A Monocular Vision Sensor-Based Efficient SLAM Method for Indoor Service Robots

Tae-jae Lee, Chul-hong Kim, and Dong-il Dan Cho, *Member, IEEE*

***Abstract*—This paper presents a new implementation method for efficient simultaneous localization and mapping using a forward-viewing monocular vision sensor. The method is developed to be applicable in real time on a low-cost embedded system for indoor service robots. In this paper, the orientation of a robot is directly estimated using the direction of the vanishing point. Then, the estimation models for the robot position and the line landmark are derived as simple linear equations. Using these models, the camera poses and landmark positions are efficiently corrected by a local map correction method. The performance of the proposed method is demonstrated under various challenging environments using dataset-based experiments using a desktop computer and real-time experiments using a low-cost embedded system. The experimental environments include a real home-like setting. These conditions contain low-textured areas, moving people, or changing environments. The proposed method is also tested using the robotics advancement through web publishing of sensorial and elaborated extensive datasets benchmark dataset.**

***Index Terms*—Efficient simultaneous localization and mapping (SLAM), embedded system, indoor service robot, monocular vision.**

I. Introduction

ONE of the goals in robotics is to develop a mobile robot that can act autonomously in the real world. Localization is the most important prerequisite for this goal. Without using a global positioning system or preconstructed ad-hoc infrastructures, localization can be performed using the sensors on board the robot [1], [2]. Among the various methods, the simultaneous localization and mapping (SLAM) technique provides an attractive solution because it does not need user-built maps. Within various sensors for SLAM, monocular vision is highly attractive because of its low cost, lightweight, and low power consumption, especially for resource-constrained robotic platforms. In addition, it can capture rich visual information from the environment.

During the last two decades, the vision-based SLAM problem has been intensively researched. However, in order for these SLAM solutions to be practical for real applications such as consumer robots, some important factors need to be considered. First, the computational requirement of the SLAM algorithm should be low enough to be executed on a low-cost microprocessor. A microprocessor on a robot should execute SLAM in parallel with other algorithms such as path and motion planning in real time to conduct various missions. The computational requirements of conventional SLAM algorithms are too high to be used on a low-cost processor. Second, the environment where a robot is used should be considered in developing and applying a SLAM method.

From the perspective of vision-based SLAM, the home environment is quite challenging. It contains plenty of less-textured areas. When the robot with a camera moves close to the obstacles or objects, all tracked features are easily lost because of occlusion and severe scale changes in the image domain. This situation occurs frequently for home-service robots, such as robotic vacuums, because they must move through every inch of the environment. Input images from robotic vacuums with a forward-viewing monocamera that moves through every inch of the house contain only an average of 151 corner features at 320×240 image resolution per image (see Section IV-A for a detailed explanation of the home dataset). When compared with popular benchmark datasets, this is a very challenging situation for SLAM. Kitti (outdoor, sequence 00) [3] and robotics advancement through web publishing of sensorial and elaborated extensive datasets (RAWSEEDS) (indoor, sequence 25 b frontal) [4], which are popular benchmark datasets, contain an average of 4443 corners at 1241×376 image resolution per image (731 for 320×240 image size ratio) and 657 corners at 320×240 image resolution per image, respectively. For corner detection, the algorithm [5] with the threshold 20 is used. Another challenge is that the home environment is highly dynamic because of human activities. There are moving people and objects that affect the performance of the SLAM. In addition, the environment can be changed during the SLAM process. For example, the locations of the objects or the illumination can be changed. Fig. 1 shows these challenging situations, which are present in the home dataset in this study.

Manuscript received July 21, 2017; revised October 25, 2017, February 12, 2018, and March 13, 2018; accepted March 24, 2018. Date of publication April 16, 2018; date of current version August 31, 2018. This work was supported in part by LG Electronics, Inc., Seoul, South Korea, and in part by Brain Korea 21 Plus Project funded by National Research Foundation of Korea. *(Corresponding author: Dong-il Dan Cho.)*

T.-j. Lee and C.-h. Kim are with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 151-742, South Korea, and also with the Automation and Systems Research Institute, Seoul National University, Seoul 151-744, South Korea (e-mail: ltj88@snu.ac.kr; chkim89@snu.ac.kr).

D.-i. D. Cho is with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 151-742, South Korea, and also with the Interuniversity Semiconductor Research Center and Automation and Systems Research Institute, Seoul National University, Seoul 151-744, South Korea (e-mail: dicho@snu.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2018.2826471

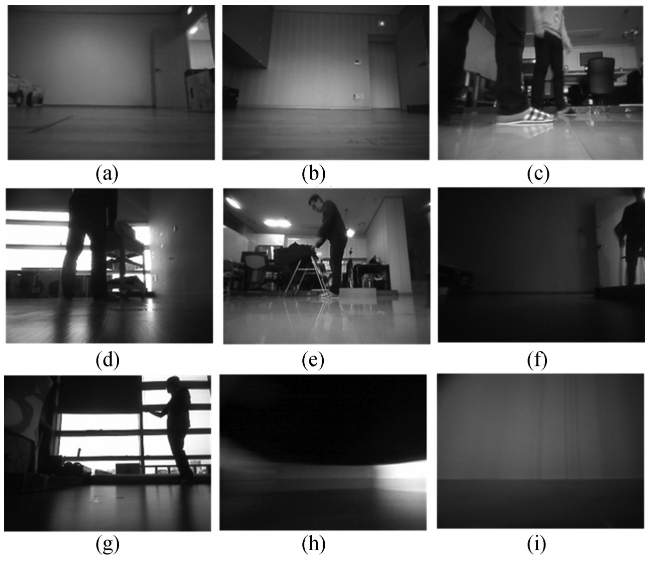

Fig. 1. Challenging situations from the home dataset in this work. Images are captured from a robotic vacuum with a forward-viewing monocular vision sensor. (a) and (b) Less-textured areas. (c) and (d) Moving people and object. (e) Moving person who hang out the wash. (f) Changing illumination when a person turns off the light. (g) Changing environment where a person pull down a roller blind. (h) No visual information when robot is too close to a wall. (i) No visual information when robot moved under a sofa.

This paper presents an efficient implementation method for forward-viewing monocular vision-based SLAM. The method is applicable on a low-cost embedded system for indoor service robots, such as home service robots. We have assumed that the robotic motions are planar, and the images captured by a forward-viewing monocamera and odometer are used as sensory inputs. The proposed method is quite robust in challenging indoor environments, which contain low-textured areas, moving people, or changing environments. For robust performance in challenging indoor environments, the proposed method adopts the vanishing point (VP) and orthogonal structure assumptions, which are mainly inspired by the work of Zhou *et al.* [6].

The main contribution of this paper is proposing a new implementation method using line features and VP for efficient SLAM. The details of the contribution are as follows. First, VP is utilized in both line landmark parametrization and direct estimation of the robot orientation which improves the accuracy. Second, new estimation models for robot's translation and landmarks are newly derived in a simple form for efficient implementation. Finally, using these models, robot poses and landmarks are separately corrected during the local map correction process, which significantly reduces the computational requirement of the proposed method such that it can be installed on a low-cost embedded system and integrated in a real-time autonomous robot navigation system.

The rest of this paper is organized as follows. In Section II, we present studies related to this topic. The proposed method is described in Section III. In Section IV, we present the experimental result. This is followed by conducting remarks and plans for future research in Section V.

II. Related Works

In the fields of robotics and computer vision, visual SLAM has been studied intensively. Researchers have mainly used feature points as a landmark for SLAM. Current SLAM methods can be classified into two categories: filtering-based methods [7], [8] and optimization-based methods [9], [10]. In filtering-based methods, every frame is processed by the filter to jointly estimate the camera pose and landmark positions in a probabilistic way. On the other hand, optimization-based methods estimate the camera pose and landmark positions in a deterministic way, usually through numerical optimizations, called bundle adjustments. The representative work of this kind is parallel tracking and mapping (PTAM) [9]. The PTAM method proposed the concept of tracking the camera pose and mapping the environment in two simultaneous threads. Recently, Mur-Artal *et al.* [10] proposed an oriented FAST and rotated BRIEF (ORB) point-feature-based monocular SLAM called ORB-SLAM. This method used the same ORB features for all SLAM tasks: tracking, mapping, relocalization, and loop closing.

