This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

StructSLAM : Visual SLAM with Building Structure Lines

Huizhong Zhou, Danping Zou*, Ling Pei, Rendong Ying, Peilin Liu, *Member, IEEE,* Wenxian Yu

***Abstract*—We propose a novel 6 DoF visual SLAM method based on the structural regularity of man-made building environments. The idea is that we use the building structure lines as features for localization and mapping. Unlike other line features, the building structure lines encode the global orientation information that constrains the heading of the camera over time, eliminating the accumulated orientation errors and reducing the position drift in consequence. We extend the standard EKF visual SLAM method to adopt the building structure lines with a novel parametrization method that represents the structure lines in dominant directions. Experiments have been conducted in both synthetic and real-world scenes. The results show that, our method performs remarkably better than the existing methods in terms of position error and orientation error. In the test of indoor scenes of the public RAWSEEDS datasets, with the aid of wheel odometer, our method produces bounded position errors about 0.79 meter along a 967 meter path even though no loop closing algorithm is applied.**

***Keywords*—Visual SLAM, Indoor Scenes, Manhattan-World Assumption, Line Features**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a critical issue of autonomous vehicle navigation that has been studied over many years in both computer vision and robotics communities. Visual SLAM, which uses cameras as the sensor inputs, is favorable to be applied in the platforms where the requirement of cost, energy, and weight of the system is limited, e.g. the micro aerial vehicle (MAV) systems. Lots of visual SLAM methods have been proposed in the last decade. Many of them use points as features to estimate both the camera pose and the map of the surroundings represented by sparse point clouds. The state-of-the-art visual SLAM algorithms [1][2] can produce results that rival laser range scanner accuracy when the scene contains stable feature points. However, if there are few feature points in the scene, visual SLAM algorithms usually produce a large drift error both in position and orientation or even fail to work. It usually happens in scenes that consist of mainly texture-less surfaces. Some also use line features for visual SLAM [3][4][5][6]. The line features are good complements when there are not enough point features in the scene. However, the line-based visual SLAM methods exhibit essentially no significant improvement on performance and sometimes yield worse results due to the difficulty in tracking line robustly. No matter using point or line features, visual SLAM methods have the well-known drift problem, which means that the localization errors are accumulated over time. Without some special treatments, e.g. loop closing algorithms [7], or some aiding sensors, it is difficult to reach desirable accuracy in a large scale environment.

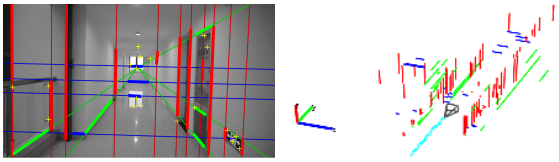


Fig. 1. **Left:** Building structure lines used for Visual SLAM. **Right:** Estimated 3D structures and camera trajectories.

Man-made buildings are everywhere in our living environments, exhibiting strong structural regularity. In most cases, the man-made buildings can be abstracted as blocks that are stacked together with three dominant directions, which is known as Manhattan-world assumption [8]. Researchers have used the observation as a prior knowledge for applications such as indoor modeling [9][10], scene understanding [11] and heading estimation [12]. Inspired by this strong regularity of man-made buildings, we propose a novel visual SLAM method, which adopts building structure lines for localization and mapping. By building structure lines, we mean the lines aligned with three dominant directions of the buildings as shown in Fig. 1.

The major difference between the structure line features and the line features used in existing visual SLAM systems [6] is that, the structure lines encode the global orientation information. The benefit of adopting the building structure lines as the landmarks is obvious: the accumulated orientation error can be eliminated. In consequence, it will reduce the accumulated position error and improve the accuracy of the SLAM system. As we can see in the experiments, the proposed method outperforms the state-of-the-art visual SLAM methods in both accuracy and robustness. In the test of using the RAWSEEDS datasets, by using odometer as the motion predictor, our method produces bounded position errors less than 1 meter along a 967 meter path without applying any loop closing algorithms. We also show that the robustness of our method under challenging cases where the existing visual SLAM methods fail. We highlight the main contributions of our work in the following:

We employ building structure lines of three dominant directions for 6DoF SLAM through a novel parametrization method, which constrains the camera orientation globally and reduces the overall drift error consequently, making the 6DoF visual SLAM in the environments with man-made buildings

*Corresponding author: Danping Zou, Email : dpzou@sjtu.edu.cn
All the authors are with Shanghai Key Laboratory of Navigation and Location Based Services, Shanghai Jiao Tong University.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

much more stable and accurate.

The maps constructed by the 3D structure lines provide extra information of the scene, which could be useful for scene modeling and understanding. The proposed visual SLAM method is of potential use in applications of indoor vehicular technology, such as indoor car parking, and autonomous service robots.

II. RELATED WORK

Most visual SLAM methods use feature points as the input to estimate the ego-motion of the sensor body and the map of the surroundings. They can be classified into two categories: structure-from-motion (SFM) method [13][2] and filter-based method [14]. In SFM methods, the ego-motion and map are estimated in a deterministic way, usually through optimizations operated locally and globally. The representative work of this kind is the PTAM system [2]. PTAM system can handle hundreds of feature points in real time by putting the time consuming bundle adjustment into a separated thread. The filter-based methods [14][15], however, work in a probabilistic way, where the accuracy of the estimates is maintained by covariances. Although filter-based methods are less efficient than SFM methods, they are more favorable to be used in fusing with other sensors since the covariances are available.

Some methods use line features for localization and mapping [3][4] [5][6]. In their work, line features are represented by segments with two end points. Other methods using line features adopt different settings rather than monocular pinhole camera. The authors in [16] proposed a SLAM method using vertical line features in indoor environments captured by an omni-directional camera. In [17], authors used the stereo camera to capture line features. Line feature, as a higher level structure representation than a point feature, can be used to improve the quality of the output maps. In [10][18], authors attempted to automatically discover 3D lines from the output of point-based SLAM algorithms, and then used them to detect planar structures. The final model was represented by planar structures which are helpful for scene modeling and understanding. In [19], the authors gave an excellent review of different parametrization methods for both point and line features.

The disadvantage that line features are more difficult to be tracked than point features makes line-based SLAM methods less popular than point-based ones, as lines are often partially occluded in the image and their end points are difficult to be located. In [20], a novel method was proposed to improve EKF SLAM by using vanishing points as global features in indoor environments. Their experimental results show that with the help of vanishing points, the SLAM performance is significantly improved. In [21], authors adopted vertical lines and floor lines as features for solving 3DoF SLAM problem. To reduce the accumulated heading error, the authors employed vanishing points as an extra measurements and also used them for loop closing. Their work shows impressively if the structural regularity is properly used, the accuracy of SLAM can be significantly improved.

Inspired by the previous work, we seek to use the structural regularity to solve the 6DoF SLAM problem with a single pinhole camera in environments containing man-made buildings. We propose to use the building structure lines that lie in three dominant orientations under the Manhattan-World assumption for localization and mapping. We will show that, the extension of EKF visual SLAM method that incorporates building structure lines can achieve remarkable robustness and accuracy in large-scale indoor environments, and produce maps consisting of 3D building line structures that could be useful for 3D CAD map generation and scene understanding.

III. BUILDING STRUCTURE LINES

Most man-made buildings exhibit Manhattan-World property, namely surfaces are aligned with three dominant directions. In other words, the intersecting lines of those surfaces are also aligned with three dominant directions. We say that the lines that can be aligned with those dominant directions as *structure lines* as shown in Fig. 1. Each structure line is associated with a dominant direction that can be represented by a vanishing point in the image.

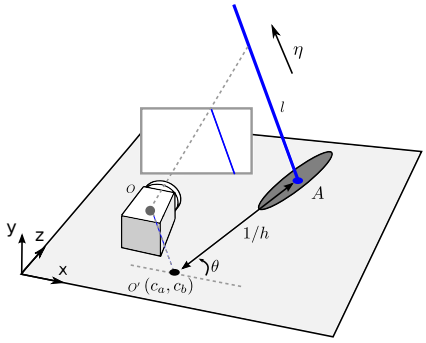


Fig. 2. A structure line l with the dominant direction η is represented in the parameter plane (XZ plane in the world frame) by a point A together with its dominant direction η, where A is the intersection of the structure line and the parameter plane. O' is the camera center projected onto the parameter plane along the dominant direction. The shadow around A indicates the covariance of the structure line on the parameter plane.

