x}}_{tra,k-1} + Rot(\hat{\phi}_{r,k-1}) \begin{bmatrix} {}^{k-1}\hat{x}_k \\ {}^{k-1}\hat{y}_k \end{bmatrix} \tag{29}$$

$$\hat{\phi}_{r,k} = \hat{\phi}_{r,k-1} + \Delta\hat{\phi}_{r,k-1} \tag{30}$$

$$= \hat{\phi}_{r,k-1} +^{k-1} \hat{\phi}_k \tag{31}$$

where *Rot* indicates the rotation matrix. The errors of $\Delta\hat{\mathbf{x}}_{tra,k-1}$ and that of $\Delta\hat{\phi}_{r,k-1}$ are accumulated for the global translation and the global rotation, respectively. Note that $\Delta\hat{\mathbf{x}}_{tra,k-1}$ is affected by the error of $\hat{\phi}_{r,k-1}$.

4.1.2 Kalman filter

A. State prediction

DL-SLAM only uses directional landmarks. Therefore, the state consists of current robot angle ϕ_r and the n registered DLmarks $\theta_{(1:n)}$, and the accumulated error due to the error of $\Delta\hat{\phi}_{r,k-1}$ can only be corrected by the Kalman filter. The state $\mathbb{x}$ and the covariance matrix $\boldsymbol{P}$ of the Kalman filter are defined as follows:

$$\mathbb{x} = \begin{bmatrix} \phi_r \\ \theta_{(1:n)} \end{bmatrix} \tag{32}$$

$$\hat{\mathbf{u}}_{k-1} = {}^{k-1}\hat{\phi}_k \tag{33}$$

$$\hat{\mathbb{x}}_k^- = f(\hat{\mathbb{x}}_{k-1}^+, \hat{\mathbf{u}}_{k-1}) = \hat{\mathbb{x}}_{k-1}^+ + \begin{bmatrix} \Delta\hat{\phi}_{r,k-1} \\ \mathbf{0}_{n\times 1} \end{bmatrix} \tag{34}$$

$$\boldsymbol{P}_k^- = \boldsymbol{E}[(\mathbb{x}_k^- - \hat{\mathbb{x}}_k^-) \cdot (\mathbb{x}_k^- - \hat{\mathbb{x}}_k^-)^T] \tag{35}$$

$$= \boldsymbol{P}_{k-1}^+ + \begin{bmatrix} \sigma_{\Delta\phi}^2 & \mathbf{0}_{1\times n} \\ \mathbf{0}_{n\times 1} & \mathbf{0}_{n\times n} \end{bmatrix} \tag{36}$$

where $f(\cdot,\cdot)$ is a state transition equation, $\hat{\mathbf{u}}_{k-1}$ is a control input at step $k-1$, and $\sigma_{\Delta\phi}^2$ is the variance of the $\Delta\hat{\phi}_{r,k-1}$.

Here, the superscript '$-$' denotes *before the update* and the superscript '$+$' denotes *after the update*.

B. Observation function

The observation function of 2D DL-SLAM is defined as follows:

$$z = \mathbb{H}\mathbb{x} + w_k, \quad \boldsymbol{E}[w_k^2] = \boldsymbol{R} \tag{37}$$

where w_k is zero-mean uncorrelated Gaussian noise.

$$\hat{z} = \mathbb{H}\hat{\mathbb{x}} \tag{38}$$

$$= \begin{bmatrix} \boldsymbol{H}_r & \boldsymbol{H}_i \end{bmatrix} \begin{bmatrix} \hat{\phi}_r^- \\ \hat{\theta}_{(1:n)}^- \end{bmatrix} = -\hat{\phi}_r^- + \hat{\theta}_i^- \tag{39}$$

$$\boldsymbol{H}_r = -1 \tag{40}$$

$$\boldsymbol{H}_i = \begin{bmatrix} 0 & 0 & \ldots & 1 & \ldots & 0 & 0 \end{bmatrix}, i\text{th element is } 1 \tag{41}$$

Note that there is no inconsistency problem due to the linearization error [23] in the correction process because the observation function is linear.

C. State update

The update procedure of state $\mathbb{x}$ and covariance $\boldsymbol{P}$ by Kalman gain $\boldsymbol{K}$ is as follows:

$$\hat{\mathbb{x}}_k^+ = \hat{\mathbb{x}}_k^- + \boldsymbol{K} \cdot \nu \quad (\nu: \text{ innovation}) \tag{42}$$

$$\boldsymbol{K} = \boldsymbol{P}_k^- \mathbb{H}^T \cdot (\mathbb{H}\boldsymbol{P}_k^- \mathbb{H}^T + \boldsymbol{R})^{-1} \tag{43}$$

$$= \boldsymbol{P}_k^- \mathbb{H}^T \cdot (\sigma_\nu^2)^{-1} \quad (\sigma_\nu^2: \text{ variance of } \nu) \tag{44}$$

$$\boldsymbol{P}_k^+ = (\boldsymbol{I}_{n+1} - \boldsymbol{K}\mathbb{H})\boldsymbol{P}_k^-(\boldsymbol{I}_{n+1} - \boldsymbol{K}\mathbb{H})^T + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^T \tag{45}$$

where $\nu = z - \hat{z}$.

D. State augmentation

$$\hat{\mathbb{x}}_k^+ = \begin{bmatrix} \hat{\phi}_r^+ \\ \hat{\theta}_{(1:n)}^+ \end{bmatrix} \rightarrow \hat{\mathbb{x}}_k^+ = \begin{bmatrix} \hat{\phi}_r^+ \\ \hat{\theta}_{(1:n)}^+ \\ \hat{\theta}_a \end{bmatrix} \tag{46}$$

$$\boldsymbol{P}^+ = \begin{bmatrix} \sigma_\phi^2 & E[\tilde{\phi} \cdot \tilde{\Theta}^T] \\ E[\tilde{\Theta} \cdot \tilde{\phi}^T] & E[\tilde{\Theta} \cdot \tilde{\Theta}^T] \end{bmatrix} \tag{47}$$

$$\rightarrow \boldsymbol{P}^+ = \begin{bmatrix} \sigma_\phi^2 & E[\tilde{\phi} \cdot \tilde{\Theta}^T] & E[\tilde{\phi} \cdot \tilde{\theta_a}] \\ E[\tilde{\Theta} \cdot \tilde{\phi}^T] & E[\tilde{\Theta} \cdot \tilde{\Theta}^T] & E[\tilde{\Theta} \cdot \tilde{\theta_a}] \\ E[\tilde{\theta_a} \cdot \tilde{\phi}^T] & E[\tilde{\theta_a} \cdot \tilde{\Theta}^T] & E[\tilde{\theta_a} \cdot \tilde{\theta_a}] \end{bmatrix} \tag{48}$$

4.2 3D DL-SLAM equations

This section explains the entire procedure of 3D DL-SLAM, which is composed of the initial estimate and a unscented Kalman filter.

4.2.1 Initial estimate

The robot pose in 3D can be represented as follows [24,25]:

$$\mathbf{x}_r = \left[\mathbf{x}_{tr}\ \mathbf{x}_{ro}\right]^T \tag{49}$$

$$= \left[x_r\ y_r\ z_r\ q_0\ q_1\ q_2\ q_3\right]^T \tag{50}$$

Here, subscripts tr and ro denote robot translation and rotation part, respectively. The rotation of a robot is represented using a quaternion representation, which composes of scalar part q_0 and vector part (q_1, q_2, q_3). A transformation between the adjacent steps is acquired directly from the odometry as follows:

$$^{k-1}\hat{\mathbf{x}}_{r,k} = \left[^{k-1}\hat{\mathbf{x}}_{tr,k}\ ^{k-1}\hat{\mathbf{x}}_{ro,k}\right]^T = \left[^{k-1}\hat{x}_k\ ^{k-1}\hat{y}_k\ ^{k-1}\hat{z}_k\ ^{k-1}\hat{\boldsymbol{Q}}_{r,k}\right]^T \tag{51}$$

This transformation indicates a local translation and a local rotation. The transition equations of the robot pose are as follows:

$$\hat{\mathbf{x}}_{tra,k} = \hat{\mathbf{x}}_{tra,k-1} + \Delta\hat{\mathbf{x}}_{tra,k-1} \tag{52}$$

$$= \hat{\mathbf{x}}_{tra,k-1} + \hat{\boldsymbol{R}}_{k-1} \cdot^{k-1} \hat{\mathbf{x}}_{tr,k} \tag{53}$$

$$\hat{\boldsymbol{Q}}_{r,k} = \hat{\boldsymbol{Q}}_{r,k-1} \otimes \Delta\hat{\boldsymbol{Q}}_{r,k-1} \tag{54}$$

$$= \hat{\boldsymbol{Q}}_{r,k-1} \otimes^{k-1} \hat{\boldsymbol{Q}}_{r,k} \tag{55}$$

where $\hat{\boldsymbol{R}}_{k-1}$ and $\hat{\boldsymbol{Q}}_{r,k-1}$ mean rotation matrix and quaternion representation of a robot at time step $k-1$, respectively. The symbol $\otimes$ means a quaternion multiplication. Note that $\Delta\hat{\mathbf{x}}_{tra,k-1}$ is affected by the error of $\hat{\boldsymbol{R}}_{k-1}$.

4.2.2 Unscented Kalman filter

A. State prediction

3D DL-SLAM only uses directional landmarks. Therefore, the state consists of current robot rotation $\boldsymbol{Q}_r$ and the n registered DLmarks $\overrightarrow{\boldsymbol{a}}_{(1:n)}$, and the accumulated error due to the error of $\Delta\hat{\boldsymbol{Q}}_{r,k-1}$ can only be corrected by the Kalman filter. The state $\mathbb{x}$ and the covariance matrix $\boldsymbol{P}$ of the Kalman filter are defined as follows:

$$\mathbb{x} = \begin{bmatrix} \boldsymbol{Q}_r \\ \overrightarrow{\boldsymbol{a}}_{(1:n)} \end{bmatrix} \tag{56}$$

$$\hat{\mathbf{u}}_{k-1} = {}^{k-1}\hat{\boldsymbol{Q}}_{r,k} \tag{57}$$

$$\hat{\mathbb{x}}_k^- = f(\hat{\mathbb{x}}_{k-1}^+, \hat{\mathbf{u}}_{k-1}) \tag{58}$$

$$\hat{\mathbb{x}}_{k,(1:4)}^- = \hat{\mathbf{x}}_{ro,k}^- = \hat{\boldsymbol{Q}}_k^- = \hat{\boldsymbol{R}}_k^- \tag{59}$$

$$= \left[\hat{q}_{0,k}^-\ \hat{q}_{1,k}^-\ \hat{q}_{2,k}^-\ \hat{q}_{3,k}^-\right]^T \tag{60}$$

$$= \hat{\mathbb{x}}_{k-1,(1:4)}^+ \otimes^{k-1} \hat{\boldsymbol{Q}}_{r,k} \tag{61}$$

$$= \hat{\boldsymbol{Q}}_{k-1}^+ \otimes^{k-1} \hat{\boldsymbol{Q}}_{r,k} \tag{62}$$

Note that the stored landmarks are not changed as follows:

$$\hat{\mathbb{x}}_{k,(5:4+(n\times3))}^- = \hat{\mathbb{x}}_{k-1,(5:4+(n\times3))}^+ \tag{63}$$

The updated covariance matrix $\boldsymbol{P}_k^-$ is as follows:

$$\boldsymbol{P}_k^- = \boldsymbol{E}[(\tilde{\mathbb{x}}_k^-) \cdot (\tilde{\mathbb{x}}_k^-)^T] \tag{64}$$

$\hat{\mathbf{x}}_{ro,k}^-$ and $\boldsymbol{P}_k^-$ are calculated by using unscented transform [26] because the 3D equations are nonlinear in contrast to the equations of 2D case.