Various graph optimization methods in the least-square sense for optimization-based SLAM methods are proposed. The g2o or iSAM2 are widely used methods [11], [12]. Recently, Khosoussi *et al.* [13] proposed an efficient graph optimization algorithm that takes advantage of the separable structure of SLAM for reliable and faster convergence. To deal with false constraints problem from loop closure detections or data association in graph optimization Vertigo, dynamic covariance scaling (DCS) and Max-mixture methods were proposed [14]–[16].

Several studies have been presented to cope with a dynamic environment. To deal with moving objects while performing SLAM, simultaneous localization, mapping and moving object tracking was proposed [17]. The method distinguishes stationary and moving points based on the difference of hypotheses and augments those points into SLAM state vector. Tan *et al.* [18] proposed a monocular vision sensor-based online keyframe representation and updating method to adaptively model the dynamic environments, where the appearance or structure changes can be effectively detected. Lee and Myung [19] proposed an error metric and node grouping rules to detect low dynamic situations for RGB-D SLAM. Li and Lee [20] proposed a static weighting method for edge points in RGB-D SLAM to reduce the influence of dynamic objects. Apart from these point methods, line feature-based SLAM is generally more robust for dynamic environments, because they are not steadily extracted in moving people or objects [6]. Several studies about image analysis can be used to remove the moving objects from images in a dynamic environment. Li *et al.* [21] proposed a foreground extraction algorithm by using superpixels. Li *et al.* [22] proposed a foreground extraction algorithm by using path alignment manifold matting. Moving objects can be excluded by removing the extracted foreground from the images.

Numerous studies have been presented on arbitrary three-dimensional (3-D) line feature-based SLAM [23]–[25]. These methods used the filtering method to formulate the SLAM problem where line segments are parameterized with two endpoints [23] or as an infinite line [24] in a small 3-D space. Recently, Zhang *et al*. [25] proposed a 3-D line-based stereo SLAM system. The method used the *Plücker* line coordinates for line parameterization.

In the indoor environment, several studies have used ceiling line features as landmarks for SLAM. They usually require a upward-viewing camera because data association, geometrical modeling, and implementation are much easier. In addition, they use corner features as additional landmarks for SLAM [26], [27]. On the other hand, the proposed method uses a forward-viewing camera, which has much technical difficulties in feature extraction and tracking. Nevertheless, a forward-viewing method has significant commercial advantages in that the same camera can be used for obstacle avoidance, home monitoring, or as a user interface.

Several studies have applied VP information to attitude estimation and SLAM as is done in this study. These studies usually used the orthogonal structure assumption. Huttunen and Robert [28] proposed a monocular camera-based orientation estimation method using the VPs. The method detects orthogonal VPs and estimates the three-axis orientation of the camera by using the assumption of structural regularity. Camposeco and Marc [29] proposed a visual-inertial odometry method that used VPs to reduce angular drift in camera pose estimations. The method used direction of VP observations to update the orientation of the camera and the directions of the VP. Zhou *et al*. [6] proposed visual SLAM using building structure lines called StructSLAM. In this method, the orthogonal structures and the directions of VPs are used for parameterizing the line landmarks. Ji *et al.* [30] proposed an RGB-D SLAM using VP and door plate in an indoor environment. The method used VP and door plate as landmarks to increase the stability of the SLAM process. Lee *et al*. [31] proposed an algorithmic compass which uses VP to estimate the heading information of a mobile robot in an indoor environment. Zhang *et al.* [32] proposed a VP-based loop closure method in line-based mono-SLAM. The method used VP to correct the heading angle and line landmark to correct poses.

In this study, we have utilized the VP to reduce the orientation error by assuming the structural regularity of the indoor environment. In addition, line landmarks are aligned with the directions of the VPs as in [6]. Unlike the previous studies, the robot poses and landmarks are separately corrected during the local map correction process. In order to implement this method, the estimation model for the robot's orientation, robot's translation, and landmarks are newly derived in a simple form. In addition, the robot angle measurements which are directly calculated from the VP are utilized in the local map correction method.

III. Methodology

In this work, the robot is assumed to move around the flat ground. As sensory inputs, images captured from a forward-viewing monocamera and odometry are used. The camera is slightly tilted 8.7° upward to provide various consumer services. The overall architecture of the proposed SLAM algorithm is shown in Fig. 2. The proposed method runs three threads in parallel as in [10] (i.e., tracking, mapping, and loop closing). To avoid redundant computation, new images are only captured when the robot moved more than a predefined distance or rotated more than a predefined angle from the previous frame based on the odometry data. The tracking thread extracts lines and VPs and estimates the robot's orientation from the VPs. From the data association of lines, line landmark observations are made, and the position of new landmarks are estimated. By using the VP and line landmarks, the proposed method can be made more robust for dynamic environments, because they are not steadily extracted in moving people or objects. The mapping thread creates new line landmarks using the estimated positions in the tracking thread and inserts them in a map database. Afterward, local map correction is conducted on an active window of selected frames. The loop-closing thread finds large loops for every input frame. After a loop is detected, a pose graph is optimized using relative pose constraints between frames.

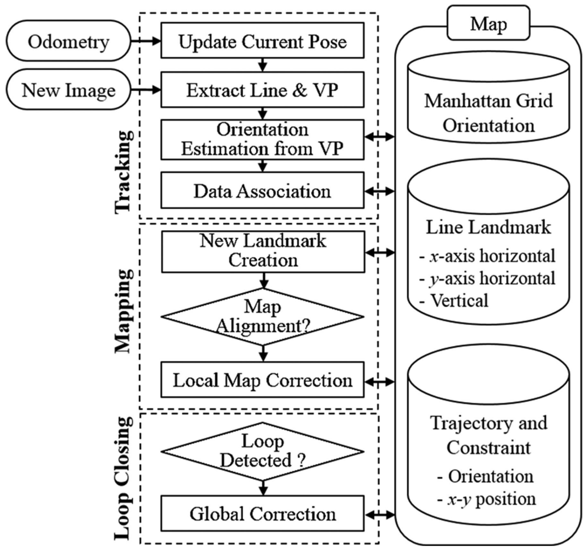

Fig. 2. Flowchart of the proposed SLAM algorithm.

A. Manhattan Frame and System Initialization

Most indoor environments can be abstracted as blocks that are stacked together in three dominant directions, which are referred to as a Manhattan grid [33]. This grid gives a natural reference frame for the viewer. The advantage of adopting the Manhattan world assumption is that both the robot's orientation error and the line landmark's orientation error can be eliminated. Under this assumption, the extracted VPs in the image plane correspond to the three dominant directions of the Manhattan grid.

From the first pose of the robot, the world frame is set according to the initial pose of the robot. At the system initialization step, the orientation of the Manhattan frame with respect to the world frame is estimated. To do this, the estimated angle of the Manhattan frame with respect to the world frame is averaged up to some predefined number. The initialization is conducted only when the variance of the orientation of the Manhattan frame with respect to the world frame is smaller than some predefined angle. In the initialization step, the robot poses are estimated using the odometry data only. The detailed method for estimating the angle between the robot and the Manhattan

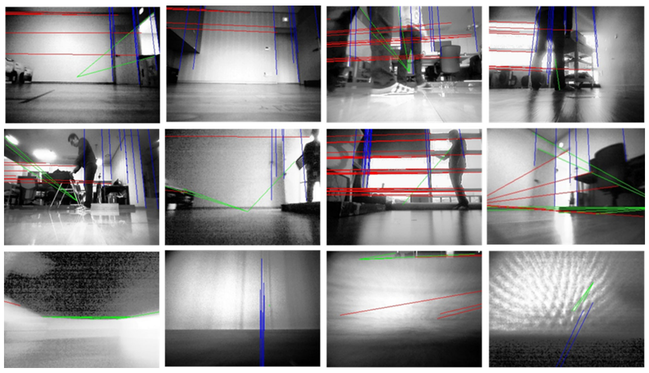
Fig. 3. Examples of VP extraction results in a typical home environment. The images of the third row are failure cases (no visual information when the robot moved under a sofa or the robot is too close to a wall).

frame is explained in the next section. Usually, typical indoor environments can be modeled using one Manhattan frame, but sometimes multiple Manhattan frames are required (known as "Atlanta world"). The proposed method resolves this situation by setting up several Manhattan frames. The additional Manhattan frame is configured when a dominant direction of the VP is changed in a partitioned grid area.