To parameterize the structure lines, we select three orthogonal planes through the origin of the world frame, namely XY, YZ and ZX planes. A structure line can be represented by a point on a parameter plane and the related dominant direction. The plane for parameterizing is selected by comparing the angular distances between the related dominant direction and the normals of the planes. The one with the nearest angular distance is chosen as the parameter plane.

The structure line is then represented by a point in the parameter plane with inverse depth representation similar to that in [15], namely,

$$\mathbf{l} = (\ c_a, c_b, \theta, h\ )^T \tag{1}$$

Here, $[c_a\, c_b]^T$ is the camera center (O' in Fig. 2) projected onto the parameter plane along the dominant direction (η in Fig. 2). θ is an angle describing the ray through the projection of camera center and the intersection point (A in Fig. 2) of structure line and the parameter plane. h is the inverse depth

value where $1/h$ is the distance between the projection of the camera center and the intersection point on the parameter plane as shown in Fig. 2 . This inverse-depth representation is adopted to minimize the non linear effect in initialization as described in [15]. In the following sections, we'll present our SLAM framework in detail.

IV. Visual SLAM Framework with Structure Lines

The main task of visual SLAM is to estimate the camera motion and build the 3D map of the surrounding simultaneously from the features detected in the image sequences. We employ the EKF framework to solve the SLAM problem similar to that of [1], where all the variables to be estimated are put into a state vector $\mathbf{x}$. It includes the camera state vector $\mathbf{x}_c$, consisting of its orientation represented by a quaternion $\mathbf{q}^{wc}$, its position $\mathbf{p}^w$, the velocity $\mathbf{v}^w$ in the world frame, and the angular velocity ω^c in the camera frame. The remainder of the state vector $\mathbf{x}$ consists of the vectors representing the feature points $\mathbf{x}_p$. We extend the framework by appending the vectors representing the structure lines $\mathbf{x}_l$. A state vector in our SLAM framework is therefore written as

$$\mathbf{x} = \left[\begin{array}{c} \mathbf{x}_c^T, \mathbf{x}_p^T, \mathbf{x}_l^T \end{array} \right]^T, \tag{2}$$

where $\mathbf{x}_p = \left[\begin{array}{c} \mathbf{m}_1^T, \mathbf{m}_2^T, \cdots \end{array} \right]^T$ and $\mathbf{x}_l = \left[\begin{array}{c} \mathbf{l}_1^T, \mathbf{l}_2^T, \cdots \end{array} \right]^T$. Both the point and structure line features are represented by the inverse depth parametrization ($\mathbf{m}_i \in \mathbb{R}^{6\times 1}$ and $\mathbf{l}_i \in \mathbb{R}^{4\times 1}$). The estimation uncertainty is described by a covariance matrix

$$\mathbf{\Sigma} = \left[\begin{array}{ccc} \mathbf{\Sigma}_{cc} & \mathbf{\Sigma}_{cp} & \mathbf{\Sigma}_{cl} \\ \mathbf{\Sigma}_{pc} & \mathbf{\Sigma}_{pp} & \mathbf{\Sigma}_{pl} \\ \mathbf{\Sigma}_{lc} & \mathbf{\Sigma}_{lp} & \mathbf{\Sigma}_{ll} \end{array} \right], \tag{3}$$

where the diagonal block matrices are co-variances of camera, point features and structure line features. The remaining block matrices are the cross-variances between different variables.

A. Estimating dominant directions

Like points, we need to add new structure lines in the state frequently to grow the map. Before initializing any structure lines, it requires to estimate the dominant directions in the world frame. This can be done by back projecting the vanishing points detected in the image into the world frame.

The vanishing point $\mathbf{v}$ can be computed by intersecting the parallel lines in the image. Mathematically, this can be achieved by solving the linear system

$$\mathbf{s}^T\mathbf{v} = 0, \tag{4}$$

where $\mathbf{s}$ is a $3 \times M$ matrix of which the columns represent the equation of the parallel lines. Here we use 3×1 homogeneous coordinates to represent the vanishing point $\mathbf{v}$.

We use the J-linkage to classify the parallel lines in the image as described in [22] [23] and then compute a rough estimate of vanishing point using (4). Then we refine the result by a nonlinear least square optimization that minimizes the distances from the end points of the observed line segment to the line connecting the vanishing point and the mid point of the observed line segment.

After we get the vanishing point $\mathbf{v}$ in the image, the dominant direction in three-dimensional space can be obtained:

$$\eta \propto \mathbf{R}^{wc}\mathbf{K}^{-1}\mathbf{v}. \tag{5}$$

Here, η is a 3×1 vector representing the dominant direction in 3D space and $\mathbf{R}^{wc}$ is the camera rotation matrix corresponding to $\mathbf{q}^{wc}$, indicating the rotation transform from the camera frame to the world frame. $\mathbf{K}$ is the intrinsic matrix of the camera that can be calibrated ahead. The distortion in the image can be removed after calibration.

Conversely, if the dominant directions are known, their corresponding vanishing points in the image can be obtained

$$\mathbf{v} = \mathbf{K}\mathbf{R}^{cw}\eta. \tag{6}$$

Here $\mathbf{R}^{cw} = (\mathbf{R}^{wc})^T$ represents the rotation transform from the world frame to the camera frame. The vanishing point $\mathbf{v}$ in the image can be used to determine the dominant of the newly detected line segments in the image, which will be described in the next section.

As the dominant directions are assumed to be perpendicular to each other, it requires at least two sets of parallel lines to estimate them (The third direction can be obtained by the cross product of the first two directions). We use (4) and (5) to estimate rough values of the dominant directions and orthogonalize them using cross product. Then the three directions are slightly adjusted by a rotation minimizing the distances between the vanishing points computed from (6) and the observed ones obtained from (4). If there are more than three sets of parallel lines detected, we use the three sets with the largest number of members for estimation.

B. Initialization of structure lines

New structure lines are initialized when some line segments newly appear in the image. Before initializing new structure lines, we attempt to determine the dominant directions related to the new line segments. We project the dominant direction η_i onto the image to get the corresponding vanishing points $\mathbf{v}_i$ using (6). By drawing a line through the vanishing point and the midpoint of the observed line segment, if the line can be aligned with the observed line segment closely (within 4-pixels alignment error in our experiments), the dominate direction of the observed line segment is set to be η_i.

If there is one vanishing point inside the image, a line segment may be aligned with two possible vanishing points. It will cause ambiguity in initializing structure lines. We simply discard these line segments to avoid generating structure lines assigned with false dominant directions.

It is not necessary to initialize structure lines for all new line segments, since it may increase the computational cost and make data association difficult. We select only long segments that are not close to predicted structure lines in the image for initialization.

The initialization process has two phases. The first one is to compute the projection of the camera center onto the parameter plane along the dominant direction, namely, (c_a, c_b).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

The second phase is to obtain the orientation from the projection of the camera center to the intersection of the structure line on the parameter plane, represented by a normal vector $[\cos(\theta), \sin(\theta)]^T$. The state of the structure line is initialized as $\mathbf{l} = [c_a, c_b, \theta, h]^T$ with a default depth value $1/h$. The covariance matrix $\mathbf{\Sigma}$ is also required to be augmented after one new structure line is added. The details of initialization are described in Appendix A.

After a structure line is initialized, we also keep the image patch around the the midpoint of the line segment, used for data association introduced in the latter section. In our experiments, the image patch is set to be of size 11×11 pixels.