B. Observation function

The observation function of 3D DL-SLAM is as follows:

$$z = \mathbb{H}\mathbb{x} + \mathbf{w}_k, \quad E[\mathbf{w}_k\mathbf{w}_k^T] = \boldsymbol{R} \tag{65}$$

where $\mathbf{w}_k$ is zero-mean uncorrelated Gaussian noise vector.

$$\hat{z} = \mathbb{H}\hat{\mathbb{x}} \tag{66}$$

$$= (\hat{\boldsymbol{R}}_k^-)^{-1} \cdot \overrightarrow{\boldsymbol{a}}_i, \tag{67}$$

where $\overrightarrow{\boldsymbol{a}}_{f,i}$ is the stored ith landmark.

C. Unscented Kalman Correction

The update procedure of state $\mathbb{x}$ and covariance $\boldsymbol{P}$ by Kalman gain $\boldsymbol{K}$ is as follows:

$$\hat{\mathbb{x}}_k^+ = \hat{\mathbb{x}}_k^- + \boldsymbol{K} \cdot \overrightarrow{\nu} \quad (\nu:\ \text{innovation vector}) \tag{68}$$

$$\boldsymbol{K} = \boldsymbol{P}_k^- \mathbb{H}^T \cdot (\mathbb{H}\boldsymbol{P}_k^- \mathbb{H}^T + \boldsymbol{R})^{-1} \tag{69}$$

$$= \boldsymbol{P}_k^- \mathbb{H}^T \cdot \mathbf{S}^{-1} \quad (\mathbf{S}:\ \text{covariance matrix of } \overrightarrow{\nu}) \tag{70}$$

$$\boldsymbol{P}_k^+ = (\boldsymbol{I}_{3n+4} - \boldsymbol{K}\mathbb{H})\boldsymbol{P}_k^-(\boldsymbol{I}_{3n+4} - \boldsymbol{K}\mathbb{H})^T + \boldsymbol{K}\boldsymbol{R}\boldsymbol{K}^T \tag{71}$$

$$= \boldsymbol{P}_k^- - \boldsymbol{K}\mathbf{S}\boldsymbol{K}^T \tag{72}$$

where $\overrightarrow{\nu} = z - \hat{z}$. In 3D case, the rotation part of the state has to be calculated by another method. When a quaternion representation is used, the update equation is not plus but quaternion multiplication. Other three equations are the same as the original equations. The procedure of updating the state is as follows. First, we calculate the $\boldsymbol{K} \cdot \overrightarrow{\nu}$ value. Second, we conduct the add operation for the feature part and normalize the results to one because the landmarks are unit vectors. Third, we calculate the quaternion multiplication $\hat{\boldsymbol{Q}}_{r,k}^+ = \hat{\boldsymbol{Q}}_{r,k}^- \otimes \boldsymbol{K}_q \cdot \overrightarrow{\nu}$. In this procedure, UKF [26] is used.

D. State augmentation

When a measurement ${}^{r}\vec{a}_{a}$ is decided as a new landmark, the feature is registered in the state as follows:

$$\vec{a}_{a} = \hat{R}_{k}^{+} \cdot {}^{r}\vec{a}_{a} \tag{73}$$

4.3 Discussion

The global consistency can be validated by showing the global translation and rotation error. The global translation error depends on both the local translation error and the global rotation error as shown in (29) and (53), whereas the global rotation error depends only on the local rotation error as shown in (31) and (55). DL-SLAM is an algorithm that corrects the global rotation error by the Kalman filter. Therefore, the accumulated local rotation error is erased frequently. However, the local translation error and the global translation error caused by the global rotation error cannot be corrected, and these errors are accumulated. Note that there are some steps that do not detect DLmarks at all or have false-positive data associations. In these steps, the global rotation error is not corrected. For these reasons, the DLmark detection rate and false-positive rate influence the *accuracy*.

In the Experiment section, we show two things using extensive datasets. First, the local translation error of the odometry is small. Second, the global translation error caused by the global rotation error is minimized because of the frequent DLmark detection and low false-positive rate.

5 Experiment

5.1 2D experiments

We conducted off-line and real-time experiments using 20 datasets that were acquired from Radish [27], SLAM benchmarking [28], Rawseed [29,30], and our own datasets. The information about the datasets that were used for the experimental verification is explained in Table 2. The 15 datasets (a to o) mapped in Figs. 7, 8, and 9 were used to show the performance of DL-SLAM, and 5 datasets (p to t) mapped in Fig. 10 were used to indicate the types of environments where DL-SLAM cannot be used. In Fig. 7, 8, 9, and 10, the blue-dotted lines and the green ellipses on the reference map indicate the ideal DLmarks and the curved parts of the environments, respectively. The blue circles in the DL-SLAM results indicate the loop closure points, which were manually selected to calculate the error. The far-right column of Table 2 shows the information about the wheeled mobile robot platform of each dataset. The platform type of Pioneer-DX, Tetra, and Robocom is a two-wheel differential drive, whereas that of B21r and Magellan Pro is a four-wheel synchro drive.

Table 2 Information about the 20 datasets

	Dataset name	Source	Robot platform
a	Chosun Univ.	Radish	Pioneer-3DX
b	CMU NSH	Radish	Pioneer-2DX
c	USC SAL	Radish	Pioneer-2DX
d	SDR site B	Radish	Pioneer-2DX
e	SRI A bldg.	Radish	Pioneer-2DX
f	Intel Oregon	Radish	Pioneer-2DX
g	Stanford Gates	Radish	Pioneer-2DX
h	Freiburg bldg 79	Radish	Pioneer-2DX
i	Aces	Radish	Magellan Pro.
j	MIT Killian Court	SLAM benchmarking	B21r
k	Bicocca Indoor	Rawseeds	Robocom
l	KU 2nd Engr.	Our dataset	Pioneer-3DX
m	KU CJ Int.	Our dataset	Tetra
n	KINTEX[a]	Our dataset	Tetra
o	Gangnam Station[a]	Our dataset	Tetra
p	KU Station	Our dataset	Tetra
q	KU Lyceum	Our dataset	Pioneer-3DX
r	Fr101 explored	Radish	Pioneer-2DX
s	MIT CSAIL	Radish	B21r
t	Intel Lab.	Radish	Pion eer2DX

[a] Real-time experiment

Table 3 shows the results of the quantitative analysis for the 15 datasets.

The real-time experiments were conducted at the (n) KINTEX hall and (o) Gangnam Station during the daytime to show the computational efficiency of DL-SLAM. KINTEX is the biggest exhibition hall in Korea, and Gangnam Station is one of the most crowded places in Korea. The average velocity of the robot was 0.5 m/s apart from the stationary state. The smartphone used for the real-time experiments was a Samsung Galaxy SII SHV-E110S. The configuration of the hardware is given in Fig. 11.

In this section, we analyze the off-line and real-time experimental results in terms of three main perspectives: adaptability, accuracy, and lightness.

5.1.1 Adaptability

Various environments We used various types of environments to show the *adaptability* of DL-SLAM. These were simple corridors, overcrowded environments, small rooms, and partially curved environments. These types are indicated using superscripts in column 1 of Table 3.

- Simple corridor: In this type, information about the structure is abundant. Therefore, it is easy to detect accurate DLmarks.

