Hand-held Monocular SLAM Based on Line Segments

Lilian Zhang, Reinhard Koch
Institute of Computer Science
University of Kiel, Germany
Email:{lz,rk}@informatik.uni-kiel.de

***Abstract*—This paper presents an approach of visual SLAM using line segments as primitives. We choose Plücker lines to represent spatial lines and derive a novel fast line projection function based on the relationship between Plücker line and Plücker matrix. During normal SLAM procedure, the Nearby Line Tracking (NLT) method is adopted to track lines and an Extended Kalman Filter (EKF) is used to predict and update the state of the camera and line landmarks. After each update step, a robust spatial line reconstruction algorithm is used to find new line landmarks and to add them into the map. The SLAM procedure is under supervision and when failure is detected, a recovery method based on the angle histogram of keyframes is adopted to relocalize the camera and re-start the SLAM procedure. We demonstrate that our monocular SLAM system is robust to illumination change, partial occlusion and fast camera motion.**

***Keywords*—Line Projection Function, Visual SLAM, Failure Recovery.**

I. Introduction

Since a real time monocular SLAM (Simultaneous Localization And Mapping) implementation was demonstrated by Davison in 2003 ([3] is an up-to-date version), the research on vision based SLAM has been booming both in computer vision and robotics. Many applications in Augmented Reality (AR) for industry or entertainment [8] and Autonomous Navigation (AN) for robots to explore partially known or unknown environment [12] are implemented and demonstrated in practice. The real time performance is a challenging task, however, due to the increasing availability of computer capability, this challenge is tackled by these properly designed systems.

Among these visual SLAM systems, point features are favourable to be taken as landmarks for their desirable properties: Point feature selection and description is well studied, and they are easily represented both in 2D image and 3D map. Unfortunately, they are also easily occluded and blurred by fast camera motion. In this paper, we propose to use line segments as landmarks due to the following advantages: First, such primitives are very numerous in structured environments (indoor or urban outdoor); Second, a map of line segments gives higher level of relevant information on the structure of environment; Finally, line segments are more robust to partial occlusion and more resilient to motion blur.

Though highly invariant to lighting, line segments are difficult to distinguish from each other locally, and this increases the complexity of matching line features in different images. Recently, Wang *et al*. [17] investigate the problem of automatically matching line segments only from their neighbourhood appearance, without resorting to any other constraints or priori knowledge. They design a so called mean-standard deviation line descriptor (MSLD) which is similar to the SIFT descriptor for point features. The MSLD descriptor is quite computationally expensive but very robust under rotation, illumination change, image blur, viewpoint change and partial occlusion. In our work, we adopt this approach to recover from tracking failure. During the normal SLAM procedure, a simple line tracking method based on searching for nearby lines in line parameter space replace MSLD.

Another barrier for incorporating line landmarks into a visual SLAM system is the representation of spatial lines. Because a line with 4 degrees of freedom would be a homogeneous 5-vector, this causes some problems when it is used in mathematical expressions together with the 4-vectors representing points and planes. Inspecting the related work in line or edge based visual SLAM, various approaches are taken to tackle this issue. Andrew and Walterio [1] and Smith *et al*. [13] represent a line as two endpoints, so the line measurement equations are similar to point's. The two systems are easily integrated with point features, but determining line endpoints may be difficult because of partial occlusion or fragmentation. Eade and Drummond [4], as well as Klein and Murray [9] use a 3D point corresponding with the center of line and a 3D unit vector describing the direction of line. They give a set of not compact equations to model the relationship between image line and spatial line. Lemaire and Lacroix [10] represent a line by its Euclidean Plücker coordinates. Again, the formulation of their line observation function is challenging because of the complicated camera projection matrix for a Plücker line. This representation which includes line direction and line moment is also used in [6], [15]. The relationship between Plücker line and Plücker matrix introduced by Hartley and Zisserman [7] is revealed in the sequel. Based on this relationship, a new representation of line projection function is derived.

Although point and line based SLAM systems differ in their primitives, they are similar in terms of data fusion and association. Batch optimisation techniques are popular in Structure-from-Motion (SfM), however, due to their large time consumption, they are not usually associated with real-time operation. Instead, filter optimisation techniques are favourable

in visual SLAM. The improvement of computer power facilitates the situation a little bit. Inspired by former successful applications of bundle adjustment to real-time visual odometry and tracking, Klein and Murray [8] split tracking and mapping into two separate tasks which are processed in parallel threads on a dual-core computer. This allows the use of a batch optimisation method in map building. More recently, Strasdat *et al*. [14] perform an analysis of the relative advantages of filtering and batch optimisation for sequential monocular SLAM. Their analysis shows that it is worth filtering when the tracking cost is high and the overall processing budget is small. Considering the line extraction and tracking time, we propose a batch optimisation reconstruction algorithm to reconstruct 3D lines given a set of tracked image lines but we don't update all the line landmarks by this. Instead, we take a camera motion model to predict camera states and use an Extended Kalman Filter (EKF) to update camera's and line landmarks' states similar to Davison's work [3].

