Parallel Line-based Structure from Motion by Using Omnidirectional Camera in Textureless Scene

Ryosuke Kawanishi, Atsushi Yamashita, Toru Kaneko and Hajime Asama

Abstract

In this paper, we propose a reconstruction method for a 3D structure using sequential omnidirectional images in an artificial environment. The proposed method is fundamentally categorized into the Structure from Motion (SfM) technique. The conventional point-based SfM using a standard camera is, however, likely to fail to recover a 3D structure in an artificial and textureless environment such as a corridor. To tackle this problem, the proposed technique uses an omnidirectional camera and line-based SfM. Line features, such as a borderline of a wall and a floor or a window frame, are easy to discern in an artificial environment comparing point features, even in a textureless scene. In addition, an omnidirectional camera can track features for a long period because of its wide field-of-view. Extracted line features in an artificial environment are often mutually parallel. Parallel lines provide valuable constraints for camera movement estimation. Directions and locations of lines are estimated simultaneously with 3D camera movements. A 3D model of the environment is constructed from measurement results of lines and edge points. Experimental results show the effectiveness of our proposed method.

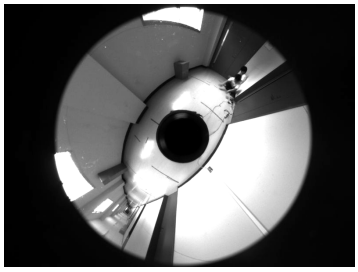
keywords: Structure from Motion, Parallel lines, Textureless scene, Omnidirectional camera.

1 INTRODUCTION

In this paper, we propose a reconstruction method of a 3D structure using sequential images acquired with a monocular omnidirectional camera in an artificial environment.

The accurate and efficient estimation of the camera movement is very important for scene reconstruction using the monocular stereo method. As a method of monocular stereo, approaches based on Structure from Motion (SfM) [1, 2] or vSLAM [3, 4] have been proposed. Camera movement estimation from an image sequence acquired using a single camera is difficult because a camera movement matrix (such as the essential matrix) has at least 6 degrees of freedom (DOFs) (3 DOFs for rotation and 3 DOFs for translation). Moreover, the estimation is a nonlinear problem. When locations and orientations of many viewpoints are estimated simultaneously using previous methods, the processing cost is high and the probability of falling into a local minimum is great.

(a) Omnidirectional camera.



(b) Acquired image.

Figure 1: An omnidirectional camera equipped with a hyperboloid mirror and the acquired image.

Figure 2: Example of a textureless scene (corridor).

An omnidirectional camera is used for a self-localization and a scene reconstruction in our proposed method. Self-localization methods using monocular stereo often necessitate feature tracking (KLT tracker [5], SIFT [6], etc.) to obtain correspondence between different viewpoints. However, features will be lost easily because of a camera swing, if a typical camera which has a narrow field of view is used. Therefore, an omnidirectional camera with a wide field of view is effective for a self-localization [7]. An omnidirectional camera in this paper has a hyperboloid mirror (Fig. 1). The omnidirectional camera can be regarded as a pinhole camera. Self-localization and scene reconstruction methods using an omnidirectional camera have been proposed [8–10].

Among previous monocular stereo methods, some approaches have used correspondences of feature points [2, 3, 8–12], straight lines [13–15], or both features [1]. Point-based methods have the benefit of a fast calculation by a linear solution (8-point algorithm [16], 5-point algorithm [2], etc.). However, the number of feature points is insufficient in textureless scenes (such as an indoor environment shown in Fig. 2). On the other hand, line-based methods are available for textureless scenes. Textureless scenes contain anthropogenic objects. Many anthropogenic objects consist of a linear shape. Therefore, line correspondences are obtainable in textureless scenes.

Our proposed method for the textureless scene reconstruction is based on SfM using line correspondences. Lines which consist of anthropogenic objects are often mutually parallel. Parallel lines can be extracted from an omnidirectional image easily and constantly because of its wide field of view. Parallel lines provide valuable constraints for camera movement estimation. Therefore, the proposed method uses constraints obtained from parallel lines. As previous methods, SLAM using parallel lines and their vanishing point [17], SfM in urban environment (building scene) [18], rotation estimation by video compass [19] and so on have been proposed. However, the computation complexity of these methods increases as the number of viewpoints or lines increases [17, 18]. In addition, some previous methods have been based on the assumption that the camera movement is only a horizontal movement [18, 19]. Our proposed method estimates 3D camera movements using correspondences of parallel lines and non-parallel lines among no fewer than three images.

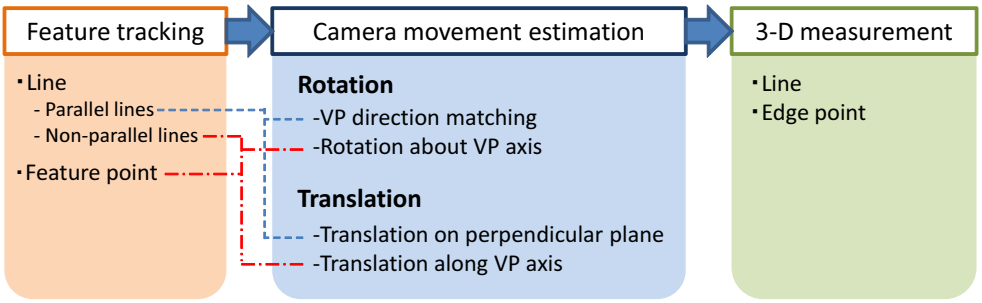


Figure 3: Procedures used for our proposed method.

Although there are linear computation algorithms for a line-based SfM using a trifocal tensor [20,21], it is only for image triplets. A bundle adjustment [22] is used in many previous methods to estimate the camera movement among more than three images. A bundle adjustment is a framework of the camera movement estimation based on reprojection error minimization. The Levenberg–Marquardt algorithm provides a solution of the error function minimization in a bundle adjustment. However, the algorithm often falls into a local minimum [13, 14]. Line-based EKF SLAM [15] is proposed. Nevertheless, it is known there is a problem of the error accumulation of the camera movement estimation in EKF SLAM.

The proposed line-based SfM can estimate 3D camera movements of more than three viewpoints simultaneously. The constraints obtained from parallel lines are useful for reduction in the degree of freedom of the camera movement estimation. Using the constraints, estimation of the rotation matrices and the translation vectors is divided into two procedures. Each procedure is solved as a 1 DOF problem without regard to the number of viewpoints and features. Consequently, the calculation cost is extremely low, and it can avoid falling into a local minimum. The proposed method can obtain the global minimum easily. The effectiveness of our proposed method is demonstrated in experimentally obtained results.

2 FRAMEWORK OF PARALLEL LINE-BASED SFM

The prerequisite in our proposed method is that at least 6 line correspondences (3 parallel lines and 3 non-parallel lines) exist between omnidirectional images. The image sequence must include at least three images. Parallel lines are extracted easily from a man-made structure in an indoor environment because an omnidirectional camera has a wide field of view. Therefore, the assumption is proper for general indoor environments.

The procedure of our proposed method is described below (Fig. 3). Lines and feature points are extracted and tracked along an omnidirectional video. At each viewpoint, parallel lines and the vanishing point (VP) are detected from these lines. A vector in the direction of the VP is calculated. The vector is called a “VP axis” in this paper. Pseudo-lines, which are created from a couple of feature points, are regarded as non-parallel lines.

The estimation of a rotation and a translation is separated in the proposed method. These separated

estimations are divided into two procedures.

In the first procedure of the rotation estimation, a rotation that makes the direction of the VP axes at all viewpoints the same in the world coordinate system is calculated. In the second procedure, a rotation about the VP axis is estimated. When a rotation between two viewpoints is determined, rotations at the other viewpoints are calculated by minimizing a quartic function about the rotation angle. The minimum value of a quartic function is calculated uniquely. Therefore, a 3D camera rotation movement can be estimated by solving a problem with 1 DOF about a rotation angle between two arbitrary viewpoints without regard to the number of viewpoints and features.