C. Dynamic model and state prediction

We used a constant velocity and a constant angular velocity in the camera dynamic model, the same as that of [1], that is,

$$\mathbf{f}_c(\mathbf{x}_c) = \begin{pmatrix} \bar{\mathbf{p}}^w \\ \bar{\mathbf{q}}^{wc} \\ \bar{\mathbf{v}}^w \\ \bar{\omega}^c \end{pmatrix} = \begin{pmatrix} \mathbf{p}^w + \mathbf{v}^w \Delta t \\ \mathbf{q}^{wc} \cdot \mathbf{q}(\omega^c)\Delta t \\ \mathbf{v}^w \\ \omega^c \end{pmatrix}. \tag{7}$$

Here we use a *bar* sign to identify the predicted variables. The dynamic is perturbed by the noise $\mathbf{n} = \left[\begin{array}{cc} \delta \mathbf{v}^{wT}, \delta \omega^T \end{array} \right]^T$, whose covariance is written as $\mathbf{\Sigma}_n$. The state of features (including both points and lines) remains unchanged , namely

$$\bar{\mathbf{x}} = \left[\begin{array}{ccc} \bar{\mathbf{x}}_c^T, \bar{\mathbf{x}}_p^T, \bar{\mathbf{x}}_l^T \end{array} \right]^T = \left[\begin{array}{ccc} \mathbf{f}_c(\mathbf{x}_c)^T, \mathbf{x}_p^T, \mathbf{x}_l^T \end{array} \right]^T. \tag{8}$$

Given the Jacobian of the dynamic model and the noise

$$\mathbf{F_x} = \begin{bmatrix} \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}_c} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \text{ and } \mathbf{F_n} = \begin{bmatrix} \frac{\partial \mathbf{f}_c}{\partial \mathbf{n}} \\ \mathbf{0} \end{bmatrix}, \tag{9}$$

the propagated uncertainty can be described as

$$\bar{\mathbf{\Sigma}} = \mathbf{F_x} \mathbf{\Sigma} \mathbf{F_x}^T + \mathbf{F_n} \mathbf{\Sigma_n} \mathbf{F_n}^T. \tag{10}$$

In many robotic platforms, there are usually sensors that can be used to predict the motion of the robot, such as IMU and wheel odometer. In those cases, we simply extract the velocity and angular velocity information from them and replace the $\bar{\mathbf{v}}^w$ and $\bar{\omega}^c$ respectively.

D. Measurement model of structure lines

The measurements or observations are the line segments detected by the LSD detector [24]. To compute the innovation, it requires to get the image position of the structure line and measure the distance between the structure line and its associated line segments.

The image of a structure line can be computed by connecting the image positions of the associated vanishing point and intersection of the structure line and the parameter plane as shown in Fig. 2. The detailed computation is present in Appendix B.

Suppose $\mathbf{s}_j$ is the line segment associated with the i-th structure line. We denote the projection of the structure line in the image by $\bar{\mathbf{l}}_i = (\bar{l}_i^1, \bar{l}_i^2, \bar{l}_i^3)^T$. The signed distances between the end points of the line segment, $\mathbf{s}_j^a, \mathbf{s}_j^b$, and the projected structure line $\bar{\mathbf{l}}_i$ make up the measurement function:

$$\mathbf{m}_{ij} = \begin{bmatrix} \mathbf{s}_j^a \cdot \bar{\mathbf{l}}_i / \sqrt{(\bar{l}_i^1)^2 + (\bar{l}_i^2)^2} \\ \mathbf{s}_j^b \cdot \bar{\mathbf{l}}_i / \sqrt{(\bar{l}_i^1)^2 + (\bar{l}_i^2)^2} \end{bmatrix}. \tag{11}$$

The following cases make the measurement model of a structure line be carefully designed: 1) A structure line is usually related to several segments detected in the image; 2) A measurement with a long line segment should play a *stronger* role than that with the shorter ones.

For the first case, we allow associating one structure line with multiple line segments. That means, the measurement function of a structure line $\mathbf{l}_i$ has the following form:

$$\mathbf{m}_i = \left[\begin{array}{ccc} \mathbf{m}_{i1}^T, \cdots, \mathbf{m}_{ij}^T, \cdots \end{array} \right]^T. \tag{12}$$

To make the measurement impact of line segments proportional to their lengths, we simply break the long line segment into several short segments and compute the measurement function. By doing this, longer segments will lead to more residuals and get more impact on EKF update step. In our experiments, we break long segments into short segments with a length of 30 pixels in the image.

The measurement function of the whole state is

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= \left[\begin{array}{cc} \mathbf{m}(\mathbf{x})^T, \mathbf{p}(\mathbf{x})^T \end{array} \right]^T \\ &= \left[\begin{array}{ccccccc} \mathbf{m}_1^T, \cdots, \mathbf{m}_i^T, \cdots, \mathbf{p}_1, \cdots, \mathbf{p}_k^T, \cdots \end{array} \right]^T + \mathbf{n} \end{aligned} \tag{13}$$

.

Here $\mathbf{p}(\mathbf{x})$ is the measurement function for feature points [14]. $\mathbf{n}$ is the measurement noise described by a covariance matrix of which only the dialog elements are non-zero value σ^2. The value σ is usually set to be several pixels depending on the feature detection accuracy (2 pixels in our experiments).

The innovation is computed as

$$\begin{aligned} \mathbf{r}(\mathbf{x}) &= \left[\begin{array}{cc} \mathbf{0} - \mathbf{m}(\mathbf{x})^T, \mathbf{u}^T - \mathbf{p}(\mathbf{x})^T \end{array} \right]^T \\ &= [-\mathbf{m}_1^T, \cdots, -\mathbf{m}_i^T, \cdots, \\ &\quad \mathbf{u}_1^T - \mathbf{p}_1^T, \cdots, \mathbf{u}_k^T - \mathbf{p}_k^T, \cdots]^T, \end{aligned} \tag{14}$$

where $\mathbf{u}_i$ represents the feature points detected in the image.

E. State update

The EKF update follows the standard Kalman filter formulation:

$$\begin{aligned} \text{Innovation covariance:} \quad & \mathbf{S} = \mathbf{H} \bar{\mathbf{\Sigma}} \mathbf{H}^T + \mathbf{N} \\ \text{Kalman gain:} \quad & \mathbf{K} = \bar{\mathbf{\Sigma}} \mathbf{H}^T \mathbf{S}^{-1} \\ \text{State update:} \quad & \mathbf{x} \leftarrow \bar{\mathbf{x}} + \mathbf{K}\mathbf{r} \\ \text{Covariance update:} \quad & \mathbf{\Sigma} \leftarrow \bar{\mathbf{\Sigma}} - \mathbf{K}\mathbf{S}\mathbf{K}^T. \end{aligned} \tag{15}$$

Here $\mathbf{H}$ is the Jacobian of $\mathbf{h}(\mathbf{x})$ and innovation $\mathbf{r}$ is the $\mathbf{r}(\mathbf{x})$ defined in (14). Instead of attempting to get the analytic form which requires complicated calculation, we use central difference to obtain the Jacobian matrices in our implementation.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

We also use only parts of measurements to update the state to obtain a tentative estimate, whose steps are the same as (15) except for using parts of $\mathbf{H}, \mathbf{N}$ and $\mathbf{r}$. Partial update is useful for rejecting outliers in robust association and has shown its remarkable performance in existing work [25]. We also use partial update in data association to reject outliers as stated in the following section.

F. Robust data association for structure lines

It is more difficult to associate the observations with the structure lines than the feature points, as lines are usually broken into several segments in the image and the textures along different structure lines usually resemble each other. There are some work dedicated in finding effective descriptors for line features [26] [27]. In consideration of the running time efficiency, we simply use the image patch around the midpoint of the initially associated line segment as the descriptor. We find this simple representation yields promising results while keeping high computational efficiency in practice. Though a better line descriptor may be found, it is not our main interest here. We leave this in the future work.

The first step of data association of the structure lines is to find the candidate segments that are close to the predicted structure lines and have similar appearances with the structure line by comparing the image patches. The closeness between a predicted structure line and a line segment is determined by the χ^2-distance computed as

$$\chi^2 = \mathbf{r}_i^T (\mathbf{H}_i \mathbf{H}_i^T)^{-1} \mathbf{r}_i^T. \tag{16}$$

Here, $\mathbf{r}_i$ is the 2×1 residual vector (see (11) and (14)) of the i-th structure line and $\mathbf{H}_i$ is the corresponding Jacobian matrix. The line segment with χ^2 value larger than 5.99 (with a probability less than 5%) is rejected. After that, the similarity between the line segment and the structure line is measured by ZNCC $\in [-1, 1]$ (zero mean normalized cross-correlation) between two image patches. In our experiments, ZNCC is set to 0.8 for most cases.

After obtaining the candidate segments, we use RANSAC to figure out the remaining outliers as described in the point-based visual SLAM method[25]. However, unlike point features, one straight line usually have multiple line segments associated considering the fact that lines are easily separated into several segments through line detection algorithms. So we make some changes to adapt it to our framework, which is outlined as the following:

1) A line segment is sampled randomly from all candidate ones .

2) We run a tentative EKF update using the line segment selected and use the updated state to predict all the structure lines on the image again.

3) Inliers are determined by checking the distance between the structure line and its associated segment. If the distance is smaller than a threshold (3 is used in our approach), we treat the associated line segment as an inlier. We therefore get an inlier set after above steps.

