Embedded Visual SLAM

By Seongsoo Lee and Sukhan Lee

Applications for Low-Cost Consumer Robots

A camera poses a highly attractive choice as a sensor in implementing simultaneous localization and mapping (SLAM) for low-cost consumer robots such as home cleaning robots. This is due to its low cost, light weight, and low power consumption. However, most of the visual SLAMs available to date are not designed and, consequently, not suitable for use in a low-cost embedded SLAM for consumer robots. This article presents a computationally light yet performance-wise robust SLAM algorithm and its implementation as an embedded system for low-cost consumer robots using an upward-looking camera. Especially for a large-scale mapping of indoor environments, methods of pose graph optimization as well as submapping are employed. An occupancy grid map is used to integrate an efficient Kalman filter-based localization into a SLAM framework. Furthermore, an algorithmic visual compass is introduced as a means of reducing the computational complexity involved in pose graph optimization, taking advantage of the distinct geometric features of the scenes captured by an upward-looking camera. The proposed visual SLAM is implemented in a real home cleaning robot as an embedded system using an ARM11 processor. Extensive test results demonstrate the power of the proposed embedded visual SLAM in terms of not only its computational efficiency but also its performance robustness in real-world applications.

Digital Object Identifier 10.1109/MRA.2013.2283642
Date of publication: 31 October 2013

Visual SLAM—A Key Technology for Low-Cost Mobile Robots

SLAM represents a key technology for a mobile robot to self-navigate in an unknown environment. It automatically constructs a map of its environment while simultaneously determining its pose on the map being constructed [7].

Visual SLAM has been the focus of many recent studies due to the remarkable advancement in the performance of low-cost vision systems, which provide abundant information on the environment. A simple camera, with its small size, light weight, and low energy requirement, is useful as an integrated sensor that can be easily embedded in most mobile robots, especially for consumer robots.

Monocular Camera-Based Approach

Several approaches to visual SLAM that employ a monocular camera have been proposed [4], [5], [18]. In [4], a monocular camera SLAM algorithm without odometry is proposed for

1070-9932/13/$31.00©2013IEEE

real-time visual SLAM with an extended Kalman filter (EKF). This method shows an increase in localization accuracy by the use of an explicit frame-to-frame error-propagation model, as well as by the particle filter representation of feature depths for initialization. In [18], this framework is extended by proposing an inverse parameterization of feature depths. Since this allows feature depths to be better approximated with a Gaussian, no particle filter is required for initialization. However, the classical EKF-based SLAM gives rise to a fundamental limitation in the number of features that can be mapped due to the quadratic complexity of the EKF algorithm. In [5], this technique is integrated in a hierarchical SLAM framework based on the aforementioned approaches and proposed as a means of building a large-scale environment map.

Pose Graph Optimization Technique

The most successful methods currently in use for solving large SLAM problems with many loops are the pose graph optimization algorithms [13], [20], [23]. Instead of estimating the joint density of the robot pose and the landmark locations observed, these methods find a set of poses that minimizes the total error defined by the relative pose constraints, given the recorded observations and control inputs. Such a scenario is attractive because it is possible to transform the SLAM problem to satisfy a generally sparse set of pose constraints [20], [23], [24]. Like other methods, pose graph solvers have the worst-case complexity at loop closure, which is dependent on the number of poses to be estimated.

Limitation of Vision-Only SLAM

The main problem with most existing visual SLAM implementations that operate without odometry is their lack of robustness in adverse situations. Tracking systems typically rely on the prior pose used to limit the search for visual feature correspondences, yielding a rapid frame-to-frame localization. However, tracking failures occur frequently in situations where vision is no longer reliable, e.g., under temporary sensor occlusions or featureless scenes like those caused by unlit spots in the environment. For this reason, the incorporation of odometry into a visual SLAM is preferred for realistic applications.

Visual SLAM Using Odometry

Researchers who have used odometry and cameras have demonstrated reliable and accurate vision-based localization and mapping [19], [21], [22]. In particular, schemes based on cameras pointing straight up at the ceiling [6], [11], [17] have certain advantages compared to other schemes in indoor environments. First, there is no significant effect caused by dynamic obstacles such as moving people since the camera points at the ceiling. Additionally, there is no scale change of features between neighboring images in most indoor environments. Therefore, under the aforementioned conditions, visual features can be extracted quickly from an image while robust matching results are attained. In [6], a camera pointed toward the ceiling is used to match the current image to a large ceiling mosaic covering the whole operational space of the robot. The mosaic has to be constructed in advance, which involves a complex state estimation problem. Furthermore, it is not designed for a low-cost embedded system. In [17], a vision-based SLAM is proposed for low-cost applications, but it still lacks the quadratic complexity of the EKF algorithm.

Motivation, Contribution, and Organization

There are very few works with a lightweight and low-cost approach targeting commercial applications such as home cleaning robots. Most general-purpose visual SLAM algorithms are not appropriate for low-cost applications because they require high-resolution images or rapid processing power. Therefore, it is worthwhile to investigate a visual SLAM application for lightweight, low-cost applications. To this end, a simple low-resolution camera with an ARM11 processor may be a good choice for implementing such an embedded visual SLAM.

In this article, we present an efficient visual SLAM algorithm for resource-limited consumer robots such as home cleaning robots. The proposed SLAM algorithm uses odometry and a single camera with a wide-angle lens that yields low-resolution images of 320×240 pixels. The camera points in the zenith direction perpendicular to the ground plane and is restricted to approximate only two-dimensional (2-D) planar motion. Under these conditions, the upward-looking camera dramatically reduces the computational complexity needed for extraction of features and matching compared to the front-view cameras. The original contributions of this article are:

- a visual algorithmic compass that accurately estimates the robot's orientation using the orthogonal lines of an environment detected in upward-looking scenes
- computing the three-dimensional (3-D) locations of features and the loop closure constraint using an upward-looking camera
- reducing the linearization errors and the computational complexity in the pose graph optimization technique described in [23] using the visual algorithmic compass
- integrating a standard Kalman filter-based localization method into the SLAM framework for efficiency
- development of a lightweight and robust SLAM algorithm on an embedded system for practical applications.

The article from here is organized into the following sections. An overview of the proposed SLAM algorithm is described in the "Algorithm Overview" section. Methods for defining spatial constraints based on the upward-looking camera and correcting the robot trajectory using the spatial constraints are presented in the "Visual Slam Framework" section. A number of experimental examples to demonstrate the validity of the proposed method are given in the "Experimental Results" section, while the results are detailed in the "Conclusions" section.

Algorithm Overview

An overview of the proposed visual SLAM algorithm consisting of six main modules is shown in Figure 1. The six main

modules that form an operational chain are loaded into our embedded vision board of a home cleaning robot, as shown in Figure 2.

In this article, we employ a pose graph optimization technique [13], [20], [23] that successfully solves large SLAM problems with many loops. There is no privileged pose, and the recovering landmark estimates seem to require a graph traversal. Each node of the graph represents a robot pose at which a sensor measurement was acquired. The edges in the graph represent spatial constraints between the nodes. In this article, the constraint $z_{i,j}$ is obtained using sensor measurements and describes a relative transformation (composed of the 3-D vector $z_{i,j} = [\Delta x_{(z_{i,j})}, \Delta y_{(z_{i,j})}, \Delta\theta_{(z_{i,j})}]^T$) between two poses $x_i = [x_{(x_i)}, y_{(x_i)}, \theta_{(x_i)}]^T$ and $x_j = [x_{(x_j)}, y_{(x_j)}, \theta_{(x_j)}]^T$ to represent the relative pose of x_j with respect to the pose x_i. Let $(z_{i,j} - x_j \ominus x_i)$ be the error introduced by the constraint $z_{i,j}$, where $\ominus$ means the inverse of compounding operation $\oplus$, and it is the same operation as defined in [16]. Assuming that the constraints are independent, the poses $\{x_1, \ldots, x_k\}$ as nodes of the graph are corrected by minimizing the sum of weighted square costs introduced by the error $(z_{i,j} - x_j \ominus x_i)$, as

$$X^* = \underset{X=\{x_1,\ldots,x_k\}}{\operatorname{argmin}} \sum_{i,j} (z_{i,j} - x_j \ominus x_i)^T W_{i,j} (z_{i,j} - x_j \ominus x_i), \quad (1)$$

where $W_{i,j}^{-1}$ is the covariance matrix of $z_{i,j}$.

There are two types of constraints to be considered for a pose graph optimization technique to SLAM: the incremental constraint and the loop closure constraint. The incremental constraint describes a relative transformation between successive poses that have continuity with the time. The loop closure constraint models the spatial relationship between nonconsecutive poses.

