2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Indoor Mapping Using Planes Extracted from Noisy RGB-D Sensors*

Tae-kyeong Lee, Seungwook Lim, Seongsoo Lee, Shounan An, and Se-young Oh, *Senior Member, IEEE*

Abstract— **This paper presents a fast and robust plane feature extraction and matching technique for RGB-D type sensors. We propose three algorithm components required to utilize the plane features in an online Simultaneous Localization and Mapping (SLAM) problem: fast plane extraction, frame-to-frame constraint estimation, and plane merging. For the fast plane extraction, we estimate local surface normals and curvatures by a simple spherical model and then segment points using a modified flood fill algorithm. In plane parameter estimation, we suggest a new uncertainty estimation method which is robust against the measurement bias, and also introduce a fast boundary modeling method. We associate the plane features based on both the parameters and the spatial coverage, and estimate the stable constraints by the cost function with a regulation term. Also, our plane merging technique provides a way of generating local maps that are useful for estimating loop closure constraints. We have performed real-world experiments at our lab environment. The results demonstrate the efficiency and robustness of the proposed algorithm.**

I. Introduction

There has been much interest in building 3D maps of indoor environments, which are needed for a wide range of robotic applications such as navigation [1], manipulation [2], augmented reality [3], etc. RGB-D sensors, such as Microsoft Kinect, are well suited for this purpose because it can obtain both RGB and depth images in VGA resolution at a frame rate of 30 Hz. For RGB-D sensor based mapping, many researchers have adopted graph-based approaches to Simultaneous Localization and Mapping (SLAM) due to its consistency over a long run [4][5]. Many of these approaches utilize visual features such as the Scale Invariant Feature Transform (SIFT) [6] and the Speeded-Up Robust Features (SURF) [7] extracted from RGB data to obtain the initial transformation between camera frames, and then calculate the refined constraint by matching raw point clouds through the Iterative Closest Point (ICP) algorithm [4]. These approaches result in dense 3D maps of the environments represented by millions of points. However, due to a high computational cost for matching large sensor data, they are hard to be implemented as real-time without the help of additional parallel computing resources such as GPU [8]. They also require a heavy rendering process for visualization due to a large number of points to draw. For this reason, there have been various approaches to reduce the data dimension by extracting geometric features from the raw sensor data.

Many approaches assume that major indoor structures, like walls, floors, cupboards, etc., can be represented by sets of lines or planes [9][10][11]. Using planes instead of raw point cloud has several advantages including data reduction, fast matching, and fast rendering in visualization. However, for online SLAM, they need to be extracted very fast. Some earlier approaches in plane extraction used normal vectors to segment points [12] but the computational load was high due to nearest neighborhood search and eigen-decomposition of local covariance matrixes. Poppinga et al. [13] suggested a fast region growing method by exploiting the sequential nature of 3D data acquisition. However, they needed to update the plane model every time a new point is added to the plane segment. Recently, Holz et al. [14] suggested a real-time plane segmentation algorithm, which calculates the local surface normals using integral images and then cluster points in both normal and distance space. Although this algorithm processes up to 30 Hz (with QQVGA data), it does not give a way to estimate parameter uncertainty.

In order to make use of plane features in SLAM, we need proper estimation of the parameter uncertainty. In literatures, it is usually assumed that the point segmentation for planes (or lines) is perfect and that the measurements contain Gaussian noise. Under this assumption, the covariance matrix can be estimated from a maximum likelihood formulation [13]. However, these assumptions are usually violated in real-world applications due to algorithmic and physical limitations, such as pixel quantization and lens distortions (See Appendix for analysis on the Kinect measurement bias). In this situation, the conventional approaches could significantly underestimate the parameter uncertainty, resulting in matching failures at data association. Another requirement of plane features is a way of merging multiple planes measured from the same planar region. This process helps reducing both the data association ambiguity and the data size. However, not much literatures deal with this problem.

In this paper, we present three necessary components for utilizing the plane features, extracted from noisy RGB-D sensors, in online SLAM frameworks. First, we present a fast plane extraction method. We estimate the local normal vectors and edges using a local spherical approximation, and then segment the depth image through a flood fill algorithm. To handle the sensor bias, we recommend a new way of estimating the covariance of plane parameters. A fast plane boundary approximation method is also presented, which

*Resrach supported by Future IT Laboratory of LG Electronics Inc.

T. Lee is with the Department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Kyungbuk, Korea (corresponding author to provide phone: +82-54-279-2904; fax: +82-54-279-5594; e-mail: devilee@postech.ac.kr).

S. Lim is with the Pohang University of Science and Technology (POSTECH), Pohang, Kyungbuk, Korea (e-mail: ysw0536@postech.ac.kr).