In the first procedure of the translation estimation, translations on the plane perpendicular to the 3D direction of the VP axis are estimated. When the translation direction between two viewpoints is determined, translations at the other viewpoints are calculated uniquely by solving a simultaneous equation. Therefore, the estimation is a problem with 1 DOF about the translation direction on the plane between arbitrary two viewpoints. In the second procedure, translations along the VP axis are estimated. In this procedure, when a translation along VP axis between two viewpoints is determined, translations at the other viewpoints are calculated as a translation minimizing a quartic function about the translation. Therefore, the estimation is also a problem with 1 DOF about the translation along VP axis between arbitrary two viewpoints.

The proposed method estimates 3D camera movements by solving three problems with 1 DOF. The calculation cost is extremely low and it is easy to avoid falling into local minimum. Moreover, the proposed method can obtain the global minimum solution easily.

Locations and directions of 3D lines are estimated simultaneously with camera movement. For a dense 3D reconstruction, edge points are measured using the estimated camera movement. A mesh model is constructed from the measurement results of lines and edge points. Textures obtained from omnidirectional images are added for mesh surfaces.

3 OMNIDIRECTIONAL CAMERA COORDINATE SYSTEM

The coordinate system of an omnidirectional camera is shown in Fig. 4. A hyperboloid mirror reflects a ray from the camera lens to image coordinates (u, v). In this paper, the reflected ray is called a "ray vector". The extension lines of all ray vectors intersect at the focus of the hyperboloid mirror. The ray vector $\mathbf{r}$ is calculated using the following equations.

$$\mathbf{r} = \frac{\hat{\mathbf{r}}}{\|\hat{\mathbf{r}}\|}, \tag{1}$$

$$\hat{\mathbf{r}} = \begin{bmatrix} \lambda(u - c_x)p_x \\ \lambda(v - c_y)p_y \\ \lambda f - 2\gamma \end{bmatrix}, \tag{2}$$

$$\lambda = \frac{\alpha^2 \left(f\sqrt{\alpha^2 + \beta^2} + \beta\sqrt{u^2 + v^2 + f^2} \right)}{\alpha^2 f^2 - \beta^2 (u^2 + v^2)}, \tag{3}$$

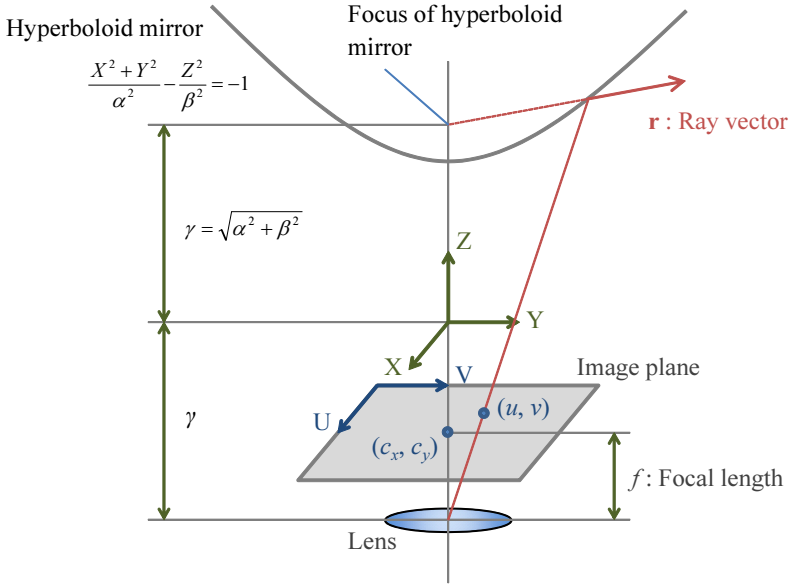


Figure 4: Coordinate system of the omnidirectional camera.

where c_x and c_y are the center coordinates of the omnidirectional image, p_x and p_y are pixel size, f is the focal length of a camera lens, and α, β, and γ are hyperboloid parameters. In the proposed method, these parameters are calibrated in advance.

4 FEATURE DETECTION AND TRACKING

4.1 FEATURE POINT TRACKING

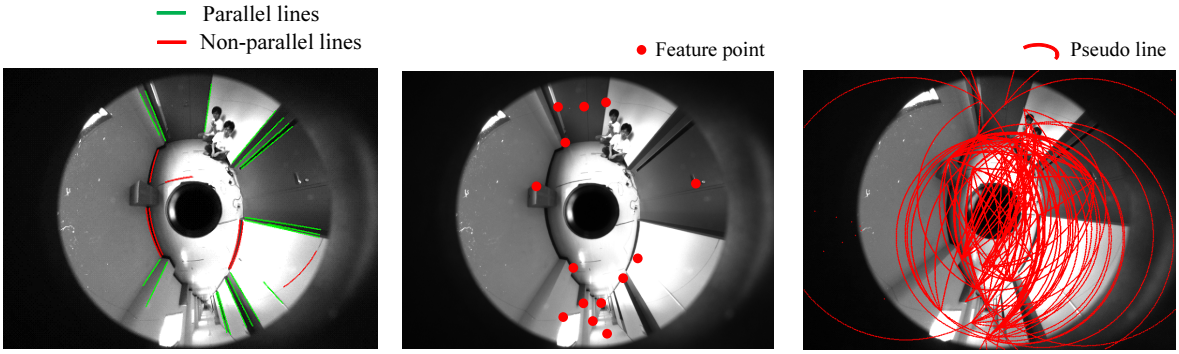
The proposed method needs at least three non-parallel lines in addition to three parallel lines. In textureless scenes, many parallel lines are perpendicular to the floor. However, there are often insufficient non-parallel lines for camera movement estimation, although an omnidirectional camera has a wide field of view (Fig. 5(a)). Therefore, the proposed method uses feature points.

Feature points are tracked along an omnidirectional video by KLT tracker [5]. Pseudo-lines are created from a couple of feature points. Pseudo-lines are regarded as non-parallel lines. Examples of feature points and pseudo-lines are shown in Figs. 5(b) and 5(c). Pseudo-lines in Fig. 5(c) are curved because a straight line is projected as a curved line in an omnidirectional image.

4.2 LINE TRACKING

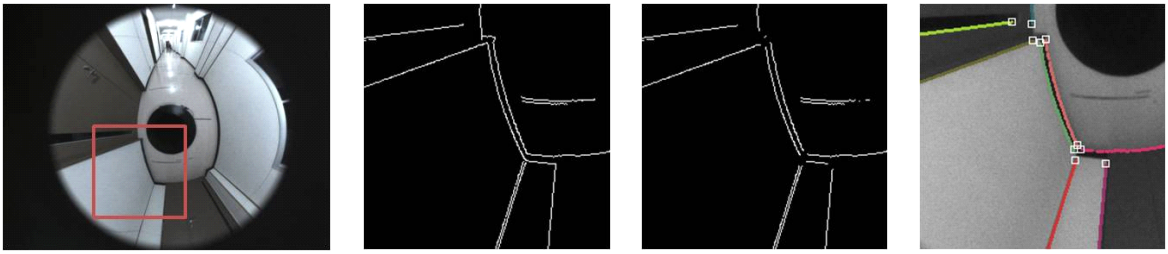
Straight lines are extracted from distorted omnidirectional images. The proposed method obtains edge points using a Canny edge detector [23]. Examples of edge point detection are shown in Figs. 6(a) and 6(b).

To separate each line, corner points are removed as shown in Fig. 6(c). Corner points are detected



(a) Non-parallel lines. (b) Feature points. (c) Pseudo-lines.

Figure 5: Creation of pseudo-lines.



(a) Input image. (b) Canny edge points. (c) Corner point removal. (d) Extracted lines.