We do these steps recursively until reaching the maximum number of RANSAC iterations. Finally the inlier set with the largest number of members is our inlier set. The remaining line segments are treated as outliers and excluded from the EKF update in (15). After that, we run EKF update using all the inlier segments.

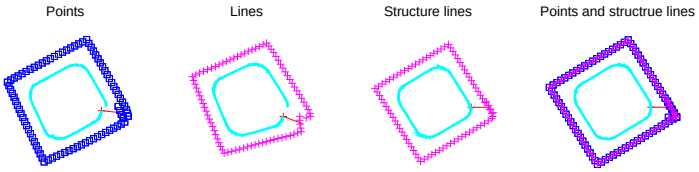


Fig. 3. Top view of the camera trajectory (cyan) and feature location projections in XZ-plane. Blue squares represent point features, while red crosses stand for vertical lines. Here we don't mark the horizontal lines.

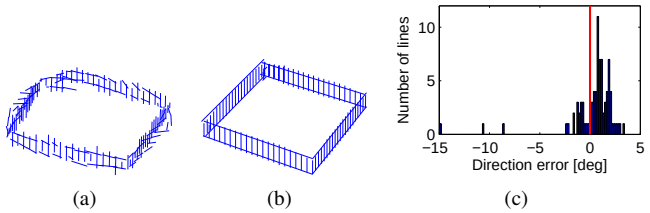


Fig. 4. The final maps produced by using line features (a) and structure lines (b). (c) shows the error distribution of the orientations of the reconstructed vertical line features (Blue: using line features; Red: using structure lines, partly visible).

Outlier rejection works in a similar way when both points and structure lines are used in the framework. As we can see in the experimental results in Fig. 17, the false associations produced by ZNCC matching will be removed by outlier rejection, which increases the robustness of the algorithm.

G. Feature Management

Since the number of features in the map grows over time, it will reduce the run time efficiency significantly as the dimension of the state becomes large. Like managing point features, we also set a maximum number of structure lines that are kept in the state for each dominant direction. New structure lines are initialized only when the number of structure lines is less than the expected one.

To limit the number of line features within a reasonable level, we need to remove the old features that do not appear in the image. To manage the features, we keep a number of matching failure (NoF) variable for each feature to indicate the number of successive steps that it fails to be associated with any observations. NoF is reset to zero when the feature succeeds in matching some observations again. When the number of features exceeds a predefined threshold (two times of the expected feature number of 15 is used in our experiments), the features with the largest NoF are removed from the state. The covariance matrix is also updated accordingly.

V. Experiment

A. Synthetic scenes

To better understand the characters of different algorithms, we synthesize a simple scene for evaluation. The synthetic scenario consists of 20×20 m four-side barriers, including 88 lines

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

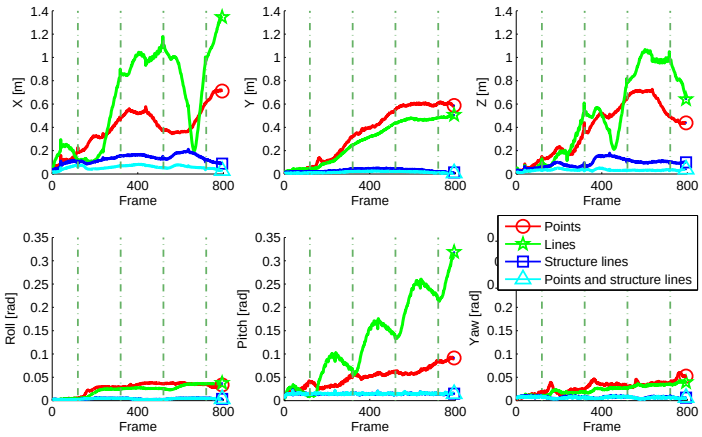


Fig. 5. RMSE of the camera pose (Translation in X, Y, Z and Orientation in Roll, Pitch, Yaw) using point-based method (red, circle), line-based method (green, star), structure-line-based method (blue, square), structure line and point hybrid method (cyan, triangle), over 25 Monte Carlo runs. The dashed vertical lines mark the frames (Frame 120, 320, 520 and 720) where the camera starts to turn into a corner (4 corners altogether).

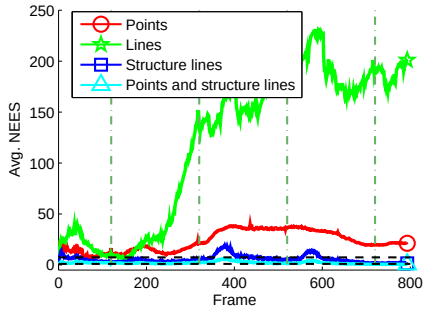


Fig. 6. The averaged NEES of the camera pose (Translation in X, Y, Z and Orientation in Roll, Pitch, Yaw) over 25 runs. The dashed black lines mark the consistency band limit of 95% confidence for 6 DOF and $N = 25$ runs, namely $\bar{\eta} = 7.432$ and $\underline{\eta} = 4.719$ ($\bar{\eta} = \chi^2_{25\times6}(1-0.975)/25 = 7.432, \underline{\eta} = \chi^2_{25\times6}(1-0.025)/25 = 4.719$)[28]. If the averaged NEES is larger than $\bar{\eta}$ for more than 2.5% of the whole time, the filter is considered inconsistent [29].

(80 vertical, 8 horizonal) and 160 points. A perspective camera (90° FOV, 640×320 pixels, added with a 2-pixel Gaussian measurement noise) moves inside the barriers and completes a square trajectory in the XZ plane, with the heading towards the barriers. 794 sequential images are generated and in each frame we let the number of lines and points be approximately the same.

Four methods are used for comparison: the point-based one [15], the line-based one [3], the proposed method using only structure lines and the proposed method using both points and structure lines. All initialization conditions and related parameters are kept the same for the four approaches. The estimated trajectories are shown in Fig. 3. It is clear that the methods using only points and only lines perform worse than our method. They fail to construct the square shape of the barriers and have a large close loop error. The advantage of using structure lines is obvious: with the global orientation information encoded in the structure lines, both the trajectories and the 3D landmarks coincide well with the ground truth.

Fig. 4 presents the side-by-side comparison between the 3D maps of the line-based method and our method. As shown in the figure, the orientations of many lines are incorrect in the line-based method. We take the vertical lines as an example, whose direction error distribution is shown in Fig. 4(c). There are 15 out of 86 vertical lines with errors more than 2 degree, while our method do not have such problems: all lines are well aligned with their true directions.

To provide a quantitative analysis we perform Monte Carlo experiments of 25 runs. At each frame, the root mean square error (RMSE) for each camera pose component (Translation in X, Y, Z and Orientation in Roll, Pitch, Yaw) over 25 runs is calculated. The results are shown in Fig. 5. When the camera turns around the first corner, both point-based method and line-based method start to show an increase in the estimation error. Especially for the line-based method, the orientation error in pitch largely increases at each corner and finally reaches as high as 0.3180 rad (more than ten times of the proposed method 0.0143 rad). It is maybe due to the redundant degree of freedom of the line representation by 2-points[3] : the two non-overlap points can locate anywhere along the line, making it more difficult to let the line to converge to the real pose.

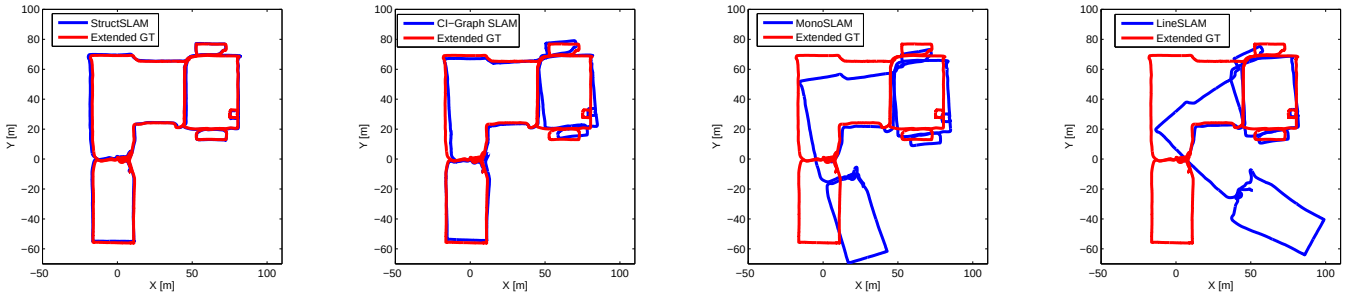
In contrast, the position errors of the proposed method are always kept within a small value (0.2 m) during the whole trajectory (4×20 m), and it increase slightly if using structure lines alone (e.g. RMSE in X, structure line only: 0.1311 m, hybrid: 0.0576 m). It therefore clearly shows that structure lines play a critical role in our proposed method, where both points and structure lines are adopted. To test the consistency of our algorithm, we use the averaged normalized estimation error squared (NEES), as explained in [29]. As shown in Fig. 6, the methods using only lines and only points are inconsistent, mostly because of their inaccuracy. The averaged NEES of our methods is however constantly under the upper limit and considered as consistent.