S. Lee is with the Future IT Laboratory, LG Electronics Inc., Seoul, Korea (e-mail: seongsu.lee@lge.com).

S. An is with the Future IT Laboratory, LG Electronics Inc., Seoul, Korea (e-mail: ethan.an@lge.com).

S. Oh is with the Department of Creative IT Excellence Engineering and Future IT Innovation Laboratory, Pohang University of Science and Technology (POSTECH), Pohang, Kyungbuk, Korea (e-mail: syoh@postech.ac.kr).

enables a real-time rendering by reducing the data size. Second, we suggest a robust frame-to-frame constraint estimation method. The planes are associated in both the parameter space and the spatial coverage (defined as the spatial distribution of the segmented points) to reduce the perceptual aliasing. The stability of constraint estimation is improved by adding a regulation term to the cost function. Third, we propose a method to merge planes of multiple fames and generate a local map frame. The local maps can be used to detect a loop closure, which is a revisiting of the same place after a long distance of travel. To validate our method, we have implemented a SLAM system that utilizes only the depth information from the RGB-D sensor and the odometry information from the robot. The algorithm was tested on a real-world dataset obtained at our laboratory. In the result section, we present a computation time analysis, a matching performance comparison with a conventional approach, and the resulting map of the environment.

In the next section we present the plane model and its spatial transformation. The three algorithm components are presented from Section III to V. Our SLAM framework is briefly described in Section VI. The experiments in Section VII demonstrate the effectiveness of our algorithm.

II. PLANE MODEL AND ITS SPATIAL TRANSFORMATION

We use a spherical parameterization, $\psi = (\alpha, \beta, d)$, of a plane as illustrated in Fig. 1. We also represent the spatial coverage of a plane using mean ($\bar{\mathbf{p}}$) and covariance ($C_{\mathbf{p}}$) of the segmented points. Given $\bar{\mathbf{p}}$ and the plane-normal vector ($\mathbf{n}$), the plane parameters are obtained as:

$$\psi = \begin{bmatrix} \alpha \\ \beta \\ d \end{bmatrix} = \begin{bmatrix} \Theta(\mathbf{n}) \\ \bar{\mathbf{p}}^{\top}\mathbf{n} \end{bmatrix}, \tag{1}$$

where $\Theta(\cdot)$ is a function that transforms a normal vector into angels in the spherical coordinate.

Suppose $C_{\mathbf{p}}$ has eigenvalues $(\lambda_0, \lambda_1, \lambda_2)$ and the corresponding orthonormal eigenvectors $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$, where $\lambda_0 < \lambda_1 < \lambda_2$ and $\mathbf{e}_0 \times \mathbf{e}_1 = \mathbf{e}_2$. We define the plane coordinate frame (Fig. 1) such that: the origin is $\bar{\mathbf{p}}$; (x_p, y_p, z_p) axes correspond to $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$. In the plane coordinate frame, plane parameters become:

$$(\alpha_p, \beta_p, d_p) = (0,0,0); \quad \bar{\mathbf{p}}_p = (0,0,0).$$

Given a transformation, $T = (\mathbf{R}, \mathbf{t})$, between two camera frames, we can transform the plane parameters as follows:

$$\psi' = \begin{bmatrix} \alpha' \\ \beta' \\ d' \end{bmatrix} = \begin{bmatrix} \Theta(\mathbf{R}\,\mathrm{N}(\alpha, \beta)) \\ \mathbf{t}^{\top}\mathbf{R}\,\mathrm{N}(\alpha, \beta) + d \end{bmatrix}, \tag{2}$$

$$\bar{\mathbf{p}}' = \mathbf{R}\,\bar{\mathbf{p}} + \mathbf{t}, \tag{3}$$

where the angles are first transformed to a normal vector by a function $\mathrm{N}(\cdot)$ for the rotation. We can also transform the error

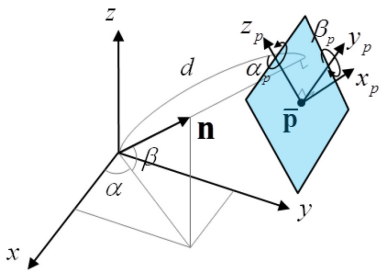


Figure 1. Plane parameters. (x, y, z) and (x_p, y_p, z_p) represent the camera and plane coordinate frame, respectively.

covariance of the plane parameters (C_{ψ}) and the point distribution covariance ($C_{\mathbf{p}}$) as follows:

$$C_{\psi'} = \mathbf{J}_{T,\psi} \begin{bmatrix} C_T & 0 \\ 0 & C_{\psi} \end{bmatrix} \mathbf{J}_{T,\psi}^{\top}, \tag{4}$$

$$C_{\mathbf{p}'} = \mathbf{J}_{T,\mathbf{p}} \begin{bmatrix} C_T & 0 \\ 0 & C_{\mathbf{p}} \end{bmatrix} \mathbf{J}_{T,\mathbf{p}}^{\top}. \tag{5}$$

where C_T is the covariance matrix of the transformation, T; $\mathbf{J}_{T,\psi}$ and $\mathbf{J}_{T,\mathbf{p}}$ represent the Jacobian matrix of (2) and (3), respectively. Estimation of C_{ψ} and $C_{\mathbf{p}}$ is described in Sec. III. D.