Despite efforts to make tracking as robust as possible, eventual tracking failure is inevitable. Williams *et al*. [18] present a system which automatically detects and recovers from tracking failure by training a fast classifier with map point features. Klein and Murray [9] store a set of keyframes and relocalize directly from matching current frame and stored keyframes. They take the complete image as a descriptor. We follow the keyframe idea, but instead of saving images we compute and store compact angle histograms of keyframes from their Hough space. If a tracking failure is detected, we compute the correlation between angle histograms of the current image and the keyframes, and take the image pair with largest correlation value as the nearest correspondence pair, then relocalize the camera.

The next section describes image line detection and tracking. In section 3, we introduce mathematic representations of the spatial line and their relationships, then present our novel and fast line projection function. In section 4 we discuss the monocular SLAM in detail. A variety of simulation results as well as real experimental results are presented in section 5 to verify the efficiency of the system. Finally, conclusions are drawn in the last section.

II. Image Line

There is no doubt that measurement is always the vital step for a successful running system. Since we take line segments as primitives for our visual SLAM system, approaches to detect and track lines are first presented in this section.

A. Line segment detection

Line segment detection is an old and recurrent topic in computer vision. In recent work of von Gioi *et al*. [16], they propose a linear-time line segment detector which directly extracts lines from original gray image rather than from canny edge binary image. This line detector gives accurate results with a controlled number of false detections, but the real time performance still needs to be improved. Compared to von Gioi's work, Fernandes and Oliveira [5] present a line

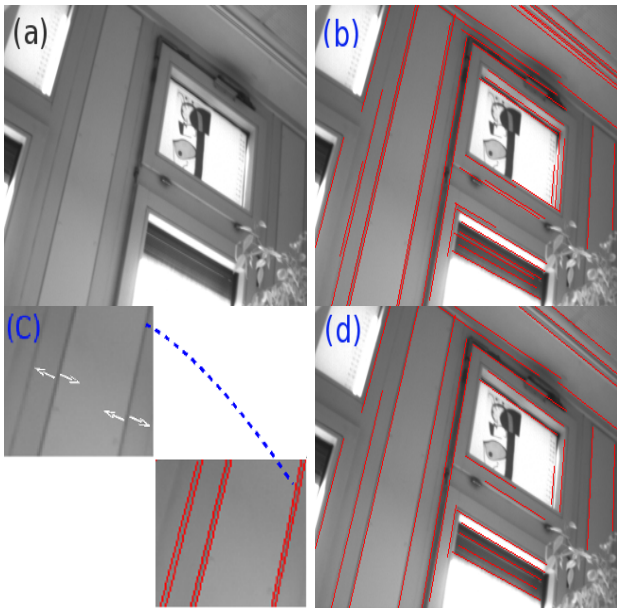

Fig. 1. Line segements detection. (a) Input image with 320x240 pixels. (b) Originally detected line segements. 41 lines are detected in total. (c) Amplification of close line pairs. Note that the gradients of close line pairs are just opposite to each other, so we keep one of close lines for each pair based on their gradients rather than arbitrarily. (d) Detected line segements after deleting redundant close lines. 28 lines are left. The total process of line detection takes 8ms per frame.

detection approach based on standard methods which first apply canny edge detector followed by a Hough Transform (HT). They improve the voting scheme of traditional HT that allows a software implementation of the algorithm to perform in real time on a desktop PC. We follow their approach: First, apply the canny edge detector to get binary image; Second, identify clusters of approximately collinear pixel segments; Then, for each cluster, use an oriented elliptical-Gaussian kernel to cast votes for only a few lines in Hough space; Last, identify peaks of votes in the Hough space to get detected lines. Instead of taking line segments as infinite lines, we compute the middle point and length of each kernel so as to get two endpoints of relevant line segment. Another modification we do is to delete redundant close lines in order to keep the reconstructed 3D lines more distinguishable in the map. Fig. 1 shows an example of detected line segments in our experimental scenery.