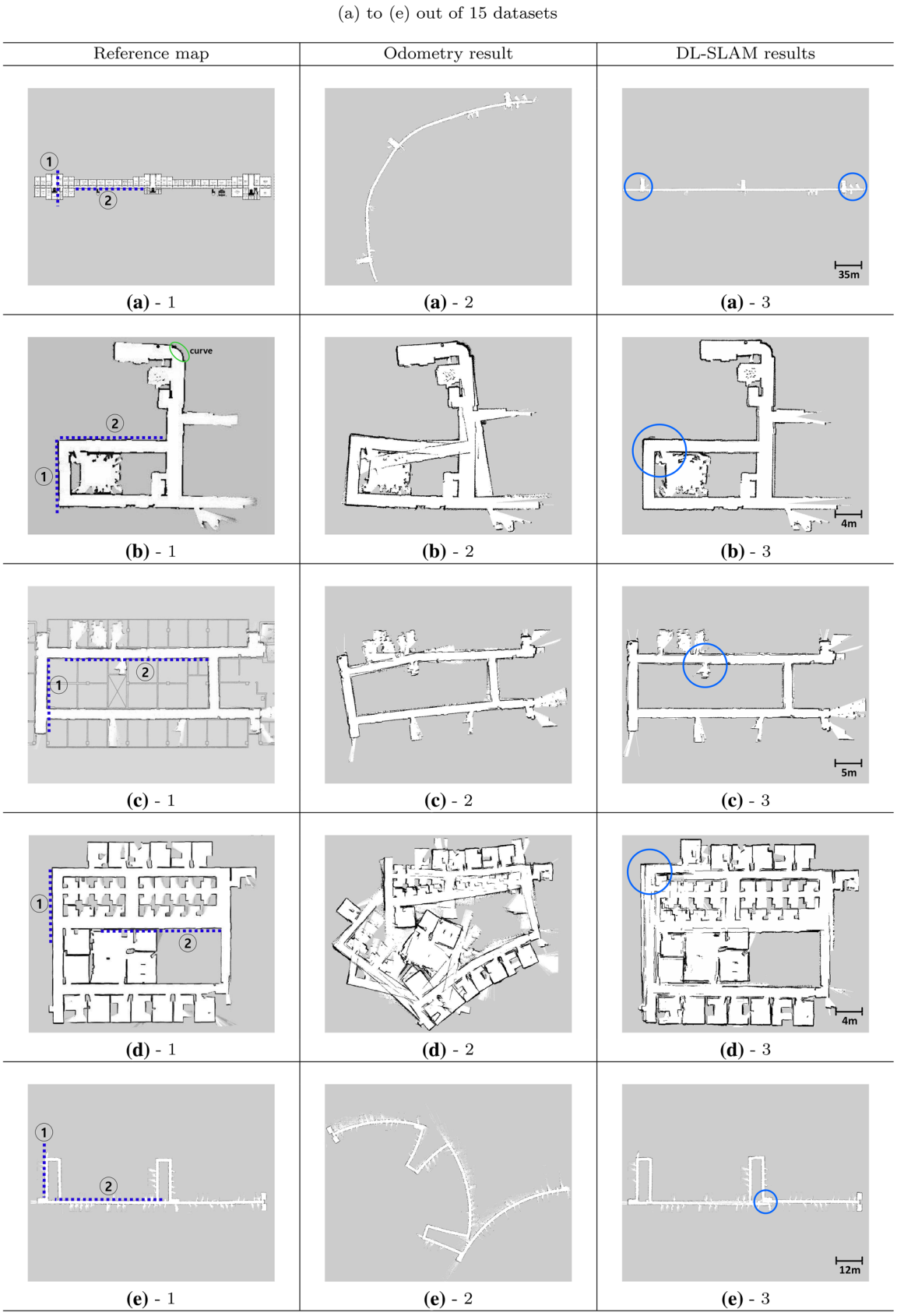


Fig. 7 Occupancy grid mapping results showing the performance of DL-SLAM

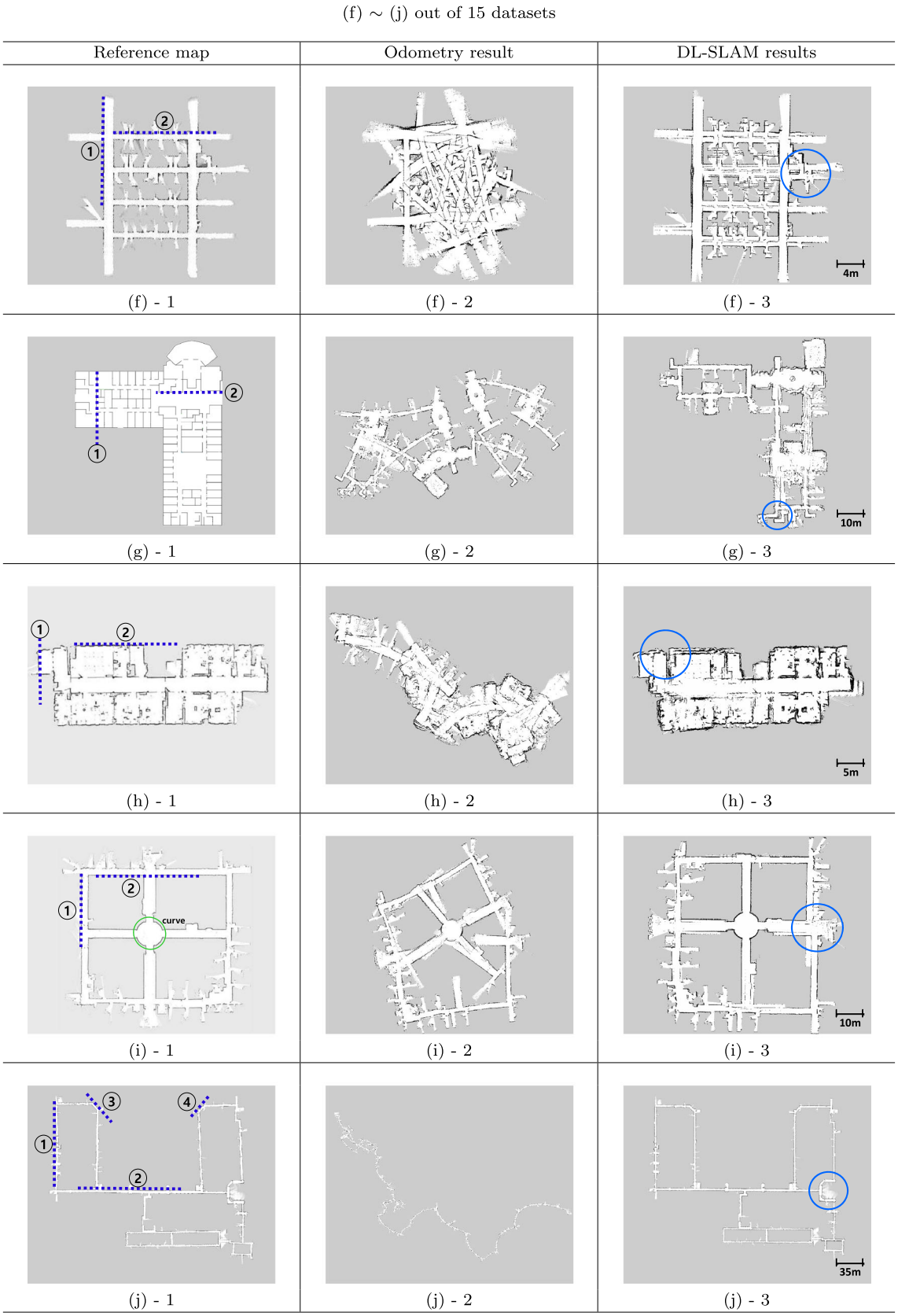


Fig. 8 Occupancy grid mapping results showing the performance of DL-SLAM

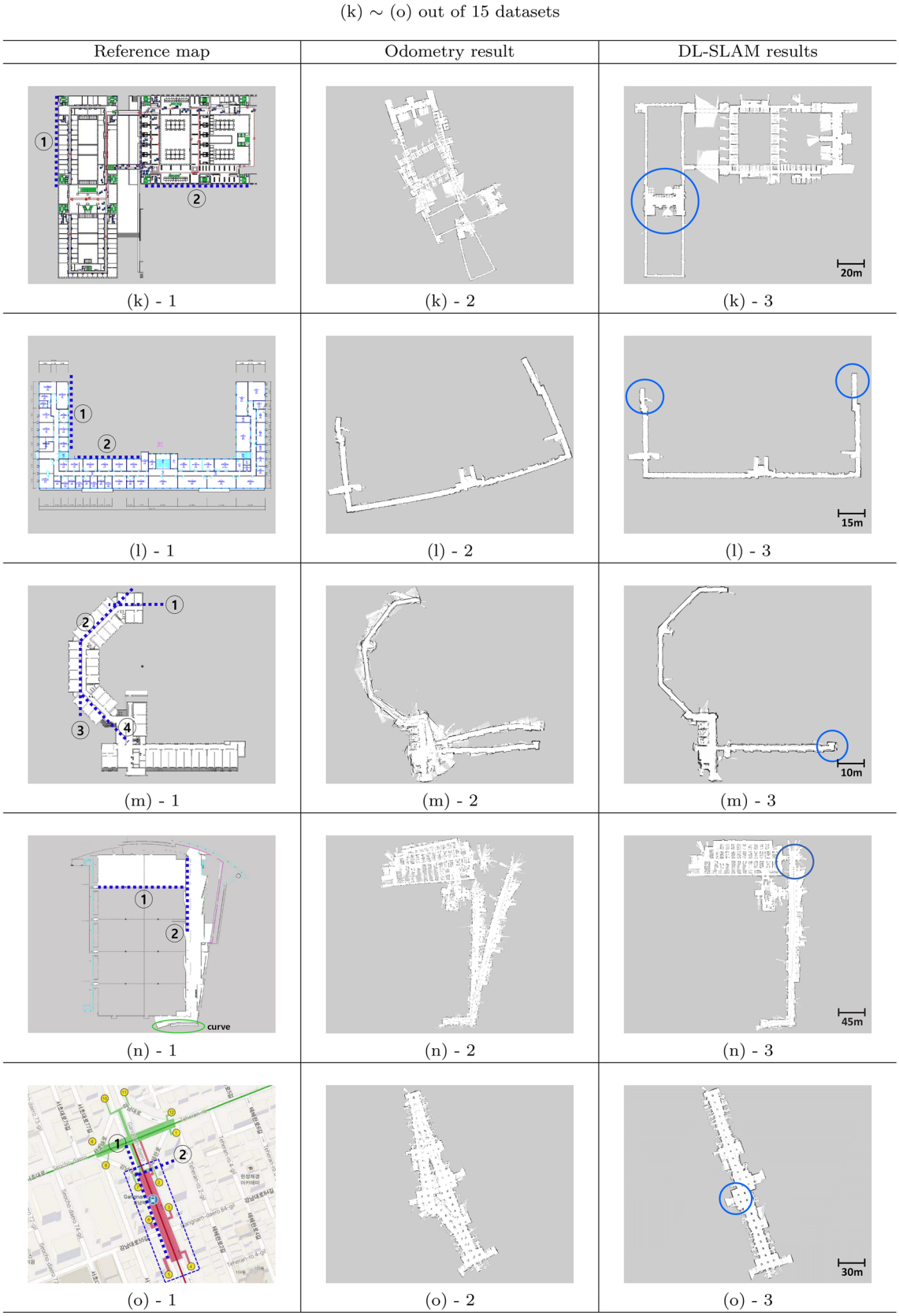


Fig. 9 Occupancy grid mapping results showing the performance of DL-SLAM

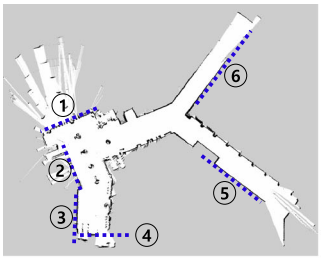
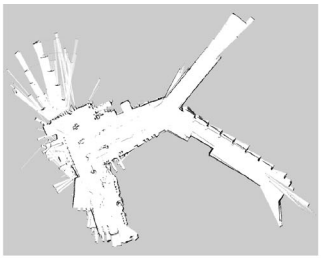
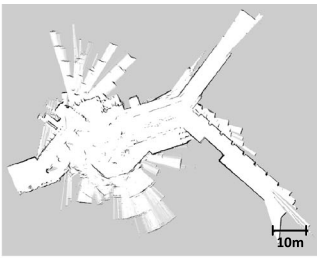
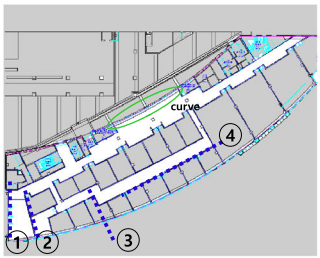
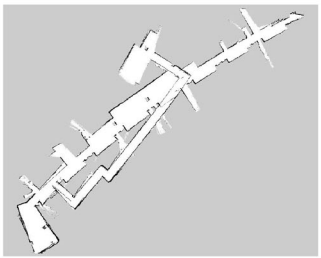
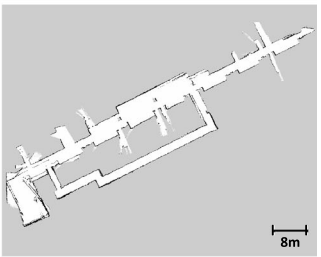
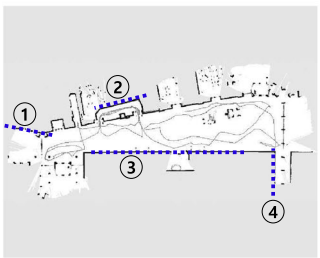
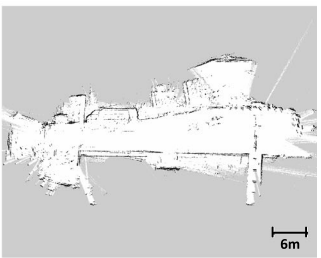
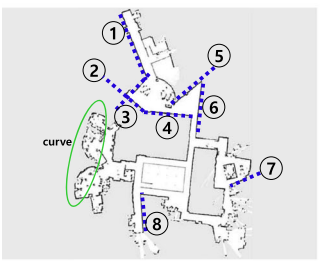
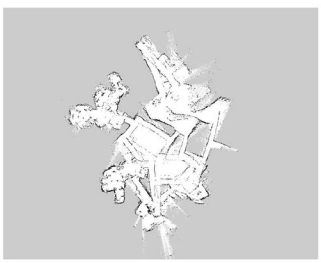
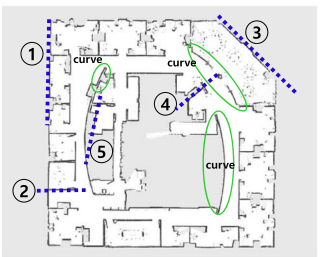
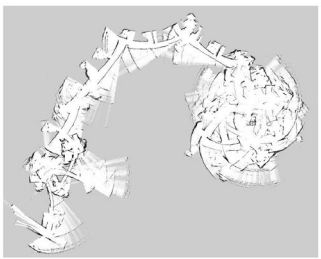
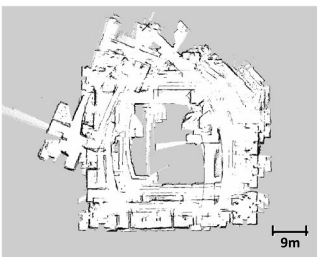
Reference map	Odometry result	DL-SLAM results (fail)
		
(p) - 1	(p) - 2	(p) - 3
		
(q) - 1	(q) - 2	(q) - 3
		
(r) - 1	(r) - 2	(r) - 3
		
(s) - 1	(s) - 2	(s) - 3
		
(t) - 1	(t) - 2	(t) - 3

Fig. 10 The environments where the DL-SLAM algorithm cannot be used

Table 3 Quantitative analysis results for all the datasets

	① Dataset name	② Feature detection rate (%)	③ False-positive rate (%)	④ Travel length (m)	⑤ Translation error (m)	⑥ Translation error rate for travel length (%)
a	Chosun Univ.[a]	99.93	0	325.79	0.24	0.07
b	CMU NSH Level A[a,d]	92.31	0.11	116.72	0.50	0.43
c	USC SAL[a]	99.53	0.04	122.06	0.45	0.37
d	SDR site B[a]	99.72	0.10	357.15	0.65	0.18
e	SRI A bldg.[a]	100	0	298.58	1.10	0.37
f	Intel Oregon[a]	98.44	0.14	239.70	1.90	0.79
g	Stanford Gates[a]	91.29	0.12	517.38	2.90	1.76
h	Freiburg bldg 79[c]	72.01	0.07	392.31	1.20	0.31
i	Aces[a,d]	95.71	0.60	325.88	2.00	0.61
j	MIT Killian Court[a]	97.47	0.16	1912.02	1.00	0.05
k	Bicocca Indoor[a]	98.69	0.03	1036.83	1.00	0.10
l	KU 2nd Engr.[a]	99.59	0.16	181.31	0.88	0.49
m	KU CJ Int.[a]	99.99	0	291.93	0.45	0.15
n	KINTEX[b,d]	99.67	3.53	1527.96	2.60	0.17
o	Gangnam Station[b]	98.35	0.53	411.81	0.55	0.13
	Average value	96.18	0.37	513.66	1.16	0.40

[a] Simple corridor, [b] overcrowded environment, [c] small room, [d] partially curved environment

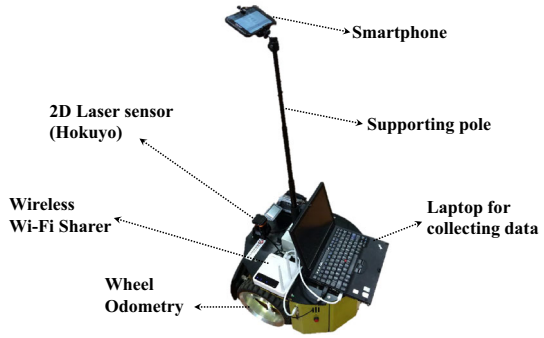


Fig. 11 Hardware configuration for the real-time experiments. First, the wheel-based odometry and the Hokuyo LIDAR data are collected using a laptop and then are transmitted to a smartphone via a wireless Wi-Fi sharer in real time. Then, the DL-SLAM application installed in the smartphone estimates the trajectory and shows it on the screen. Note that all calculations are carried out in the smartphone

- Overcrowded environment: This environment has many static and dynamic objects. The attached video illustrates the degree of crowdedness of this environment.
- Small room: When a robot navigates a small room, the amount of information that it can detect is small. Therefore, the DLmark detection is sensitive to the existence of various objects compared to when in a simple corridor.
- Partially curved environment: This environment generally consists of linear structures. However, some parts are curved. In this environment, it is possible to extract wrong features from the curved parts.

We used the same parameter setting for all these types of environments and identified that all the results satisfied the global consistency.

Parameters

In this subsection, we explain all parameters used in the proposed algorithm.

(Sensor parameter)

– SensorStdRange = 0.01 m: Standard deviation of sensor range data.
– SensorStdBearing = 0.1°: Standard deviation of sensor bearing data.

These parameters are set by considering the performance of the sensor used. In the case of using general laser range finder, these values are valid.

(Odometry parameter)

– OdoRotStd = 5°: The maximum standard deviation of odometry rotation between adjacent steps.

This parameter means the maximum angle error expected when the robot moves one step.

(DLmark detection algorithm parameter)