III. FAST PLANE EXTRACTION

Fig. 2 shows the overall process of the proposed plane extraction algorithm. It comprises five main procedures: 1) normal and edge extraction; 2) depth image segmentation; 3) plane model initialization using the Random Sample Consensus (RANSAC) [15]; 4) model optimization; and 5) boundary extraction. As a preprocessing step, we estimate the normal vectors and extract the edge pixels from the depth data obtained from the RGB-D sensor. Then we segment the depth image as a group representing a plane candidate. When the group has more number of pixels than a threshold (0.5 percent of the total number of pixels), we apply RANSAC to obtain the initial plane model. To reduce the computational cost, we randomly sample small number of points, and check inliers that fit the model. If the percentage of the inliers is higher than

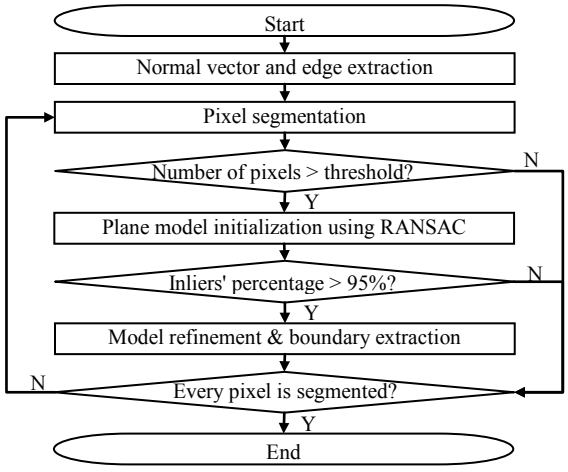


Figure 2. Flow chart of the plane extraction algorithm.

a certain threshold (95%), we refine the plane model and extract a polygonal plane boundary. Otherwise, the group is discarded and no plane is extracted. These procedures are repeated until every depth pixel is segmented.

A. Normal and edge extraction

Existing normal estimation methods can be divided into *optimization-based* methods and *averaging* methods [16]. Although the former, such as local plane fitting, is more accurate, its computational complexity is higher than the latter's. For fast calculation, we follow the latter approach. For a fast neighborhood search, we consider the fixed pixel neighborhood (Fig. 3 (a)) by utilizing the organized structure of RGB-D camera data [14].

In our approach, the local surface is regarded as a part cut from a sphere, approximated as four pieces of triangles as shown in Fig. 3 (b). We calculate the normal vector, $\mathbf{n}_i$, of each triangle by a cross product of the two side vectors. All the vectors are summed and normalized to obtain the normal vector of interest. By exploiting the sphere assumption, curvature at the point (which is inverse of the sphere radius) can be approximated by the following equation:

$$\kappa = 1/R = \sqrt{8\pi(A-B)}/A, \quad \left(A = \sum_i^4 |\mathbf{n}_i|, \; B = \left| \sum_i^4 \mathbf{n}_i \right| \right) \quad (6)$$

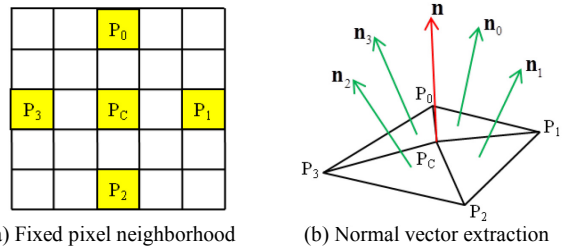
where a half of A represents area of the sphere surface; a half of B represents area of a circle which is projection of the sphere surface onto the tangential plane. Then (6) is exactly derived under the sphere assumption. If the curvature is larger than a predetermined threshold, we mark the pixel as an edge and exclude from the plane extraction.

B. Depth image segmentation

For depth image segmentation, we modify a simple eight-way flood fill algorithm. Starting from a seed point, the segment expands just like a wave propagation. The wave is blocked in several ways: by other segment pixels; edge pixels; and discontinuity of normal vector. The normal vector discontinuity is detected if the inner product between a normal and a reference vector is smaller than an empirically defined threshold value. The seed pixel's normal vector is used as an initial reference vector. During the segmentation, we obtain the average normal vector of the segmented points and replace the reference vector with this one. Fig. 4 depicts an example of the depth image segmentation.

C. Plane model initialization using RANSAC

After the segmentation, we apply a RANSAC algorithm to obtain the initial plane model. During a RANSAC iteration, we randomly choose three points within the segment and obtain the plane model that passes through these points. Then we calculate a score of the model by counting the number of inliers and computing the variance of distances to the plane. In our implementation, we use only a hundred random points to compute the score. After fifty iterations, we select the plane model with the best score as the initial plane model.



(a) Fixed pixel neighborhood (b) Normal vector extraction

Figure 3. Normal vector estimation method.

(a) RGB data (b) Segmented depth image

Figure 4. Example of depth image segmentation. RGB data in (a) is not used in the algorithm. In (b), segments are represented by different colors, and the plane boundaries are shown in white.