The environment is divided into multiple submaps [8] for large-scale mapping of indoor environments. The following two independent SLAM estimates must be maintained at all times:

$$M = [{}^G x_{m_1}, \ldots, {}^G x_{m_j}]^T, \quad m_j = [{}^L x_1, \ldots, {}^L x_k]^T, \quad (2)$$

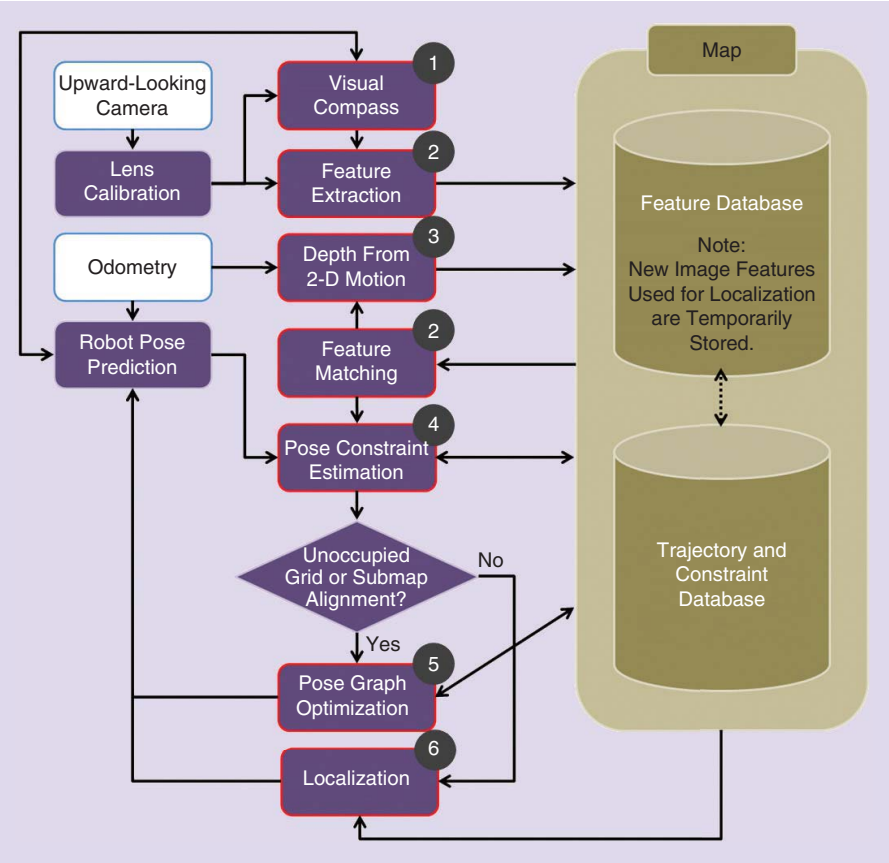

Figure 1. An overview of our visual SLAM algorithm.

where M denotes a set of the global reference poses of the submap coordinate frames $[{}^G x_{m_1}, \ldots, {}^G x_{m_j}]^T$, and m_j is the jth local submap with locally referenced robot poses $[{}^L x_1, \ldots, {}^L x_k]^T$. With a locally referenced pose ${}^L x_i$ in the local submap m_j, a globally referenced pose ${}^G x_i$ is represented by ${}^G x_{m_j} \oplus {}^L x_i$. In this study, $[{}^G x_{m_1}, \ldots, {}^G x_{m_j}]^T$ and $[{}^L x_1, \ldots, {}^L x_k]^T$ are corrected using the same pose graph optimization method that solves (1). The current map is independent of any prior map because it is built relative to the initial robot pose and depends only on the sequence of odometry readings and sensor data obtained during the involved steps. Therefore, all of the submaps are statistically independent and are not correlated to each other. The decision to close the current submap and create a new submap will be made once the number of poses reaches a maximum or the uncertainty of the robot pose with respect to the base reference of the current submap reaches a limit.

Visual SLAM Framework

Visual Compass

In our previous work [3], orthogonal lines of indoor environments were utilized to solve a localization problem within a given map under significant illumination changes. However, this approach is limited, focusing on computing the relative

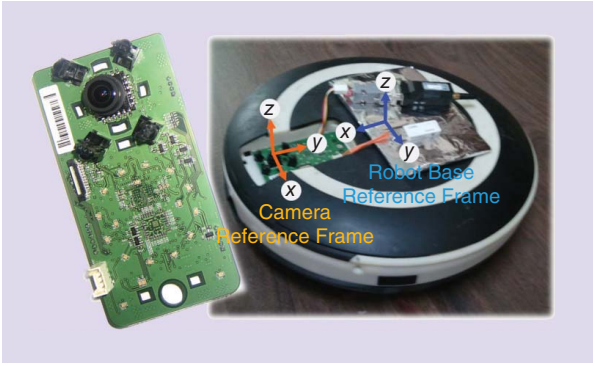

Figure 2. A home cleaning robot with our embedded vision board, including a camera, an ARM11 processor, and 64 MB RAM. (Photo courtesy of the Future IT Laboratory of LG Electronics, Inc.)

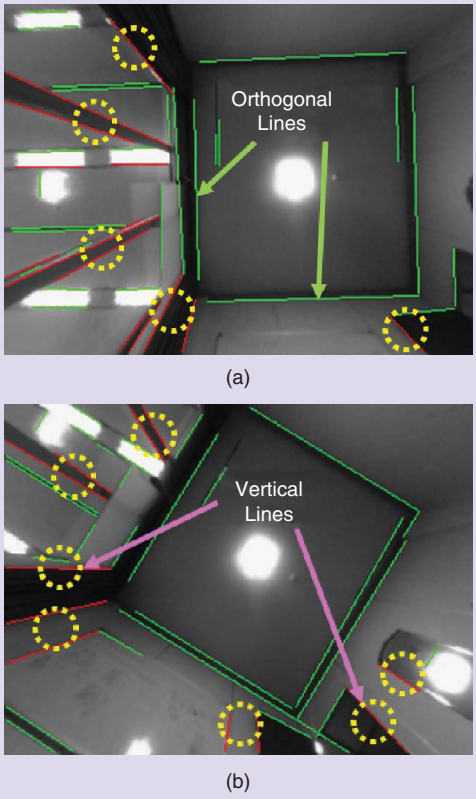

Figure 3. Extracted lines in 320 × 240 resolution images captured from a single camera pointing in the upward direction. The red and green lines represent vertical lines and nonvertical lines, respectively. The vertical lines are highlighted with yellow dotted circles. (a) Image 1. (b) Image 2. (Images courtesy of the Future IT Laboratory of LG Electronics, Inc.)

robot orientation with respect to a pose stored in a visual map after building the visual map in advance. In this section, we present an algorithmic visual compass, simply referred to as the *visual compass*, as a virtual sensor to correct an absolute orientation of the mobile robot to within ±45° under an orthogonal structure assumption in an indoor environment.

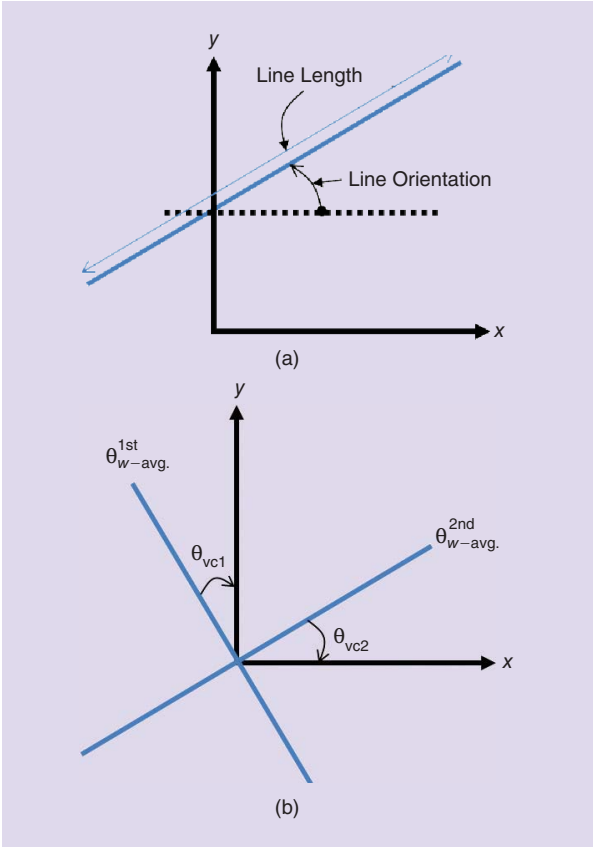

Figure 4. The visual compass concept: (a) line length and angle in the camera reference frame and (b) two rotational angles obtained from two dominant orientations $\theta_{w-\text{avg.}}^{1\text{st}}$ and $\theta_{w-\text{avg.}}^{2\text{nd}}$ of the orthogonal lines.

In [12], a visual compass algorithm is proposed that estimates a camera orientation in an EKF-based SLAM framework based on distant points, ideally at infinity, as features. However, the visual compass algorithm described in [12] is inappropriate for visual SLAM in indoor environments, since it is effective for a large scale of outdoor scenes where a number of distant points are available as features for a visual compass.

Figure 3 shows two images captured from the upward-looking camera (shown in Figure 2) to detect the orthogonal lines. The lines are extracted from an image based on the edge-following approach [25] to extract all lines needed to detect the orthogonal lines. All vertical lines to the ground are easily detected and removed since vertical lines projected in the image plane vanish to the optical center. The red vertical lines are highlighted with yellow dotted circles in Figure 3. The remaining lines outlined in green are considered for detecting the orthogonal lines.