B. VP-Based Robot Orientation Estimation

To extract a VP, we extract the line segments using the line segment detector [34]. For a robust estimation of the VP, we eliminate line segments with lengths less than 15 pixels because short line segments tend to be noisy observations. Then, we extract the VP using the algorithm proposed by Zhang *et al.* [35]. Examples of VP extraction results in a typical home environment are illustrated in Fig. 3. The images shown in the third row are failure cases when the robot moved under a sofa or the robot is too close to a wall. Although the VP extraction fails in these situations, the local map correction method automatically handles the situation (detailed explanation in Section III-E). In this study, VPs of horizontal lines in the image are used to estimate the orientation between the robot and the Manhattan frame. These VPs are reliable when there are several lines appearing at the top of the image. Hence, images without any horizontal lines at the top region of the image are skipped.

From the VP extraction, we obtain three dominant orthogonal line direction vectors with respect to the camera frame. Then, the three direction vectors are transformed with respect to the robot frame. The robot rotates only with respect to its z-axis; therefore, the orientation of the robot with respect to the Manhattan frame can be calculated using the estimated VP direction vectors. First, from the three VP direction vectors, we find the one VP whose direction is closest to the y-axis of the robot frame. We call this ${}^{R}\mathbf{vp}_{y'}$, which is shown by a red arrow in Fig. 4. The direction vector ${}^{R}\mathbf{vp}_{y'}$ is parallel or orthogonal to the Manhattan frame, which can be considered as a wall in front of the robot. Second, we compute the robot's orientation with respect to the Manhattan frame structure by projecting the direction vector ${}^{R}\mathbf{vp}_{y'}$ to the xy plane of the robot frame. Finally, we calculate the robot's orientation with respect to the world frame.

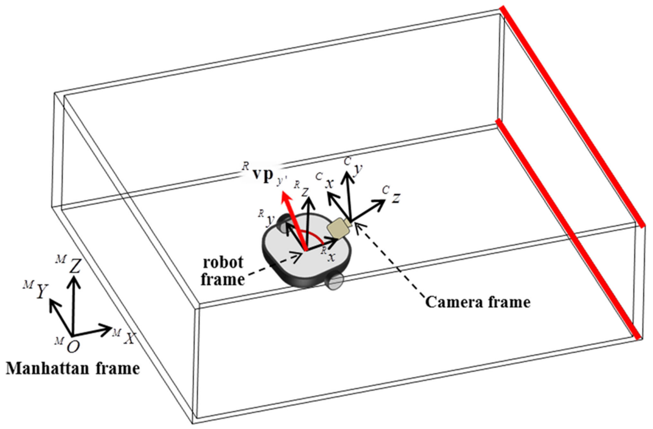

Fig. 4. Schematic diagram of the mobile robot coordinate system in a Manhattan grid.

The angle between the Manhattan frame and the world frame is known from the initialization; therefore, this angle is added to the robot's orientation in the Manhattan frame, resulting in the robot's orientation in the world frame.

C. Line Landmark Estimation

Considering the tilting angle θ_t of the camera, the camera matrix for point projection can be expressed as follows:

$$\mathbf{P} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} -\sin\theta_c & \cos\theta_c & 0 & x_c\sin\theta_c - y_c\cos\theta_c \\ -c_t\cos\theta_c & -c_t\sin\theta_c & s_t & c_t x_c\cos\theta_c + c_t y_c\sin\theta_c \\ s_t\cos\theta_c & s_t\sin\theta_c & c_t & -s_t x_c\cos\theta_c - s_t y_c\sin\theta_c \end{pmatrix} \tag{1}$$

where (f_x, f_y), (c_x, c_y), (x_c, y_c), θ_c, s_t, and c_t are focal length of the camera, principal point of the camera in image domain, the x–y position of the camera, the orientation of the camera, $\sin\theta_t$, and $\cos\theta_t$, respectively. Using the camera matrix, the line projection model can be expressed as follows [36]:

$$\tilde{\mathbf{l}} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} = \begin{pmatrix} (\mathbf{p}^{2T}\cdot\tilde{\mathbf{a}})\cdot(\mathbf{p}^{3T}\cdot\tilde{\mathbf{b}}) - (\mathbf{p}^{2T}\cdot\tilde{\mathbf{b}})(\mathbf{p}^{2T}\cdot\tilde{\mathbf{a}}) \\ (\mathbf{p}^{3T}\cdot\tilde{\mathbf{a}})\cdot(\mathbf{p}^{1T}\cdot\tilde{\mathbf{b}}) - (\mathbf{p}^{3T}\cdot\tilde{\mathbf{b}})(\mathbf{p}^{1T}\cdot\tilde{\mathbf{a}}) \\ (\mathbf{p}^{1T}\cdot\tilde{\mathbf{a}})\cdot(\mathbf{p}^{2T}\cdot\tilde{\mathbf{b}}) - (\mathbf{p}^{1T}\cdot\tilde{\mathbf{b}})(\mathbf{p}^{2T}\cdot\tilde{\mathbf{a}}) \end{pmatrix} \tag{2}$$

where $\tilde{\mathbf{l}}$ is the projected line in an image plane, $\tilde{\mathbf{a}}, \tilde{\mathbf{b}} \in \mathbb{R}^4$ are the two endpoints of the line, and $\mathbf{p}^{iT}$ is the ith row vector of camera matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$. As landmarks for SLAM, three types of line features are used in this work. These are composed of the vertical line, the x-axis horizontal line, and the y-axis horizontal line with respect to the Manhattan frame. To parameterize the line landmarks, both endpoints are used. For the vertical line, the endpoints $\tilde{\mathbf{a}}_v$ and $\tilde{\mathbf{b}}_v$ can be expressed as follows:

$$\tilde{\mathbf{a}}_v = \begin{pmatrix} x_v & y_v & z_1 & 1 \end{pmatrix}^T \text{ and } \tilde{\mathbf{b}}_v = \begin{pmatrix} x_v & y_v & z_2 & 1 \end{pmatrix}^T \tag{3}$$

where x_v and y_v are variables that we estimate during the SLAM. Variables z_1 and z_2 are calculated using the pin-hole camera model, and the observed line endpoints in the image plane after x_v and y_v are estimated. For the x-axis horizontal line, the endpoints $\tilde{\mathbf{a}}_{hx}$ and $\tilde{\mathbf{b}}_{hx}$ can be expressed as follows:

$$\tilde{\mathbf{a}}_{hx} = \left(x_1\ y_{hx}\ z_{hx}\ 1\right)^T \text{ and } \tilde{\mathbf{b}}_{hx} = \left(x_2\ y_{hx}\ z_{hx}\ 1\right)^T \quad (4)$$

where y_{hx} and z_{hx} are variables that we estimate during the SLAM. Similarly, the endpoints $\tilde{\mathbf{a}}_{hy}$ and $\tilde{\mathbf{b}}_{hy}$ of the y-axis horizontal line can be expressed as follows:

$$\tilde{\mathbf{a}}_{hy} = \left(x_{hy}\ y_1\ z_{hy}\ 1\right)^T \text{ and } \tilde{\mathbf{b}}_{hy} = \left(x_{hy}\ y_2\ z_{hy}\ 1\right)^T \quad (5)$$

where x_{hy} and z_{hy} are variables that we estimate during the SLAM.