D. Model optimization

Due to biased characteristics of the Kinect range data (Appendix), using more points does not always guarantee better plane model. Therefore, we randomly sample a small number, $N_\mathbf{p}$ (we use 100 in our implementation), of inliers for the calculation. First, we calculate the vector $\mathbf{q}$ and the matrix $\mathbf{Q}$, which are defined as:

$$\mathbf{q} = \sum_i^{N_\mathbf{p}} \mathbf{p}_i, \quad \mathbf{Q} = \sum_i^{N_\mathbf{p}} \mathbf{p}_i \mathbf{p}_i^\top, \quad (7)$$

where $\mathbf{p}_i$ is the ith sampled point. Then we can obtain the mean ($\bar{\mathbf{p}}$) and covariance ($C_\mathbf{p}$) of the point distribution as follows:

$$\bar{\mathbf{p}} = \mathbf{q}/N_\mathbf{p}, \; C_\mathbf{p} = \mathbf{Q}/N_\mathbf{p} - \mathbf{q}\mathbf{q}^\top/(N_\mathbf{p})^2. \quad (8)$$

The minimum mean square error solution to the plane normal ($\mathbf{n}$) estimation is obtained as the normalized eigenvector corresponding to the minimum eigenvalue of $C_\mathbf{p}$. Then the plane parameters are calculated as in (1). Here, we make the distance parameter (d) positive by adjusting the direction of $\mathbf{n}$ (as well as α and β). This process helps a lot in data association by disambiguating planes that are parallel but observed at different sides, such as the different sides of a wall.

Due to the sensor bias, the conventional plane covariance estimation method can cause underestimation of the uncertainty. Therefore, we suggest a conservative method by treating the plane as a single measurement instead of a set of many point measurements. The covariance of the plane parameters is first obtained in the plane coordinate frame and then transformed into the camera coordinate frame. The transformation from the camera to the plane coordinate frame is easily obtained by the definition of the plane coordinate

frame (Sec. II). Note: in the following, we use the same notation as in Sec. II.

In the plane coordinate frame, all the parameters are independent to each other. We obtain the depth variance ($\sigma_{\bar{\mathbf{p}}}^2$) of the center of mass ($\bar{\mathbf{p}}$) from a sensor noise model, which we can generate by several experiments. By treating the plane as a single point measurement, we can obtain the depth variance of the plane as:

$$\sigma_{d_p}^2 = \sigma_{\bar{\mathbf{p}}}^2 \left| \mathbf{e}_0 \cdot \left(\bar{\mathbf{p}} / |\bar{\mathbf{p}}| \right) \right|. \quad (9)$$

The square roots of the eigenvalues represent the lengths of a covariance ellipse in the direction of the corresponding eigenvectors. From these values, we can intuitively define the error variance of the rotation parameters:

$$\sigma_{\alpha_p}^2 = \left(\text{atan}\left(\sigma_{d_p} / \sqrt{\lambda_1} \right) \right)^2 , \ \sigma_{\beta_p}^2 = \left(\text{atan}\left(\sigma_{d_p} / \sqrt{\lambda_2} \right) \right)^2 . \quad (10)$$

This seems rather heuristic, but it works well in practice as we have verified experimentally. Then, the error covariance is represented in the plane coordinate frame as follows:

$$C_{\psi_p} = \begin{bmatrix} \sigma_{\alpha_p}^2 & 0 & 0 \\ 0 & \sigma_{\beta_p}^2 & 0 \\ 0 & 0 & \sigma_{d_p}^2 \end{bmatrix}. \quad (11)$$

Then we can transform C_{ψ_p} into the camera frame by (4). Here, C_T is simply set to zero. The obtained plane parameter set is then treated as a single plane measurement with additive Gaussian noise.

E. Boundary extraction

As the final process of the plane extraction, we model the plane boundary as a polygon. The overall process of the boundary extraction is shown in Fig. 5. First we sample all the boundary points of the segment in a boundary following sequence. Then we generate a graph by defining the sampled points as *nodes* and the connections between the neighborhood *nodes* as *edges*. A *node*-weight is defined as the area change that occurs when we remove it, and an *edge*-weight as the total area change that has occurred between the neighborhood nodes. The *edge*-weights are initialized as zeros, and the *node*-weights are calculated as follows:

$$w_i^p = \left(\left(\mathbf{p}_{i-1} - \mathbf{p}_i \right) \times \left(\mathbf{p}_{i+1} - \mathbf{p}_i \right) \right) \cdot \mathbf{n} / 2 + w_{i-1,i}^e + w_{i,i+1}^e , \quad (12)$$

where $\mathbf{p}_i$ and w_i^p are the (i)th *node* and its weight, respectively; $\mathbf{n}$ is the normal vector of the plane; and $w_{i-1,i}^e$ is the *edge*-weight between the ($i-1$)th and the (i)th node. Note that the weights can be either positive or negative according to the type of the *nodes* (either convex or concave). If the number of boundary points is larger than a desired number, we find the *node*-weight which is the nearest to zero and remove the *node*. The *edge*-weight between the two neighborhoods is set to the removed *node*'s weight. Then, the neighborhood *nodes*' weights are updated by (12). This is repeated until the desired abstraction level is satisfied. Then we project the remaining nodes onto the plane to assure that all the points are on the plane.