Figure 4(a) shows the information used to detect the orthogonal lines, while Figure 4(b) provides an example of computing the visual compass orientation using the two dominant orientations $\theta_{w-\text{avg.}}^{1\text{st}}$ and $\theta_{w-\text{avg.}}^{2\text{nd}}$ of the orthogonal lines. The two dominant orientations $\theta_{w-\text{avg.}}^{1\text{st}}$ and $\theta_{w-\text{avg.}}^{2\text{nd}}$ are

formed by the orthogonal structures of an environment and represent a direction of the x or y axis of the orthogonal environment (OE) reference frame in the camera reference frame. Here, two rotational angles θ_{VC1} and θ_{VC2} are obtained from the two dominant orientations $\theta^{1st}_{w-avg.}$ and $\theta^{2nd}_{w-avg.}$, respectively. The two rotational angles θ_{VC1} and θ_{VC2} represent an orientation of the upward-looking camera in the OE reference frame. In other words, $-\theta_{VC1}$ and $-\theta_{VC2}$ represent how much the x and y axis of the OE reference frame are rotated in the camera reference frame.

In this article, θ_{VC1} and θ_{VC2} are represented in the range of $(-45° \le \theta < 45°)$ because the x or y axis of the OE reference frame cannot be explicitly distinguished with only orthogonal lines that are detected in a single image captured from an upward-looking camera. Ideally, the two rotational angles θ_{VC1} and θ_{VC2} must be identical.

To obtain the two rotational angles θ_{VC1} and θ_{VC2}, line angles are used to form an orientation histogram that has 180 bins covering from $-45°$ to 134° and are represented in the range of $(-45° \le \theta < 135°)$ by iteratively adding (if line angle $< -45°$) or subtracting (if line angle $\ge 135°$) 180°. Each sample added to the histogram is weighted by a line length.

Figure 5 shows the orientation histogram generated from image two in Figure 3. The highest peak in the histogram is detected, and the weighted average orientation $\theta^{1st}_{w-avg.}$ is then calculated in the vicinity of the peak. To identify the perpendicular lines to $\theta^{1st}_{w-avg.}$, the second peak is obtained in the region of $(\theta^{1st}_{w-avg.} + 90° - \theta_{err} < \theta < \theta^{1st}_{w-avg.} + 90° + \theta_{err}$ or $(\theta^{1st}_{w-avg.} - 90° - \theta_{err} < \theta < \theta^{1st}_{w-avg.} + 90° + \theta_{err}$, where θ_{err} refers to the maximum error of line angle and was set to 10° in this article. In a similar manner, the weighted average orientation of the second peak $\theta^{2nd}_{w-avg.}$ is also calculated in the vicinity of the peak. If a peak is detected in the vicinity of $-45°$ or 134°, the adjacent bins that lie on the border between $-45°$ and 134° are carefully used to compute the weighted average orientation of the peak. Because the orientation histogram has 180 bins covering $-45°$ to 134°, one of the two weighted average orientations $\theta^{1st}_{w-avg.}$ and $\theta^{2nd}_{w-avg.}$ is greater than 45°, so it is again represented in the range of $(-45° \le \theta < 45°)$ by subtracting 90°. For our camera configuration shown in Figure 2, the visual compass orientation θ_{VC} in the OE reference frame is then given by

$$\theta_{VC} = \frac{\sum_i \text{line}(i)^{1st}_{length} \times \theta_{VC1} + \sum_j \text{line}(j)^{2nd}_{length} \times \theta_{VC2}}{\sum_i \text{line}(i)^{1st}_{length} + \sum_j \text{line}(j)^{2nd}_{length}},$$
$$\text{if}\,(\theta^{1st\text{ or }2nd}_{w-avg.} \ge 45°), \theta_{VC1\text{ or }2} = -(\theta^{1st\text{ or }2nd}_{w-avg.} - 90°)$$
$$\text{otherwise}, \theta_{VC1\text{ or }2} = -\theta^{1st\text{ or }2nd}_{w-avg.}, \quad (3)$$

where $\sum_i \text{line}(i)^{1st}_{length}$ and $\sum_j \text{line}(j)^{2nd}_{length}$ represent the sum of the line lengths used to calculate $\theta^{1st}_{w-avg.}$ and $\theta^{2nd}_{w-avg.}$, respectively.

To estimate the robot orientation $\theta_{(x_{i+k})}$ at time $i+k$ using the visual compass, 0° of the visual compass is aligned with that of the global reference frame. More specifically, the

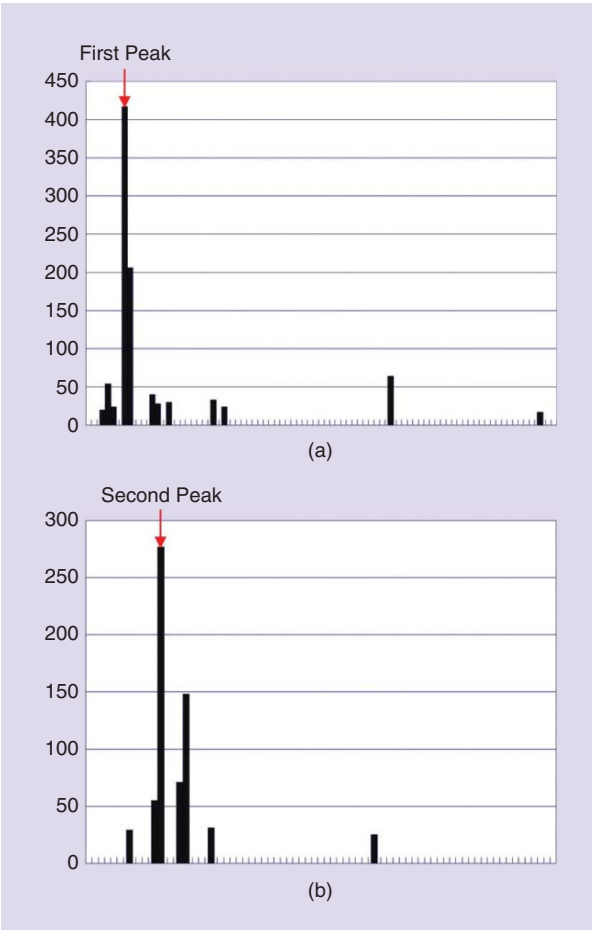

Figure 5. An orientation histogram for Image 2. (a) $50° \le \theta < 135°$. (b) $-45° \le \theta < 31°$.

x or y axis of the OE reference frame is aligned with the x axis of the global reference frame. Generally, the origin of the global reference frame is initialized using the initial robot pose. Assuming that the x axis of the camera reference frame is parallel to the y axis of the robot base reference frame, as in our camera configuration shown in Figure 2, the 0° alignment between the global and OE reference frames is performed by measuring the first visual compass orientation $\theta_{VC(x_i)}$ at the robot pose x_i as a globally referenced pose and then subtracting the robot orientation $\theta_{(x_i)}$ to $\theta_{VC(x_i)}$. Therefore, an offset angle $\Delta\theta_{VC\_align.}$ used for the 0° alignment is given by

$$\Delta\theta_{VC\_align.} = \theta_{VC(x_i)} - \theta_{(x_i)}. \quad (4)$$

Here, the offset angle $\Delta\theta_{VC\_align.}$ is represented in the range of $(-45° \le \theta < 45°)$ by iteratively adding (if $\Delta\theta_{VC\_align.} < -45°$) or subtracting (if $\Delta\theta_{VC\_align.} \ge 45°$) 90°.

Utilizing the offset angle $\Delta\theta_{VC\_align.}$ and the next visual compass orientation $\theta_{VC(x_{i+k})}$ at the robot pose x_{i+k} as a globally referenced pose, the correction angle $\Delta\theta_{VC\_corr(x_{i+k})}$ is added to the robot orientation $\theta_{(x_{i+k})}$, as

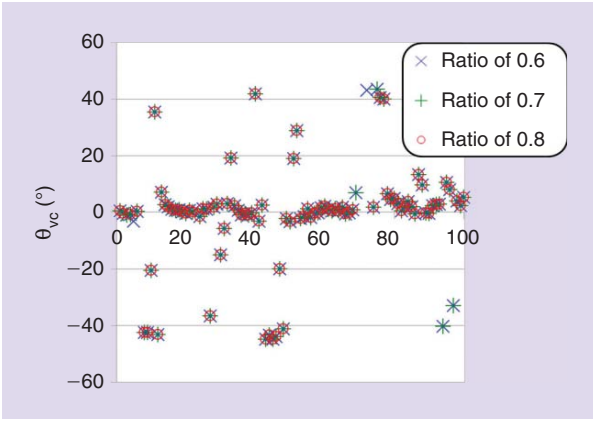

Figure 6. Visual compass orientations under different ratios of the orthogonal lines to the total lines. The minimum line length is set to 120 pixels.