– SplitDistErrThres = 0.05 m: Minimum distance between a line segment and a most distant point for conducting split procedure.

Table 4 Quantitative results of three datasets on the translation error (2D)

	Translation error m (abs)/m^2 (sqr)	① Graph mapping	② Proposed alg.
Ⓐ	(h) Freiburg bldg 79		
	Average of absolute errors	0.056 ± 0.042	0.263 ± 0.352
	Average of squared errors	0.005 ± 0.011	0.193 ± 0.404
	Maximum absolute error of a relation	0.459	1.489
Ⓑ	(i) Aces		
	Average of absolute errors	0.044 ± 0.044	0.094 ± 0.277
	Average of squared errors	0.004 ± 0.009	0.085 ± 0.483
	Maximum absolute error of a relation	0.347	2.224
Ⓒ	(j) MIT Killian court		
	Average of absolute errors	0.050 ± 0.056	0.141 ± 0.275
	Average of squared errors	0.006 ± 0.029	0.095 ± 0.289
	Maximum absolute error of a relation	0.765	1.380
Ⓓ	Mean of absolute errors for 3 datasets	0.05	0.17

– SplitNPointThres = 5: Mininum number of points that can make a line segment.
– SplitLengthThres = 0.1 m: Minimum length of a valid line segment.
– FilterThresSameGroup (β) = 5°: The maximum angular difference for admitting as the same group.
– FilterMinLengthThres = 2 m: The minimum descriptor of a DLmark.

(Data association parameter)

– DataAssoMinAngDiff (α) = 35°: The minimum angular difference among DLmarks of the environment that can be applied to DL-SLAM algorithm.

We found the proper value by considering the performance of sensor and the error of odometry. Most of the ordinary indoor spaces satisfy this condition.

5.1.2 Accuracy

Global translation error

The *accuracy* of the global rotation can be showed through the results of the occupancy grid mapping shown in Figs. 7, 8, and 9. On the other hand, the translation error of each trajectory should be analyzed quantitatively because DL-SLAM only uses directional features as landmarks. We used two methods to analyze the result of DL-SLAM in terms of translation error. First, we measured the translation error of the relative displacement between the beginning and the end of the trajectory using the CAD information of datasets (a) and (l), which did not have a loop closure in their trajectories. Second, we measured the translation error at the loop closure position of the trajectory for the other datasets. For these two methods, the positions for measuring the translation error are indicated by blue circles in the map of DL-SLAM (Figs. 7, 8, 9 third column). The errors measured using these two methods are organized in columns 5 and 6 of Table 3. The average of the translation errors for the 15 datasets was 1.16 m, and the rate for the travel length was 0.40%. It means that a 40-cm translation error occurred per 100 m on average.

Additionally, we conducted the comparison with the previous algorithm by using the representative paper [31] which is used for comparison of the accuracy for SLAM algorithm. Table 4 shows the translational error, and Table 5 shows the rotational error. In the paper [31], 6 datasets are used for evaluation. Among those, we only used 3 indoor datasets that are possible of directional segmentation. Maximum absolute error for both translational and rotational errors of DL-SLAM is much larger than that of Graph Mapping. Especially, the rotational error is very large value of 22.678° (Table 5-A2). This large error is measured at the step which does not conduct correction by using registered landmarks. That is, it is natural for EKF framework. D row of Table s4 and 5 shows the mean value for the 3 absolute errors. The difference between Graph Mapping and DL-SLAM is 0.12 m which can be considered to small error even though DL-SLAM algorithm only uses directional feature.

These quantitative results show two things. First, the local translation error of a wheel-based odometry is small in practice. Second, the global translation error due to the global rotation error is minimized. In conclusion, the greater part of the global translation error is erased by the DL-SLAM algorithm.