B. Line segment tracking

As introduced in the first section, during the normal SLAM procedure, we take a simple line tracking method based on searching nearby lines in parameter space. We name it Nearby Line Tracking (NLT). For a visual system which works at more than 25Hz, it is reasonable to assume that the correponding lines in two consecutive images have similar parameters which include: line parameters in Hough space, *i.e*. the distance from the image center to the line and the angle between the image's x-axis and the normal to the line; the length and middle point of line segment; and the average pixel gray value and gradient

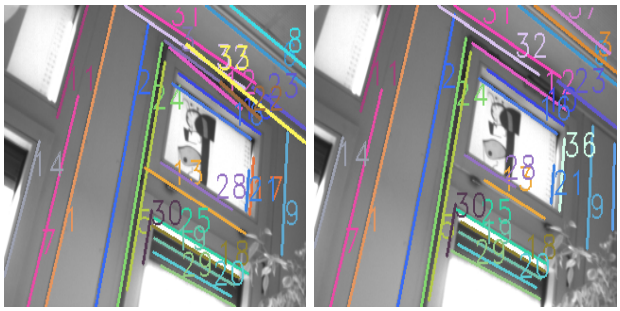

Fig. 2. Two snapshots of line tracking result of NLT method during normal SLAM procedure. The line tracking time is less than 0.2ms, and all the 24 matches shown in the same numbers with same color are correct. It's only failure to find the counterpart of line 8 on left image.

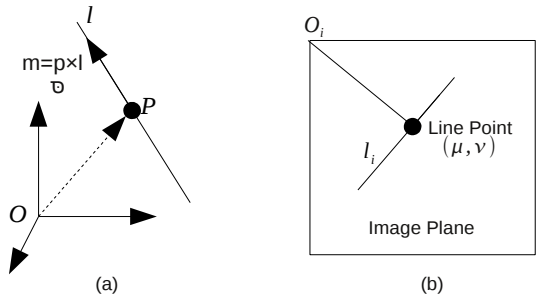

Fig. 3. Plücker line and line point. (a) Plücker line. $\mathbf{p}$ is a point on the line, and the line direction is $\mathbf{l}$. Line moment $\mathbf{m} = \mathbf{p} \times \mathbf{l}$. (b) Line point. The image origin $\mathbf{O}_i$ is the top left corner of image plane. The line point (μ, ν) is the closest point on the image line $\mathbf{l}_i$ to $\mathbf{O}_i$

value of points on line segment. The distance between these line parameters with weighting coefficients is our minimization function. For each line in the previous image, we search its nearest line in the current image according to the minimization function, if the minimal distance is smaller than a global threshold, then these two lines are taken as corresponding pair. Fig. 2 gives two snapshots of line tracking results using our NLT method during normal SLAM procedure. Admittedly, the above assumption does not always hold, especially when the SLAM procedure fails. In this situation, we use the MSLD [17] tracking approach although it is computationally expensive.

III. Spatial Line

There are many mathematical forms to represent lines in 3D space, and in this section we first give a brief introducation of two forms: Plücker line and Plücker matrix. Then we show their relationship, and based on this, we deduce our line projection function. At last, as its application, we give an algorithm to reconstruct spatial lines according to a set of observed image lines.

A. Spatial line representation

In this paper, we choose Plücker lines to represent spatial lines [19]. A Plücker line in space with direction $\mathbf{l}$ through a point $\mathbf{p}$ can be represented as

$$\hat{\mathbf{l}} = \mathbf{l} + \epsilon\mathbf{m}, \quad with \quad \epsilon^2 = 0 \quad but \quad \epsilon \neq 0, \tag{1}$$

where $\mathbf{m}$ is called the line moment and is equal to $\mathbf{p} \times \mathbf{l}$ as shown in Fig. 3(a). The line moment is normal to the plane through the line and the coordinate origin, with the magnitude equal to the distance from the line to the origin. The constraints

$$\| \mathbf{l} \| = 1, \quad \mathbf{l}^T\mathbf{m} = 0 \tag{2}$$

guarantee that the degrees of freedom of an arbitrary line in space are four.

In order to reveal the relationship between Plücker line and Plücker matrix introduced by Hartley and Zisserman [7], a brief survey of Plücker matrix is given. Here, a line is defined by the conjunction of two points or the intersection of two planes and is represented by a 4×4 skew-symmetric homogenous matrix. In particular, the line joining the two points $\mathbf{A}$ and $\mathbf{B}$ is represented by Plücker matrix $\mathbf{L}$ in vector notation as

$$\mathbf{L} = \mathbf{A}\mathbf{B}^T - \mathbf{B}\mathbf{A}^T. \tag{3}$$

A dual Plücker matrix $\mathbf{L}^*$ is obtained for a line formed by the intersection of two planes Π_1 and Π_2:

$$\mathbf{L}^* = \Pi_1\Pi_2^T - \Pi_2\Pi_1^T, \tag{4}$$

and has similar properties to $\mathbf{L}$. The matrix $\mathbf{L}^*$ can be obtained directly from $\mathbf{L}$ by a simple rewrite rule:

$$\frac{l^*_{34}}{l_{12}} = \frac{l^*_{42}}{l_{13}} = \frac{l^*_{23}}{l_{14}} = \frac{l^*_{14}}{l_{23}} = \frac{l^*_{13}}{l_{42}} = \frac{l^*_{12}}{l_{34}}, \tag{5}$$

where l_{ij} and l^*_{ij} are the i^{th} row and j^{th} column elements of matrices $\mathbf{L}$ and $\mathbf{L}^*$, respectively.