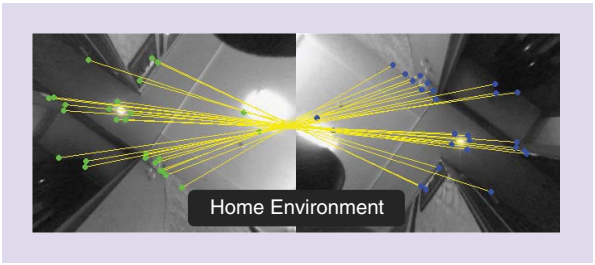

Figure 7. The matching result between two images. (Images courtesy of the Future IT Laboratory of LG Electronics, Inc.)

$$\Delta\theta_{VC\_\mathrm{corr.}(x_{i+k})} = (\theta_{VC(x_{i+k})} - \theta_{(x_{i+k})}) - \Delta\theta_{VC\_\mathrm{align.}} \\ (= (\theta_{VC(x_{i+k})} - \theta_{VC(x_i)}) - (\theta_{(x_{i+k})} - \theta_{(x_i)})) \\ \theta^{+}_{(x_{i+k})} = \theta_{(x_{i+k})} + \Delta\theta_{VC\_\mathrm{corr.}(x_{i+k})}. \quad (5)$$

Here, $\Delta\theta_{VC\_\mathrm{corr.}(x_{i+k})}$ is also represented in the range of $(-45° \leq \theta < 45°)$ when computing $\Delta\theta_{VC\_\mathrm{corr.}(x_{i+k})}$. Under normal conditions, the correction angle $\Delta\theta_{VC\_\mathrm{corr.}(x_{i+k})}$ must be represented in the range of $(-45° \leq \theta < 45°)$. If this assumption is not satisfied, it indicates that the visual compass is not available. In this study, $\Delta\theta_{VC\_\mathrm{align.}}$ computed at the pose x_i is not changed to provide the absolute orientation information with respect to, x_i. For this reason, we refer to θ_{VC} as the *visual compass orientation*.

There are two thresholds to consider when evaluating the validity of the visual compass:

- the minimum line length used to calculate the weighted average orientation
- the ratio of the orthogonal lines to the total lines to decide the two thresholds.

We examined the sensitivity of the visual compass with respect to variations of the two thresholds using 100 images captured from an upward-looking camera with a 160° wide-angle lens. In this test, we found that the visual compass accuracy almost does not change when the minimum line length is 100–150 pixels and the ratio is between 0.6 and 0.8.

Figure 6 depicts a test result of the visual compass using the 100 images given three different ratios and the minimum line length of 120 pixels. The three ratios of 0.6, 0.7, and 0.8 were satisfied in 95, 93, and 88 images, respectively. In the test, the visual compass output changed relatively little compared to variations of the ratio threshold. This result indicates that the visual compass is not sensitive to parameter tuning even though parameter tuning for the visual compass depends on the camera's field of view and image resolution.

In this study, we used a minimum line length of 120 pixels and the ratio of 0.7 because no false positives existed in various home environmental experiments that satisfy an orthogonal structure assumption. Using these thresholds, the two visual compass orientations obtained from the images displayed in Figure 3 were −1.06° and 32.86°, respectively.

Feature Extraction and Matching

The Harris corner detector method [15], is employed to extract cornerlike features as points of interest for feature matching. The midpoints of straight lines are also considered to be points of interest because they improve the feature matching performance. In this article, the straight lines are extracted when computing the visual compass orientation.

Visual features are only extracted at the original image scale because there is no need to consider the scale invariance property of features in images captured from an upward-looking camera perpendicular to the ground plane. This assumption is satisfactory in most indoor environments. The local descriptors of visual features are designed based on the speeded-up robust features (SURF) method [1].

Visual feature matching between two images is performed by checking the descriptor distance ratio (= the closest neighbor distance/the second closest neighbor distance) and applying the epipolar constraint [14].

Figure 7 shows the matching result between two images captured in a home environment.

Depth from 2-D Motion

This section describes how to initialize the 3-D locations of features from multiple pairs of corresponding features between two images using an upward-looking camera and odometry. In this study, the 3-D locations of features are simply initialized using a triangulation method [10], which is a popular method in computer vision. However, the 3-D locations are not updated by several measurements. Instead, the validity of 3-D locations is carefully evaluated. Odometry is used to predict a camera motion between successive images. Monocular ego-motion estimation based on the epipolar geometry may be additionally used to predict a camera motion. However, it was relatively inaccurate compared to the odometry in a short movement trajectory because of the use of the wide-angle lens and low-resolution image. Furthermore, the absence of scale information in the epipolar geometry makes it harder to combine odometry with monocular ego-motion estimation.

A large error of the camera motion used in the triangulation method may sometimes occur due to wheel slippage. In

this case, all 3-D locations must be considered to be outliers. To cope with this problem, three successive images are used to distinguish outliers caused by a large motion error. First, the 3-D locations of features between the first and second images are initialized using the triangulation method. A relative transformation between two poses associated with the second and third images is then computed using a set of 3-D locations in the robot base reference frame of the second image and the corresponding image features in the third image. A method to compute the relative transformation between two poses will be presented in the next section. The relative transformation between two poses is again compared with its prediction estimated by odometry to distinguish outliers caused by a large motion error, because the two transformations were far different under wheel slippage situations. For memory saving, a depth of each feature is stored in the database and the 3-D location of the feature is computed with its depth based on the pinhole camera model if necessary. Figure 8 shows a 3-D map reconstructed from successive images.

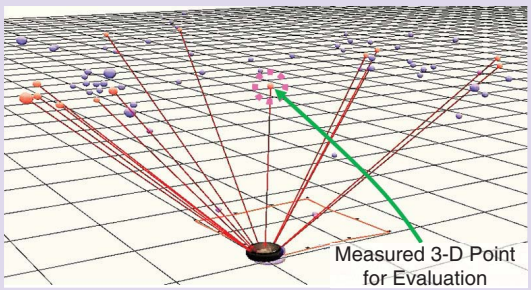

Figure 8. The 3-D reconstruction result from successive images.

Pose Constraint Estimation

For our visual SLAM, successful data association refers to the attainment of an accurate transformation between neighboring poses when producing a spatial pose constraint. In this study, odometry is primarily utilized to obtain the incremental constraint between successive poses and we adopt the odometry motion model algorithm described in [9] to obtain the covariance matrix $W_{i,j}^{-1}$ for the incremental constraint $z_{i,j}$. In the previous section, we have explained how to obtain a set of the valid 3-D locations of features at each pose using a triangulation method. The loop closure constraint $z_{i,j} = [\Delta x_{(z_{i,j})}, \Delta y_{(z_{i,j})}, \Delta\theta_{(z_{i,j})}]^T$ between two nonconsecutive poses is obtained based on a nonlinear optimization technique given 2-D image features in the image coordinate system of the pose x_i and the corresponding 3-D points in the robot base reference frame of the pose x_j. The cost function minimized by adjusting the optimized variables $\{\Delta x_{(z_{i,j})}, \Delta y_{(z_{i,j})}, \Delta\theta_{(z_{i,j})}\}$ is given by:

$$\sum_s \Big(\underbrace{\psi_{(x_i)}[s] - C_{\text{calib.}}\, T_{\text{Robot}}^{\text{Camera}}\,(Rp_{(x_j)}[s] + T)}_{=\varepsilon(s)}\Big)^T \times \Big(\psi_{(x_i)}[s] - \underbrace{C_{\text{calib.}}\, T_{\text{Robot}}^{\text{Camera}}\,(Rp_{(x_j)}[s] + T)}_{=h(s)}\Big), \quad (6)$$

where

$$R = \begin{pmatrix} \cos(\Delta\theta_{(z_{i,j})}) & -\sin(\Delta\theta_{(z_{i,j})}) & 0 \\ \sin(\Delta\theta_{(z_{i,j})}) & \cos(\Delta\theta_{(z_{i,j})}) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (7)$$

$$T = \begin{pmatrix} \Delta x_{(z_{i,j})} \\ \Delta y_{(z_{i,j})} \\ 0 \end{pmatrix}. \quad (8)$$

$C_{\text{calib.}}$ and $T_{\text{Robot}}^{\text{Camera}}$ are a known camera calibration matrix and a known coordinate transformation from the 3-D robot base coordinates to the 3-D camera coordinates, respectively, $\psi_{(x_i)}[s]$ is the sth measured feature location $[u, v, 1]^T$ in the image coordinate system of the pose x_i, and $p_{(x_j)}[s]$ is the sth 3-D feature point $[x, y, z]^T$ in the robot base reference frame of the pose x_j.

Algorithm 1. Relative transformation calculation.

Input: 2-D image features at the pose x_i, 2-D image features and their 3-D points at the pose x_j.