Figure 6: Line extraction procedure. Panels (b), (c), and (d) portray enlarged views of the red rectangle in panel (a).

using two eigenvalues of the Hessian $\mathbf{H}$ of the image. The Hessian matrix is defined as

$$\mathbf{H} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}, \tag{4}$$

where the derivatives I_{xx}, I_{xy}, and I_{yy} are calculated by taking differences of neighboring edge points. If the ratio of eigenvalues is sufficiently high, then the edge point is regarded as line-like. The ratio is set to 10 using the trial-and-error method.

A least squares plane is calculated from ray vectors of segmented edge points. If the edge segment consists of a straight line, then these ray vectors are located on the same plane (Fig. 7). Therefore, an edge segment which has a small least-squares error is regarded as a straight line. The proposed method can extract straight lines, even if an edge segment resembles a curve in an omnidirectional image. If over half the edge points of the edge segment i satisfy the following equation, then the segment is determined as a straight line (Fig. 6(d)).

$$\left({\mathbf{r}_{i,j}}^{\mathrm{T}}\mathbf{n}_i\right)^2 < l_{th}. \tag{5}$$

Therein, l_{th} is a threshold. $\mathbf{r}_{i,j}$ is a ray vector to the edge point j from the mirror focus included in the

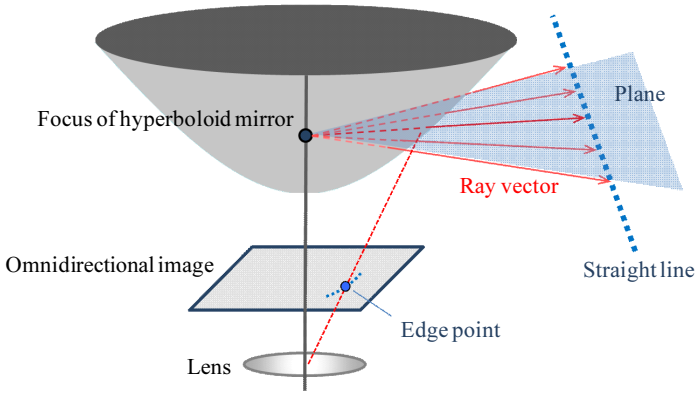


Figure 7: Relation between a straight line and ray vectors.

line i. $\mathbf{n}_i$ is the normal vector of the least-squares plane calculated from the line i. $\mathbf{n}_i$ is a unit vector. The vector is called "NV" in this paper. In the detection, edge points which do not constitute the line are rejected as noises by RANSAC [24]. The threshold l_{th} is determined from the image resolution.

Lines are tracked along the omnidirectional image sequence. The proposed method obtains sampling points located on a straight line. Sampling points are extracted at constant intervals (Fig. 8(b)). Edge segments are extracted in the next frame (Fig. 8(c)). The points extracted in Fig. 8(b) are tracked to the next frame by KLT tracker (Fig. 8(d)). The edge point closest to the tracked point is selected as a corresponding edge point (Fig. 8(e)). The edge segment with the maximum number of corresponding edge points is regarded as a corresponding edge segment (Fig. 8(f)). If an edge segment corresponds to several lines, then a line having a larger number of corresponding edge points is selected.

An aperture problem [25] exists in matching the point search on the line. However, it is not difficult for the proposed method to obtain the corresponding edge segment because it does not require point-to-point matching. By continuing the processes explained above, straight lines are tracked along the omnidirectional image sequence.

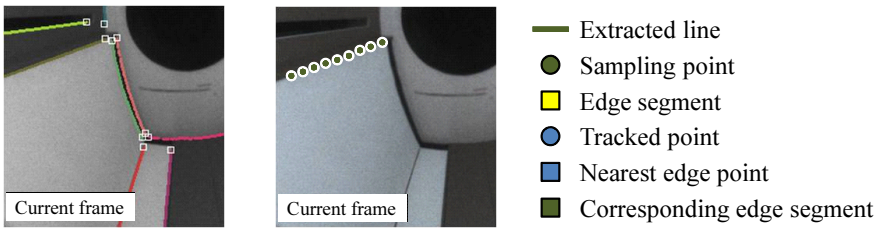
4.3 DETECTION OF PARALLEL LINES AND A VANISHING POINT

Parallel lines and their associated vanishing point are detected from tracked lines. A VP axis $\mathbf{v}^{c_j}$ and NVs $\mathbf{n}_i^{c_j}$ for parallel lines i at the viewpoint c_j satisfy the following equation.

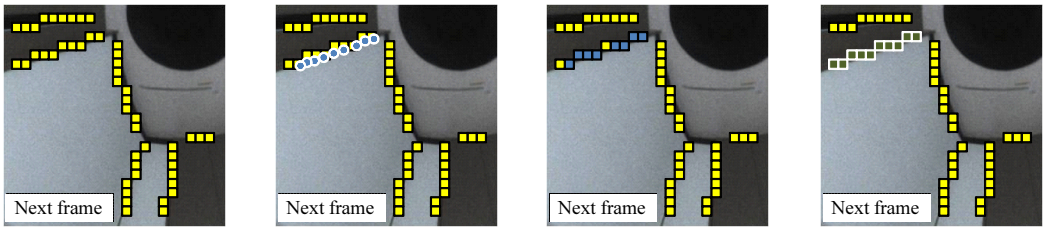
$$\mathbf{v}^{c_j\,\mathrm{T}}\mathbf{n}_i^{c_j} = 0. \tag{6}$$

Here, the superscript of the vector signifies its reference coordinate system. The reference coordinate system of the vector without a superscript is the world coordinate system. Three lines are necessary for the parallel line detection from an image. The proposed method selects three lines randomly from tracked lines. A vector $\mathbf{v}_{\mathrm{rand}}^{c_j}$ that minimizes E_v in Eq. (7) is calculated using the least squares method.

$$E_v = \sum_i^{n_l} \left(\mathbf{v}_{\mathrm{rand}}^{c_j\,\mathrm{T}} \mathbf{n}_i^{c_j} \right)^2. \tag{7}$$



(a) Extracted lines. (b) Sampling points.



(c) Edge segments. (d) Tracked points. (e) Nearest edge points. (f) Corresponding edge segment.

Figure 8: Searching for a corresponding edge segment in the subsequent frame.

In that equation, n_l is the number of lines. If selected lines satisfy the following equation in the entirety of an input image sequence, then these are regarded as parallel lines.

$$E_v < p_{th}, \tag{8}$$

where p_{th} is a threshold. Lines that are regarded as mutually parallel are integrated. A line group that has the maximum number of lines is used for the following process as parallel lines. The VP axis $\mathbf{v}^{c_j}$ is calculated from integrated parallel lines as a vector $\mathbf{v}^{c_j}_{\mathrm{rand}}$ that minimizes E_v in Eq. (7).

5 PARALLEL LINE BASED SFM

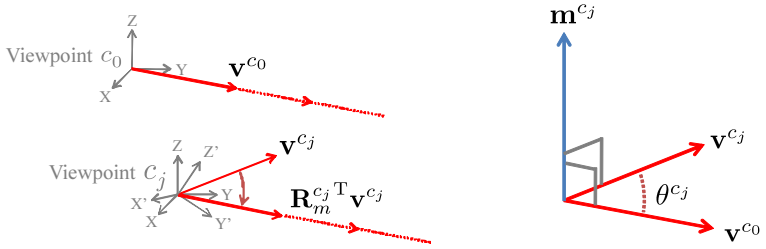
5.1 ESTIMATION OF CAMERA ROTATION AND LINE DIRECTION

The camera rotation estimation process is divided into two procedures. In the first procedure, the method calculates a camera rotation matrix that makes the direction of VP axes among all viewpoints the same. In the second procedure, a rotation about the VP axis is estimated using at least three lines. These lines must have a different 3D direction from parallel lines.