Using the line projection model of (2) and the previously defined landmark parameterization from (3) to (5), the equation for the position estimation of landmarks can be derived as a simple linear model. The projection equation of a vertical line is as follows:

$$(l_1 a_{v,1} - l_3 a_{v,2})x_v + (l_1 b_{v,1} - l_3 b_{v,2})y_v = (l_3 c_{v,1} - l_1 c_{v,2}) \quad (6)$$

where

$$\begin{aligned} a_{v,1} &= \mathbf{P}_{11}\mathbf{P}_{23} - \mathbf{P}_{13}\mathbf{P}_{21} \\ a_{v,2} &= \mathbf{P}_{21}\mathbf{P}_{33} - \mathbf{P}_{23}\mathbf{P}_{31} \\ b_{v,1} &= \mathbf{P}_{12}\mathbf{P}_{23} - \mathbf{P}_{13}\mathbf{P}_{22} \\ b_{v,2} &= \mathbf{P}_{22}\mathbf{P}_{33} - \mathbf{P}_{23}\mathbf{P}_{32} \\ c_{v,1} &= \mathbf{P}_{24}\mathbf{P}_{33} - \mathbf{P}_{23}\mathbf{P}_{34} \\ c_{v,2} &= \mathbf{P}_{14}\mathbf{P}_{23} - \mathbf{P}_{13}\mathbf{P}_{24}. \end{aligned} \quad (7)$$

Similarly, the projection equation of the x-axis horizontal line is as follows:

$$\begin{aligned} &(l_2 a_{hx,1} - l_3 a_{hx,2})y_{hx} + (l_2 b_{hx,1} - l_3 b_{hx,2})z_{hx} \\ &= (l_3 c_{hx,1} - l_2 c_{hx,2}) \end{aligned} \quad (8)$$

where

$$\begin{aligned} a_{hx,1} &= \mathbf{P}_{12}\mathbf{P}_{21} - \mathbf{P}_{11}\mathbf{P}_{22} \\ a_{hx,2} &= \mathbf{P}_{11}\mathbf{P}_{32} - \mathbf{P}_{12}\mathbf{P}_{31} \\ b_{hx,1} &= \mathbf{P}_{13}\mathbf{P}_{21} - \mathbf{P}_{11}\mathbf{P}_{23} \\ b_{hx,2} &= \mathbf{P}_{11}\mathbf{P}_{33} - \mathbf{P}_{13}\mathbf{P}_{31} \\ c_{hx,1} &= \mathbf{P}_{11}\mathbf{P}_{34} - \mathbf{P}_{14}\mathbf{P}_{31} \\ c_{hx,2} &= \mathbf{P}_{14}\mathbf{P}_{21} - \mathbf{P}_{11}\mathbf{P}_{24}. \end{aligned} \quad (9)$$

Finally, the projection equation of the y-axis horizontal line is as follows:

$$\begin{aligned} &(l_2 a_{hy,1} - l_3 a_{hy,2})x_{hy} + (l_2 b_{hy,1} - l_3 b_{hy,2})z_{hy} \\ &= (l_3 c_{hy,1} - l_2 c_{hy,2}) \end{aligned} \quad (10)$$

where

$$\begin{aligned} a_{hy,1} &= \mathbf{P}_{11}\mathbf{P}_{22} - \mathbf{P}_{12}\mathbf{P}_{21} \\ a_{hy,2} &= \mathbf{P}_{12}\mathbf{P}_{31} - \mathbf{P}_{11}\mathbf{P}_{32} \\ b_{hy,1} &= \mathbf{P}_{13}\mathbf{P}_{22} - \mathbf{P}_{12}\mathbf{P}_{23} \\ b_{hy,2} &= \mathbf{P}_{12}\mathbf{P}_{33} - \mathbf{P}_{13}\mathbf{P}_{32} \\ c_{hy,1} &= \mathbf{P}_{12}\mathbf{P}_{34} - \mathbf{P}_{14}\mathbf{P}_{32} \\ c_{hy,2} &= \mathbf{P}_{14}\mathbf{P}_{22} - \mathbf{P}_{12}\mathbf{P}_{24}. \end{aligned} \quad (11)$$

The position of the landmarks can be easily estimated from the line matching results obtained from different locations of the robot because (6), (8), and (10) are linear. For data association of line features, the normalized image patch of size 11×11 pixels around the midpoint of the extracted line is used for computational efficiency. When the same line is matched over several images, the position of the line can be calculated using the linear least-squares method.

D. Camera Position Estimation

From the extracted line landmark with a known position, the camera position $\mathbf{x} = (x_c\ y_c)^T$ can be estimated. These estimates of the camera x–y positions are used in the local map correction step in the mapping thread. Using the previously defined landmark parameterization and the line projection model, we can obtain the equations for estimating the camera position. Using the vertical line projection, the camera position can be expressed as follows:

$$(a_v \cdot l_1 - b_v \cdot l_3)x_c + (c_v \cdot l_1 - d_v \cdot l_3)y_c = e_v \cdot l_3 - f_v \cdot l_1 \quad (12)$$

where

$$\begin{aligned} a_v &= f_x f_y s_t \sin\theta_c + f_x c_y c_t \sin\theta_c - f_y c_x \cos\theta_c \\ b_v &= f_y \cos\theta_c \\ c_v &= -f_x f_y s_t \cos\theta_c - f_y c_y c_t \cos\theta_c - f_y c_x \sin\theta_c \\ d_v &= f_y \sin\theta_c \\ e_v &= -f_y(x_v \cos\theta_c + y_v \sin\theta_c) \\ f_v &= (f_x f_y s_t y_v + f_x c_y c_t y_v + f_y c_x x_v)\cos\theta_c \\ &\quad + (f_y c_x y_v - f_x f_y s_2 x_v - f_x c_y c_t x_v)\sin\theta_c. \end{aligned} \quad (13)$$

Similarly, using the x-axis horizontal line projection, the camera position can be expressed as follows:

$$(a_{hx} \cdot l_3 - b_{hx} \cdot l_2)y_c = c_{hx} \cdot l_2 - d_{hx} \cdot l_3 \quad (14)$$

where

$$\begin{aligned} a_{hx} &= f_x s_t \\ b_{hx} &= f_x f_y c_t - f_x c_y s_t \\ c_{hx} &= (f_x c_y s_t - f_x f_y c_t)y_{hx} \\ &\quad + (f_x f_y s_t \sin\theta_c + f_x c_y c_t \sin\theta_c - f_y c_x \cos\theta_c)z_{hx} \\ d_{hx} &= -f_x s_t y_{hx} - f_x c_t \sin\theta_c z_{hx}. \end{aligned} \quad (15)$$

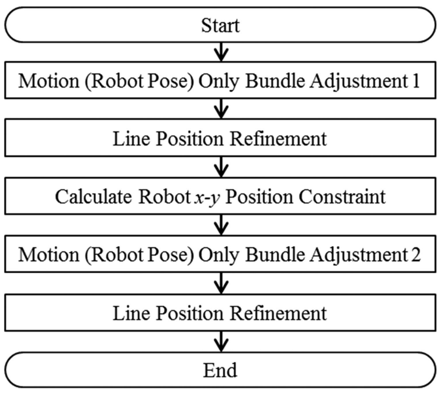
Start
Motion (Robot Pose) Only Bundle Adjustment 1
Line Position Refinement
Calculate Robot x-y Position Constraint
Motion (Robot Pose) Only Bundle Adjustment 2
Line Position Refinement
End

Fig. 5. Flowchart of the proposed local map correction process.

Finally, using the y-axis horizontal line projection, the camera position can be expressed as follows:

$$(a_{hy} \cdot l_3 - b_{hy} \cdot l_2)x_c = c_{hy} \cdot l_2 - d_{hy} \cdot l_3 \quad (16)$$

where

$$\begin{aligned} a_{hy} &= -f_x s_t \\ b_{hy} &= f_x c_y s_t - f_x f_y c_t \\ c_{hy} &= (f_x f_y c_t - f_x c_y s_t)x_{hy} - (f_y c_x \cos\theta_c \\ &\quad + f_x f_y s_t \cos\theta_c + f_x c_y c_t \cos\theta_c)z_{hy} \\ d_{hy} &= f_x s_t x_{hy} + f_x c_t \cos\theta_c z_{hy}. \end{aligned} \quad (17)$$

Using these three types of extracted line landmarks, the camera position can be estimated using the linear least-squares method.