1) Match 2-D image features between the two poses x_i and x_j using the feature matching module.
2) Obtain multiple pairs between the 2-D image features at the pose x_i and the corresponding 3-D points at the pose x_j with the matching result.
3) Choose five random pairs among the multiple pairs.
4) Compute the relative transformation $z_{i,j}$ using a nonlinear optimization algorithm.
5) Check the number of inliers and their re-projection error.
6) Repeat lines 3–5 until n relative transformations for $z_{i,j}$ are obtained.
7) Determine the best hypothesis among the n relative transformations.

An approach for minimizing (6) is to iteratively adjust the optimized variables $\{\Delta x_{(z_{i,j})}, \Delta y_{(z_{i,j})}, \Delta\theta_{(z_{i,j})}\}$ with a correction term $\delta_{z_{i,j}}$ computed by the Gauss-Newton form [2], as

$$\delta_{z_{i,j}} = (J_h^T J_h)^{-1} J_h^T \varepsilon, \quad (9)$$

where J_h is the Jacobian matrix $\partial h/\partial z_{i,j}$, and ε is the matrix form of $\varepsilon(s)$. Here, $(J_h^T J_h)^{-1}$ computed in the final iteration process means the 3×3 covariance matrix of the solution and is used as the covariance matrix $W_{i,j}^{-1}$ of $z_{i,j} = [\Delta x_{(z_{i,j})}, \Delta y_{(z_{i,j})}, \Delta\theta_{(z_{i,j})}]^T$. An algorithm for computing the loop closure constraint is briefly summarized in Algorithm 1. In our tests, the high map accuracy was achieved when diagonal terms of the covariance matrix for the loop closure constraint were about one to four times greater than those for the incremental constraint. For satisfying this condition, the odometry motion model parameters described in [9] were heuristically adjusted.

For the alignment of local submaps, two locally referenced poses obtained from two different submaps are used in constructing the loop closure constraint between nonconsecutive submaps. First, the relative transformation

between the two locally referenced poses is computed using Algorithm 1 and is then converted to the relative transformation between the global reference poses of the two submap coordinate frames.

To reduce the computational complexity of SLAM, we introduce two strategies. First, the current pose stored in the map can have a single loop closure constraint using Algorithm 1. Second, the largest loop closure constraint among multiple loop closure candidates is selected for the current pose. These strategies are necessary to apply a pose graph optimization technique in a low-cost embedded system because the computational complexity is proportional to the number of poses and the number of loop closure constraints.

In this study, the visual compass output obtained in a certain short trajectory $\{x_i, ..., x_{i+k}\}$ is used to construct a loop closure constraint that describes only a rotational angle between the two poses x_i and x_{i+k}. This is because the proposed visual SLAM algorithm corrects the robot trajectory to be estimated as nodes of the graph using the spatial constraints identifying the relative transformation between poses. Assume that a set of correction angles of the visual compass $\{\Delta\theta_{VC\_corr.(x_{idx(j)})},\ j = 1, \ldots, n\}$ is obtained in a part of the short trajectory $\{x_i, ..., x_{i+k}\}$, where $idx(j)$ is an index of the pose with the correction angle obtained by the visual compass. Here, the average correction angle $\Delta\theta_{VC\_avg.corr.}$ for the set of correction angles $\{\Delta\theta_{VC\_corr.(x_{idx(j)})}, j = 1, \ldots, n\}$ is used to obtain the loop closure constraint that describes a rotational angle $\Delta\theta_{(z_{i+k,i})}$ between the two poses x_i and x_{i+k}, as

$$\Delta\theta_{(z_{i+k,i})} = \theta_{(x_i)} - (\theta_{(x_{i+k})} + \Delta\theta_{VC\_avg.corr.}), \tag{10}$$

where $\theta_{(x_i)}$ and $\theta_{(x_{i+k})}$ are the orientations of the two poses x_i and x_{i+k}, respectively. In our tests, the weight of the rotational angle $\Delta\theta_{(z_{i+k,i})}$ required to compute (1) was set to a relatively high value compared to that of the rotational angle term in the loop closure constraint obtained using Algorithm 1 because the visual compass orientation was comparatively accurate.

This scheme allows the visual compass to smoothly reduce the orientation errors of the robot trajectory to be estimated as nodes of the graph.

Pose Graph Optimization

Whenever the current submap is closed, the alignment of local submaps is performed by minimizing a sum of weighted square costs expressed by spatial constraints between the global reference poses of the submap coordinate frames. Such a formula is equal to (1) because all of the constraints between the submaps mean the relative transformations between poses. Therefore, the locally referenced poses in each submap and the global reference poses of the submap coordinate frames can be corrected by the same pose graph optimization method.

In (1), $\ominus$ (which is the inverse of compounding operation $\oplus$) [16] is a nonlinearity function. In this article, the constraint $z_{i,j}$ and the nonlinearity function $h(x_i, x_j)$ are given by

$$z_{i,j} = \begin{bmatrix} \Delta x_{(z_{i,j})} \\ \Delta y_{(z_{i,j})} \\ \Delta\theta_{(z_{i,j})} \end{bmatrix}$$
$$\Leftrightarrow h(x_i, x_j) = \begin{bmatrix} \Delta X_{i,j}\cos(\theta_{(x_i)}) + \Delta Y_{i,j}\sin(\theta_{(x_i)}) \\ -\Delta X_{i,j}\sin(\theta_{(x_i)}) + \Delta Y_{i,j}\cos(\theta_{(x_i)}) \\ \theta_{(x_j)} - \theta_{(x_i)} \end{bmatrix}, \tag{11}$$

where $x_i = [x_{(x_i)}, y_{(x_i)}, \theta_{(x_i)}]^T$, $x_j = [x_{(x_j)}, y_{(x_j)}, \theta_{(x_j)}]^T$, $\Delta X_{i,j} = (x_{(x_j)} - x_{(x_i)})$, and $\Delta X_{i,j} = (y_{(x_j)} - y_{(x_i)})$. The nonlinearity of (11) arises from the orientation $\theta_{(x_i)}$ of the pose x_i. An approach for efficiently solving (1) is to linearize a constraint $z_{i,j}$ with respect to the orientation $\theta_{(x_i)}$ [23]. The linearized form of (11) is given by

$$z_{i,j(\text{linearized})} = \begin{bmatrix} \Delta x_{(z_{i,j})}\cos(\theta_{(x_i)}) - \Delta y_{(z_{i,j})}\sin(\theta_{(x_i)}) \\ \Delta x_{(z_{i,j})}\sin(\theta_{(x_i)}) + \Delta y_{(z_{i,j})}\cos(\theta_{(x_i)}) \\ \Delta\theta_{(z_{i,j})} \end{bmatrix}$$
$$\Leftrightarrow h(x_i, x_j)_{(\text{linearized})} = \begin{bmatrix} x_{(x_j)} - x_{(x_i)} \\ y_{(x_j)} - y_{(x_i)} \\ \theta_{(x_j)} - \theta_{(x_i)} \end{bmatrix}. \tag{12}$$

The covariance matrix $W^{-1}_{i,j(\text{linearized})}$ of $z_{i,j(\text{linearized})}$ can also be represented by

$$W^{-1}_{i,j(\text{linearized})} = J_z W^{-1}_{i,j} J_z^T, \tag{13}$$

where J_z is the Jacobian matrix $\partial z_{i,j(\text{linearized})}/\partial z_{i,j}$, and $W^{-1}_{i,j}$ is the covariance matrix of $z_{i,j}$.

Using (12) and (13), (1) is represented in a linearized version and can be solved by a linear solver that handles a linear least-square problem such as $AX = B$ [23]. The covariance matrix of the solution is used to define the covariance matrix $W^{-1}_{i,}$ of any pose x_i in the graph. There is a major problem in using the linearized version of (1), which is that the linearization errors cannot be avoided due to orientation errors of the poses. The orientation errors will have a severe effect on the performance of SLAM. Therefore, the linearized version of (1) will hold for real-world applications, when the orientation errors are small. To overcome this problem, the visual compass has been developed using the orthogonal structure characteristics of an indoor environment and is applied to accurately make the loop closure constraint that represents a rotational angle between nonconsecutive poses.

To more efficiently solve the linearized version of (1), the location and orientation terms of the poses are separately estimated in this article. First of all, the orientation terms of the poses in (1) are directly corrected by a linear solver because the correction of the orientation terms in this article is a linear estimation problem. The corrected orientations are again used to compute (12) and (13), and the linearized version of (1) is obtained. Here, the location terms are only considered in estimation and are corrected by a linear solver that computes the linearized version of (1).

In summary, the pose graph optimization method presented in this section solves two linear estimation problems to

correct the robot trajectory (that is represented based on a hierarchical map structure), and its performance depends on the visual compass accuracy.

Localization

The simple addition of new poses to the graph causes the system to grow unacceptably large over time. Reducing the graph size is necessary if large-scale map making is to be complete in a reasonable amount of time. The basic idea for reducing the map size is to uniformly distribute the poses to be estimated in a space to spatially minimize the number of overlapping poses. To achieve this objective, an occupancy grid map is used. A new pose is added to the graph when the robot is located in an unoccupied grid. This approach is possible because feature matching between two adjacent images captured from an upward-looking camera is possible regardless of a rotation of the robot about the z-axis of the robot base reference frame. If the robot moves in occupied grids, the localization process is performed instead of the pose graph optimization process described in the "Pose Graph Optimization" section.