Now, suppose two 3D points $\mathbf{A}$ and $\mathbf{B}$ are represented using homogeneous coordinate as $[\mathbf{a}^T, 1]^T$ and $[\mathbf{b}^T, 1]^T$ respectively. Then, according to the definition of the Plücker matrix (3), yield

$$\begin{aligned} \mathbf{L} &= \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix} [\mathbf{b}^T \quad 1] - \begin{bmatrix} \mathbf{b} \\ 1 \end{bmatrix} [\mathbf{a}^T \quad 1] \\ &= \begin{bmatrix} -[(\mathbf{a} \times \mathbf{b})]_\times & \mathbf{a} - \mathbf{b} \\ (\mathbf{b} - \mathbf{a})^T & 0 \end{bmatrix}, \end{aligned} \tag{6}$$

where $\mathbf{a} \times \mathbf{b}$ is the cross product of two vectors and $[\mathbf{a}]_\times$ is a skew-symmetric matrix. On the other hand, according to the definition of Plücker line, we have

$$\hat{\mathbf{l}} = \frac{\mathbf{a} - \mathbf{b}}{\| \mathbf{a} - \mathbf{b} \|} + \epsilon\frac{\mathbf{a} \times (\mathbf{a} - \mathbf{b})}{\| \mathbf{a} - \mathbf{b} \|} = \frac{\mathbf{a} - \mathbf{b}}{\| \mathbf{a} - \mathbf{b} \|} - \epsilon\frac{\mathbf{a} \times \mathbf{b}}{\| \mathbf{a} - \mathbf{b} \|}. \tag{7}$$

Combine (1), (6) and (7) to obtain the relationship between Plücker matrix and Plücker line as follows

$$\mathbf{L} = \| \mathbf{a} - \mathbf{b} \| \begin{bmatrix} [\mathbf{m}]_\times & \mathbf{l} \\ -\mathbf{l}^T & 0 \end{bmatrix}. \tag{8}$$

B. Line measurement model

Before building the line measurement model, we first introduce a related lemma which is used in the derivation of our line projection function.

Lemma 1: For a vector $\mathbf{v} \in \Re^3$, and a matrix $\mathbf{Q} \in \Re^{3\times 3}$,

$$\mathbf{Q}[\mathbf{v}]_\times \mathbf{Q}^T = \frac{1}{\mid \mathbf{U} \mid \cdot \mid \mathbf{V} \mid}[(\mathbf{U}\bar{\mathbf{D}}\mathbf{V})\mathbf{v}]_\times. \tag{9}$$

$(\mathbf{U}, \Lambda, \mathbf{V})$ are the Singular Value Decomposition (SVD) results of $\mathbf{Q}$, $\mid \mathbf{U} \mid$ means the determinant of matrix $\mathbf{U}$, and $\bar{\mathbf{D}}$ is computed from the three eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of $\mathbf{Q}$:

$$\bar{\mathbf{D}} = \begin{bmatrix} \lambda_2\lambda_3 & 0 & 0 \\ 0 & \lambda_1\lambda_3 & 0 \\ 0 & 0 & \lambda_1\lambda_2 \end{bmatrix}. \tag{10}$$

Especially, if $\mathbf{Q}$ is an orthogonal matrix, then

$$\mathbf{Q}[\mathbf{v}]_\times \mathbf{Q}^T = \frac{1}{\mid \mathbf{Q} \mid}[\mathbf{Q}\mathbf{v}]_\times. \tag{11}$$

We now turn to the forward projection of lines. The camera projective matrix is $\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I} \mid -\mathbf{c}]$, where $\mathbf{K}$ is the camera calibration matrix, $\mathbf{R}$ is rotation matrix representing the orientation of the camera coordinate frame and $\mathbf{c}$ represents the coordinates of the camera center in the world coordinate frame. A line in 3-space represented as a Plücker matrix $\mathbf{L}$ is imaged as the line $\mathbf{l}_i$, where [7]

$$[\mathbf{l}_i]_\times = \mathbf{P}\mathbf{L}\mathbf{P}^T. \tag{12}$$

Substituting the relationship between Plücker line and Plücker matrix (8) into Plücker matrix mapping equation (12), yield