E. Local Map Correction

The local map correction process corrects the estimates of a bundle of camera poses and landmark positions. It is only conducted for the recent N frames and the observed line landmarks at the corresponding frames. When measurements from the camera (i.e., robot orientation from the VP and robot x–y position measurement) are not valid because of low-textured areas, occlusions, or changing environments, the local map correction is skipped. Subsequently, when valid measurements are available, local map correction is conducted, which includes the skipped frames. The flowchart of the local map correction process is shown in Fig. 5.

The motion-only bundle adjustment 1 corrects the camera poses using the odometry measurements and the VP-based orientation measurements obtained in the tracking thread. The cost function to be minimized is formulated as follows:

$$E(\mathbf{x}_{c,s}, \ldots, \mathbf{x}_{c,k}) = \sum_i ||(\mathbf{z}_{o,i} - (\mathbf{x}_{c,i} \ominus \mathbf{x}_{c,i-1}))||^2_{\Sigma_{\text{odo}}} + \sum_i R_i (z_{\text{VP},s,i} - (\theta_i - \theta_s))^2 \quad (18)$$

where $\mathbf{x}_{c,s} \in \mathbb{R}^3$ is the sth 2-D camera poses; $\mathbf{x}_{c,k} \in \mathbb{R}^3$ is the current camera pose; $\mathbf{z}_{o,i} \in \mathbb{R}^3$ is the odometry measurements which is the relative pose between the (i–1)th camera pose and ith camera pose; $\ominus$ is the inverse pose composition operator; $z_{VP,s,i}$ is the VP-based orientation measurements with respect to the sth camera orientation; θ_i is the ith camera orientation estimation; Σ_{odo} is the covariance matrix of odometry measurement; and R_i is the covariance of VP-based orientation measurement. The variable s indicates the start index of the local map correction. The covariance of the VP-based orientation is inversely proportional to the ratio of inliers in estimating the VPs. The inverse pose composition operator is a relative transformation between the two camera poses $\mathbf{x}_i$ and $\mathbf{x}_j$ defined as follows:

$$\mathbf{z}_{ij} = \mathbf{x}_i \ominus \mathbf{x}_j = \begin{bmatrix} (x_i - x_j)\cos\theta_j + (y_i - y_j)\sin\theta_j \\ -(x_i - x_j)\sin\theta_j + (y_i - y_j)\cos\theta_j \\ \theta_i - \theta_j \end{bmatrix}. \quad (19)$$

The Levenberg–Marquart algorithm using a g2o framework [11] is executed to minimize (18). We can obtain the estimates of the bundle of camera poses from the motion-only bundle adjustment 1 step; therefore, the accuracy of the corresponding landmark position estimation can be improved. In this regard, line position is reestimated using the method proposed in Section III-C. The next step is to calculate the x–y position of cameras using the method given in Section III-D followed by the motion-only bundle adjustment 2. The cost function to be minimized is formulated as follows:

$$E(\mathbf{x}_{c,s}, \ldots, \mathbf{x}_{c,k}) = \sum_i ||(\mathbf{z}_{o,i} - (\mathbf{x}_{c,i} \ominus \mathbf{x}_{c,i-1}))||^2_{\Sigma_{\text{odo}}} + \sum_i R_i (z_{\text{VP},s,i} - (\theta_i - \theta_s))^2 + \sum_i ||(\mathbf{z}_{xy,i} - \mathbf{x}_{xy,i})||^2_{\Sigma_{xy}} \quad (20)$$

where $\mathbf{z}_{xy,i} \in \mathbb{R}^3$ is the x–y position measurement at the ith camera obtained from the line matching results, $\mathbf{x}_{xy,i} \in \mathbb{R}^3$ is the current estimation of the x–y position at ith camera, and $\sum_{xy}$ is the covariance matrix of the x–y position measurement. $\sum_{xy}$ is determined proportional to the inverse of the residual error for estimating the x–y position. Finally, the line position is again estimated using the method proposed in Section III-C.

F. Whole Image Descriptor-Based Loop Detection

The loop-closing thread finds large loops for every input frame. The state-of-the-art SLAM methods usually adopt the loop-detection technique based on bag-of-visual-words (BoVW) [10], [25]. To convert an input image into BoVW representations for loop detection, the system has to load a huge vocabulary tree (e.g., 250-Mb memory for DBoW2 [37]). As an alternative, the BRIEF-Gist [38] method is used in this work considering the applicability to a low-cost embedded system. To robustly detect the revisited place even under the viewpoint change, we extract three BRIEF-Gist descriptors from a single image, that is, the left 80% of the image, the right 80% of the image, and the original image as in [39]. When comparing two

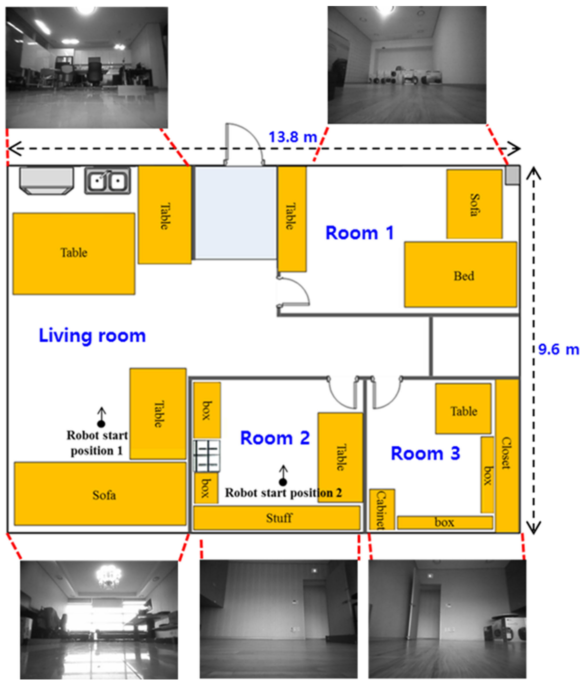

Fig. 6. Blueprint of home environment and example images.

TABLE I
HOME DATASET CHARACTERISTICS

Dataset	START POSITION	Environment	# of images	# of images containing moving people or objects
1	1	Static	8393	None
2	2	Static	8780	None
3	1	Dynamic	7734	1859
4	2	Dynamic	8431	1466

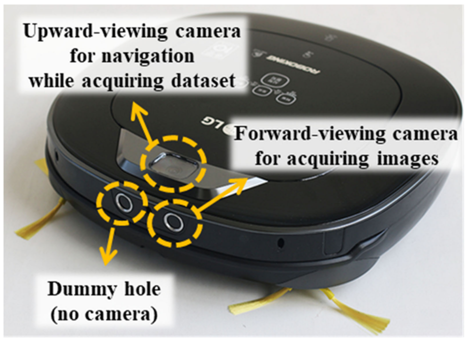
Upward-viewing camera for navigation while acquiring dataset
Forward-viewing camera for acquiring images
Dummy hole (no camera)

Fig. 7. Robot platform for acquiring home environment dataset.

images captured from the two places, three descriptors are compared with one another. For fast comparison, we also adopt data structure as proposed in [39].