Shown in Figure 9 is an example of localizing the overlapping pose x_k. Figure 9(a) is a given graph to correct the pose x_k using the loop closure constraint $z_{k,j-2}$. In this study, the overlapping pose x_k is updated based on the Kalman filter concept. Figure 9(b) describes the prediction and observation for the pose x_k with respect to the base reference frame. The prediction of the pose x_k and its covariance matrix are given by

$$\begin{aligned} x_{k(\text{pred.})} &= x_{k-1} \oplus z_{k-1,k} \\ W^{-1}_{k(\text{pred.})} &= W^{-1}_{k-1} + J_{\oplus} W^{-1}_{k-1,k} J^{T}_{\oplus}, \end{aligned} \tag{14}$$

where $J_{\oplus}$ is the Jacobian matrix of compounding operation $\oplus$. In a similar manner, with the loop closure constraint $z_{k,j-2}$ and the pose x_{j-2}, the observation of the pose x_k and its covariance matrix $W^{-1}_{k(\text{observ.})}$ can be computed as

$$\begin{aligned} x_{k(\text{observ.})} &= x_{j-2} \oplus (\ominus z_{k,j-2}) \\ W^{-1}_{k(\text{observ.})} &= W^{-1}_{j-2} + J_{\oplus}(J_{\ominus} W^{-1}_{k,j-2} J^{T}_{\ominus}) J^{T}_{\oplus}, \end{aligned} \tag{15}$$

where $J_{\ominus}$ is the Jacobian matrix of $\ominus$ (which is the inverse of compounding operation $\oplus$). After computing (14) and (15), the pose x_k and its covariance matrix are updated using the standard Kalman filter formulation:

$$\begin{aligned} x_{k(\text{update})} &= W^{-1}_{k(\text{update})} \left(W_{k(\text{pred.})} x_{k(\text{pred.})} + W_{k(\text{observ.})} x_{k(\text{observ.})} \right) \\ W^{-1}_{k(\text{update})} &= \left(W_{k(\text{pred.})} + W_{k(\text{observ.})} \right)^{-1}. \end{aligned} \tag{16}$$

The prediction of the next pose x_{k+1} and its covariance matrix are also given by

$$\begin{aligned} x_{k+1(\text{pred.})} &= x_{k(\text{update})} \oplus z_{k,k+1} \\ W^{-1}_{k+1(\text{pred.})} &= W^{-1}_{k(\text{update})} + J_{\oplus} W^{-1}_{k,k+1} J^{T}_{\oplus}. \end{aligned} \tag{17}$$

Two types of overlapping poses can be considered: one for local localization before closing the current submap and the other for global localization after closing the current submap. In a real scenario, the robot may move to its destination far from the current submap. Before closing the current submap, the robot pose as a locally referenced pose is updated in the current submap using (14)–(17). However, the robot is gradually far from the current submap, and the pose correction in the current submap is finally impossible. In this case, the current submap is automatically closed when the uncertainty of the robot pose with respect to the base reference of the current submap reaches a limit. After that, the robot tries to make the loop closure constraint between the current robot pose and any pose stored in the map. To update the robot pose with the loop closure constraint, poses expressed in (14)–(17) are represented as globally referenced poses.

If the pose x_{k+p} is located in an unoccupied grid and is thus permanently added to the graph, the incremental constraint $z_{k-1,k+p}$ between the two poses x_{k+p} and x_{k-1} is obtained by simply computing $x_{k+p} \ominus x_{k-1}$ with the two poses x_{k+p} [estimated using (16) and (17)] and x_{k-1} [corrected using (1)]. In this study, the covariance matrix $W^{-1}_{k-1,k+p}$ of the incremental constraint $z_{k-1,k+p}$ is heuristically defined by the error covariance matrix predicted from the odometry motion model without regard to W^{-1}_{k+p}, updated using (16) and (17). This scheme prevents the covariance matrix $W^{-1}_{k-1,k+p}$ from becoming smaller than those for other incremental constraints. In our tests, the incremental constraint $z_{k-1,k+p}$ with an overly optimistic covariance matrix sometimes resulted in an inconsistent global map. This problem was more serious when a trajectory between the two poses x_{k+p} and x_{k-p} was long. If the pose x_{k+p} is used as a base reference of new submap, the incremental constraint between the just previous and new submaps can be obtained by representing the pose x_{k+p} as a locally referenced pose in the just previous submap with the aforementioned $z_{k-1,k+p}$ and $W^{-1}_{k-1,k+p}$.

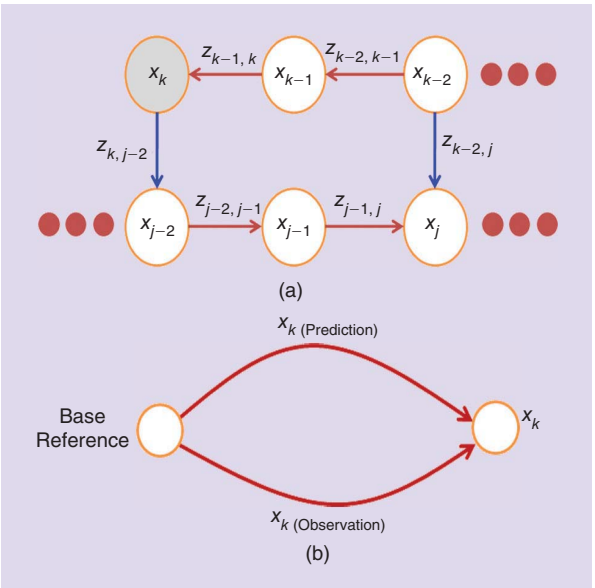

Figure 9. An example of localizing the overlapping pose x_k using the loop closure constraint $z_{k,j-2}$. (a) Given graph. (b) Prediction and observation for x_k.

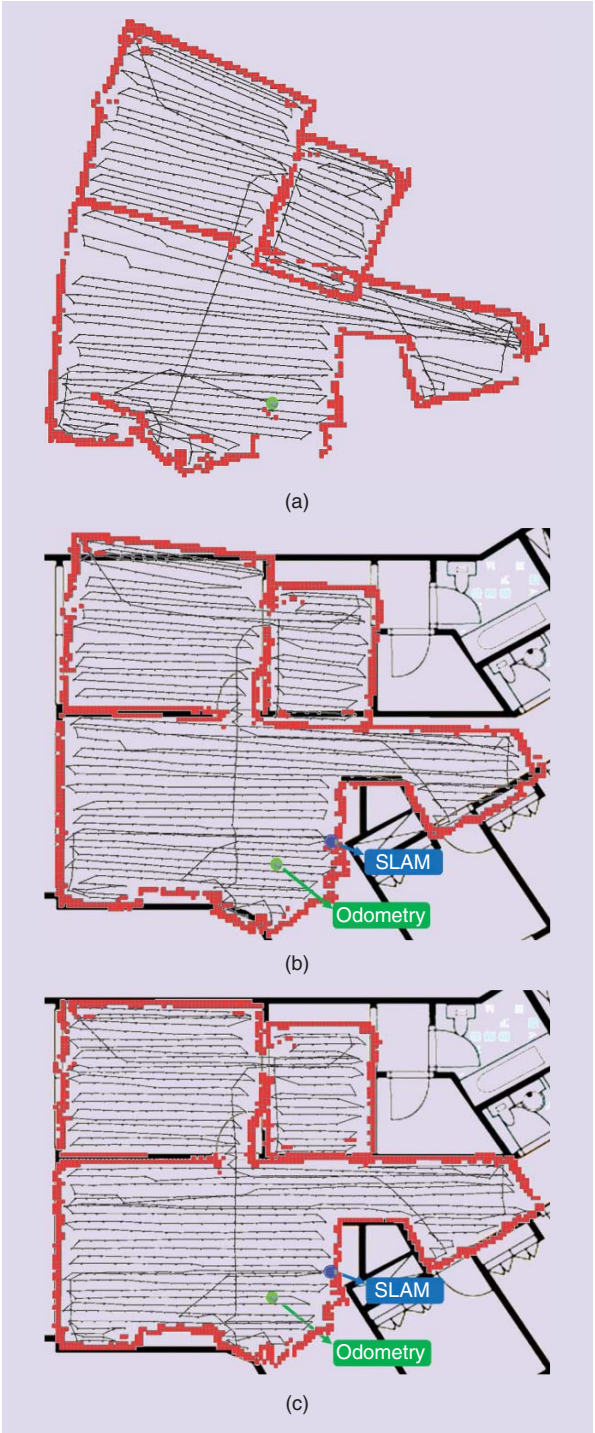

Figure 10. Constructed obstacle maps (in red) "before" and "after" applying the proposed visual compass for the embedded visual SLAM. The black lines inside the map show the trajectory the robot explored. (a) The odometry map without visual SLAM. (b) Visual SLAM without the proposed visual compass. (c) Visual SLAM with the proposed visual compass.