5.1.1 VP DIRECTION MATCHING

This procedure requires VP axes, which should have the same 3D direction in the world coordinate because a vanishing point lies at an infinite distance from the viewpoint, theoretically. Therefore, the proposed method calculates a rotation matrix $\mathbf{R}^{c_j}_m$ satisfying the following equation.

$$\mathbf{v}^{c_0} = {\mathbf{R}^{c_j}_m}^{\mathrm{T}} \mathbf{v}^{c_j}. \tag{9}$$



(a) VP direction matching. (b) Rotation axis and VP axes.

Figure 9: Relation among the rotation axis $\mathbf{m}^{c_j}$ and VP axes $\mathbf{v}^{c_0}$ and $\mathbf{v}^{c_j}$.

In that equation, $\mathbf{R}_m^{c_j}$ is a rotation matrix that makes the direction of the VP axis $\mathbf{v}^{c_j}$ the same as that of the VP axis $\mathbf{v}^{c_0}$ at the initial camera coordinate system. Here, the initial camera coordinate system is equal to the world coordinate system in this paper. $\mathbf{R}_m^{c_j}$ is calculated as a rotation about an axis $\mathbf{m}^{c_j}$ using the Rodrigues rotation formula. The rotation axis $\mathbf{m}^{c_j}$ and angle θ^{c_j} are calculated by the following equations. A relation between $\mathbf{m}^{c_j}$ and θ^{c_j} is shown in Fig. 9.

$$\mathbf{m}^{c_j} = \mathbf{v}^{c_0} \times \mathbf{v}^{c_j}, \tag{10}$$

$$\theta^{c_j} = \arccos({\mathbf{v}^{c_0}}^{\mathrm{T}}\mathbf{v}^{c_j}). \tag{11}$$

The 3D directions of parallel lines and the VP axis $\mathbf{v}^{c_0}$ are the same. Therefore, the vector $\mathbf{v}^{c_0}$ also represents a 3D direction of parallel lines in the following explanation.

5.1.2 ESTIMATION OF ROTATION ABOUT THE VP AXIS

In the second procedure, a rotation matrix about the VP axis $\mathbf{v}^{c_0}$ is estimated. This procedure requires at least three lines. The 3D direction of these lines must not be equal to the VP axis $\mathbf{v}^{c_0}$.

In the proposed method, the only remaining unknown parameter of 3D rotation is rotation $\mathbf{R}_v^{c_j}$ about the VP axis $\mathbf{v}^{c_0}$ because the other two parameters are obtainable from the constraints of the vanishing point. Therefore, this procedure estimates a rotation matrix $\mathbf{R}_v^{c_j}$, namely, the rotation angle ϕ^{c_j} (Fig. 10).

The camera rotation matrix $\mathbf{R}^{c_j}$ between the initial viewpoint c_0 and a j-th viewpoint c_j is defined using two rotation matrices as the following equation.

$$\mathbf{R}^{c_j} = \mathbf{R}_m^{c_j}\mathbf{R}_v^{c_j}. \tag{12}$$

The camera rotation matrix $\mathbf{R}^{c_j}$ and 3D line direction $\mathbf{d}_i$ satisfy the following equation.

$$\left({\mathbf{R}^{c_j}}^{\mathrm{T}}\mathbf{n}_i^{c_j}\right)^{\mathrm{T}}\mathbf{d}_i = 0. \tag{13}$$

In that equation, $\mathbf{n}_i^{c_j}$ is the NV of the line i at the viewpoint c_j. $\mathbf{d}_i$ is a unit vector. NVs in the world coordinate system ${\mathbf{R}^{c_j}}^{\mathrm{T}}\mathbf{n}_i^{c_j}$ is perpendicular to the 3D line i. When a rotation angle ϕ^{c_j} is given, the

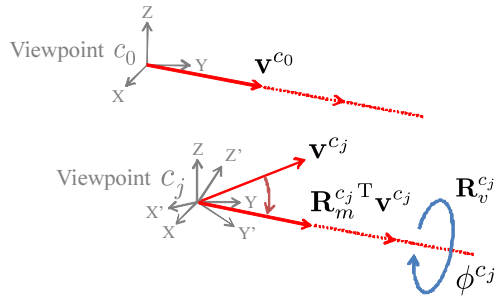


Figure 10: Rotation about the VP axis $\mathbf{v}_{c_0}$.

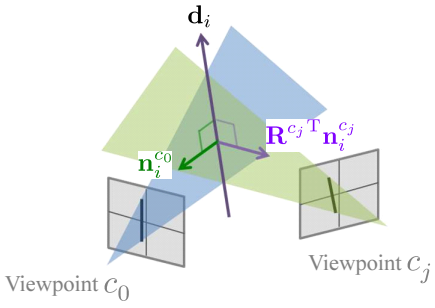


Figure 11: 3D line direction calculation.

3D direction $\mathbf{d}_i$ of the line i is calculated by cross product of NVs at viewpoint c_0 and c_j (Fig. 11).

$$\mathbf{d}_i = \mathbf{n}_i^{c_0} \times \left(\mathbf{R}^{c_j\,\mathrm{T}}\mathbf{n}_i^{c_j}\right). \tag{14}$$

Using the 3D line direction, a rotation matrix $\mathbf{R}^{c_k}$ between the initial viewpoint c_0 and the other viewpoint c_k is calculated by solving the following equation.

$$e_{\mathrm{rot}}(\phi^{c_k}) = \sum_i^{n_l} \left| \left(\mathbf{R}^{c_k\,\mathrm{T}}\mathbf{n}_i^{c_k}\right)^{\mathrm{T}} \mathbf{d}_i \right|^2 \rightarrow \min, \tag{15}$$

where n_l is the number of non-parallel lines. Although the function is nonlinear, it is solvable easily because $e_{rot}(\phi^{c_k})$ is just a quartic function about ϕ^{c_k}. Consequently, if a rotation angle ϕ^{c_j} at a viewpoint c_j is given, then rotation angles ϕ^{c_k} at the other viewpoints c_k are determined. The proposed optimization of rotations is represented as the following equation.

$$E_{\mathrm{rot}}(\phi^{c_j}) = \sum_k^{n_c} e_{\mathrm{rot}}(\phi^{c_k}) \rightarrow \min, \tag{16}$$

where n_c represents the number of viewpoints. In the proposed method, the rotation estimation is a search problem with 1 DOF about the rotation angle ϕ^{c_j} without regard to the number of viewpoints and features.

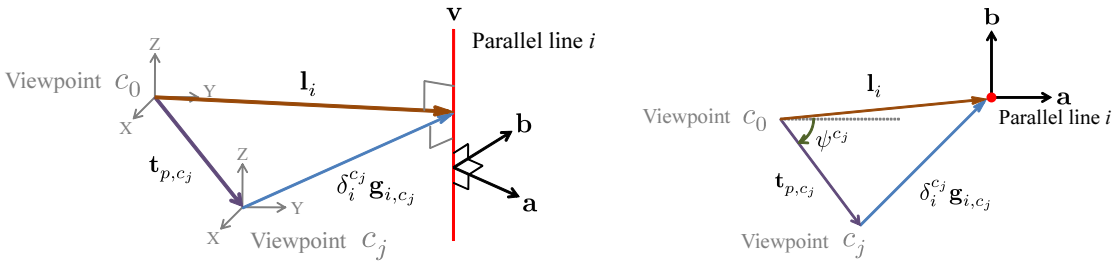
In the following explanation, $\mathbf{v}$ expresses the VP axis $\mathbf{v}^{c_0}$ or 3D direction of parallel lines.

5.2 ESTIMATION OF CAMERA TRANSLATION AND LINE LOCATION

Camera translations are estimated using two procedures. In the first, translations on the plane perpendicular to parallel lines are estimated. In the second, translations directed along parallel lines are estimated. As is true also for rotation estimation, these two procedures of the translation estimation are solvable as problems with 1 DOF.