G. Global Pose Correction

To close the loop, a pose graph optimization is performed after the loop detection. Before the pose optimization, the relative position is computed between the matched frame from the loop detection and the current frame using the method proposed in Section III-D. The pose graph optimization is conducted for all poses between the matched frame from the loop detection and the current frame. The cost function to be minimized is as follows:

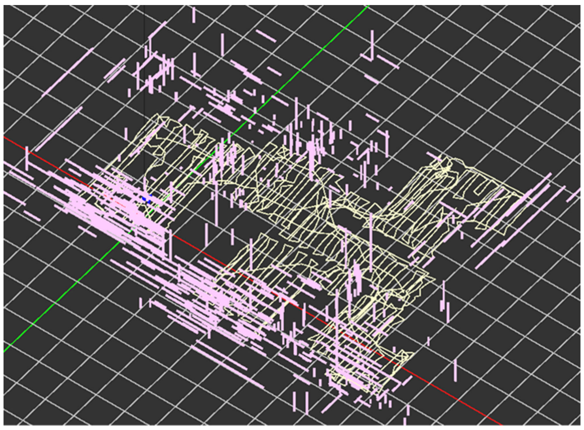

Fig. 8. Result of the proposed SLAM using home dataset 3 (dynamic). Purple lines represent reconstructed line landmarks, and yellow line represents the SLAM-based robot trajectory.

$$E(\mathbf{x}_{\text{matched}}, \ldots, \mathbf{x}_{\text{curr}}) = \sum_{i=\text{matched}}^{\text{curr}} ||(\mathbf{z}_{\text{incre},i} - (\mathbf{x}_i \ominus \mathbf{x}_{i-1}))||^2_{\Sigma_{\text{incre}}} + ||(\mathbf{z}_{\text{matched,curr}} - (\mathbf{x}_{\text{matched}} \ominus \mathbf{x}_{\text{curr}}))||^2_{\Sigma_{\text{LD}}} \quad (21)$$

where $\mathbf{z}_{\text{incre},i} \in \mathbb{R}^3$ is the relative incremental pose of $\mathbf{x}_i$ with respect to the $\mathbf{x}_{i-1}$ frame, $\mathbf{z}_{\text{matched,curr}} \in \mathbb{R}^3$ is the relative pose of $\mathbf{x}_{\text{curr}}$ with respect to $\mathbf{x}_{\text{matched}}$, Σ_{incre} is the covariance matrix of the incremental pose estimation, and $\sum_{\text{LD}}$ is the covariance matrix of the relative pose from the loop detection. The covariance matrices are determined proportional to the inverse of the covisibility of line landmarks. In case, there is low covisibility between incremental poses, we set the upper bound to determine the covariance matrix. After the optimization, each map line is transformed according to the correction of each corresponding frame that observes it.

IV. EXPERIMENTS

Datasets-based experiments on a desktop computer and real-time experiments on an embedded system are performed. Experiments are conducted using the home datasets which we acquired and the public RAWSEEDS benchmark dataset [4]. For real-time embedded experiments, the proposed algorithm is implemented in NXP4330Q embedded board (NEXELL Co., Republic of Korea), and experiments are conducted in a home environment. In our comparative experiments, we compare the proposed method with three approaches as given below.

First, a two-dimensional (2-D) version of the ORB-SLAM [10] method is implemented. There are several differences between the original ORB-SLAM and the implemented 2-D version of ORB-SLAM. First, the camera poses are parameterized in the 2-D space. Second, the odometry data are used to calculate the initial pose of the robot in the tracking thread. The odometry

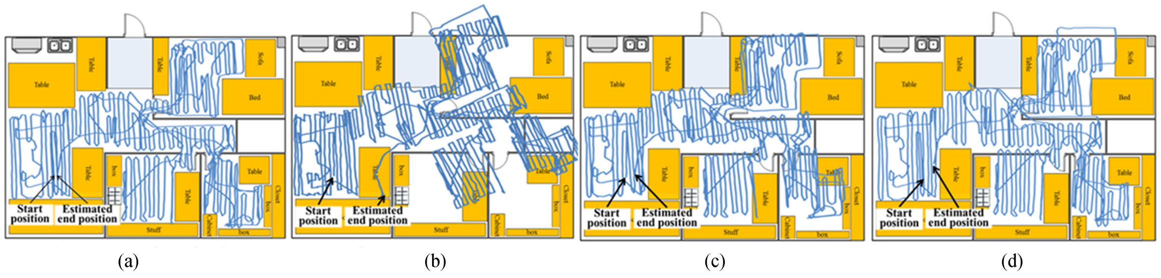

Fig. 9. Estimated robot trajectories of various methods using the home dataset 3.

data are used as an edge in the local bundle adjustment step. This algorithm is denoted as *ORB-SLAM 2D* in the following.

Second, the VP-based line SLAM with the standard bundle adjustment method is implemented. The overall algorithm is very similar to the proposed method. The line extraction, VP extraction, data association, line parameterization, line observation model, line initialization, and loop closing methods are the same as those in the proposed method. In the tracking thread, the VP-based robot orientation estimation procedure is skipped. In the mapping thread, the standard local bundle adjustment is conducted. The cost function to be minimized is formulated as follows:

$$E(\mathbf{x}_{c,s},\ldots,\mathbf{l}_l,\ldots) = \sum_i ||(\mathbf{z}_{o,i} - (\mathbf{x}_i \ominus \mathbf{x}_{i-1}))||^2_{\Sigma_{\text{odo}}} + \sum_i \sum_j ||\mathbf{z}_{i,j} - h(\mathbf{x}_i, \mathbf{l}_j)||^2_{\Sigma_{\text{line}}} \quad (22)$$

where $\mathbf{l}_j \in \mathbb{R}^3$ is the position of the landmark, which is one of the three lines (i.e., the vertical line, the x-axis horizontal line, or the y-*axis* horizontal line); $\mathbf{z}_{i,j} \in \mathbb{R}^3$ is the measurement of line landmark in the image plane; and $h(\cdot)$ is the line projection model. This version of implementation is intended to examine the effect of the proposed estimation method for VP-based robot orientation and the local map correction method compared with the standard optimization-based method. This algorithm is denoted as *VP standard BA* in the following. Third, the VP-based orientation correction method without any other process is implemented. This version estimates VP-based robot orientation only in the tracking thread and conducts motion-only bundle adjustment 1 in the mapping thread with no loop-closing thread. This version is intended to examine the performance of the proposed VP-based orientation estimation method only. This method is denoted as *VP only* in the following.

A. Home Dataset

Home datasets are acquired in a typical home environment, as shown in Fig. 6. The robot explored the experimental environment in a trajectory greater than 400 m while collecting images from a forward-viewing monocamera, along with the robot odometry data. These steps are performed four times with different conditions. The characteristics of each home datasets are summarized in Table I. The sample images of the dynamic environment situations are illustrated in Figs. 1(c)–(g). For our robotic platform, a robotic vacuum is used as shown in Fig. 7. An upward-viewing monocamera is used for autonomous

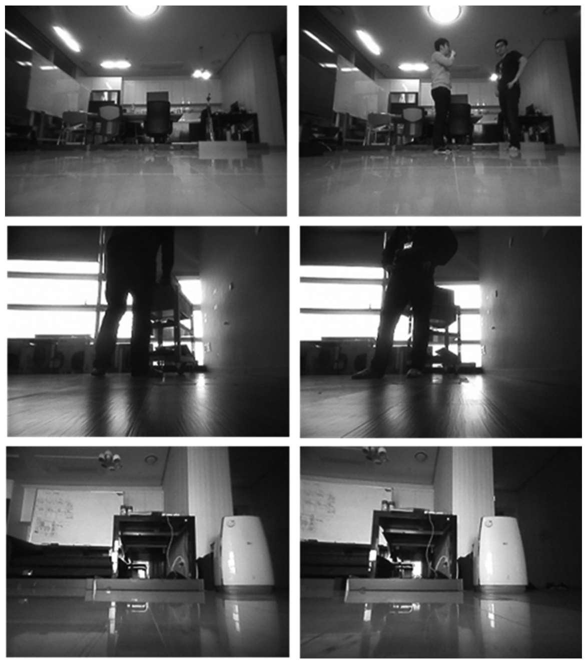

Fig. 10. Examples of matched frames from loop closure in home dataset 3.

navigation [27] while acquiring the datasets. As a path-planning strategy, the Boustrophedon method [40] is used to cover the whole environment, and ultrasonic sensors are used to avoid obstacles. At the end of the drive, we manually returned the robot to the starting point using the remote controller. Ideally, the estimated final pose of the robot must be the origin, that is, $(0, 0, 0°)^T$. We use the distance between the estimated final position and the origin to quantitatively measure the accuracy of the SLAM algorithm, and we call this error as a closed-loop error.