Odometry error

The *accuracy* of wheel-based odometry is determined by various conditions: floor condition, collision with obstacles, accuracy of wheel encoders, length of trajectory, and moving velocity of a robot. In this paper, we assert that the greater part

Table 5 Quantitative results of three datasets on the rotation error (2D)

	Rotation error deg (abs)/deg^2 (sqr)	① Graph mapping	② Proposed alg.
Ⓐ	(h) Freiburg bldg 79		
	Average of absolute errors	0.6 ± 0.6	1.410 ± 1.869
	Average of squared errors	0.7 ± 1.7	5.478 ± 27.945
	Maximum absolute error of a relation	5.4	28.979
Ⓑ	(i) Aces		
	Average of absolute errors	0.4 ± 0.4	0.630 ± 0.730
	Average of squared errors	0.3 ± 0.8	0.929 ± 3.159
	Maximum absolute error of a relation	3.5	7.7
Ⓒ	(j) MIT Killian court		
	Average of absolute errors	0.5 ± 0.5	0.679 ± 0.870
	Average of squared errors	0.9 ± 0.9	1.218 ± 6.976
	Maximum absolute error of a relation	5.4	16.2
Ⓓ	Mean of absolute errors for 3 datasets	0.50	0.91

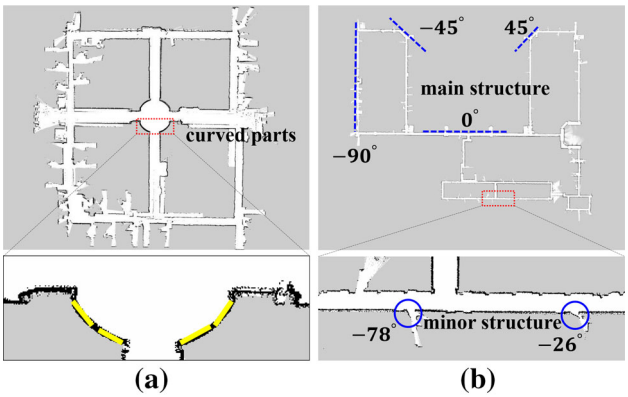


Fig. 12 Non-DLmarks can be detected from **a** partially curved parts or **b** non-dominant directions that are extracted not from the main but from a minor structure of the environment

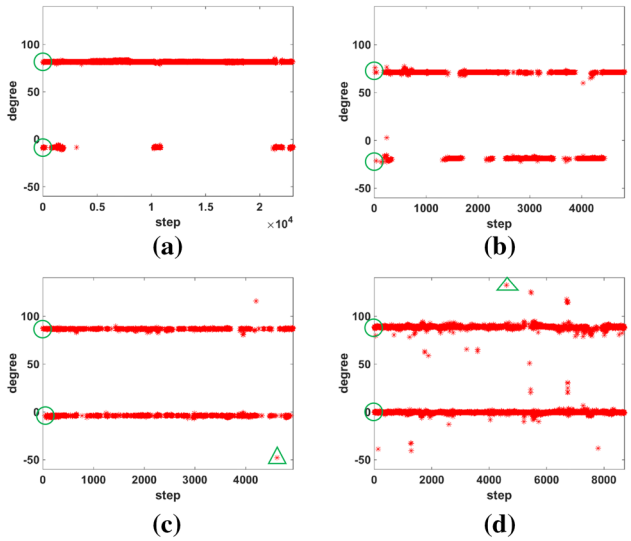


Fig. 13 We marked all the extracted features at each robot position to visualize the performance of the DLmark detection for **a** simple corridor, **b** partially curved environment, **c** small room, **d** overcrowded environment. *Green circles* and *green triangles* indicate the registration spot of the DLmarks and the spurious landmarks, respectively (colour figure online)

of the odometry error is caused by the global rotation error regardless of the variation of these conditions. Our extensive experiments using datasets from various mobile robot platforms, Web sites, lengths of trajectories, and indoor environments support the assertion.

Effect of the DLmark detection rate

The global rotation error causes the global translation error. Therefore, the DLmark detection rate should be high. The average detection rate for the 15 datasets was 96.18%, and this high rate made the global translation accurate. Note that the detection rate means the ratio of the number of steps that have extracted DLmarks more than once over the total number of steps.

False-positive data association

Figure 12a illustrates the directional features that were detected in the curved part of dataset (i), whereas Fig. 12b illustrates the non-dominant directions of dataset (j). These were not DLmarks because the directions were not extracted from the main structure of the environment. Therefore, the features acted as false positives in DL-SLAM. Figure 13 illustrates all the extracted features at each robot position. From the figure, it can be seen that the registered DLmarks were detected consistently for most of the datasets. However, there were some scattered features that are different from the DLmarks. These features were detected either from partially curved parts or from non-dominant directions. We call such features as non-DLmarks, which are not located in the range of $\pm 5°$ according to the ideal DLmark. The non-DLmarks acted as false positives in the data association process, and a false-positive rate is defined as follows:

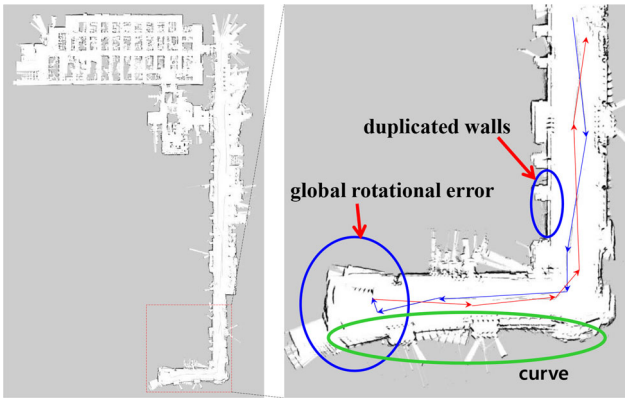


Fig. 14 Non-DLmarks detected from the curved part act as false positives that cause a global translation error. The error is identified in the form of duplicated walls on the map

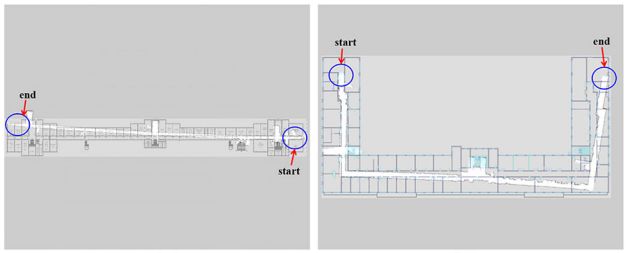


Fig. 15 **a** and **b** Results of a general SLAM algorithm for a loop-less trajectory. The SLAM algorithm cannot reduce the error at the end position because there is no physical loop

$$\text{false-positive rate} = \frac{\#\text{of false positives}}{\#\text{of all the extracted features}} \tag{74}$$

For the 15 datasets, the false-positive rate calculated by (74) was 0.37% on average and 3.53% at maximum. These non-DLmarks caused a global rotation error, which, in turn, caused a global translation error. Figure 14 illustrates the duplicated walls due to the non-DLmarks that were detected in the curved part. Nevertheless, the DL-SLAM algorithm can keep the global consistency owing to the two characteristics of the algorithm. First, at least one of the DLmarks is registered at the initial step. That is, if a DLmark is detected once again, the rotation error is diminished to the initial error value like a loop closure. Second, these loop-closing effects occur frequently. In Fig. 13, it can be seen that at least one of the DLmarks was detected in almost all steps. Therefore, DL-SLAM can maintain the global consistency even though there are wrong updates owing to the false positives.

Loop-less trajectory

The trajectories of datasets (a) and (l) did not have any loop. For these datasets, the estimate error of the general SLAM algorithm increased as time went on. Figure 15 illustrates the results of the open SLAM algorithm (GMapping) [32].

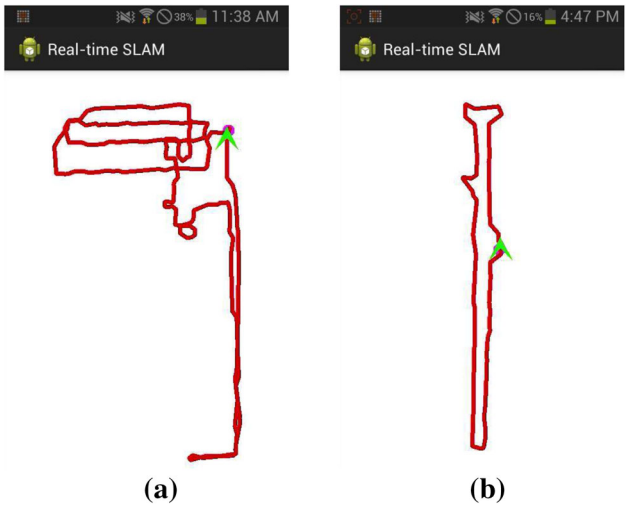


Fig. 16 These figures illustrate the trajectories that were made with DL-SLAM in real time on an Android-based smartphone. The *green arrow in each trajectory* indicates the current position and rotation of a robot. **a** KINTEX hall, **b** Gangnam Station (colour figure online)

The figure illustrates the growing estimate error as time went on. On the other hand, the results using DL-SLAM were accurate as shown in Figs. 7a and 9l because loop closing with respect to the direction occurred almost continuously even though there were no physical loops.

Environments that cannot use DL-SLAM

Datasets (p) to (t) were used to indicate the types of environments where the DL-SLAM algorithm cannot be used. Figure 10 illustrates such environments. The assumption of DL-SLAM is that the environments have distinguishable directional patterns. The datasets in Fig. 10 had directional patterns. However, these patterns are not distinguishable. Similar but different directional patterns caused a wrong matching among DLmarks, and the results of DL-SLAM diverged as shown in the far-right column of Fig. 10.

5.1.3 Lightness

In general, the number of landmarks increases as the coverage that a robot has to manage increases [33]. In other words, the quantity of the information to manage grows as time goes on. Therefore, when the coverage is vast, it is difficult to keep up in real time. On the other hand, DL-SLAM uses only four landmarks for dataset (j), whose coverage was about 250 (w) $\times$ 200 (h). That is, it was verified that the number of landmarks was small regardless of the coverage. The smartphone experiments showed that our algorithm is lightweight enough to operate in real time for a low-end hardware. The results were plotted on the screen of the smartphone in real time as shown in Fig. 16.

5.2 3D experiments

We used 12 datasets acquired from three sensor systems to verify the proposed 3D DL-SLAM algorithm. The sensor systems are shown in Fig. 17. Note that Fig. 18 shows the procedure of using the 3D Velodyne LIDAR as an initial estimate. The information about the 12 datasets that were used for the experimental verification is explained in Table 6. The 10 datasets (a to j) mapped in Figs. 19, 20 and 21 were used to show the performance of 3D DL-SLAM, and 2 datasets (k and l) mapped in Fig. 22 were used to indicate the types of environments where 3D DL-SLAM cannot be used. Bird-eye views of the 3D mapping results for 10 datasets (a to j) are shown in Fig. 23. The analysis method is the same as the analysis of 2D experimental results. In this section, we analyze the 3D experimental results in terms of the three requirements as follows.