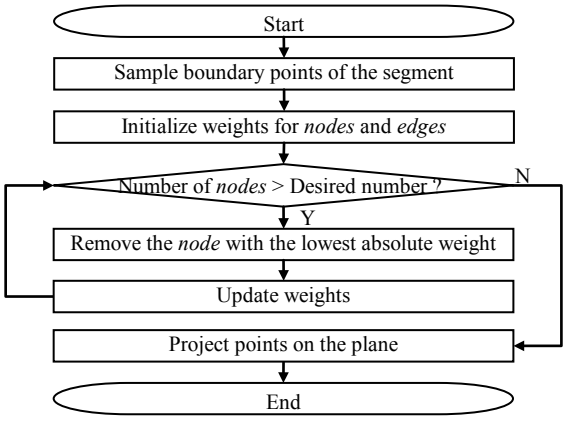


Figure 5. Flow chart of the boundary polygon extraction algorithm.

IV. PLANE MATCHING AND CONSTRAINT ESTIMATION

In this work, we present a preliminary work using only plane features for mapping without using the RGB information obtained from the Kinect sensor. Because of the narrow field of view and the short reliable range of the sensor, using only planes make it difficult to estimate the correct matching as well as the transformation between camera frames. Therefore, we assume the initial transformation T_0 and the covariance matrix C_{T_0} between two frames are given. Between two consecutive frames, odometry sensors, such as wheel encoders and an inertial measurement unit (IMU), can be utilized. Otherwise, we can apply a relative transformation to the posterior estimate obtained so far. Then we can transform the planes in the second frame into the first frame by (2) and (4). We also transform the spatial coverage of the plane, $\bar{\mathbf{p}}$ and $C_{\mathbf{p}}$, as in (3) and (5), respectively.

After the transformation, we obtain the candidate matching pairs by applying the chi-square test, which calculates the Mahalanobis distances between the parameters as follows:

$$\mathrm{X}_{\psi_{i,j}}^2 = \left(\psi_i - \psi_j \right)^{\mathrm{T}} \left(C_{\psi_i} + C_{\psi_j} \right)^{-1} \left(\psi_i - \psi_j \right), \quad (13)$$

$$\mathrm{X}_{\bar{\mathbf{p}}_{i,j}}^2 = \left(\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j \right)^{\mathrm{T}} \left(C_{\mathbf{p}_i} + C_{\mathbf{p}_j} \right)^{-1} \left(\bar{\mathbf{p}}_i - \bar{\mathbf{p}}_j \right). \quad (14)$$

We accept the match only if both $\mathrm{X}_{\psi_{i,j}}^2$ and $\mathrm{X}_{\bar{\mathbf{p}}_{i,j}}^2$ satisfy the desired confidence level. For the matched pairs, the likelihood of matching can be calculated as follows:

$$L_{i,j} = \eta \left(1 / \sqrt{\left| C_{\psi_i} + C_{\psi_j} \right|} \right) \exp\left(-\mathrm{X}_{\psi_{i,j}}^2 / 2 \right), \quad (15)$$

where η is a constant value.

To filter out false-positive matching pairs, we apply RANSAC algorithm. At each iteration, we select three

matching pairs and calculate the relative transformation as follows:

$$T_C = \arg\min_T \left\{ \left(\sum_k \mathrm{X}^2_{\psi_k} \right) + (T - T_o)^\top \mathbf{C}^{-1}_{T_o} (T - T_o) \right\}, \quad (16)$$
$$\mathbf{C}_{T_C} = \mathbf{H}^{-1}_{T_C}$$

where $\mathrm{X}^2_{\psi_k}$ is the Mahalanobis distance between the kth pair of planes as in (13), $\mathbf{H}$ is the Hessian matrix of the cost function. Note that we add the Mahalanobis distance between the new (T) and the initial (T_o) transformations to the cost function as a regulation term to bound the transformation estimate. This term enhances stability compared to using only sparse and noisy plane features, because a plane contains information about only 3 degrees of freedom while a transformation has all 6 degrees. We can solve (16) using iterative linearization methods such as Gauss-Newton method. The score for the estimated transformation is obtained by adding all the matching likelihoods of the inliers. After finding the best constraint estimate, we refine the estimate using all the inliers.

V. Plane Merging

Given a set of camera frames, their relative transformations, and the correspondences of the planes, we can generate a local map frame (Fig. 6). We first transform all the planes into a single reference frame by (2) and (4), and then merge the associated planes. We can derive the merged plane estimate using the maximum likelihood formulation [17] as follows:

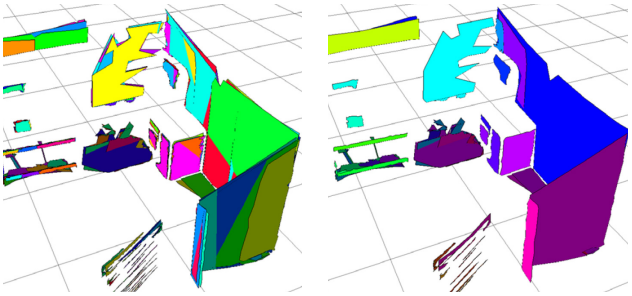
$$\psi_m = C_{\psi_m} \left(\sum_k C^{-1}_{\psi'_k} \psi'_k \right), \; C_{\psi_m} = \left(\sum_k C^{-1}_{\psi'_k} \right)^{-1}. \quad (17)$$