The algorithms are tested on a desktop computer with Intel Core i7-2600. Fig. 8 shows the result of the proposed SLAM from the home dataset 3 (dynamic environment). Fig. 9 shows the estimated robot trajectories for various methods for the home dataset 3. For the *ORB-SLAM 2D* case, a large error drift occurred. This is mainly due to the error drift in the low-textured areas of the environment and the difficulty in feature tracking. Furthermore, no large loop closure between the early stage and the last stage occurred. The loop closure occurred only twice between the 1035 frame and the 1206 frame, and between the 3060 frame and the 6320 frame. On the other hand, in the

TABLE II
CLOSED-LOOP ERROR OF VARIOUS METHODS IN HOME DATASET

Dataset	PROPOSED METHOD (cm)	ORB-SLAM 2D (cm)	VP Standard BA (cm)	VP-only (cm)
1	**6.6**	147.0	52.0	93.5
2	**7.0**	67.6	42.2	129.8
3	**13.0**	226.1	59.0	106.2
4	**6.1**	98.4	47.4	91.3
Avg.	**8.2**	134.8	50.2	105.2
Std.	**3.2**	69.1	7.1	17.7

The best results are denoted in bold type.

TABLE III
TIMING RESULTS OF VARIOUS METHODS PER EACH THREADS

Version	THREAD	Avg. (ms)	Std. (ms)
Proposed Method	Tracking	6.96	5.03
	Mapping	5.71	4.19
	Loop Closing	4.72	**1.38**
ORB-SLAM 2D	Tracking	16.41	7.67
	Local Mapping	41.92	34.22
	Loop Closing	**3.08**	26.37
VP Standard BA	Tracking	6.92	5.01
	Mapping	20.31	11.20
	Loop Closing	4.81	1.72
VP-only	Tracking	**3.02**	**0.96**
	Mapping	**0.3**	**0.14**

The best results are denoted in bold type.

TABLE IV
AVERAGE MEMORY USAGE OF VARIOUS METHODS IN HOME DATASET

PROPOSED METHOD	ORB-SLAM 2D	VP Standard BA	VP-only
114.1 MB	1160.5 MB	113.9 MB	**2.9 MB**

The best results are denoted in bold type.

TABLE V
ABSOLUTE POSITION ERROR OF VARIOUS METHODS IN RAWSEEDS DATASET

	PROPOSED METHOD	ORB-SLAM 2D	VP Standard BA	VP-only
Avg. (m)	**0.71**	2.51	0.88	2.43
Max. (m)	**1.36**	5.14	2.24	3.61
Std. (m)	**0.31**	1.21	0.54	0.73

The best results are denoted in bold type.

proposed method, the loop closure occurred six times. The examples of the matched frames from the loop closure detection are illustrated in Fig. 10. Clearly, the orientation estimation of the proposed method is more accurate than that of the *VP Standard BA* method. The difference lies in the fact that the robot's orientation is corrected directly from the VP in the proposed method. The robot's orientation is corrected by extracting the landmarks in the *VP Standard BA* method. Although the orientation error in the line landmark estimation can be eliminated, the accuracy of the *VP Standard BA* method reduces when the extraction of the line landmark is limited in low-textured areas. In case of *VP only*, the orientation error is effectively eliminated, but the integrated translation error exists. Table II shows the measured closed-loop error of the various methods for the four home datasets. The proposed method shows the lowest closed-loop error compared with other methods. Especially, the average of the closed-loop errors of the proposed method is 6.1 times lower than that of the *VP Standard BA* method.

With respect to the computational speed, the average running time for processing a single frame of each thread is measured and summarized in Table III. The difference between the proposed method and *VP Standard BA* is mainly in the mapping thread. The proposed local map correction method is 3.6 times faster than the standard bundle adjustment while showing better accuracy. The *ORB-SLAM 2-D* shows the slowest result.

After finishing the SLAM, the total memory usages are measured. The average memory usages are summarized in Table IV for the four methods. The memory usage for the *VP-only* method is the lowest because it does not estimate the landmark. For *ORB-SLAM 2-D* case, it requires more than 250 MB to load the vocabulary tree at the start of the SLAM. In the experiments, the *ORB-SLAM 2-D* method has reconstructed an average of 25 892 point features, whereas the proposed method has reconstructed an average of 1072 line features.

B. RAWSEEDS Benchmark Dataset

To evaluate the accuracy in large-scale indoor environments, experiments are conducted using the RAWSEEDS benchmark dataset (Biccoca25b) [4]. The dataset is collected in an office building by a wheeled robot with multiple sensors, including laser range finders, cameras, an inertial measurement unit, and wheel encoders. The dataset also provides ground truth data of the robot's pose for evaluation. In this experiment, the image sequences from the frontal camera and odometry data are used. The dataset consists of 52 695 images, and the whole path is approximately 774 m long. Even though this dataset contains low-textured areas and dark corridors in some areas, it has sufficient features in comparison with the home dataset. In addition, the path contains several loops and revisited regions. Absolute position errors for whole robot trajectories are compared in Table V. The average of the absolute position error for the whole trajectory of the proposed method *ORB-SLAM 2-D*, *VP Standard BA*, and *VP only* are 0.71, 2.51, 0.88, and 2.43 m, respectively. Interestingly, the *VP*-only method is quite accurate with no effort required to execute complicated landmark estimation procedures.

C. Embedded Real-Time SLAM in a Home Environment

The proposed SLAM algorithm is implemented in a low-cost embedded board of NXP4330Q, which is equipped with a Cortex A9 processor and a 512-MB memory. Fig. 11 illustrates the robot platform equipped with the NXP4330Q board. The proposed method is integrated in an autonomous robot navigation system. When the robot explores the environment, the obstacle grid map is constructed using the localization result from the proposed SLAM and the detected obstacles from the

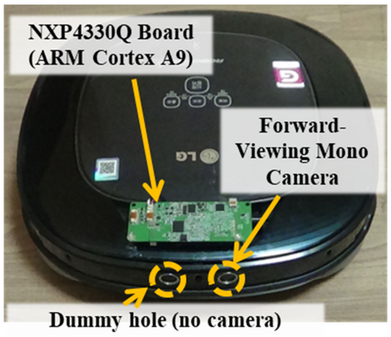

Fig. 11. Robot platform equipped with NXP4330Q board for real-time SLAM experiment.

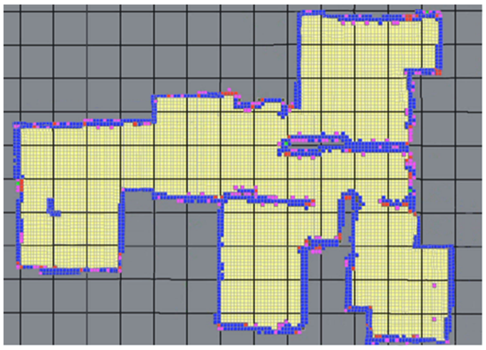

Fig. 12. Generated obstacle grid map for autonomous navigation using the proposed SLAM on NXP4330Q board in a home environment (sequence 3).

ultrasonic sensor. This obstacle grid map is used in path and motion planning for autonomous robot navigation. The Boustrophedon method [40] is used for the path planning strategy. All services, including data acquisition, the proposed SLAM, navigation, and motion planning are simultaneously executed in the NXP4330Q board in real time. Considering the limited computational resources, images are captured when the robot is moved more than 30 cm or rotated more than 30° compared with the previous frame based on the odometry data. The driving and rotation velocities of the mobile robot are 0.35 m/s and 30°/s, respectively. At the end of the drive, we manually returned the robot to the starting point along with the remote controller for measuring the closed-loop error. Similar to the home dataset-based experiments in the previous section, we have performed real-time SLAM experiments four times. The scenarios of the experiments are the same as those for acquiring home datasets. Fig. 12 illustrates the generated obstacle grid map in the real-time experiment sequence 3, which is a dynamic environment. The yellow grid indicates the area where the robot has driven. The blue and pink grids indicate the detected wall and obstacle from the ultrasonic sensor, respectively. Compared with the blueprint of the environment in Fig. 6, the map is accurately built using the proposed method. Table VI shows the measured closed-loop error of the real-time SLAM experiments for the four sequences. The accuracy is similar to the dataset-based experimental results where the algorithm is executed in the desktop PC. For the computation time, the average time for the embedded processor to process one frame for tracking, mapping, and loop-closing thread is 243.2, 211.8, and 138.1 ms, respectively.