Experimental Results

The proposed visual SLAM algorithm has been implemented in an embedded vision board with an ARM11 processor installed and tested extensively in indoor environments using a home cleaning robot developed by the LG Electronics R&D laboratory, as illustrated in Figure 2. A single camera, pointing upward in the zenith direction toward the ceiling, was employed to capture a 320 × 240-pixel resolution of images with a wide view-angle of 160°. For experimentation, the robot was set to move at a speed of 0.4 m/s, while two incremental encoders were used for robot odometry. All the sensor information and SLAM output were collected through a wireless local area network (LAN). The logged time-stamps were again used to recreate and analyze the experimental results on a laptop computer so that the results did not differ from those obtained by the real robot. Because there was no ground truth available, an obstacle map built with a range sensor, such as a position-sensitive device (PSD), was used instead for the verification of localization accuracy. To this end, the PSD raw data were captured along the robot trajectory.

Performance Analysis on a Laptop Computer

Experiments were conducted in a home environment 10 × 13 m in size. A number of images were collected with the home cleaning robot exploring the room in a zigzag motion. This type of navigation was effective for exploring the entire experimental environment. An experiment corresponds approximately to 406 m of trajectory run with 1,132 images captured. To compare the proposed method with other methods, we tested using the logged data on a laptop computer.

First, we evaluated the accuracy of the proposed method under a significant amount of odometry error. To clarify how much the proposed visual compass contributes to the performance of our embedded visual SLAM, we divided the experimental evaluation into two for comparison: 1) the performance "before (or without)" and 2) the performance "after (or with)" the application of the proposed visual compass. Note that when the visual compass module is omitted in the proposed method, it is comparable to the CV-SLAM technique proposed by [17] since both resort to an upward-looking camera configuration as well as Harris corner features. But, unlike our embedded visual SLAM, the CV-SLAM is not designed for large-scale mapping of indoor environments.

Figure 10 shows the result of experimental investigation on the performance comparison between "before" and "after" applying the proposed visual compass. Figure 10(a) illustrates a large amount of odometry error to be accumulated with no visual SLAM present. Figure 10(b) shows that such a large amount of odometry error accumulated can be considerably reduced based on the visual SLAM even without the proposed visual compass implemented. However, as seen from the red obstacle map of Figure 10(b) that is built without the aid of visual compass, there still remain quite significant, an unacceptable level of, distortions. As shown in Figure 10(c), after the proposed visual compass is applied, the map distortions illustrated in Figure 10(b) without visual compass practically disappear such that the resulting map becomes quite close to the ground truth data. In this experiment, an area that

violates an orthogonal structure assumption existed (see the right part of maps in Figure 10). However, the visual compass did not suffer from this problem because the ratio of 0.7 (as a threshold for the visual compass) used in this study was not satisfied in the vicinity of the area (refer to the "Visual Compass" section for more details). Figure 10 illustrates a typical performance comparison between "before" and "after" the proposed visual compass, consistent among a large set of experiments. This indicates that the proposed visual compass provides a visual SLAM with an excellent means of enhancing its performance, as long as the environment where the visual SLAM is applied allows proper features for the visual compass. It would be interesting to see the visual compass used as a virtual sensor for global orientation in the future for various other robot applications.

Second, we evaluate experimentally the computational efficiency associated with the embedded visual SLAM with the proposed visual compass in comparison with the conventional graph-based SLAM. More specifically, we measure the execution time for the proposed method and for the conventional pose graph optimization method based on an Intel Core i7-2620M CPU running at 2.7 GHz. The conventional pose graph optimization method is to correct all poses stored in a global map using their loop closure constraints such that its computational complexity is similar to the standard EKF SLAM.

Figure 11 shows the computational cost per image frame in terms of the number of frames (up to 1,132 frames) to be processed for the conventional pose graph optimization method (in yellow) and for the proposed method (in red). As seen in Figure 11, the computational cost of the proposed method is very low compared to the conventional pose graph optimization method. More importantly, the execution time of the proposed method does not increase in terms of the number of frames to be processed. This is due to the fact that the proposed method resorts to 1) an efficient hierarchical pose graph optimization process with the two linear estimation problems separately applied to orientation and translation, where the proposed visual compass is to minimize the linearization errors and 2) a localization process for eliminating overlapping poses based on an occupancy grid map. Note that the computational burden associated with the hierarchical management of pose graph optimization turns out to be negligible since there are only 39 submaps generated at the end of the exploration. The high level of computational efficiency as well as the complexity of the proposed method, as demonstrated in Figure 11, clearly verifies why the proposed method offers a computationally feasible approach to the development of low-cost embedded visual SLAM.

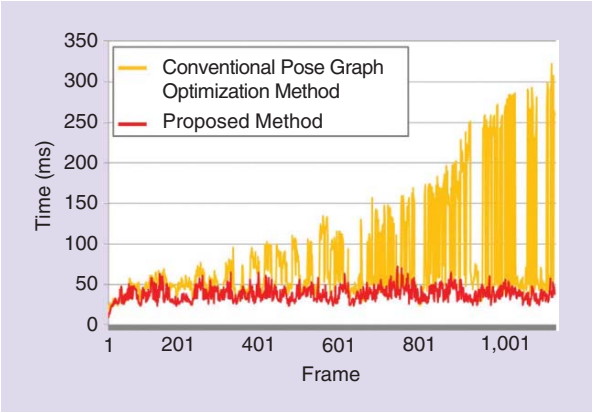

Figure 11. The execution time per image frame in terms of the number of frames for the conventional pose graph optimization method (in yellow) and for the proposed method (in red).

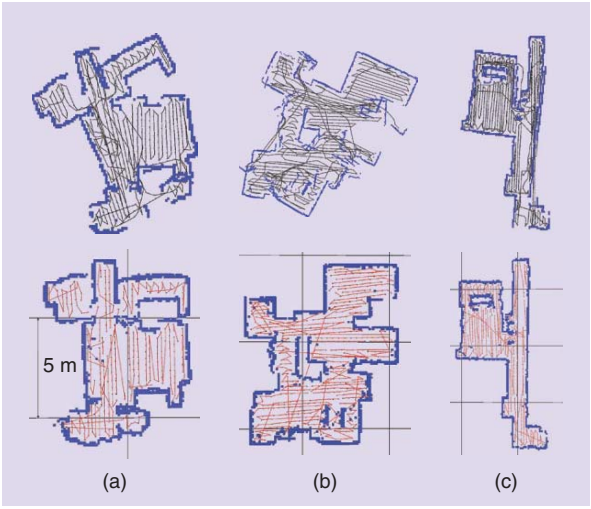

Figure 12. Maps constructed at the end of the exploration of the three experimental environments. Top row: the obstacle grid maps generated using odometry. The black lines show the robot's movement trajectories during exploration. Bottom row: the obstacle grid maps built using our visual SLAM. The red lines show the robot trajectory for each submap in our hierarchical map structure. (a) First place. (b) Second place. (c) Third place.

Processing Speed on an Embedded Vision Board

Three online experiments were conducted in different indoor environments to analyze the computational cost of the proposed method on our embedded vision board. Numerous images were collected by the home cleaning robot moved based on the aforementioned zigzag motion-based navigation.

When the zigzag motion-based navigation was used to build a map, there were submaps with a few poses during exploration. However, the submaps could not be used to obtain a loop closure constraint. For this reason, the submaps with a few poses (which have continuity with the time) merged into a single submap to reduce the number of unnecessary submaps, and they were disregarded in representing the whole map. Note that this scheme does not influence the map accuracy.

Figure 12 illustrates the obstacle maps generated using odometry and the proposed method at the end of the exploration of the three experimental environments. The robot trajectory stored as nodes of the graph for each submap is drawn with red lines, and the obstacle grid maps built with the PSD sensors are represented using blue grids. Although there was a

Table 1. Attained results and computational costs in the three experiments.

Experiment	First	Second	Third
Length	203.3 m	332.7 m	314 m
Total poses	614	983	948
Remaining poses	323	511	462
Submaps	13	27	26
Average processing speed in unoccupied grids	1.99 Hz	1.97 Hz	1.83 Hz
Average processing speed in occupied grids	3.92 Hz	4.52 Hz	4.01 Hz

Figure 13. The average computational time for the runs of each experimental environment.

significant amount of odometry error in the three experiments, the proposed method successfully built accurate maps.

The meaningful data and computational costs for the proposed method in all three experiments are summarized in Table 1. First, the remaining poses as nodes of the graph in all submaps did not increase with the continued traversal of a region due to the overlapping pose elimination in occupied grids and they were much smaller than the total poses. Additionally, the number of submaps was very small. Therefore, the hierarchical pose graph optimization process could be performed in a constant time. For this reason, the proposed visual SLAM algorithm could operate at about 2 Hz for correcting the robot trajectory stored as nodes of the graph in unoccupied grids and about 4 Hz for performing the robot localization in occupied grids. These processing speeds were measured in our embedded vision board with an ARM11 processor running at 533 MHz while multiple tasks, such as motion control, path planning, and obstacle map building, were being concurrently performed.