In summary, the simple case demonstrates that adopting structure lines in the SLAM framework will lead to a remarkable improvement in the estimation accuracy of both orientation and position. In the following experiments, we will test our method in real-world scenes with or without the ground truth.

B. Benchmark with large scale indoor scenes

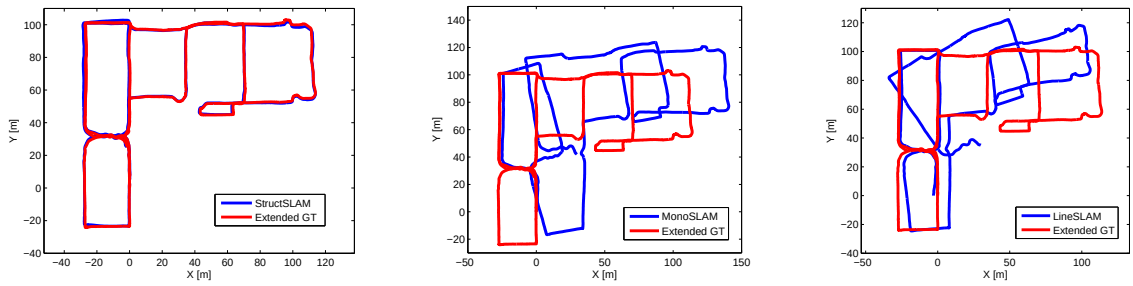
We test our methods on the dataset of *Biccoca_2009-02-25b* and *Biccoca_2009-02-27a* from the *RAWSEEDS* 2009 [30], which have been proposed as benchmarks for SLAM problems. The datasets are collected by a wheel robot mounted with multiple sensors in an office building. The indoor scenarios consist of a wide range of different architectural features, e.g. halls, wide and narrow corridors, passageways between buildings and libraries and the light conditions are changing significantly among different places. In our experiments the image sequences from the frontal camera (The image resolution is 320×240.) are used for tests. To evaluate the result, we align the resultant trajectory with the extend ground truth

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology



(a) StructSLAM vs GT (b) CI-Graph vs GT (c) MonoSLAM vs GT (d) LineSLAM vs GT

Fig. 7. Results of different methods (blue trajectories) aligned with the ground truth (red trajectories) of *Biccoca_2009-02-25b*, where the traveling distance is about $774m$.



(a) StructSLAM (b) MonoSLAM (c) LineSLAM

Fig. 8. Results of different methods (blue trajectories) and the ground truth (red trajectories) of *Biccoca_2009-02-27a*. The traveling distance is 967 m.

	Biccoca 25b [774 m]						Biccoca 27a [967m]					
	Position[m]			Yaw[rad]			Position[m]			Yaw[rad]		
	Mean	Max	Std	Mean	Max	Std	Mean	Max	Std	Mean	Max	Std
MonoSLAM	12.22	36.12	9.469	0.154	0.371	0.102	24.85	37.90	11.66	0.187	0.392	0.104
LineSLAM	27.63	102.7	29.65	0.509	1.393	0.420	12.77	32.35	10.17	0.208	0.706	0.206
CI-GraphSLAM	2.236	6.453	1.675	0.055	0.155	0.035	-	-	-	-	-	-
StructSLAM	**0.797**	**1.916**	**0.447**	**0.012**	**0.129**	**0.011**	**0.793**	**1.913**	**0.493**	**0.017**	**0.214**	**0.017**

TABLE I. ABSOLUTE ERROR COMPARISON.

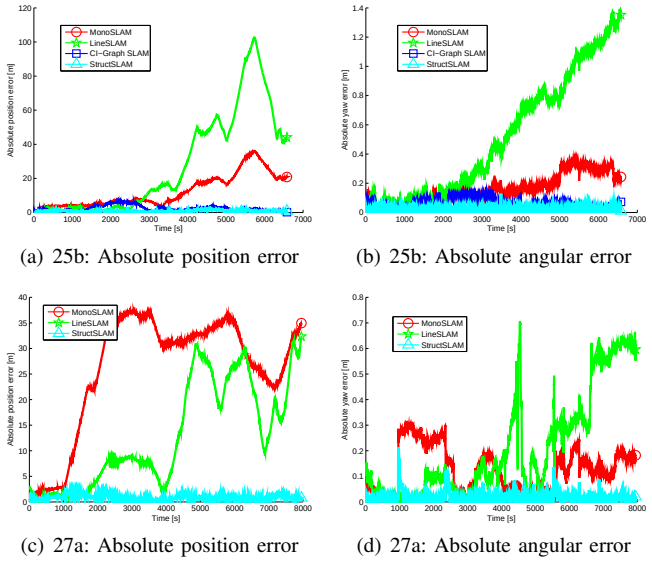
[30] by setting the starting point to the ground truth one and rotating the trajectory toward the ground truth by minimizing the distance between them. Existing visual SLAM methods that use only points (MonoSLAM) [14], lines (LineSLAM) [3] are compared with the proposed method. Wheel odometry is adopted for motion predictions (see Section IV-C) for all methods, which has also been adopted in the benchmark solution [31].

1) Biccoca_2009-02-25b : This dataset consists of $52,671$ image frames captured in 30 minutes at 15 FPS. The whole path is as long as 774 m with several loops and revisited regions. The sequence is quite challenging for the lack of textures in some areas and several sharp turns in very narrow passageways. Apart from comparing with the MonoSLAM and LineSLAM methods, we also compare with the published results of the benchmark solution [31] in this case. The benchmark solution, named as *CI-Graph SLAM*, uses two other cameras on the robot for estimation. It is the state-of-the-art visual slam method that performs well in large scale scenes by frequently applying loop closure detection and loop closing algorithms.

The results are shown in Fig. 7. Notice that for MonoSLAM and LineSLAM, the orientation errors increase at every turning around the textureless corners. The errors are accumulated and finally lead to a large position error. Since CI-Graph SLAM method adopts loop closing algorithms to correct the accumulated errors when a loop is detected, it yields much better result than MonoSLAM and LineSLAM. But there is still a clear orientation drift after the narrow corner in the second building because of lacking features as shown in Fig. 7(b).

Our StructSLAM method exploits the orientation information of the structure lines as a global constraints on the estimates, which means the error at each time step will be rectified globally. Therefore, the errors are not accumulated but bounded (within 2 m) during the whole trajectory. For better illustration, we show the absolute position error and absolute yaw error of all the approaches in Fig. 9. As shown in Table I our method presents a precise estimates with a mean absolute error of 0.79 m and 0.01 rad, outperforming the results of 2.24 m and 0.06 rad by CI-Graph SLAM and far more better than the results by MonoSLAM and LineSLAM. It is also

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology



(a) 25b: Absolute position error (b) 25b: Absolute angular error

(c) 27a: Absolute position error (d) 27a: Absolute angular error

Fig. 9. Comparison of the absolute position error and absolute angular error over time of different methods for the *Biccoca_2009-02-25b* and the *Biccoca_2009-02-27a* datasets. The errors of our method are bounded in small values. Better viewed on the screen. Red circle: MonoSLAM; Green star: LineSLAM; Blue square: CI-Graph SLAM; Cyan triangle: StructSLAM.