5.2.1 TRANSLATION ON A PERPENDICULAR PLANE

In the first procedure, translations on a plane perpendicular to parallel lines are estimated. This procedure requires at least three parallel lines. Translations on the plane and 3D locations of parallel lines are optimized simultaneously.



(a) Bird's-eye view.

(b) Top view.

Figure 12: Relation among 3D direction of parallel lines, basis vector, 3D location of the parallel line, a vector $\mathbf{g}_{i,c_j}$ and the translation on the plane.

Here, the method introduces basis vectors $\mathbf{a}$ and $\mathbf{b}$ ($\mathbf{a} \perp \mathbf{b}$, $\mathbf{a} \perp \mathbf{v}$, and $\mathbf{b} \perp \mathbf{v}$). A unit vector $\mathbf{g}_{i,c_j}$ from the viewpoint c_j to the parallel line i is calculated using Eq. (17). The vector $\mathbf{g}_{i,c_j}$ is perpendicular to parallel lines as

$$\mathbf{g}_{i,c_j} = \mathbf{v} \times (\mathbf{R}^{c_j\,\mathrm{T}} \mathbf{n}_i^{c_j}). \tag{17}$$

Using these vectors, $\mathbf{a}$ and $\mathbf{b}$ elements of the true translations can be estimated. A translation vector $\mathbf{t}_{p,c_j}$ between the initial viewpoint c_0 and a viewpoint c_j, the location $\mathbf{l}_i$ of parallel lines i and a vector $\mathbf{g}_{i,c_j}$ satisfy the following equation. The relation among these vectors is shown in Fig. 12.

$$\delta_i^{c_j} \mathbf{g}_{i,c_j} + \mathbf{t}_{p,c_j} - \mathbf{l}_i = 0. \tag{18}$$

In that expression, $\delta_i^{c_j}$ is a fixed number representing the depth of the line at the viewpoint. $\delta_i^{c_j}$ is calculated as

$$\delta_i^{c_j} = \frac{\left(\mathbf{t}_{p,c_j} - \mathbf{l}_i\right)^{\mathrm{T}} \mathbf{g}_{i,c_j}}{\mathbf{g}_{i,c_j}{}^{\mathrm{T}} \mathbf{g}_{i,c_j}}. \tag{19}$$

Here, the translation $\mathbf{t}_{p,c_j}$ is expressed as Eq. (20),

$$\mathbf{t}_{p,c_j} = \mathbf{a} \cos\psi^{c_j} + \mathbf{b} \sin\psi^{c_j}, \tag{20}$$

where ψ^{c_j} denotes the translation direction from the initial viewpoint c_0 to the viewpoint c_j. The absolute scale is unknown in the SfM approach. Consequently, the distance between these two viewpoints is set to 1 in the proposed method. When the direction ψ^{c_j} is given, the locations of parallel lines are calculated using the following equation.

$$\mathbf{l}_i = \frac{\zeta_i^{c_0} \mathbf{g}_{i,c_0} + \zeta_i^{c_j} \mathbf{g}_{i,c_j} + \mathbf{t}_{p,c_j}}{2}, \tag{21}$$

where $\zeta_i^{c_0}$ and $\zeta_i^{c_j}$ are fixed factors representing the depth of the line from each viewpoint. Fixed factors $\zeta_i^{c_0}$ and $\zeta_i^{c_j}$ are calculated as factors satisfying the following expression.

$$\left\| \zeta_i^{c_0} \mathbf{g}_{i,c_0} - \zeta_i^{c_j} \mathbf{g}_{i,c_j} - \mathbf{t}_{p,c_j} \right\|^2 \to \min. \tag{22}$$

Using 3D locations of parallel lines, translations at the other viewpoint c_k are calculated by minimizing the following equations.

$$e_{t_1}\left(\lambda_a^{c_k}, \lambda_b^{c_k}\right) = \sum_i^{n_l} \left\| \delta_i^{c_k} \mathbf{g}_{i,c_k} + \mathbf{t}_{p,c_k} - \mathbf{l}_i \right\|^2, \tag{23}$$

$$\mathbf{t}_{p,c_k} = \lambda_a^{c_k} \mathbf{a} + \lambda_b^{c_k} \mathbf{b}. \tag{24}$$

When the line location $\mathbf{l}_i$, namely, the translation direction ψ^{c_j} is given, $E_{t_1}\left(\lambda_a^{c_k}, \lambda_b^{c_k}\right)$ is solvable as a simultaneous equation about $\lambda_a^{c_k}$ and $\lambda_b^{c_k}$. Therefore, translations on the plane vertical to parallel lines are optimized by minimizing the following function about ψ^{c_j}.

$$E_{t_1}\left(\psi^{c_j}\right) = \sum_{k=1}^{n_c} e_{t_1}\left(\lambda_a^{c_k}, \lambda_b^{c_k}\right). \tag{25}$$

Consequently, Eq. (25) is solvable as a search problem with 1 DOF about the translation direction ψ^{c_j} on the plane perpendicular to parallel lines.

5.2.2 TRANSLATION ALONG THE VP AXIS

In the second procedure of the translation estimation, translations along the parallel line direction are estimated. This procedure requires at least three non-parallel lines. The camera translation vector $\mathbf{t}_{c_j}$ is represented as

$$\mathbf{t}_{c_j} = \mathbf{t}_{p,c_j} + \omega^{c_j} \mathbf{v}, \tag{26}$$

where ω^{c_j} represents the distance of translation along parallel line direction. Translations and non-parallel line locations are optimized by minimizing the sum of reprojection errors in Eqs. (27)–(29).

$$e_{t_2} = \sum_i^{n_l} \left(1 - \mathbf{q}_{i,c_j}{}^{\mathrm{T}} \mathbf{g}'_{i,c_j}\right)^2, \tag{27}$$

$$\mathbf{q}_{i,c_j} = \frac{\mathbf{l}'_i - \mathbf{t}_{c_j} + \tau_i^{c_j} \mathbf{d}_i}{\left\| \mathbf{l}'_i - \mathbf{t}_{c_j} + \tau_i^{c_j} \mathbf{d}_i \right\|}, \tag{28}$$

$$\tau_i^{c_j} = \frac{\left(\mathbf{t}_{c_j} - \mathbf{l}'_i\right)^{\mathrm{T}} \mathbf{d}_i}{\mathbf{d}_i{}^{\mathrm{T}} \mathbf{d}_i}. \tag{29}$$

In those expressions, $\mathbf{q}_{i,c_j}$ is a unit vector crossed at a right angle to the non-parallel line i from the viewpoint c_j. $\mathbf{g}'_{i,c_j}$ is a unit vector from the viewpoint c_j to the non-parallel line i. $\mathbf{g}'_{i,c_j}$ is calculated using the same procedures as those described in Eq. (17). The relation between these vectors is shown in Fig. 13. If no errors exist, then these two vectors $\mathbf{q}_{i,c_j}$ and $\mathbf{g}'_{i,c_j}$ will be the same. However, in fact, these have different direction because of various errors. The angle error is almost identical to a reprojection error.

When ω^{c_j}, namely, translation from the initial viewpoint c_0 to the viewpoint c_j is determined, 3D line locations $\mathbf{l}'_i$ are calculated in the same way as Eq. (18). Using the locations, Eq. (27) at the other viewpoint c_k is solvable as a quartic function about ω^{c_k}. Therefore, the translation estimation is a search problem with 1 DOF about ω^{c_k}. The optimum translations are estimated by minimizing the following

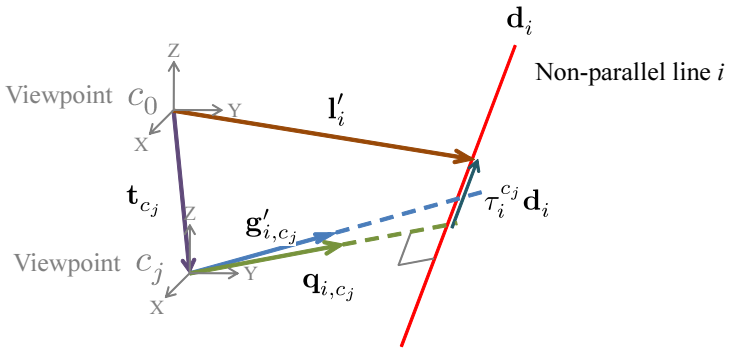


Figure 13: Reprojection error of a straight line.

equation.

$$E_{t_2}(\omega^{c_j}) = \sum_{k=1}^{n_c} e_{t_2}(\omega^{c_k}). \tag{30}$$

The proposed method can measure straight lines using the processes described above. Moreover, edge points are reconstructed to measure environments densely. The edge point reconstruction is based on [26].