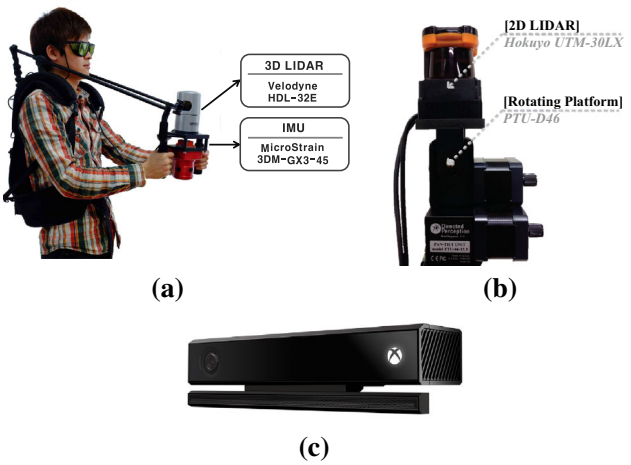


Fig. 17 Three sensor systems for 3D experiments three sensor systems that are used for acquiring the datasets for experiments. **a** 3D Velodyne LIDAR (max. range 80 m), **b** 2D hokuyo LIDAR (max. range 30 m) attached to the top of the pan-tilt unit, **c** Kinect II (max. range 8 m)

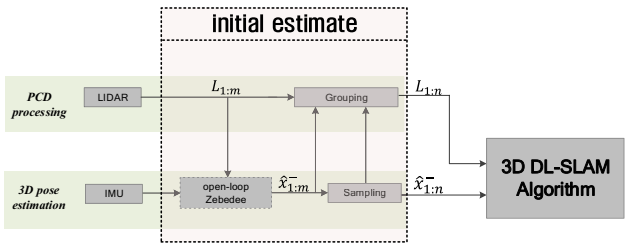


Fig. 18 Flowchart of using Velodyne Open-loop Zebedee algorithm estimates a continuous trajectory by using continuous depth data acquired from 3D LIDAR and IMU data. To extract geometric features from the depth data, the continuous trajectory and depth data are sampled and grouped, respectively

Table 6 Information about the 12 datasets that are used for 3D experiments information about the 12 datasets that are used for experiments

	Dataset name	Environmental property
a	KU Central Plaza[a]	Partially curved
b	Gangnam Station[a]	Overcrowded
c	KU Hana-Science Bld.[a]	Multi-story
d	KITECH Factory[a]	Unstructured
e	K Hotel[a]	Complex structure
f	KU 2nd Engr.[a]	Narrow corridor
g	KINTEX Hall[a]	Wide exhibit hall
h	KU Innovation Hall[b]	Narrow corridor
i	KU R.C. Bld. Corridor[b]	Narrow corridor
j	KU R.C. Bld. Office[c]	Simple room
k	KINTEX Corridor[a]	Unusual structure
l	KU Subway Station[a]	Curved

The superscripts explain the sensor system used for each dataset
[a] 3D Velodyne, [b] 2D hokuyo + pan-tilt unit, [c] kinect

5.2.1 Adaptability

We used various types of environments and three types of sensor system to show the adaptability of 3D DL-SLAM. In conclusion, the 3D DL-SLAM also can be used in various environments with the same parameter setting. However, if the computational time is important consideration, parameter tuning is necessary. When the Velodyne LIDAR sensor is used, the average number of 3D points per step is more than 943,580, while the average number of 3D points of kinect depth data and tilted hokuyo is 229,906. The second column of Table 6 shows the various types of environments. The experimental results show that the 3D algorithm can also be used in various environments like 2D case. The additional type of environment is multi-story space. Figure 24 shows the usefulness of the 3D DL-SLAM algorithm for the multi-story building.

(Parameters)
(Initial estimate parameter) InitialRotStd3D $= 10^{-3}$: The maximum standard deviation of each component of relative quaternion between adjacent steps. For 2D SLAM, the robot rotation is expressed by one parameter. However, for 3D SLAM four parameters are needed. Therefore, the uncertainty is expressed by 4 x 4 covariance matrix. The final covariance matrix is $(\text{InitialRotStd3D})^2 \times \boldsymbol{I}_{4\times4}$

(3D DLmark detection algorithm parameters)
FilterThresSameGroup $= 5°$ (common for 2D/3D): The maximum angle difference for admitting as the same group. FilterMinAreaThres $= 5\,\text{m}^2$: The minimum area descriptor of a DLmark. All parameters about split-and-filter are set by experiments.

(Data association parameter)

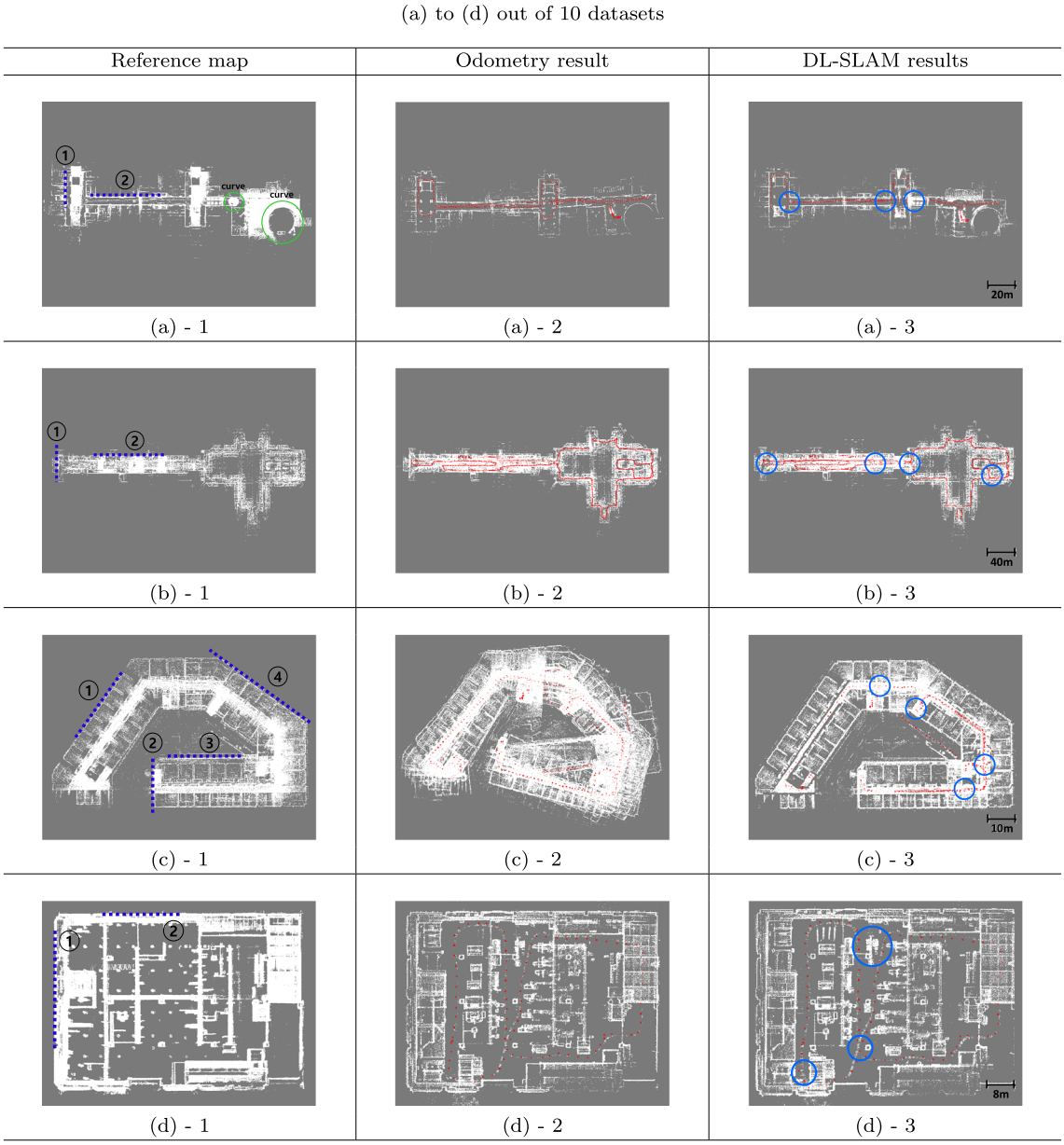


Fig. 19 Top views to show the performance of the 3D DL-SLAM (I)

DataAssoMinAngDiff = 35° (common for 2D/3D): The minimum angle difference among DLmarks of the environment that can be applied to DL-SLAM algorithm.

5.2.2 Accuracy

A. Global translation error

The global translation error of 3D DL-SLAM is analyzed with the same method of 2D case, and the results are organized in Table 7. The average translation error rate for travel length was 0.14%. It means that 0.14 m translation error occurred per 100 m on average.

B. Odometry error

As explained in the 2D experimental results, DL-SLAM can only correct the local and global rotation error. Therefore, the accumulated local translation error is remained. Experimental results show that the initial estimates of open-loop Zebedee and plane registration with tilted hokuyo have small translation error. While the initial estimate using kinect has large translation error even though the feature detection rate is high enough to correct almost translation error due to global rotation error.

C. Effect of the DLmark detection rate

The degree of freedom (DoF) of 3D rotation is three, and the DoF of one 3D DLmark is two. Therefore, the DL-SLAM