$$\begin{aligned} [\mathbf{l}_i]_\times &\simeq \mathbf{K}\mathbf{R}[\mathbf{I} \mid -\mathbf{c}] \begin{bmatrix} [\mathbf{m}]_\times & \mathbf{l} \\ -\mathbf{l}^T & 0 \end{bmatrix} (\mathbf{K}\mathbf{R}[\mathbf{I} \mid -\mathbf{c}])^T \\ &= \mathbf{K}\mathbf{R}\left[\mathbf{m} + [-\mathbf{c}]_\times \mathbf{l}\right]_\times \mathbf{R}^T\mathbf{K}^T \\ &\simeq \mathbf{K}\left[\mathbf{R}\left(\mathbf{m} + [-\mathbf{c}]_\times \mathbf{l}\right)\right]_\times \mathbf{K}^T \\ &\simeq \left[(\mathbf{U}\bar{\mathbf{D}}\mathbf{V})\mathbf{R}(\mathbf{m} + [-\mathbf{c}]_\times \mathbf{l})\right]_\times, \end{aligned} \tag{13}$$

where $\simeq$ means two sides are equal up to a scale, $(\mathbf{U}, \Lambda, \mathbf{V})$ are the SVD results of the camera calibration matrix $\mathbf{K}$, and $\bar{\mathbf{D}}$ is calculated by (10). The third equation is based on equation (11) and the last equation is based on equation (9). Finally, we get the Plücker line mapping equation which projects a spatial line into image plane as

$$\mathbf{l}_i \simeq (\mathbf{U}\bar{\mathbf{D}}\mathbf{V})\mathbf{R}(\mathbf{m} + [-\mathbf{c}]_\times \mathbf{l}). \tag{14}$$

Notice the difference between equation (13) and equation (14): the former is in matrix form but the later directly gives the image line in vector form. To the author's knowledge, the Plücker line projection function (14) is the most compact representation to project a spatial line to an image line under the perspective camera model. Taylor and Kriegman [15] and Goddard [6] give a similar line projection function derived from rigid transformation, however, their camera model only takes camera's focal length as its internal parameter and ours is their generalization. Comparing to Plücker matrix mapping equation (12) or line observation function introduced by Lemaire and Lacroix [10], though they are mathematically equivalent with ours, using equation (14) to compute image line $\mathbf{l}_i$ is much faster than using equation (12) when the camera calibration matrix $\mathbf{K}$ is known and coefficient matrix $\mathbf{U}\bar{\mathbf{D}}\mathbf{V}$ is pre-computed. It is important to compute the line projection function fast because it will be called during each iteration of the filter for each line. To test our result, we run equation (12) and equation (14) 10^6 times respectively, and the later is 15 times faster than the former. Another advantage of our line projection function is that it is easier to generate the Jacobian matrix of image line w.r.t. camera rotation and position parameters for EKF because it is less nonlinear than equation (12).

Since an image line is restricted to the image plane, only two degrees of freedom are present for each measured line segment. In our work, we choose the closest point on the measured line segment to the image origin as observation, which is named line point and is similar to Goddard's [6] definition. Fig. 3(b) illustrates the definition of the line point on the image plane. For the observed value, we can directly measure the line point from the input image. For the predicted value, we first use the line projection function (14) to predict image line $\mathbf{l}_i = (l_x, l_y, l_z)^T$, then use line point equation (15) to get the predicted line point (μ, ν).

$$\mu = -l_x l_z l_x^2 + l_y^2, \quad \nu = \frac{-l_y l_z}{l_x^2 + l_y^2}. \tag{15}$$

Combine (14) and (15) to obtain our line measurement model which is used for spatial line reconstruction and system state update.

C. Spatial line reconstruction

Hartley and Zisserman [7] introduce a way to reconstruct spatial lines from two views. For a pair of corresponding image lines in two views, first, back-project lines to get two planes Π_1 and Π_2; second, according to the definition of the dual Plücker matrix (4) get $\mathbf{L}^*$; then according to the relationship between Plücker matrix and dual Plücker matrix (5) get $\mathbf{L}$; at last, according to the relationship between Plücker Matrix and Plücker line (8), we get the initial estimate of corresponding spatial line.