Figure 13 shows the average computational time per image frame for each experiment. The lowest computational time is shown in the second experimental place because the processing time to obtain a valid loop closure constraint in the second experimental place was lower than the other experimental places. However, the computational time difference between the three experiments is comparatively small, as shown in Figure 13. This result verifies that the performance of the proposed method is not sensitive to a change in the environment.

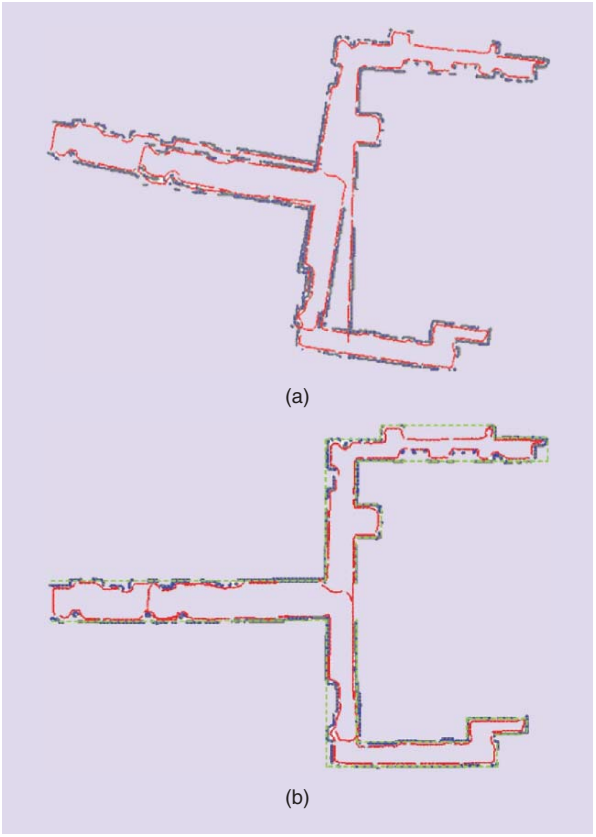

Figure 14. A large-scale mapping result; the red lines show the robot trajectories estimated by odometry and our visual SLAM, while the blue obstacle grid maps built using the PSD sensors are used to visualize the localization accuracy. The ground truth (including big obstacles such as doors and walls) is shown by the green dotted lines. (a) The odometry map. (b) Visual SLAM.

Large-Scale Map Building on an Embedded Vision Board

An online experiment was conducted to evaluate the performance of the proposed method in a large-scale indoor building. The environment was 28 × 40 m, and the sensor data were saved by the home cleaning robot that was driven manually along the walls. The robot's path covered about 233 m, and 706 images were collected.

A significant amount of odometry error was accumulated in this experiment, as shown in Figure 14(a). However, the proposed method built an accurate map in real time. Figure 14(b) shows the comparison result between the final map at the end of exploration and the ground truth data drawn with green dotted lines.

In the experiment, the processing speed of the proposed method was almost the same as that of Table 1 because the robot pose error was not significantly accumulated when closing loops between local submaps and the hierarchical pose graph optimization process could be performed in a constant time.

This experimental result demonstrates that the proposed method is a practical approach for real-time large-scale mapping on low-cost robot platforms with limited sensory and computational equipment.

Conclusion

In this article, we showed that the proposed visual SLAM algorithm implemented on an embedded system can be successfully applied to resource-limited consumer robots such as home cleaning robots. The proposed algorithm is made efficient and robust, in particular, by taking advantage of the orthogonal structure of indoor environments and the invariance property offered by an upward-looking camera.

First, we proposed an algorithmic visual compass that yields the absolute orientation information of a mobile robot by utilizing the orthogonal lines of an environment detected in an image captured from an upward-looking camera. Using the algorithmic visual compass, we were able to simplify a SLAM problem to two linear estimation problems (that correct the robot trajectory represented based on a hierarchical map structure) because the robot orientation error can be significantly reduced. This resulted in the SLAM algorithm to be operated in real time with the map accuracy well maintained even in large-scale indoor environments.

Second, in contrast to a frontal view camera, the scale invariance property of features that goes with the upward-looking camera in most indoor/home environments allowed us to use only the 2-D image features extracted at the original image scale. This helped speed up the feature matching process.

Finally, to further reduce the computational burden, we integrated an additional localization process based on the standard Kalman filter into the SLAM framework because the computational complexity of the pose graph optimization process based on a hierarchical map representation is still proportional to the number of poses (or nodes) in the graph. In the proposed method, the pose graph optimization process was only performed in unoccupied grids. This approach was possible because feature matching between two adjacent images captured from an upward-looking camera can be performed regardless of a rotation of the robot about the z axis of the robot base reference frame.

We hope this article will attract the attention of researchers in this field toward the advancement of lightweight, low-cost, yet highly reliable and real-time, embedded SLAMs, promoting a wide spread of SLAM-based consumer robots in the coming years.

Acknowledgments

This work was supported in part by the Future IT Laboratory of LG Electronics Inc., Korea, KORUS-Tech Program (KT-2008-SW-AP-FSO-0004) of the Ministry of Knowledge Economy, Korea, and NRF-2013M1A3A02042335, the Ministry of Science ICT and Future Planning, Korea.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features," *Int. J. Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[2] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer-Verlag, 1999.

[3] S. Lee, S. Lee, and J. J. Yoon, "Illumination-invariant localization based on upward looking scenes for low-cost indoor robots," *Adv. Robot.*, vol. 26, no. 13, pp. 1443–1469, 2012.

[4] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE Int. Conf. Computer Vision*, Beijing, China, 2003, pp. 1403–1410.

[5] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós, "Mapping large loops with a single hand-held camera," in *Proc. Robotics: Science Systems*, Atlanta, GA, 2007.

[6] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "MINERVA: A second generation mobile tour-guide robot," in *Proc. IEEE Int. Conf. Robotics Automation*, Detroit, MI, 1999, pp. 1999–2005.

[7] H. F. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Automat. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.

[8] T. Bailey and H. F. Durrant-Whyte, "Simultaneous localization and mapping: Part II," *IEEE Robot. Automat. Mag.*, vol. 13, no. 2, pp. 108–117, 2006.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[10] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, MA: Cambridge Univ. Press, 2000.

[11] J. Folkesson, P. Jensfelt, and H. Christensen, "Vision SLAM in the measurement subspace," in *Proc. IEEE Int. Conf. Robotics Automation*, Barcelona, Spain, 2005, pp. 30–35.

[12] J. Montiel and A. J. Davison, "A visual compass based on SLAM," in *Proc. IEEE Int. Conf. Robotics Automation*, Orlando, FL, 2006, pp. 1917–1922.

[13] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Proc. Robotics: Science Systems*, Atlanta, GA, 2007.

[14] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *Int. J. Comput. Vis.*, vol. 27, no. 2, pp. 161–195, 1998.

[15] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conf.*, Manchester, U.K., 1988, pp. 147–151.

[16] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robot.*, vol. 4, no. 4, pp. 333–349, 1997.

[17] W. Y. Jeong and K. M. Lee, "CV-SLAM: A new ceiling vision-based SLAM technique," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Systems*, Edmonton, AB, Canada, 2005, pp. 3195–3200.

[18] J. M. M. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular SLAM," in *Proc. Robotics: Science Systems*, Cambridge, CA, 2006, pp. 16–19.

[19] N. Karlsson, E. D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vSLAM algorithm for robust localization and mapping," in *Proc. IEEE Int. Conf. Robotics Automation*, Barcelona, Spain, 2005, pp. 24–29.

[20] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard, "Visual SLAM for flying vehicles," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1088–1093, 2008.

[21] P. Rybski, F. Zacharias, J. Lett, O. Masoud, M. Gini, and N. Papanikolopoulos, "Using visual features to build topological maps of indoor environments," in *Proc. IEEE Int. Conf. Robotics Automation*, Taipei, Taiwan, 2003, pp. 850–855.

[22] A. Koenig, J. Kessler, and H.-M. Gross, "A graph matching technique for an appearance-based, visual SLAM-approach using Rao-Blackwellized particle filters," in *Proc. IEEE Int. Conf. Intelligent Robots Systems*, Nice, France, 2008, pp. 1576–1581.

[23] K. Konolige, "SLAM via variable reduction from constraint maps," in *Proc. IEEE Int. Conf. Robotics Automation*, Barcelona, Spain, 2005, pp. 667–672.

[24] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to real-time visual mapping," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1066–1077, 2008.

[25] S. Lee, E. Kim, and Y. Park, "3D object recognition using multiple features for robotic manipulation," in *Proc. IEEE Int. Conf. Robotics Automation*, Orlando, FL, 2006, pp. 3768–3774.

Seongsoo Lee, Future IT Laboratory, LG Electronics Inc., Seoul, Korea. E-mail: seongsoo007@gmail.com.

Sukhan Lee, School of Information and Communication Engineering and Department of Interaction Science, Sungkyunkwan University, Suwon, Korea. E-mail: lsh@ece.skku.ac.kr.