To obtain the merged spatial coverage, the point statistics described in (7) are transformed and merged as follows:

$$\mathbf{q}' = \mathbf{R}\mathbf{q} + N_\mathbf{p}\mathbf{t},$$
$$\mathbf{Q}' = \mathbf{R}\mathbf{Q}\mathbf{R}^\top + \mathbf{R}\mathbf{q}\mathbf{t}^\top + \left(\mathbf{R}\mathbf{q}\mathbf{t}^\top\right)^\top + N_\mathbf{p}\mathbf{t}\mathbf{t}^\top, \quad (18)$$
$$\mathbf{q}_m = \sum_k \mathbf{q}'_k, \; \mathbf{Q}_m = \sum_k \mathbf{Q}'_k, \; N_{\mathbf{p}_m} = \sum_k N_{\mathbf{p}_k}$$

where $(\mathbf{R}, \mathbf{t})$ represent the transformation; $\mathbf{q}'$ and $\mathbf{Q}'$ are the transformed point statistics; $\mathbf{q}_m$, $\mathbf{Q}_m$, and $N_{\mathbf{p}_m}$ are the merged point statistics. Then we can calculate the merged point distribution, $\bar{\mathbf{p}}_m$ and $C_{\mathbf{p}_m}$ by (8).

As the final process, we merge the polygon boundaries into a single polygon. To avoid complexity of calculating all the intersection points and rejoining, we simply rasterize all the boundary lines into a single image plane, where the inverse transformation from the image to the plane coordinate is given. Then we can apply the same process presented in Sec. III. E to obtain the merged polygon boundary. Merging planes helps reducing overlapping feature data and enhancing the matching performance between local map frames. The local map matching follows the process presented in Sec. IV. To filter out false positive matches, we validate the match only if the total area of the matched planes is more than a threshold (we use twenty) percent of the entire plane area. Local map frames are more robustly matched than individual camera frames when the given initial transformation is highly uncertain, because a large variety of planes reduces perceptual aliasing in the RANSAC data association phase.



(a) Before the plane merging (b) After the plane merging

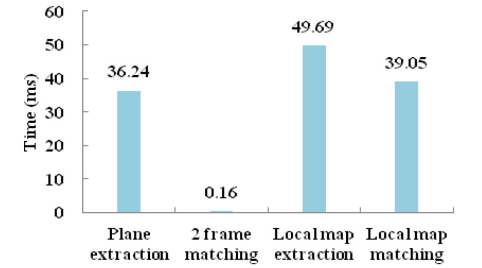
Figure 6. Plane merging.

VI. SLAM Framework

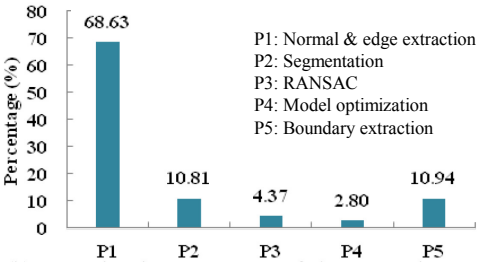
To assess the robustness of plane features, we use only the depth information from the Kinect sensor and the odometry data for SLAM. For real-time processing, we adopt and modify a sliding-window based pose-graph SLAM framework [18]. We use a hierarchical pose-graph structure, where the low-level consists of the camera frames and the high-level comprises the local map frames constructed by merging many arranged camera frames. For the low-level graph optimization, we adaptively resize the sliding window so that information loss is minimized [19]. When a new frame is added, plane features are extracted and matched with every frame in the window to generate multiple constraints, and then the low-level graph within the window is optimized. If a certain condition is satisfied (e.g. reaching the maximum window size), we generate a local map frame by merging the camera frames that are removed from the window. The local map frames are then matched, and the high-level graph is optimized. This structure has advantages of fast optimization due to the reduced number of nodes for the optimization, as well as increased matching performance by using the local maps to detect loop closures.

VII. Experimental Results

For the odometry and sensor data acquisition, we used a Pioneer mobile robot platform and a Kinect camera mounted on it. We have manually controlled the robot to obtain the data at our laboratory, a sectioned rectangular environment with 10.54 by 10.15 meters in size. From this data, we measured the computation time of all the processes in SLAM front-end and the sub-processes in plane extraction. All the processing times were measured on a notebook PC equipped with a Core i7 processor. In the plane extraction, we used 320 by 240 resolution depth images. We averaged the computation time results for each process measured during the entire SLAM process. Fig. 7 shows the result. At a single time step, two-frame matching was processed 60 times in average, resulting in 10 milliseconds for the entire matching. In summary, the total time required to process a new Kinect data was less than 50 milliseconds in average. Plane extraction