In order to obtain a more reliable and accurate reconstruction of a spatial line, a robust spatial line reconstruction algorithm is employed, which is similar to Bleser's [2] idea for robust point feature triangulation. The basic idea of the algorithm is that: First find two views which satisfy the reconstruction conditions, *i.e.* the baseline length and view angle between this two views are larger than thresholds; Then use the aforementioned two-view reconstruction approach to get the initial estimation of the spatial line; Next, check the consensus of views and remove outliers; At last, use Gauss-Markov Model (GMM) estimator [11] to get the final estimation. Alg. 1 gives the process in pseudo-code form. Assuming that a set of camera views $\{\mathbf{C}_j\}_{j=1}^m$ are associated to the projections of line $\hat{\mathbf{l}}$, the reconstruction algorithm can

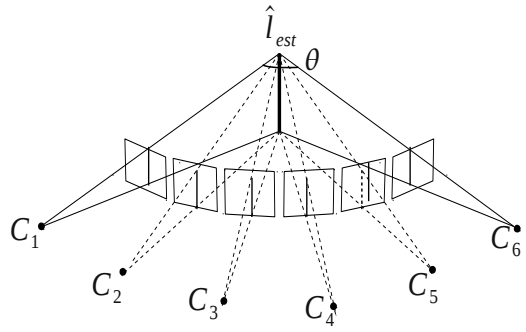

Fig. 4. Robust spatial line reconstruction. A set of camera views is used for robust reconstruction. Here, the baseline length and view angle between views $\mathbf{C}_1$ and $\mathbf{C}_6$ satisfy the reconstruction conditions, so the two-view reconstruction algorithm is used to get the initial estimation of line $\hat{\mathbf{l}}$, and the result is consistent with views $\mathbf{C}_2$, $\mathbf{C}_3$ and $\mathbf{C}_4$. Camera view $\mathbf{C}_5$ is an outlier and the other views are used to obtain the final GMM estimation $\hat{\mathbf{l}}_{est}$.

be used to obtain a GMM estimation $\hat{\mathbf{l}}_{est}$ of the spatial line. An example configuration is shown in Fig. 4.

Algorithm 1 Robust spatial line reconstruction

Require: A set of camera views $\{\mathbf{C}_j\}_{j=1}^m$

$ret \leftarrow 0$
for $j \leftarrow 2$ to m **do**
  $\theta \leftarrow$view angle between $\mathbf{C}_j$ and $\mathbf{C}_1$
  $\lambda \leftarrow$baseline length between $\mathbf{C}_j$ and $\mathbf{C}_1$
  if $(\theta > \theta_{th})$ and $(\lambda > \lambda_{th})$ **then**
    $\hat{\mathbf{l}}'_{est} \leftarrow$initial estimation obtained by reconstructing line from two views $\mathbf{C}_j$ and $\mathbf{C}_1$;
    $\mathcal{I} \leftarrow$set of inlier views which are consistent with $\hat{\mathbf{l}}'_{est}$ in $\{\mathbf{C}_j\}_{j=1}^m$;
    $\alpha \leftarrow$ ratio between inlier view number and total view number;
    if $\alpha > \alpha_{th}$ **then**
      $\hat{\mathbf{l}}_{est} \leftarrow$final estimation obtained by GMM estimator which takes $\hat{\mathbf{l}}'_{est}$ and $\mathcal{I}$ as inputs;
      $ret \leftarrow 1$;
      break;
    end if
  end if
end for
return ret, $\hat{\mathbf{l}}_{est}$

IV. Methodology of Monocular SLAM

As mentioned in the introduction, despite the difference of their primitives, different kinds of monocular SLAM systems share the same approach on data fusion and association. This section presents the issues and relevant solutions for a complete and robust SLAM system.

A. System initialization

Davison *et al*. [3] choose to aid system start-up with a small amount of prior information about the scene which makes the scale of the map certain. Klein and Murray [8] employ the five-point stereo algorithm to initialise the system and their initial map has an arbitrary scale. Since knowing the scale of the map is desirable whenever it must be related to other information or be stored for future use, we take an approach which is similar to Davison's. Normally it is easy to find a corner which is the intersection of three walls in a room, then it must be the intersection of three lines which are made by each of two walls. Actually, the corner and three lines make a natural coordinate frame. We take this frame as our world coordinate frame, then we get three lines with known directions and moments. In section (III-B) we present the line measurement model. Using these three correspondences between image lines and spatial lines can estimate camera rotation, but the position is up to scale. So we should add another spatial line with known direction and moment into the map to determine the scale. User cooperation is required at the beginning to assign the matching between these four spatial lines and their projection in the image. Then we use a GMM estimator to get the initial value of camera rotation and position.

B. State prediction and update

A probabilistic map is used to represent the estimates of the states of the camera and all reconstructed line landmarks as well as their uncertainties. Explicitly and mathematically, the camera's states $\mathbf{x}_{cam}$ comprise a metric 3D position $\mathbf{c}^w$, orientation in Euler angle form Ω^w, velocity $\mathbf{v}^w$ and angular velocity ω^b_{wb} relative to world frame w and camera body frame b: $\mathbf{x}_{cam} = [\mathbf{c}^w, \Omega^w, \mathbf{v}^w, \omega^b_{wb}]^T$. For states of reconstructed line landmarks, each line landmark $\mathbf{x}^i_l$ is represented by a direction vector $\mathbf{l}^i$ together with a moment vector $\mathbf{m}^i$ in Plücker line form: $\mathbf{x}^i_l = [\mathbf{l}^i, \mathbf{m}^i]^T$.