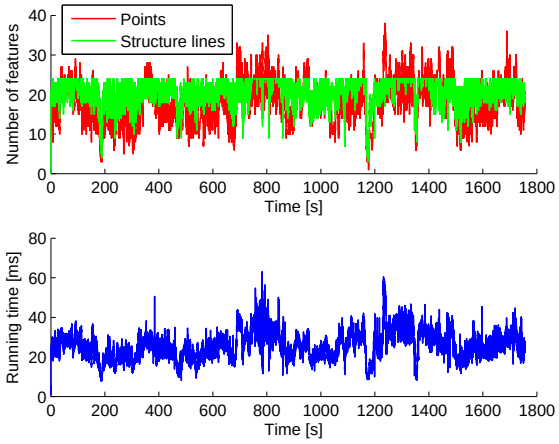


Fig. 10. Running time of StructSLAM for processing one image. Top row: the number of points and the number of structure lines in the state. Bottom row: the running time in milliseconds.

worth noting that our StructSLAM achieves this performance with only one camera and no loop closing algorithm.

2) Biccoca_2009-02-27a: This sequence (including 68,063 frames) is captured in the same scene as in *Biccoca_2009-02-25b* along a different path with a longer distance (967 m). We present the results aligned with the ground truth in Fig. 8. The proposed method still performs well in this case. The mean position error is 0.793 m and the mean orientation error is 0.017 rad. As the result of CI-Graph method is not available for this dataset, we only compared with the MonoSLAM and LineSLAM methods. As we can see in the figure, they yield

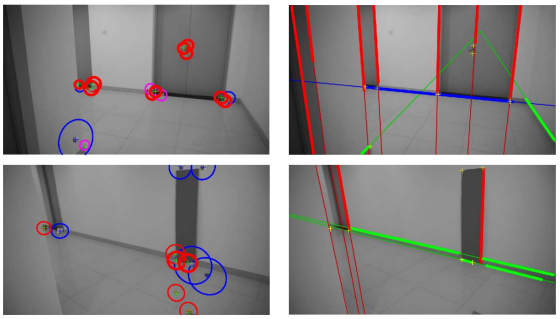
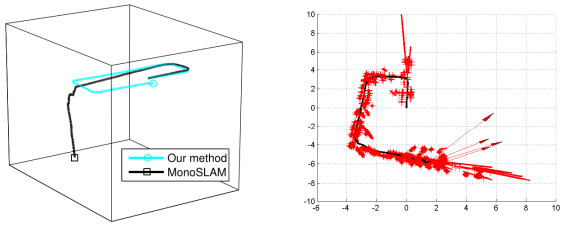


Fig. 11. Texture-less walls pose a great challenge for visual SLAM using points. **Left column:** Feature points detected in MonoSLAM [14]. Thick red circles - low innovation inliers; Thin red circle, -high innovation inliers; Magenta - outliers; **Right column**: Features detected by our method. Red, green, and blue lines represented the structure lines in three dominant directions. Thick ones stand for associated segments. Yellow crosses are the feature points used in our method.



(a) (b)

Fig. 12. **Left**: Comparison between the camera trajectories of our method (cyan, marked by a circle) and that of MonoSLAM (black, marked by a square) in 3D space. **Right**: Result of MonoSLAM in 2D. Notice that a large drift error occurs after the third turning.

very large position error due to the accumulated orientation errors. We showed the absolute errors of all methods performed on both datasets in Fig. 9 and listed the statistical results in Table I. Notice that our method yields almost the same errors for two datasets despite their different lengths. That indicates that our method produces bounded error without growing with the traveling distance.

C. Running time efficiency

The proposed method is implemented in both matlab and c++. The matlab version is slow, it takes about $0.5 \sim 1$ seconds to handle about 40 features including both points and structure lines. The c++ version is much faster, we also record the time for processing each image and the corresponding number of features in the state for *Biccoca_2009-02-27a* dataset. As shown in Fig. 10, the maximum number of points is limited to 40 and the number of structure lines is limited to 8 for each direction (24 in total). The running time increases when more features are present. The average running time for each image is about $25.8ms$ and the maximum time is 62.9ms. This is fast enough for real-time applications.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

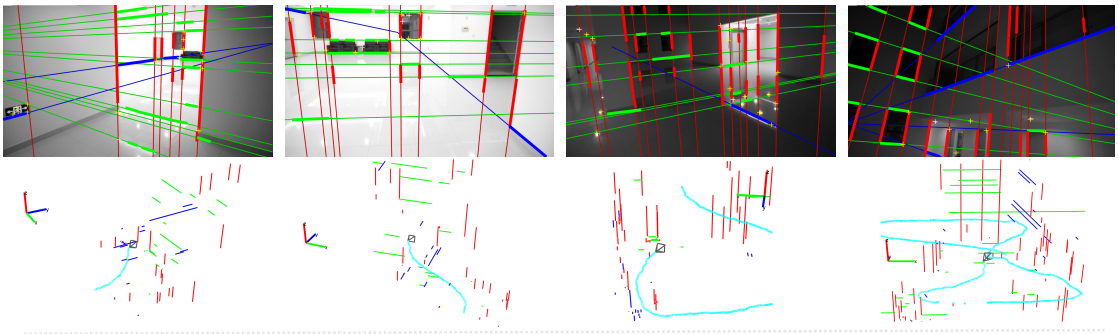


Fig. 14. Result of a complex indoor scene in the night. Notice that the illumination changes significantly between dark and bright. **Top row:** Features detected in the image. **Bottom row:** Structure lines and the camera trajectory in 3D space.

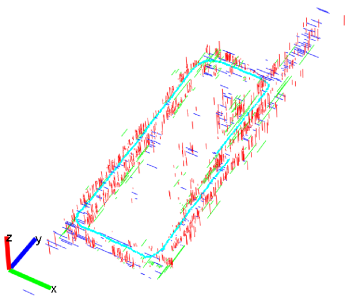


Fig. 13. Result of our method. The red lines represent the vertical lines, the green and blue lines represent the horizonal lines in Y and X direction respectively. The camera trajectory is marked as cyan. Notice that there is an obvious close-loop error. This is largely due to the little number of features when turning at the corners. However, this can be remedied by loop closing easily.

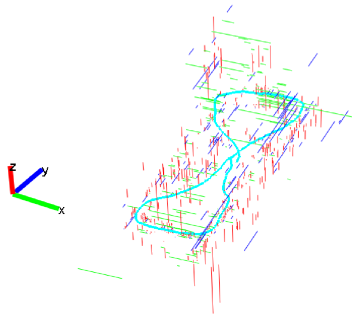


Fig. 15. Result of an indoor scene with a large hall in the night.

D. Real world using hand-held camera

We use a hand-held camera (90° FOV) to obtain image sequences with a resolution of 640×320 pixels. Three scenes were tested. The first one is a sequence of indoor corridors with closed loop. In this test, we demonstrate the robustness of our method in texture-less scenes, where the conventional visual SLAM method fails. The second test is conducted in a large hall surrounded by corridors in the nighttime, experiencing great illumination changes, performing in a relatively random route. The third test is performed in an outdoor courtyard with disturbing line features that are not aligned with the dominant directions. We refer the interested readers to the video result published online [1].

1) Closed-loop indoor corridor: In this test, the image sequence is collected following a rectangular closed-loop path (12×33 m) along the corridor. The sequence is challenging for visual SLAM because it contains mainly texture-less walls. We compared our method with the standard point-based SLAM codes (MonoSLAM) from Javier Civera's implementation [15]. The 3D camera trajectories produced by different methods are shown in Fig. 12(a), where we align the camera trajectories in the coordinate system for clearer comparison. The proposed method using both points and structure lines yields better result than MonoSLAM does. Notice that MonoSLAM produces a remarkable drift error as shown in Fig. 12(b). This is caused by the lack of feature points in the scene, particularly around the corners as shown in Fig. 11. Our method in contrast works smoothly at the texture-less corners, and produces much less drift error. The result is shown in Fig. 13. The proposed method generates a drift error of 1% of the whole travel distance when returning to the starting point. This is promising for this challenging case and this level of drift error can be easily eliminated by a loop closing algorithm. It is also worth noticing that the 3D map represented by structure lines is better at describing the structures of the scene than the one represented by points. This could be useful for indoor modeling or scene understanding applications.

2) Hall at night: In this test, the camera moves in a figure-eight path and returns to the starting point in an indoor environment with a large hall in the center. The images are captured in the night where the illumination changed significantly. Fig. 14 shows that our method handles these situations successfully . The 3D map and camera trajectory are shown in Fig. 15. Interested readers are referred to the demonstration video online.