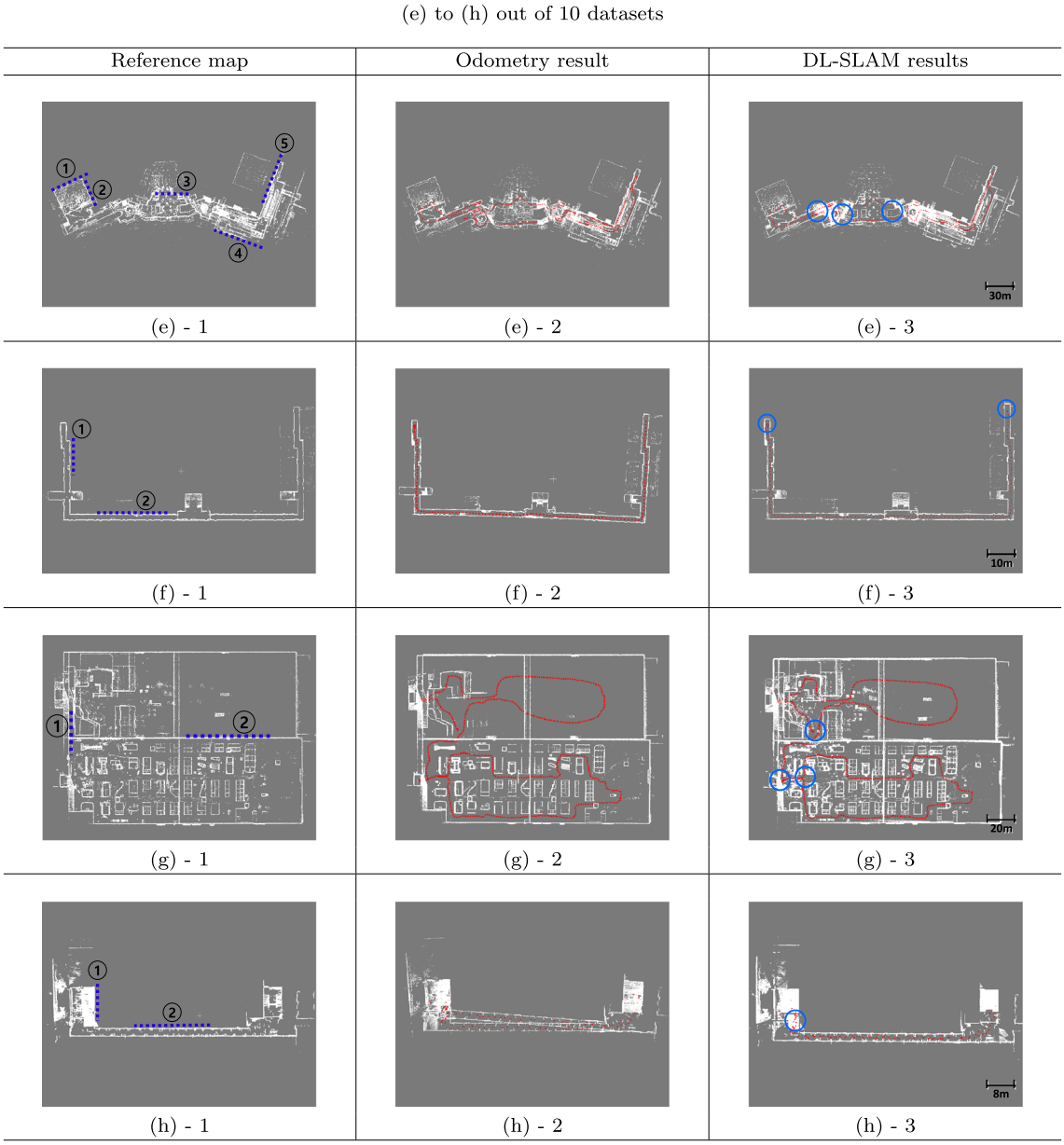


Fig. 20 Top views to show the performance of the 3D DL-SLAM (II)

algorithm needs more than two DLmarks to correct fully the 3D rotation. Table 8 shows the feature detection rate for two cases. The rate (88%) of detecting more than two features is enough high even though the rate is lower than the case of more than one feature. This means that the proposed feature detection algorithm can extract directional information in various environments. Also, the low false-positive rate (0.67%) also shows the accuracy of the algorithm.

D. Environments that cannot use DL-SLAM

Figure 22 shows the environments where the 3D DL-SLAM cannot be used. The KINTEX corridor dataset was used in 2D experimental analysis, and the result was good. However, for 3D case, the environment is classified to bad result. This is because of the characteristic of the environment. KINTEX corridor is the vast exhibit hall. Therefore, there are many large interiors such as planes of large area in the building. When the 2D sensor is used, the feature detection is not affected these kinds of interior. However, when the 3D sensor is used, the various interior was detected as other structures. For this reason, KINTEX corridor dataset shows different conclusion for 2D and 3D, respectively.

The second environment that cannot be used for 3D DL-SLAM is KU Subway Station dataset. The first column of

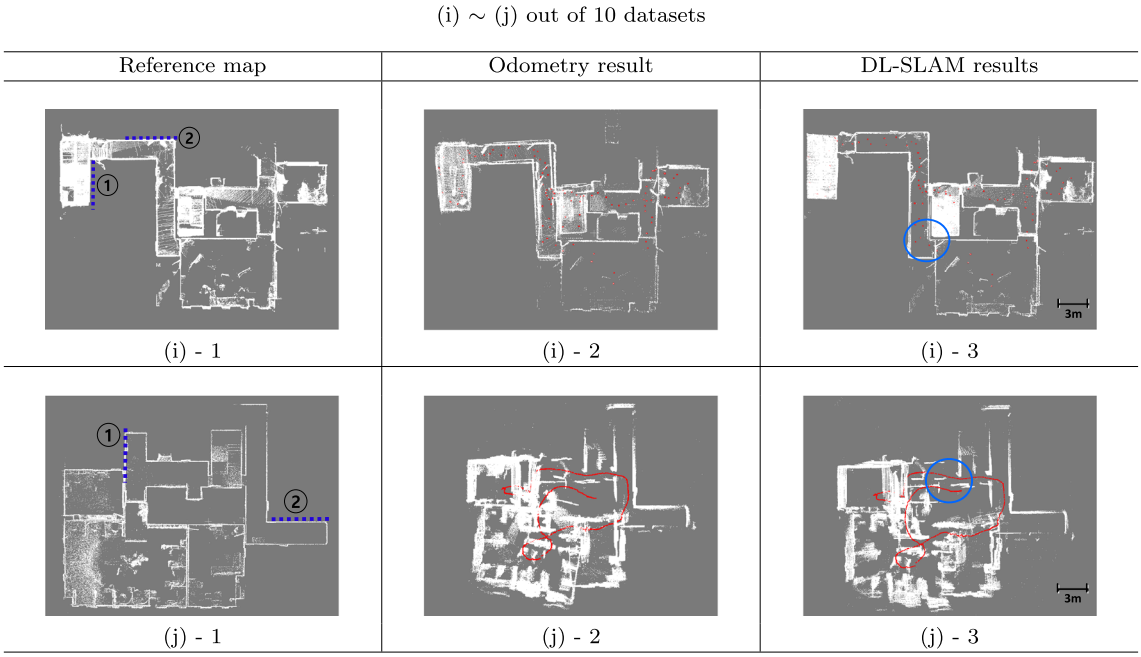


Fig. 21 Top views to show the performance of the 3D DL-SLAM (III)

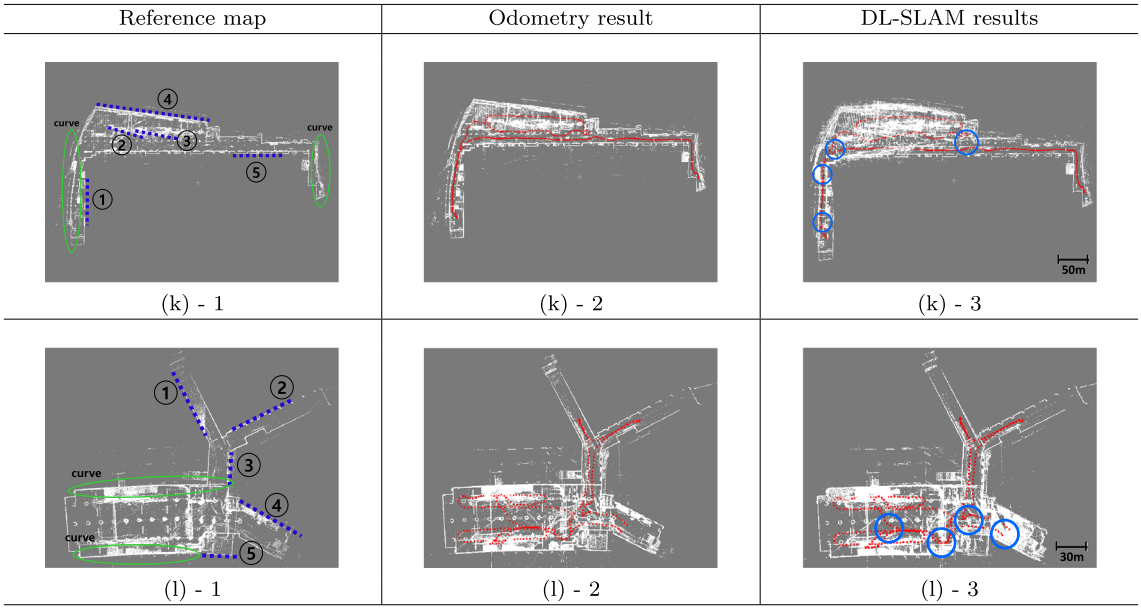


Fig. 22 Top views to show the performance of the 3D DL-SLAM (IV)

Fig. 22 shows the reference map of this dataset. Through the figure, it is difficult to distinguish the curved part and the linear part. However, the parts indicated by ellipses are actually cured parts and the length of the parts is long. Therefore, the 3D DL-SLAM cannot be used in this environment.

5.2.3 Lightness

In the case 2D DL-SLAM, we showed its lightness through the real-time smartphone experiments. However, we could not make C language version of the 3D DL-SLAM. Instead,

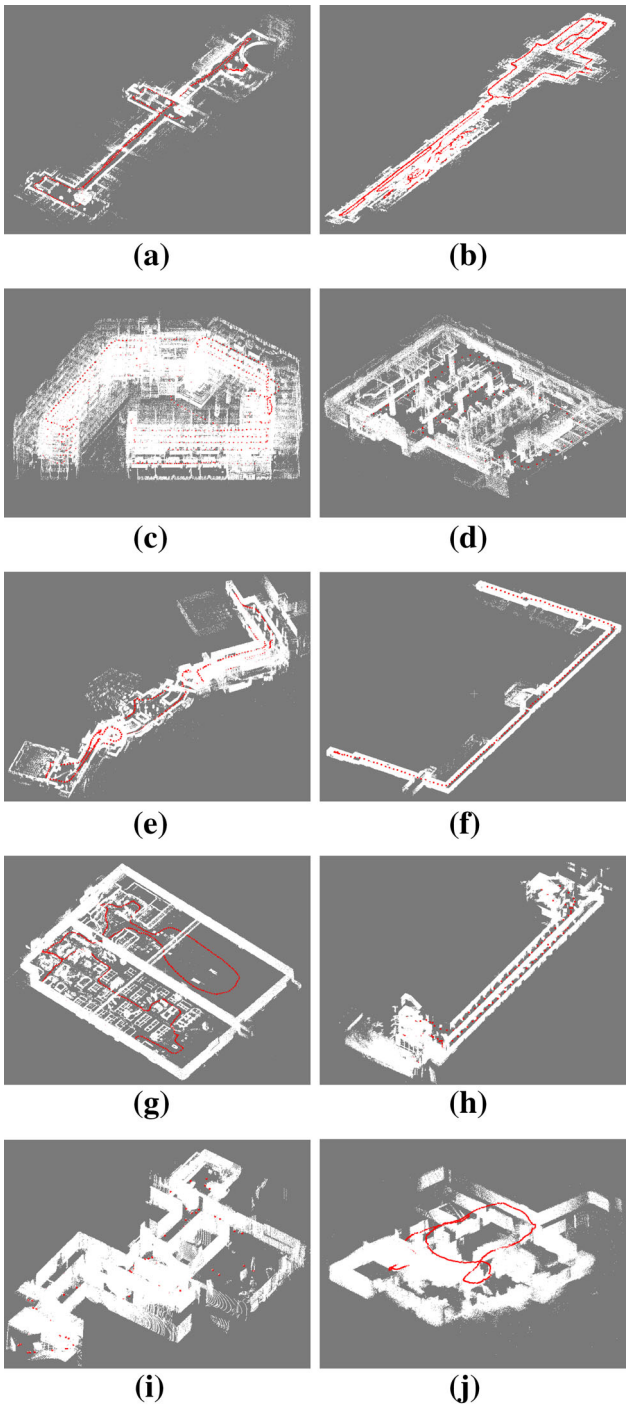


Fig. 23 Ten bird-eye views of the 3D mapping results that were made by the proposed 3D algorithm

we analyzed the processing time of MATLAB source code. The hardware specification of the computer that is used for these experiments is as follows: Intel Core i7-6700 3.40 GHz CPU, 16 GB RAM. Table 9 shows the average number of 3D