After system start-up, the state vector goes through two alternating steps: first, prediction when the camera moves in the interval between image process, then update after measurements have been observed.

For the prediction step, a camera motion model is employed which is similar to Davison's [3]. Here, Euler angles replace quaternion to represent rotation which are more suitable for using (14) to compute Jacobian matrix. The constant velocity and angular rate model assume that the unknown acceleration $\mathbf{a}^w$ and angular acceleration α^b_{wb} are zero mean and Gaussian distributed noise. The discrete camera motion model is:

$$\begin{bmatrix} \mathbf{c}^w_{(k+1)} \\ \Omega^w_{(k+1)} \\ \mathbf{v}^w_{(k+1)} \\ \omega^b_{wb(k+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{c}^w_{(k)} + \Delta t \mathbf{v}^w_{(k)} + \frac{\Delta t^2}{2}\mathbf{a}^w \\ \Omega^w_{(k)} + \mathbf{R}^w_{b(k)}(\Delta t \omega^b_{wb(k)} + \frac{\Delta t^2}{2}\alpha^b_{wb}) \\ \mathbf{v}^w_{(k)} + \Delta t \mathbf{a}^w \\ \omega^b_{wb(k)} + \Delta t \alpha^b_{wb} \end{bmatrix} \tag{16}$$

where, $\mathbf{R}^w_b$ is the rotation matrix and Δt is the prediction period. Since we assume the environment is static, states of line landmarks stay unchanged during prediction procedure.

After observation, EKF is used to update both the system states and their uncertainties. The line measurement model is presented in section (III-B), and we omit the calculation of the Jacobian matrix here. Notice that the computation complexity

of the EKF algorithm is order $O(N^2)$, where N is the number of landmarks. So the number of landmarks which can be maintained with real time processing is bounded. To improve computation efficiency, we only choose those lines which are parallel with the world coordinate axes to add into the map. Further, the directions of line landmarks are not updated by EKF, which almost cut in half the size of the state vector. This assumption is reasonable and feasible when there are enough spatial lines which are parallel with the world coordinate axes such as in indoor environments.

C. SLAM supervision and failure recovery

Failure detection and recovery are important parts of the overall algorithm to make it robust. In our work, we assume the SLAM procedure is failing when either of the following two situations are detected: First, for twenty consecutive observations, the number of measured matches between spatial lines in the map and their projections in the image is less than four in each observation; Second, the movement between current updated camera position and previous updated camera position is larger than a threshold. If the SLAM procedure is failing, then a recovery approach should be taken to relocalize the system on-the-fly. Our failure recovery algorithm is based on the angle histograms of keyframes. The angle histogram computes the image orientation of all lines in an image, which is a good indicator of the image similarity.

During the normal SLAM procedure, after each observation update, we check whether the current frame is far enough away from already stored keyframes, and whether the number of measured matches between spatial lines in the map and their projections in the current image is larger than a threshold. If both conditions are satisfied, then the current frame is taken as keyframe, and its angle histogram together with current matches, detected lines and their MSLD descriptors, and camera states are stored. Compared to Klein's recovery method [9], we don't need to store the whole image and this saves some memory. When failure of the SLAM procedure is detected , the normal prediction and update process is stopped to prevent the estimated map from being corrupted by a false update. Instead, we compute the angle histogram of the current image and then compute the correlations between angle histograms of the current image and stored keyframes, and find the largest correlation value. If this value is larger than a threshold then we use MSLD approach to track the lines between the current image and its nearest keyframe image and build the matchings between spatial lines and their projections in the current image. If the number of successful matches is larger than three, then as in the initialization method in section (IV-A), we use a GMM estimator to relocalize the camera and normal SLAM procedure is re-started.

V. Experimental results

To demonstrate that our monocular SLAM system is robust to illumination change, partial occlusion and fast camera motion, this section will test it using both synthetic and real image sequences. The following experiments are performed on a 2.4GHz Intel(R) Core(TM)2 processor with 4 GB of RAM.

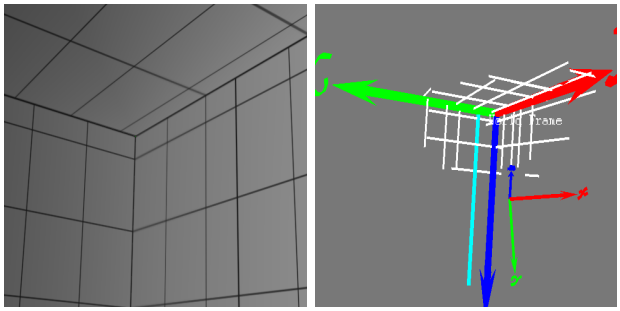

Fig. 5. Snapshot of synthetic scenery and map generated during SLAM procedure. The final map includes 25 landmarks generated from 4679 image frames and the processing rate is about 27 frames per second. The procedure is illustrated in the accompanying video.