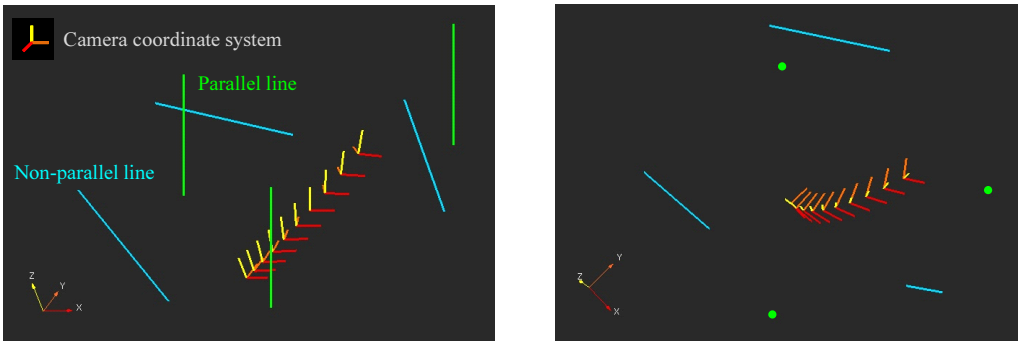
6 EXPERIMENTS

We demonstrate the proposed camera movement estimation using simulation data. All experiments are done with off-line processing. The CPU is an Intel Core i7 975 (3.33 GHz). In this experiment, the values of NVs $\mathbf{n}_{i,c_j}$ and VP axes $\mathbf{v}^{c_j}$ are given. It is known which lines are parallel. The camera movement includes a 3D rotation and translation.

We first verified that the proposed method can estimate the camera movements from 6 lines (3 parallel lines and 3 non-parallel lines). The number of viewpoints is 10. The true values of NVs are given in this experiment. The position relation between the viewpoints and lines is shown in Fig. 14. The red, yellow, and orange axes show the camera coordinate system at each viewpoint. Parallel lines are shown as green. Other lines are represented as blue. The estimation errors of camera movement and line measurement were within the rounding error.

A local minimum naturally exists in the proposed method. However, around the ground truth, Fig. 15 shows that the evaluation values of Eqs. (16), (25), and (30) are sufficiently low compared to other values. Figure 15 shows an example of the evaluation value in the rotation estimation. For that reason, local minimum avoidance is easy for the proposed method. An example of a computation time on the rotation and translation estimation by the proposed method is shown in Fig. 16. The figure shows that the computation time is proportional to the number of viewpoints.

We verified the robustness of the proposed method with noisy data. In this experiment, noisy NVs are given. The noise implies an angle error between a given vector and the true one. The noise follows a normal Gaussian distribution. We evaluated estimation errors of the camera rotations and translations



(a) Side view. (b) Top view.

Figure 14: 3D camera movement and line position in simulation.

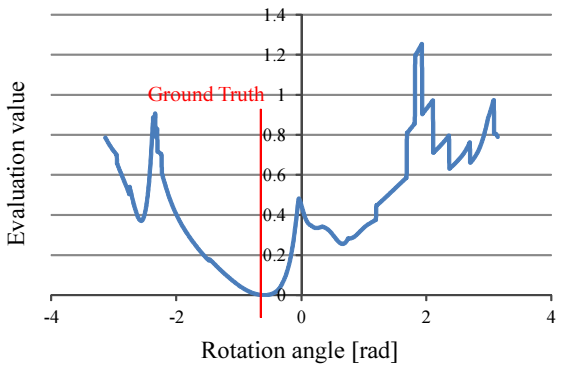


Figure 15: Evaluation value example in rotation estimation.

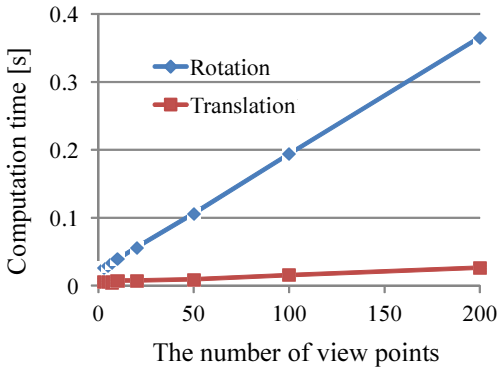


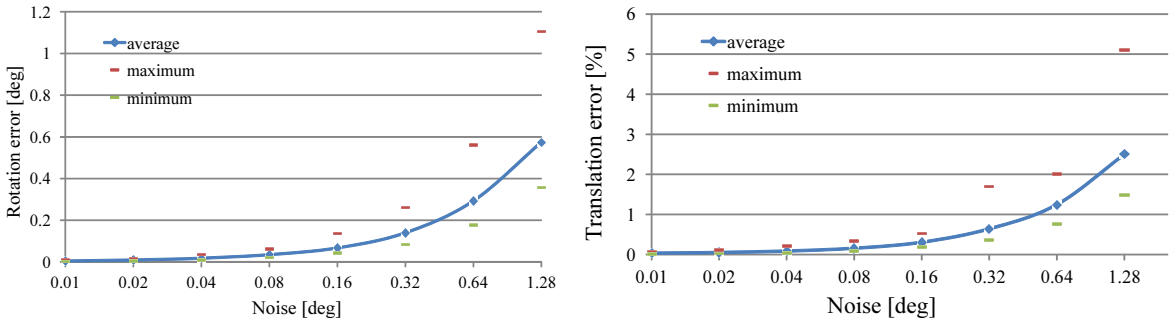
Figure 16: Computation time with the number of viewpoints.

using noisy data including 0.01 to 1.28 deg angle errors on average. Input data are 40 lines including 20 parallel lines acquired at 20 viewpoints. 100 trial runs were conducted at each noise level.

The estimation results are presented in Fig. 17. The rotation error in Fig. 17(a) represents angle errors between the axis of estimated camera coordinate system and the true one. The translation error in Fig. 17(b) represents the percentage of the distance error between the estimated camera location and the ground truth to translation distance. These values are the average of 100 times trial runs.

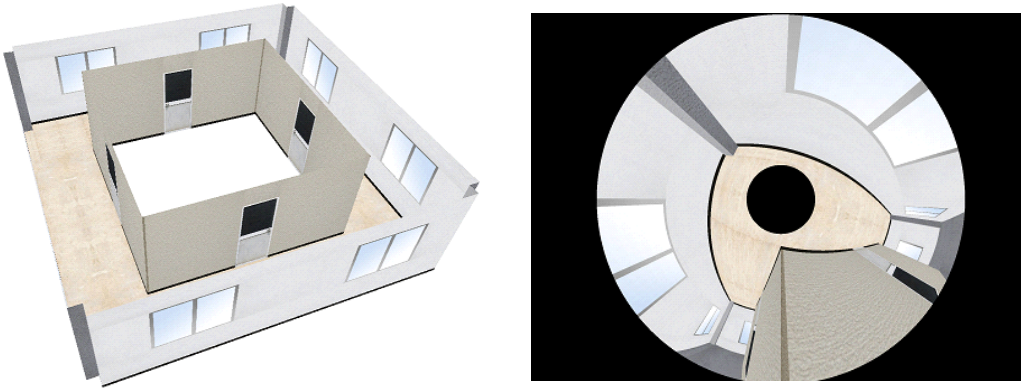
Rotation estimation error was within the given noise. The translation estimation error was within 1% against translation distance when the given noise is within 0.16 deg. According to our camera calibration, the proposed method can extract lines from an omnidirectional image (the image size is 800×600 pixels) within 0.05 deg errors. Therefore, these results show that the proposed method performs well in noisy data.