3) Corridor and courtyard: Having evaluated our proposed method with indoor environments, it would be very interesting to validate it in an outdoor scenario. We capture the scene starting from an corridor and then move into a courtyard and walk around. The final result is shown in Fig. 16. It is worth

[1] The video clip is at http://youtu.be/7HNdJEb21DQ

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

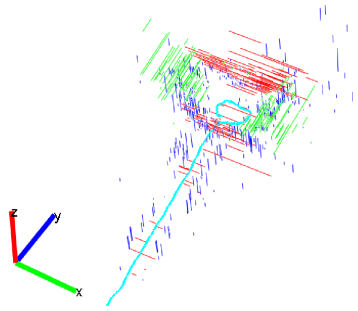


Fig. 16. Result of an outdoor scene with disturbing line features.

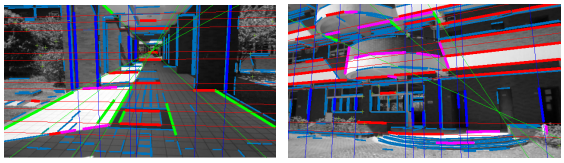


Fig. 17. Disturbing line features are rejected. Blue thin lines are the segments detected by LSD line detector [24]. Structure lines and their associated segments are represented by thin and thick color lines. The magenta segments are wrongly matched to structure line with the similar appearance, which are successfully rejected by the robust strategy. Some segments that are not aligned with any existing dominant directions are discarded (e.g. the shadow in the left image and the arc-shaped steps in the right image).

noting that the outdoor scene contains a rich set of features, which include lines that are not aligned with the dominant directions as shown in Fig. 17. Our method can discard them by robust data association.

VI. Conclusions and Future Work

In this work, we propose a visual 6DoF SLAM method using structure building lines as novel features complemented with the point features. Our method is validated in both synthetic scenes and real-world scenes. The results show that for indoor environment that lacks point features, the conventional point-based SLAM could produce a large drift error or even fail to work, while our method works well with structure lines and obtains much less drift error due to the global constraint on camera orientations. The results also show that when there are line segments that are not aligned with three dominant directions in the scene, our method still works well by rejecting them using the robust data association approach, indicating the robustness of our method.

The line-point hybrid strategy also makes our method flexible to different kinds of scenes. For indoor scenes that contains mainly texture-less walls, line segments can be extracted and used for localization and mapping. In contrast, for outdoor scenes that contain less man-made structures, point features become dominant in our method.

The assumption that there are three perpendicular directions in the scene holds for most man-made buildings, but sometimes there are also exceptions where more than three dominant directions exist or the dominant directions are not perpendicular to each other. Our method simply treat the line features that are not aligned with three perpendicular directions as outliers. This may cause problem when the number of such outliers is not negligible. This is the first issue that we leave to be addressed in the future. Another issue is that we simply use the image patch around the midpoint of the line segment for data association. Since line features are less distinguishable than point features in the image (There are usually many lines with similar appearances in the image), using this simple descriptor usually cause false matchings. In our implementation, we set the ZNCC threshold for matching as high as 0.8 to decrease false matchings. But the high threshold value also reduces the number of correct associations as the view point changes, which causes duplicated lines initialized from line segments that should be associated to one structure line. Hence we'll investigate a better descriptor for matching line features in the future.

Appendix A
Initialize a Structure Line

Suppose $\tilde{\mathbf{m}}$ is the 3×1 homogenous coordinates of the midpoint of the line segment that is used for initialization. Its 3D coordinates in the word frame is computed as

$$\mathbf{m} = \mathbf{R}^{wc}\mathbf{K}^{-1}\tilde{\mathbf{m}} + \mathbf{p}^{w}. \tag{17}$$

The structure line in the world frame can be described by the Plucker matrix $\mathbf{L} = \mathbf{m}\eta^T - \eta\mathbf{m}^T$, where $\eta \in \mathbb{R}^{3\times 1}$ is the related dominant direction. The intersection of the structure line and the parameter plane π, represented by A as shown in Fig. 2, is computed as

$$\tilde{\mathbf{l}}^{w} = \mathbf{L}\pi. \tag{18}$$

The 2D coordinates of the intersection point in the parameter plane is given by

$$\mathbf{l}^{p} = \mathbf{P}\mathbf{l}^{w}. \tag{19}$$

Here $\mathbf{l}^w$ is the 3D coordinates computed from the homogeneous representation $\tilde{\mathbf{l}}^w$ and $\mathbf{P} \in \mathbb{R}^{2\times 3}$ is a projection matrix that transforms the 3D vector into a 2D parameter vector.

In the same manner, the line through the camera center along the dominant direction intersects the parameter plane with the point (O' in Fig. 2)

$$\tilde{\mathbf{o}}^{w} = (\mathbf{c}\eta^T - \eta\mathbf{c}^T)\pi, \tag{20}$$

Hence, we get the projection of the camera center in 2D coordinates

$$\mathbf{o}^{p} = \mathbf{P}\mathbf{o}^{w}. \tag{21}$$

After that, we can derive the parameters of the initialized structure line from (19) and (21)

$$\begin{aligned}\mathbf{l} &= [\ c_a, c_b, \theta, h\ ]^T \\ &= \left[\ \mathbf{o}^p(1), \mathbf{o}^p(2), atan(\tfrac{\mathbf{l}^p(2)-\mathbf{o}^p(2)}{\mathbf{l}^p(1)-\mathbf{o}^p(1)}), h_0\ \right]^T,\end{aligned} \tag{22}$$

where h_0 is a preset inverse depth value.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2015.2388780, IEEE Transactions on Vehicular Technology

To augment the covariance matrix after initialize a new structure line, we need to compute the covariance of the newly initialized structure line , $\mathbf{l}^{(i)}$, as

$$\begin{aligned}\mathbf{\Sigma}_{ll}^{(i)} = \frac{\partial \mathbf{l}^{(i)}}{\partial \mathbf{x}_c}\mathbf{\Sigma}_{cc}\frac{\partial \mathbf{l}^{(i)}}{\partial \mathbf{x}_c}^T + \frac{\partial \mathbf{l}^{(i)}}{\partial \mathbf{s}}\mathbf{n}_{4\times 4}\frac{\partial \mathbf{l}^{(i)}}{\partial \mathbf{s}}^T \\ + \frac{\partial \mathbf{l}^{(i)}}{\partial h}\sigma_h^2\frac{\partial \mathbf{l}^{(i)}}{\partial h}^T\end{aligned} \tag{23}$$

It involves computing the Jacobian matrices of the structure line with respect to the camera variable $\mathbf{x}_c$, the line segment $\mathbf{s}$ and the inverse depth value h. Here $\partial \mathbf{l}^{(i)}/\partial h = [0, 0, 0, 1]^T$. The other Jacobian matrices are computed in a numerical approach using central difference in our implementation. The noise covariance $\mathbf{n}_{4\times 4}$ is a diagonal matrix whose diagonal elements are σ^2, which is the same as defined in (13).

The cross-variance between the newly initialized structure line and other variables in the state is computed as

$$\mathbf{\Sigma}_{lx}^{(i)} = \frac{\partial \mathbf{l}^{(i)}}{\partial \mathbf{x}_c}\mathbf{\Sigma}_{1:13,\cdots}. \tag{24}$$

Here $\mathbf{\Sigma}_{1:13,\cdots}$ represents the first thirteen rows of the covariance matrix $\mathbf{\Sigma}$. The state is augmented after a new structure line $\mathbf{l}^{(i)}$ is initialized

$$\mathbf{x} \leftarrow \left[\ \mathbf{x}^T, \mathbf{l}^{(i)T}\ \right]^T \tag{25}$$

The covariance of the whole state is then augmented as

$$\mathbf{\Sigma} \leftarrow \begin{bmatrix} \mathbf{\Sigma} & \mathbf{\Sigma}_{lx}^{(i)T} \\ \mathbf{\Sigma}_{lx}^{(i)} & \mathbf{\Sigma}_{ll}^{(i)} \end{bmatrix}. \tag{26}$$

APPENDIX B
PROJECT A STRUCTURE LINE ONTO THE IMAGE

We get the image vanishing point associated with the structure line using (6), denoted by $\mathbf{v}$. The point in the parameter plane, denoted as $\mathbf{l} = [c_a, c_b, \theta, h]^T$, is first transformed into the world frame by

$$\mathbf{l}^w h = \mathbf{P}^T\left(\left[\ c_a, c_b\ \right]^T h + \left[\ \cos(\theta), \sin(\theta)\ \right]^T\right). \tag{27}$$

Here $\mathbf{P} \in \mathbb{R}^{2\times 3}$ is a projection matrix that transforms the 3D vector into a 2D parameter vector and we multiply both sides of the equation by the inverse depth h, so that it can be safely used for infinity cases, i.e., $h = 0$.