TABLE VI
CLOSED-LOOP ERROR OF REAL-TIME SLAM EXPERIMENTS ON THE EMBEDDED SYSTEM IN A HOME ENVIRONMENT

Sequence	START POSITION	Environment	Closed loop error (cm)
1	1	Static	9.6
2	2	Static	**6.0**
3	1	Dynamic	8.4
4	2	Dynamic	10.9
	Avg.		8.7
	Std.		1.8

The best results are denoted in bold type.

V. CONCLUSION

This paper presented a new implementation method for efficient SLAM for low-cost indoor service robots. The estimation model for the robot's orientation and translation were separately derived in simple equations using VP and the line landmark, respectively. Using these models, the camera poses and landmark positions can be efficiently corrected by a local map correction process. When the robot revisits the previously mapped areas, a loop-detection procedure and a pose correction procedure are performed to obtain more accurate SLAM results. The performance of the proposed method was demonstrated under various challenging environments, which contained low-textured areas, moving people, and changing environments.

Although the proposed method was demonstrated using a production home service robot in various environments, additional studies may be necessary for actual implementation. First, we plan to investigate the case where the three dominant directions are not perpendicular to one another. Second, we also plan to conduct research on the applicability of the developed method to large-scale indoor environments such as hospitals, schools, or museums.

REFERENCES

[1] Y. Liu *et al.*, "Stereo visual-inertial odometry with multiple Kalman filters ensemble," *IEEE Trans. Ind. Electron.*, vol. 63, no. 10, pp. 6205–6216, Oct. 2016.
[2] J. Kim and W. Chung, "Localization of a mobile robot using a laser range finder in a glass-walled environment," *IEEE Trans. Ind. Electron.*, vol. 63, no. 6, pp. 3616–3627, Jun. 2016.
[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3355–3361.
[4] A. Bonarini *et al.*, "RAWSEEDS: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, vol. 6, 2006, pp. 16–23.
[5] E. Rosten and T. Drummond, "Machine learning for high speed corner detection," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 430–443.
[6] H. Zhou *et al.*, "StructSLAM: Visual SLAM with building structure lines," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2015.
[7] A. Davison, "Real time simultaneous localisation and mapping with a single camera," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, 2003, pp. 1403–1410.
[8] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 932–945, Oct. 2008.

[9] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in *Proc. IEEE ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 225–234.
[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.
[12] M. Kaess *et al.*, “iSAM2: Incremental smoothing and mapping using the Bayes tree,” *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, 2012.
[13] K. Khosoussi, S. Huang, and G. Dissanayake, “A sparse separable SLAM back-end,” *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1536–1549, Dec. 2016.
[14] N. Sünderhauf and P. Protzel, “Switchable constraints for robust pose graph SLAM,” in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2012, pp. 1879–1884.
[15] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust map optimization using dynamic covariance scaling,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 62–69.
[16] E. Olson and P. Agarwal, “Inference on networks of mixtures for robust robot mapping,” *Int. J. Robot. Res.*, vol. 32, no. 7, pp. 826–840, 2013.
[17] K. Lin and C. Wang, “Stereo-based simultaneous localization, mapping and moving object tracking,” in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2010, pp. 3975–3980.
[18] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, “Robust monocular SLAM in dynamic environments,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2007, pp. 209–218.
[19] D. Lee and H. Myung, “Solution to the SLAM problem in low dynamic environments using a pose graph and an RGB-D sensor,” *Sensors*, vol. 14, no. 7, pp. 12467–12496, 2014.
[20] S. Li and D. Lee, “RGB-D SLAM in dynamic environments using static point weighting,” *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2263–2270, Oct. 2017.
[21] X. Li, K. Liu, and Y. Dong, “Superpixel-based foreground extraction with fast adaptive trimaps,” *IEEE Trans. Cybern.*, to be published.
[22] X. Li, K. Liu, and Y. Dong, “Patch alignment manifold matting,” *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.
[23] P. Smith, I. D. Reid, and A. J. Davison, “Real-time monocular SLAM with straight lines,” in *Proc. Brit. Mach. Vis. Conf.*, 2006, pp. 17–26.
[24] J. Solà, T. Vidal-Calleja, and M. Devy, “Undelayed initialization of line segments in monocular SLAM,” in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2009, pp. 1553–1558.
[25] G. Zhang, J. Lee, J. Lim, and I. Suh, “Building a 3-D line-based map using stereo SLAM,” *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1364–1377, Dec. 2015.
[26] W. Y. Jeong and K. M. Lee, “Visual SLAM with line and corner features,” in *Proc. IEEE Int. Conf. Intell. Robot.*, 2006, pp. 2570–2575.
[27] S. S. Lee and S. H. Lee, “Embedded visual SLAM: Applications for low-cost consumer robots,” *IEEE Robot. Autom. Mag.*, vol. 20, no. 4, pp. 83–95, Dec. 2013.
[28] V. Huttunen and P. Robert, “A monocular camera gyroscope,” *Gyroscopy Navigat.*, vol. 3, no. 2, pp. 124–131, 2012.
[29] F. Camposeco and M. Pollefeys, “Using vanishing points to improve visual-inertial odometry,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 5219–5225.
[30] Y. Ji, A. Yamashita, and H. Asama, “RGB-D SLAM using vanishing point and door plate information in corridor environment,” *Intell. Serv. Robot.*, vol. 8, no. 2, pp. 105–114, 2015.
[31] Y. H. Lee, C. Nam, K. Y. Lee, Y. S. Li, S. Y. Yeon, and N. L. Doh, “VPass: Algorithmic compass using vanishing points in indoor environments,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2009, pp. 936–941.
[32] G. Zhang, D. H. Kang, and I. H. Suh, “Loop closure through vanishing points in a line-based monocular SLAM,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 4565–4570.
[33] J. M. Coughlan and A. L. Yuille, “Manhattan world: Compass direction from a single image by Bayesian inference,” in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, 1999, pp. 941–947.
[34] V. G. R. Grompone, J. Jakubowicz, J. M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.
[35] L. Zhang, H. Lu, X. Hu, and R. Koch, “Vanishing point estimation and line classification in a Manhattan world with a unifying camera model,” *Int. J. Comput. Vis.*, vol. 117, no. 2, pp. 111–130, 2015.
[36] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[37] D. Gálvez-López and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.
[38] N. Sünderhauf and P. Protzel, “BRIEF-Gist—Closing the loop by simple means,” in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2011, pp. 1234–1241.
[39] Y. Jeon, T. Lee, C. Kim, and D. Cho, “A BRIEF-gist based efficient place recognition for indoor home service robots,” in *Proc. Int. Conf. Control Autom. Syst.*, 2016, pp. 1526–1530.
[40] H. Choset and P. Philippe, “Coverage path planning: The Boustrophedon cellular decomposition,” in *Proc. Int. Conf. Field Serv. Robot.*, 1998, pp. 203–209.

Tae-jae Lee received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2011 and 2017, respectively.

He is currently with LG Electronics, Seoul. His research interests include robot vision, simultaneous localization and mapping, and multisensor fusion.

Chul-hong Kim received the B.S. degree in engineering from Australian National University, Canberra ACT, Australia, in 2014, and is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea.

His research interests include robot vision, simultaneous localization and mapping, and deep learning.

Dong-il Dan Cho (M’88) received the B.S.M.E. degree from Carnegie-Mellon University, Pittsburg, PA, USA, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1980, 1984, and 1988, respectively.

From 1987 to 1993, he was an Assistant Professor with Princeton University, Princeton, NJ. Since 1993, he has been with the Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea, where he is currently a Professor. He is the author/coauthor of more than 120 international journal articles. He is the holder/coholder of more than 110 US and Korean patents.

Dr. Cho has served on the editorial board of many international journals. He is currently a Senior Editor of IEEE JOURNAL OF MICROELECTROMECHANICAL SYSTEMS and International Federation of Automatic Control (IFAC’s) *Mechatronics*. He was the President of Institute of Control, Robotics and Systems (ICROS) in 2017, and is currently Vice President of IFAC and the Chair of the Technical Board of IFAC. He is an elected Senior Member of the National Academy of Engineering of Korea.