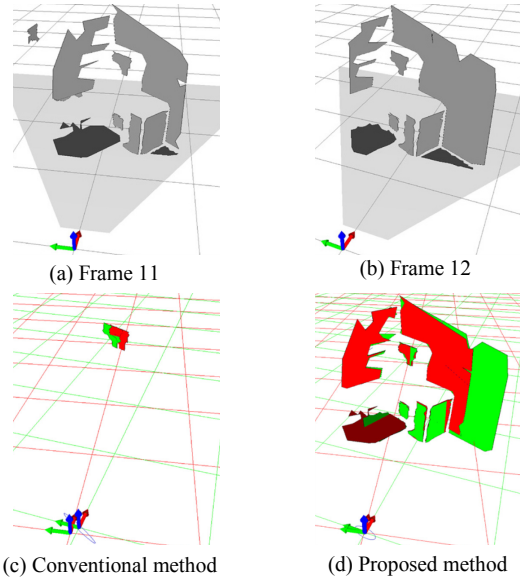


(a) Processing times of the SLAM front-end processes



(b) Processing time percentages of plane extraction steps

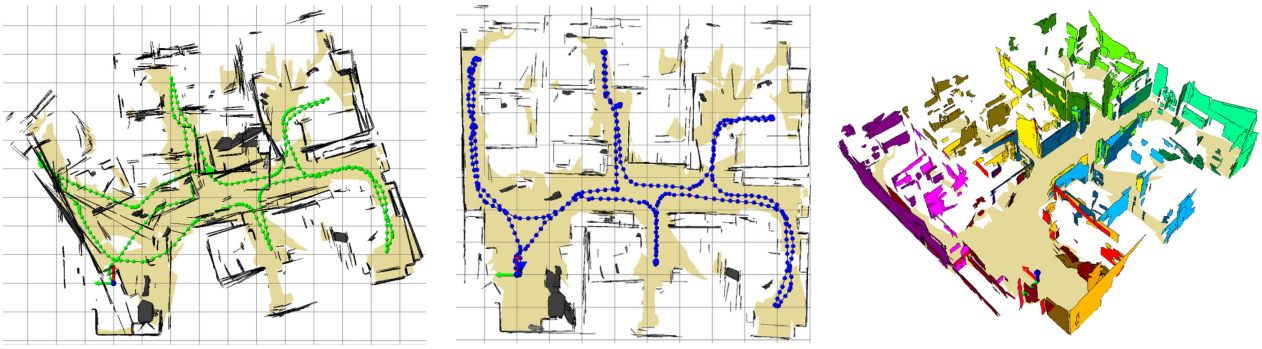
Figure 7. Statistics for the runtime of the algorithms



(a) Frame 11 (b) Frame 12

(c) Conventional method (d) Proposed method

Figure 8. Frame-to-frame matching and constraint estimation results obtained using the two different error covariance estimation methods.



(a) Odometry trajectory (b) Corrected trajectory (c) Perspective view of the entire map

Figure 9. Experimental result obtained at our lab environment.

time is further reduced if we use 160 by 120 resolution depth images (15.1 milliseconds in average). A local map matching took much more time than a two frame matching due to a large number of planes. However, because our implementation on the RANSAC data association process was not optimized, this computing time could be reduced if implemented more efficiently. Within plane extraction process, normal and edge extraction took most of the computing time, recording 68.6 percent as shown in Fig. 7 (b). The second heaviest process was boundary extraction, which took about 10.9 percent (4 milliseconds) in average.

To verify the robustness of our plane-parameter uncertainty estimation method, we compared a typical result of the two-frame matching obtained using the proposed and the conventional method: the latter is based on the assumption of Gaussian noise in the depth measurements. The experimental result is depicted in Fig. 8, where the grid-lines in the bottom are drawn at an interval of one meter. As pictured in Fig. 8 (c), the conventional model failed to match all the possible pairs, yet it results in a narrow error ellipse showing high confidence about the resulting transformation. This result comes from the fact that the conventional approach underestimates the plane-parameter uncertainty due to the sensor bias. On the other hand, our approach succeeded in matching almost all possible pairs of match. Because we regard the plane as a single measurement, the uncertainty of the plane does not get smaller even when more points are used to estimate the error covariance. Although this approach can overestimate the uncertainty, it is more robust than the conventional method when the measurement contains bias.

The final mapping result of the entire lab environment is shown in Fig. 9. In the odometry trajectory (Fig. 9 (a)), notice a significant wheel slippage around the left side sections, resulting in a significant heading error. We could handle this situation by adjusting the error covariance of the initial transformation in (16) so that the planes play a major role in the optimization. The outer lengths of the corrected map (Fig. 9 (b)) contain about 10 centimeters of errors for the both sides. Even though the local map matchings were successful, some features were misaligned because the local maps were not optimized within itself. This map can be further optimized if we add new low-level graph constraints obtained using the local-map plane correspondences and then run a full optimization of the low-level graph. Fig. 9 (c) shows a perspective view of the map rendered by openGL library, where local maps are drawn in different colors while the floor

in a single color for visibility. Due to the low dimension of the plane data, we can render this 3D map almost immediately with the help of the openGL display list functions.