After obtaining the point in world frame, we transform it into the camera frame

$$\mathbf{l}^c = \mathbf{R}^{cw}\mathbf{l}^w h - \mathbf{R}^{cw}\mathbf{p}^w h. \tag{28}$$

The point in the image is then obtained as

$$\mathbf{l}^i = \mathbf{K}\mathbf{l}^c. \tag{29}$$

So the projected structure line is computed as the cross product of the vanishing point and the projected point

$$\bar{\mathbf{l}} = \mathbf{v} \times \mathbf{l}^i. \tag{30}$$

ACKNOWLEDGMENT

The research work has been funded by Chinese National Key Project (BDZX005) and Natrual Science Foundation of China (Grant No. 6140050267).

REFERENCES

[1] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *IEEE Proc. of International Conference on Computer Vision*, 2003, pp. 1403–1410.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *IEEE & ACM Proc. of Int'l Sym. on Mixed and Augmented Reality*, 2007, pp. 225–234.

[3] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular slam with straight lines." in *BMVC*, vol. 6, 2006, pp. 17–26.

[4] T. Lemaire and S. Lacroix, "Monocular-vision based slam using line segments," in *IEEE Proc. of Robotics and Automation*. IEEE, 2007, pp. 2791–2796.

[5] J. Sola, T. Vidal-Calleja, and M. Devy, "Undelayed initialization of line segments in monocular slam," in *IEEE/RSJ Proc. of Intelligent Robots and Systems*. IEEE, 2009, pp. 1553–1558.

[6] E. Perdices, L. M. López, and J. M. Cañas, "Lineslam: Visual real time localization using lines and ukf," in *ROBOT2013: First Iberian Robotics Conference*. Springer, 2014, pp. 663–678.

[7] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *RAS*, vol. 57, no. 12, pp. 1188–1197, 2009.

[8] J. M. Coughlan and A. L. Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," in *IEEE Proc. of International Conference on Computer Vision*, vol. 2. IEEE, 1999, pp. 941–947.

[9] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Manhattan-world stereo," in *IEEE Proc. of Conference on Computer Vision & Pattern Recognition*. IEEE, 2009, pp. 1422–1429.

[10] A. Flint, C. Mei, I. Reid, and D. Murray, "Growing semantically meaningful models for visual slam," in *IEEE Proc. of Conference on Computer Vision & Pattern Recognition*. IEEE, 2010, pp. 467–474.

[11] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics," in *European Conference on Computer Vision*. Springer, 2010, pp. 482–496.

[12] L. Ruotsalainen, J. Bancroft, and G. Lachapelle, "Mitigation of attitude and gyro errors through vision aiding," in *IEEE Proc. of Indoor Positioning and Indoor Navigation*. IEEE, 2012, pp. 1–9.

[13] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," *ACM Trans. on Graphics*, vol. 25, no. 3, pp. 835–846, 2006.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[15] J. Civera, A. J. Davison, and J. Montiel, "Inverse depth parametrization for monocular slam," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[16] W. Y. Jeong and K. M. Lee, "Visual slam with line and corner features," in *IEEE/RSJ Proc. of Intelligent Robots and Systems*. IEEE, 2006, pp. 2570–2575.

[17] M. Chandraker, J. Lim, and D. Kriegman, "Moving in stereo: Efficient structure and motion using lines," in *IEEE Proc. of International Conference on Computer Vision*. IEEE, 2009, pp. 1741–1748.

[18] G. Tsai, C. Xu, J. Liu, and B. Kuipers, "Real-time indoor scene understanding using bayesian filtering with motion cues," in *IEEE Proc. of International Conference on Computer Vision*. IEEE, 2011, pp. 121–128.

[19] J. Sola, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular ekf-slam with points and lines," *Int'l Journal of Computer Vision*, vol. 97, no. 3, pp. 339–368, 2012.

[20] Y. H. Lee, C. Nam, K. Y. Lee, Y. S. Li, S. Y. Yeon, and N. L. Doh, "Vpass: Algorithmic compass using vanishing points in indoor environments," in *IEEE/RSJ Proc. of Intelligent Robots and Systems*. IEEE, 2009, pp. 936–941.

[21] G. Zhang, D. H. Kang, and I. H. Suh, "Loop closure through vanishing points in a line-based monocular slam," in *IEEE Proc. of Robotics and Automation*. IEEE, 2012, pp. 4565–4570.

[22] R. Toldo and A. Fusiello, "Robust multiple structures estimation with j-linkage," in *European Conference on Computer Vision*. Springer, 2008, pp. 537–547.

[23] J.-P. Tardif, "Non-iterative approach for fast and accurate vanishing point detection," in *IEEE Proc. of International Conference on Computer Vision*. IEEE, 2009, pp. 1250–1257.

[24] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: a line segment detector," *Image Processing On Line*, 2012.

[25] J. Civera, O. G. Grasa, A. J. Davison, and J. Montiel, "1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, no. 5, pp. 609–631, 2010.

[26] Z. Wang, F. Wu, and Z. Hu, "Msld: A robust descriptor for line matching," *Pattern Recognition*, vol. 42, no. 5, pp. 941–953, 2009.

[27] K. Hirose and H. Saito, "Fast line description for line-based slam." in *Proc. British Machine Vision Conference*, 2012, pp. 1–11.

[28] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[29] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 3562–3568.

[30] B. Andrea, B. Wolfram, F. Giulio, M. Matteo, G. S. Domenico, and D. T. Juan, "Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets," in *In proceedings of IROS'06 Workshop on Benchmarks in Robotics Research*, 2006. [Online]. Available: http://www.robot.uji.es/EURON/en/iros06.htm

[31] P. Piniés, L. M. Paz, D. Gálvez-López, and J. D. Tardós, "Ci-graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system," *Journal of Field Robotics*, vol. 27, no. 5, pp. 561–586, 2010.

Huizhong Zhou received the B.S. degree in electrical engineering from Shanghai Jiao Tong University (SJTU), China, in 2012. From 2011 to 2013, she participated an exchange double-master program in Technical University of Berlin, Germany. Since 2013, she has been a master student with Shanghai Key Laboratory of Navigation and Location Based Services, SJTU. Her research interests include computer vision and indoor localization.

Danping Zou received the B.S. degree from Huazhong University of Science and Technology (HUST) and the Ph.D. degree from Fudan University, China, in 2003 and 2010, respectively. From 2010 to 2013 he was a research fellow in the Department of Electrical and Computer Engineering at the National University of Singapore. He jointed the Department of Electronic Engineering at Shanghai Jiao Tong University as an assistant professor in 2013. His research interests include low-level 3D vision on robotics.

Ling Pei received his Ph.D. degree in test measurement technology and instruments from the Southeast University, China, in 2007. From 2007 to 2013, he worked as a specialist research scientist in the Finnish Geodetic Institute, Finland. He is currently an Associate Professor at the School of Electronics Information and Electrical Engineering in the Shanghai Jiao Tong University, China. His research interests include indoor/outdoor seamless positioning, ubiquitous computing, wireless positioning, context-aware applications and location-based services.

Rendong Ying received his B.S. degree in 1994 from East China Norm University, Shanghai, China, Master and Ph.D. degrees in 2001 and 2007 respectively from Shanghai Jiaotong University, all in electrical engineering. He is now an associate professor at the Department of Electronic Engineering at Shanghai Jiao Tong University.

Peilin Liu (M'07) received the D.Eng. degree from the University of Tokyo, Tokyo, Japan, in 1998. She was a Researcher with the University of Tokyo, Japan, in 1999. From 1999 to 2003, she was a Senior Researcher with the Central Research Institute of Fujitsu, Tokyo, Japan. She joined Shanghai Jiao Tong University, China, in 2003, and is currently a Professor with the Department of Electronic Engineering. She directs the MediaSoC Lab and co-directs the Shanghai Key Laboratory of Navigation and Location Based Services.

Wenxian Yu received the B.Sc., M.Sc., and Ph.D. degrees from the National University of Defense Technology, Changsha, China, in 1985, 1988, and 1993, respectively. He is currently with the School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, China, where he is named Distinguished Professor and Executive Dean of School. Since 2011, he directs the Shanghai Key Laboratory of Navigation and Location Based Services.