We compared the accuracy of the camera movement estimation using the proposed method with that of a common SfM using a bundle adjustment method. A bundle adjustment method is well known as an optimization of the camera movement. In this experiment, a common SfM optimizes the camera movement using feature points and lines that are the same as those of the proposed method. This



(a) Rotation. (b) Translation.

Figure 17: Estimation errors with noisy data.

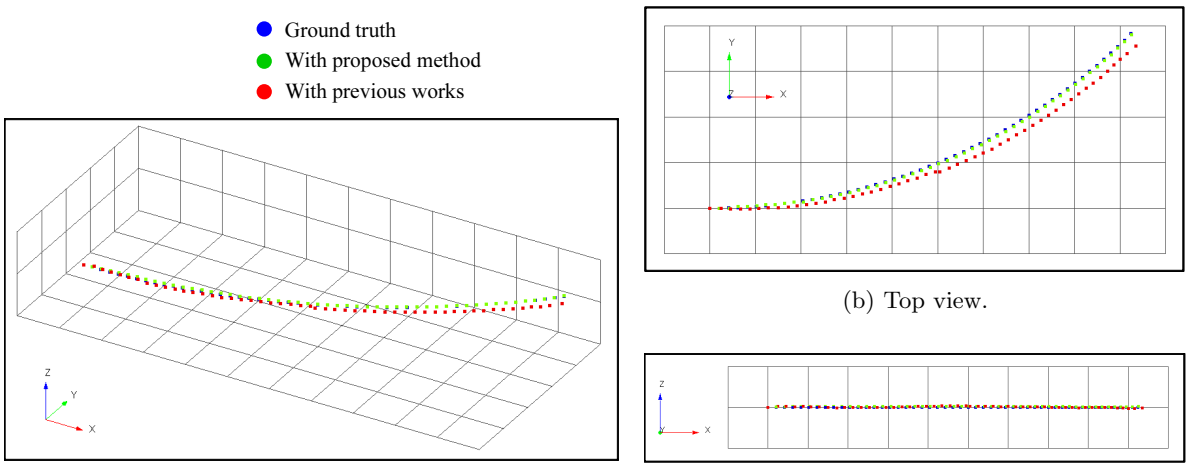


(a) Virtual environment. (b) Simulated image.

Figure 18: Virtual environment for evaluation of camera movement estimation.

experiment is demonstrated in a virtual environment as shown in Fig. 18(a) to obtain ground truth of the camera movement. Simulated omnidirectional images depending on the camera movement are created from the structure and color information of the environment (Fig. 18(b)). The image size is 800×600 pixels. The created 51 images are used as the input image sequence. In this experiment, the given information is these images only. Feature points and lines are detected automatically from the images. NVs and VP axis are calculated from the detected lines. Not only feature points but also NVs are used for the optimization by a bundle adjustment. The initial value for a bundle adjustment is obtained using a framework with eight-point algorithm and RANSAC [24].

The results of camera movement estimation are shown in Fig. 19 and Table 1. We compare the recovered camera movements with the ground truth by aligning their movement distances and initial poses. In this experiment, the ground truth of the initial camera orientation and location are known because the experiment was done in a virtual environment. The estimated camera movement by the proposed method (green marks in Fig. 19) is close to the ground truth (blue marks in Fig. 19). The reprojection error of the proposed method is within 0.5 pixels. Therefore, the result shows that the proposed method can obtain the global minimum. Although the same features are used for the experiments, camera



(a) Bird's-eye view.

(b) Top view.

(c) Side view.

Figure 19: Camera movement estimation results obtained using simulated images.

Table 1: Comparison of camera movement estimation results.

(a) Rotation errors.

	Proposed SfM	Common SfM
Average [deg]	0.119	0.177
Maximum [deg]	0.298	0.350

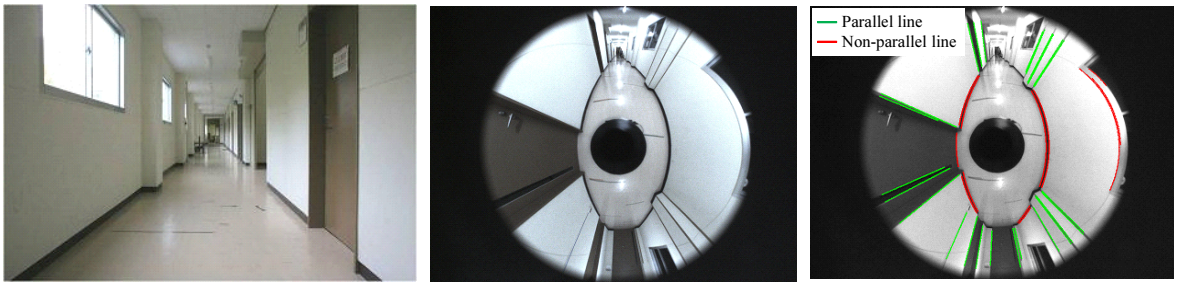
(b) Translation errors.

	Proposed SfM	Common SfM
Average [%]	0.357	1.74
Maximum [%]	0.425	3.23

movement estimation error of a common SfM is larger than the proposed method (Table 1). A bundle adjustment can obtain the optimal solution on the ideal situations. However, nonlinear optimizations such as a bundle adjustment, often fall into a local minimum in the course of experimentally obtained results.

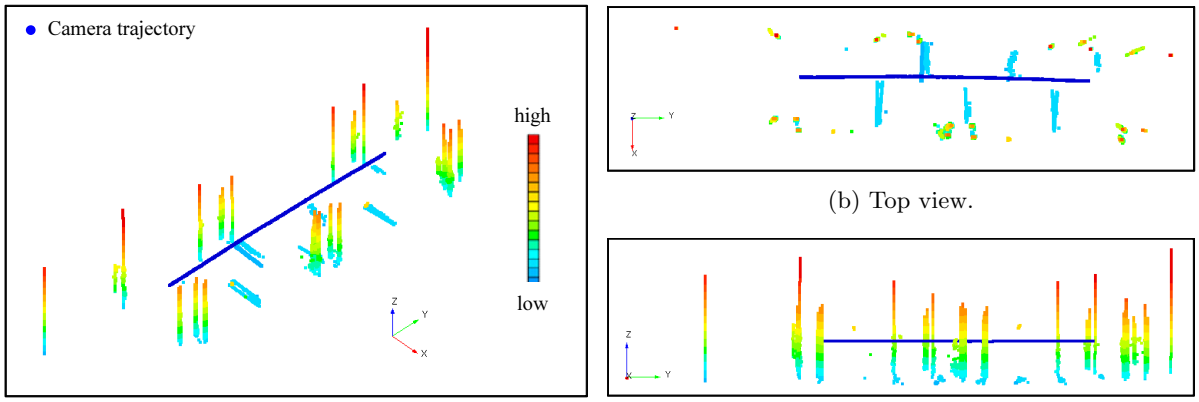
Figure 20(a) shows our verification of the proposed method using real images in a textureless scene. 800 images were acquired using a mobile robot equipped with an omnidirectional camera. The movement distance is about 12 m. The image size is 800 $\times$ 600. An input image and a parallel detection result are, respectively, shown in Figs. 20(b) and 20(c). The computing time of the line tracking was about 100 ms per frame, KLT tracker was 10 ms per frame, and the parallel lines detection was 35 ms per frame.