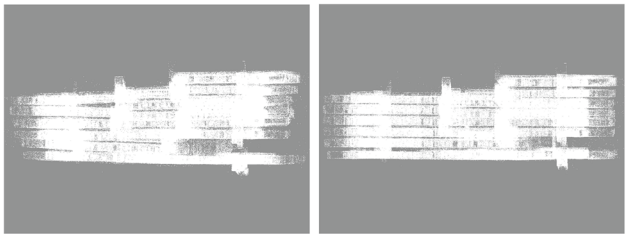


Fig. 24 Side views of 3D mapping results that are made by **a** initial estimate, **b** 3D DL-SLAM. The proposed 3D algorithm can be used usefully in multi-story building

Table 7 Quantitative analysis results to show the accuracy of the 3D DL-SLAM

① Dataset index	② Travel length (m)	③ Translation error (m)	④ Translation error rate for travel length (%)
a	510	1.05	0.21
b	1862	0.43	0.02
c	1582	0.45	0.03
d	157	0.21	0.14
e	821	0.48	0.06
f	183	1.60	0.87
g	832	1.81	0.22
h	186	0.35	0.19
i	109	0.13	0.12
j	52	2.03	3.90
Average value	629.4	0.85	0.14

Table 8 Quantitative analysis results for 3D DLmark detection algorithm

① Dataset index	② Detection rate (%) (≥ 1)	③ Detection rate (%) (≥ 2)	④ False-positive rate (%)
a	100	61.82	2.67
b	100	84.50	0.40
c	99.91	94.22	0
d	100	90.22	0
e	100	99.19	0
f	100	100	0
g	100	87.85	0.40
h	97.22	94.44	0
i	98.11	80.19	0
j	100	91.97	3.25
Average value	99.52	88.44	0.67

The second and the third columns show the number of steps which are detected more than one feature and more than two features, respectively

Table 9 Average processing time for each step and the number of average 3D points that are used for feature detection

① Dataset index	② # of 3D points per step	③ Detection time (s)	④ DL-SLAM time (s)	⑤ Total time (s)
a	794,880	2.28	0.44	2.72
b	1,069,400	4.34	1.34	5.68
c	804,300	1.92	0.15	2.07
d	1,096,100	4.20	0.23	4.43
e	972,260	3.02	0.99	4.01
f	763,420	1.42	0.25	1.67
g	1,104,700	3.86	0.50	4.36
h	338,870	0.39	0.09	0.48
i	133,760	0.17	0.06	0.23
j	217,088	0.31	0.10	0.41
Average value	729, 478	2.19	0.42	2.61

points per step and the processing time of feature detection and SLAM algorithm. The average processing time was 2.61 seconds. The reason why the time is not small is that the results include the case of using large point cloud such as Velodyne. In the case of kinect and tilted hokuyo, the average processing time was 0.37 seconds. This result means that it is possible to use the 3D algorithm in real time when the C code is implemented. Note that the kinect dataset was acquired at 10Hz.

6 Conclusion

In this paper, we proposed a practical 2D/3D SLAM algorithm for indoor environments that have distinguishable directional patterns extracted from a structure. The heart of DL-SLAM is the robust DLmark detection algorithm, which fully utilizes the property of an indoor structure by calculating the descriptors for each direction, and it enables the SLAM algorithm to have practicality.

Experiments were conducted using 27 datasets, which were composed of various environments, and it was shown that DL-SLAM showed the practicality. We also identified the types of environments where DL-SLAM cannot be used, such as those whose directional patterns are not distinguishable. Therefore, users have to know whether the directional patterns of the environment are distinguishable or not before using DL-SLAM. This is a limitation of the proposed algorithm. However, users can easily identify the directional patterns from the guide map of the building or from satellite images of Google Earth. Therefore, we can still say that DL-SLAM is useful in practice.

Additionally, in partially curved environments, we observed false positives that caused a global rotation error. In the future, we will find a method for discarding the directional features extracted from the curved part automatically to improve the accuracy of the proposed algorithm.

Acknowledgements This paper is supported by three Korean government funds of 10073166, 2012M3A6A3055700, and 2014K1A3A1A19 067398.

References

1. Almansa A, Desolneux A, Vamech S (2003) Vanishing point detection without any a priori information. IEEE Trans Pattern Anal Mach Intell 25(4):502–507
2. Wildenauer H, Vincze M (2007) Vanishing point detection in complex man-made worlds. In: International conference on image analysis and processing, pp 615–622
3. Tardif JP (2009) Non-iterative approach for fast and accurate vanishing point detection. In: IEEE international conference on computer vision, pp 1250–1257
4. Nguyen V, Harati A, Martinelli A, Siegwart R (2006) Orthogonal SLAM: a step toward lightweight indoor autonomous navigation. In: IEEE/RSJ international conference on intelligent robots and systems, pp 5007–5012
5. Lee YH, Nam C, Lee KY, Li YS, Yeon, SY, Doh NL (2009) VPass: algorithmic compass using vanishing points in indoor environments. In: IEEE/RSJ international conference on intelligent robots and systems, pp 936–941
6. Antone ME, Teller S (2000) Automatic recovery of relative camera rotations for urban scenes. In: IEEE conference on computer vision and pattern recognition, pp 282–289
7. Košecká J, Zhang W (2002) Video compass. In: European conference on computer vision, pp 476–490
8. Schindler G, Dellaert F (2004) Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In: IEEE conference on computer vision and pattern recognition, vol 1
9. Martins AT, Aguiar PM, Figueiredo MA (2005) Orientation in manhattan: equiprojective classes and sequential estimation. IEEE Trans Pattern Anal Mach Intell 27(5):822–827
10. Bazin J-C, Demonceaux C, Vasseur P, Kweon I (2012) Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment. Int J Robot Res 31(1):63–81
11. Beall C, Ta DN, Ok K, Dellaert F (2012) Attitude heading reference system with rotation-aiding visual landmarks. In: International conference on information fusion, pp 976–982
12. Kawanishi R, Yamashita A, Kaneko T, Asama H (2012) Line-based camera movement estimation by using parallel lines in omnidirectional video. In: IEEE international conference on robotics and automation. IEEE, Saint Paul, MN, USA
13. Elloumi W, Guissous K, Chetouani A, Canals R, Leconge R, Emile B, Treuillet S (2013) Indoor navigation assistance with a smartphone camera based on vanishing points. In: International conference on indoor positioning and indoor navigation. IEEE, Montbeliard-Belfort, France
14. Bosse M, Rikoski R, Leonard J, Teller S (2003) Vanishing points and three-dimensional lines from omni-directional video. Vis Comput 19(6):417–430
15. Zhang G, Kang DH, Suh IH (2012) Loop closure through vanishing points in a line-based monocular SLAM. In: IEEE international

conference on robotics and automation. IEEE, Saint Paul, MN, USA, pp 4565–4570
16. Scheggi S, Morbidi F, Prattichizzo D (2013) Uncalibrated visual compass from omnidirectional line images with application to attitude MAV estimation. In: IEEE/RSJ international conference on intelligent robots and systems. IEEE, Tokyo, Japan, pp 1602–1607
17. Camposeco F, Pollefeys M (2015) Using vanishing points to improve visual-inertial odometry. In: IEEE international conference on robotics and automation. Seattle, Washington, pp 5219–5225
18. Yeon S, Jun C, Choi H, Kang J, Yun Y, Doh NL. Robust-PCA-based hierarchical plane extraction for application to geometric 3D indoor mapping. Industrial Robot (accepted)
19. Choi Y, Lee T, Oh SY (2008) A line feature based SLAM with low grade range sensors using geometric constraints and active exploration. Auton Robots 24:13–27
20. Pfister ST (2006) Algorithms for mobile robot localization and mapping, incorporating detailed noise modeling and multi-scale feature extraction. PhD thesis, California Institute of Technology
21. Cheon YJ, Kim JH (2007) Unscented filtering in a unit quaternion space for spacecraft attitude estimation. In: IEEE international symposium on industrial electronics (ISIE). IEEE, Vigo, Spain, pp 66–71
22. Dissanayake M, Newman P, Clark S, Durrant-Whyte H, Csorba M (2001) A solution to the simultaneous localization and map building problem. IEEE Trans Robot Autom 17(3):229–241
23. Huang GP, Mourikis A, Roumeliotis SI (2010) Observability-based rules for designing consistent EKF SLAM estimators. Int J Robot Res 29(5):502–528
24. Corke P (2011) Robotics, vision and control. Springer, Berlin
25. http://euclideanspace.com
26. Julier SJ, Uhlmann JK, Durrant-Whyte H (2000) A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Trans Autom Control 45(3):477–482
27. Howard A, Roy N (2003) The robotics data set repository (radish)
28. Kümmerle R, Steder B, Dornhege, C, Ruhnke M, Grisetti G, Stachniss C, Kleiner V (2009) Slam benchmarking webpage
29. Bonarini A, Burgard W, Fontana G, Matteucci M, Sorrenti DG, Tardós JD (2006) Rawseeds: robotics advancement through web-publishing of sensorial and elaborated extensive data sets. In: IEEE/RSJ international conference on intelligent robots and systems
30. http://www.rawseeds.org
31. Kümmerle R, Steder B, Dornhege C, Ruhnke M, Grisetti G, Stachniss C, Kleiner A (2009) On measuring the accuracy of SLAM algorithms. Auton Robots 27:387–407
32. http://www.openslam.org/gmapping.html
33. Guivant JE, Nebot EM (2001) Optimization of the simultaneous localization and map-building algorithm for real-time implementation. IEEE Trans Robot Autom 17(3):242–257