VIII. Conclusions & Future Work

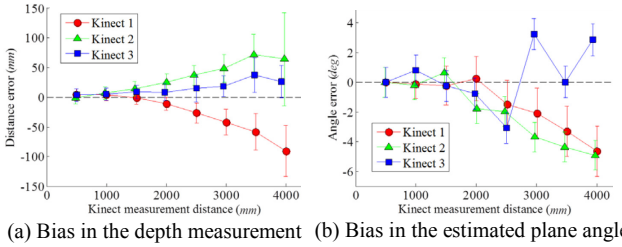
In this paper, we presented a fast plane extraction and matching method for indoor mapping based on the noisy RGB-D sensors. Our approach estimates surface normals and curvatures based on the local spherical model and segment the planar regions very fast. Then we use RANSAC to validate the planarity of the segment and select a small set of inliers to obtain the optimal plane model. Unlike the conventional approaches, we regard the plane as a single measurement, and suggest a new covariance estimation method, which is robust against bias existing in the depth measurement of the Kinect. We then abstract the boundary as a polygon of any desired number of vertices. This abstraction is useful for real-time visualization of the 3D map. For the feature registration, we match the planes based on both the parameters and the spatial coverage, and then estimate the frame-to-frame constraints by the cost function with a regulation term. We also proposed a method to merge planes and generate local map frames for both data reduction and robust global matching.

Even though planes are dominant features in indoor environments, the use of only planes causes loss of information such as edges and corners. Also, due to the limit of reliable depth range, plane by itself cannot cover large open space. Therefore, we would need combination with visual features as well as other geometric features to handle arbitrary indoor environments. For better visualization based on texture mapping, we would also need to apply bundle adjustment [20] to handle the bias inherent in the sensor measurement. We leave these issues as our future work.

Appendix

A. Analysis on the Kinect measurement bias

We conducted additional experiments to examine the bias in depth measurements of uncalibrated Kinect sensors. We placed a large planar object orthogonal to the direction of the sensors. For three Kinect sensors, we varied the distance between the plane and the sensors. For each distance, we obtained 20 frames and the ground truth measured by a laser sensor having the maximum error of 0.5 centimeters. We calculated the average and standard deviation of the depth (Fig. 10 (a)), and also estimated the statistics about the fitted plane angle (Fig. 10 (b)). If the measurement distance is more than two meters, the error becomes larger than the standard deviation, which shows a significant bias in the measurement. In this case, we may not obtain the exact plane parameters even if we use more points for the estimation.



(a) Bias in the depth measurement (b) Bias in the estimated plane angles

Figure 10. Statistics of the depth measurements and the estimated plane angle obtained from the three uncalibrated Kinect sensors. The bars located on the data points indicate the standard deviations.

References

[1] S. May, D. Droeschel, D. Holz, S. Fuchs, E. Malis, A. Nüchter, and J. Hertzberg, "Three-dimensional mapping with time-of-flight cameras," *Journal of Field Robotics*, vol. 26, no. 11-12, pp. 934-965, 2009.

[2] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz,, "Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments," in *IROS*, 2009, pp. 1-6.

[3] S. Lieberknecht, A. Huber, S. Ilic, and S. Benhimane, "RGB-D camera-based parallel tracking and meshing," available at http://wwwnavab.in.tum.de/pub/lieberknecht2011ismar/lieberknecht2011ismar.pdf.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: using depth cameras for dense 3D modeling of indoor evironments," in *the 12th International Symposium on Experimental Robotics* (*ISER*), December 2010.

[5] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *RGB-D Workshop on 3D Perception in Robotics*, Sweden, April, 2011.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, Greece, 1999, pp. 1150-1157.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: speeded up robust features," in *ECCV*, 2006.

[8] D. Neumann, F. Lugauer, S. Bauer, J. Wasza, and J. Hornegger,, "Real-time RGB-D mapping and 3-D modeling on the GPU using the random ball cover data structure," in *ICCV Workshops*, 2011, pp. 1161-1167.

[9] V. Nguyen, A. Harati, A. Martinelli, R. Siegwart, and N. Tomatis, "Orthogonal SLAM: a step toward lightweight indoor autonomous navigation," in *IROS,* 2006, pp. 5007-5012.

[10] J. Weingarten, and R. Siegwart, "3D SLAM using planar segments," in *IROS*, 2006, pp. 3062-3067.

[11] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," in *ICRA*, 2003, pp. 1304-1311.

[12] C. Wang, H. Tanahashi, H. Hirayu, Y. Niwa, and K. Yamamoto, "Comparison of local plane fitting methods for range data," in *CVPR*, 2001, pp. 663-669.

[13] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," in *IROS*, 2008, pp. 3378-3383.

[14] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using RGB-D cameras," in *Proc. 15th RoboCup International Symposium*, Istanbul, 2011.

[15] M. A. Fischler, and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, 1981.

[16] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *ICRA*, 2009, pp. 3206-3211.

[17] S. T. Pfister, "Weighted line fitting and merging," Tech. Rep., California Institute of Technology, 2002, available at: http://robotics.caltech.edu/~sam/TechReports/LineFit/linefit.pdf.

[18] K. Konolige, and M. Agrawal, "FrameSLAM: from bundle adjustment to real-time visual mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066-1077, 2008.

[19] S. Lim, T. K. Lee, S. Y. Oh, "Adaptive Sliding Window for Hierarchical Pose-Graph-Based SLAM," *in International Conference on Control, Automation and Systems (ICCAS)*, 2012, Jeju, Korea.

[20] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *Proc. the International Workshop on Vision Algorithms: Theory and Practice*, September 21-22, 1999, pp. 298-372.