The result of the camera movement estimation and line measurement is shown in Fig. 21. The computation time is about 5 min. An average of the reprojection errors of lines is within 0.5 pixels. These results show that the proposed method can estimate the camera movement and measure lines precisely. The modeling result of the textureless scene is shown in Fig. 22. The modeling method is based on [27]. The corridor shape can be reconstructed using the proposed method. However, although parallel lines are detected, it is difficult to construct correct shapes of border of the wall and the floor. Correct patches are removed or false patches remain attributable to dead angle of an omnidirectional



(a) Environment. (b) Input image. (c) Detected lines.

Figure 20: Detection result of parallel lines from omnidirectional images of a textureless scene.



(a) Bird's-eye view.

(b) Top view.

(c) Side view.

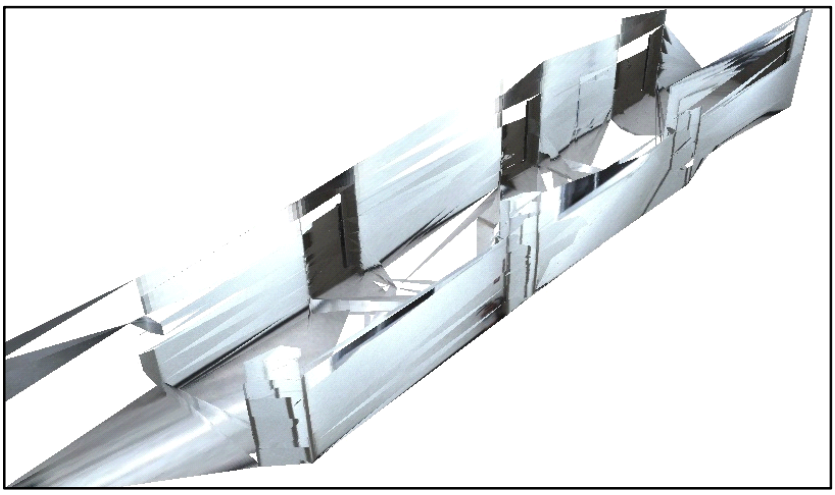
Figure 21: Results of camera movement estimation and line measurement in a textureless scene.

camera, and the difference during tracked frames of lines. Improvement of the problem is an important task for our future work. These experimentally obtained results show that the proposed method is effective for the reconstruction of a textureless scene.

7 CONCLUSION

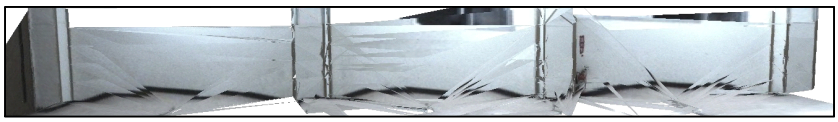
In this paper, we proposed a parallel line-based SfM for textureless scenes. Camera rotations and translations are estimated as a reasonable problem with 1 DOF using the constraints from parallel lines. Therefore, the global optimal solution is obtainable easily. Experiments underscore the effectiveness of the proposed method.

As future works, the robustness of the line detection should be improved. Illumination change makes the line detection unstable. Moreover, we should generate a framework of a parallel line-based SfM for a long image sequence. The proposed method requires lines corresponding along all images in an input image sequence.



(a) Bird's-eye view.

(b) Left side view.



(c) Right side view.

Figure 22: Modeling result of textureless scene.

ACKNOWLEDGMENTS

This work was in part supported by a MEXT KAKENHI Grant-in-Aid for Young Scientists (A), 22680017, and the Asahi-Glass Foundation.

REFERENCES

[1] D. Nister, Reconstruction From Uncalibrated Sequences with a Hierarchy of Trifocal Tensors, *Proceedings of the 6th European Conference on Computer Vision (ECCV2000)*, **1**, 649–663 (2000).

[2] D. Nister, An Efficient Solution to the Five-Point Relative Pose Problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(6), 756–770 (2004).

[3] A. J. Davison, Real-Time Simultaneous Localisation and Mapping with a Single Camera, *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV2003)*, **2**, 1403–1410 (2003).

[4] M. Tomono, 3D Object Mapping by Integrating Stereo SLAM and Object Segmentation Using Edge Points, *Advances in Visual Computing, Lecture Notes in Computer Science*, **5875**, 690–699 (2009).

[5] J. Shi and C. Tomasi, Good Features to Track, *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR1994)*, 593–600 (1994).

[6] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, **60**(2), 91–110 (2004).

[7] J. Gluckman and S. K. Nayar, Ego-motion and Omni-directional Cameras, *Proceedings of the 6th International Conference on Computer Vision (ICCV1998)*, 999–1005 (1998).

[8] R. Bunschoten and B. Krose, Robust Scene Reconstruction from an Omnidirectional Vision System, *IEEE Transactions on Robotics and Automation*, **19**(2), 351–357 (2003).

[9] C. Geyer and K. Daniilidis, Omnidirectional Video, *The Visual Computer*, **19**(6), 405–416 (2003).

[10] A. Murillo, J. Guerrero and C. Sagues, SURF Features for Efficient Robot Localization with Omnidirectional Images, *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA2007)*, 3901–3907 (2007).

[11] J. Tardif, Y. Pavlidis and K. Daniilidis, Monocular Visual Odometry in Urban Environment Using an Omnidirectional Camera, *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2008)*, 2531–2538 (2008).

[12] A. Torii, M. Havlena and T. Pajdla, From Google Street View to 3D City Models, *Proceedings of the 9th Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS2009)*, 2188–2195 (2009).

[13] A. Bartoli and P. Sturm, Multi-View Structure and Motion from Line Correspondences, *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV2003)*, 207–212 (2003).

[14] A. Bartoli and P. Sturm, Structure-from-motion using Lines: Representation, Triangulation, and Bundle Adjustment, *Computer Vision and Image Understanding*, **100**(3), 416–441 (2005).

[15] P. Smith, I. Reid and A. Davison, Real Time Monocular SLAM with Straight Lines, *Proceedings of the 2006 British Machine Vision Conference*, 17–26 (2006).

[16] R. Hartley, In Defense of the Eight-point Algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(6), 580–593 (1997).

[17] M. Bosse, R. Rikoski, J. Leonard and S. Teller, Vanishing Points and 3D Lines from Omnidirectional Video, *Proceedings of the 2002 International Conference on Image Processing (ICIP2002)*, **3**, 513–516 (2002).

[18] G. Schindler, P. Krishnamurthy and F. Dellaert, Line-Based Structure from Motion for Urban Environments, *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission*, 846–853 (2006).

[19] G. L. Mariottini and D. Prattichizzo, Uncalibrated Video Compass for Mobile Robots from Paracatadioptric Line Images, *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2007)*, 226–231 (2007).

[20] P. H. S. Torr and A. Zisserman, Robust Parameterization and Computation of the Trifocal Tensor, *Image and Vision Computing*, **15**(8), 591–605 (1997).

[21] R. Hartley, Lines and Points in Three Views and the Trifocal Tensor, *International Journal of Computer Vision*, **22**(2), 125–140 (1997).

[22] B. Triggs, P. McLauchlan, R. Hartley and A. Fitzgibbon, Bundle Adjustment -A Modern Synthesis, *Vision Algorithms: Theory and Practice, Lecture Notes in Computer Science*, **1883**, 298–372 (2000).

[23] J. F. Canny, A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-8**(6), 679–698 (1986).

[24] 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*, **24**(6), 381–395 (1981).

[25] K. Nakayama and G. Silverman, The Aperture Problem – II. Spatial Integration of Velocity Information Along Contours, *Vision Research*, **28**(6), 747–753 (1988).

[26] M. Tomono, Dense Object Modeling for 3-D Map Building Using Segment-based Surface Interpolation, *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA2006)*, 2609–2614 (2006).

[27] R. Kawanishi, A. Yamashita and T. Kaneko, Three-Dimensional Environment Model Construction from an Omnidirectional Image Sequence, *Journal of Robotics and Mechatronics*, **21**(5), 574–582 (2009).