A. Synthetic data experiments

In synthetic experiments, we build a 3D model of a simple room with some line landmarks on the wall, then use our animator tool to generate a snapshot sequence with perfectly known camera positions and rotations. The ground truth can be used to evaluate the estimated results of our system. Fig. 5 illustrates the snapshot of the room and the generated map. Notice that the endpoints of line landmarks are simply reconstructed by back-projecting the endpoints of line segments in the image to the reconstructed spatial lines, and after each observation, we update them if necessary. Fig. 6 gives the ground truth of camera position and rotation angle in X direction of each frame and their estimated values, as well as the estimated errors. The ranges of the camera translation in X direction is $[3.17m, 7.27m]$ and the rotation around X axis is $[-134.27°, -98.14°]$, respectively. The statistical values of estimated position and rotation angle errors are listed in Tab. I. We also test the robustness of our approach to image noise and Fig. 7 shows that the average estimated position and rotation angle errors do not increase with the increased image noise level. Here, image noise level is defined as the ratio of noise variance and image intensity variance.

TABLE I
THE MAXIMAL AND MEAN ESTIMATED POSE ERRORS.

	Max(m)	Mean(m)		Max(°)	Mean(°)
PosErrX	0.380	0.046	AngErrX	4.600	0.366
PosErrY	0.235	0.026	AngErrY	2.035	0.228
PosErrZ	0.243	0.014	AngErrZ	3.929	0.569

B. Real data experiments

We perform our real data experiment with a hand-held camera moving in the office with size of $5m \times 3.5m \times 3m$. The camera starts at one corner of the room and after initialization, it gradually explores half of the room. Fig. 8 shows one image of the environment and the reconstructed map. During the SLAM procedure, we also demonstrate the failure recovery.

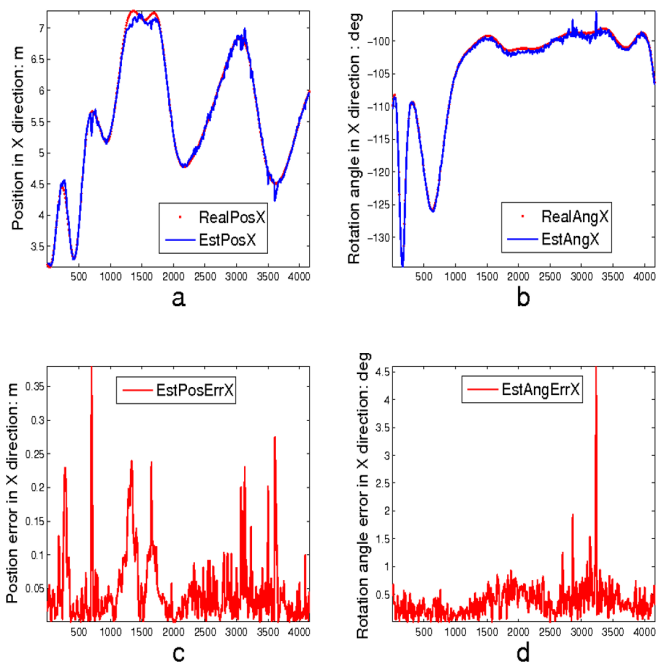

Fig. 6. Comparison of the ground truth and estimated values. The upper two graphs (a) and (b) give the ground truth and estimated results of camera position and rotation angle in X direction, and the lower two graphs (c) and (d) show the estimated errors in X direction. Notice that the pose estimation error increases in areas of fast motion changes, but the system is able to track the pose very well. The other parameters show similar behaviour.

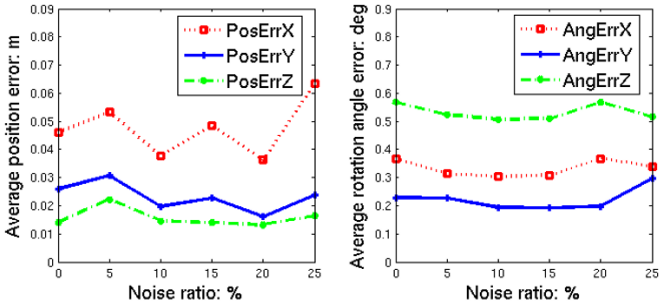

Fig. 7. The relationship between the mean estimated position and rotation angle errors and image noise level. The error is not affected by severe noise levels up to 25%.

When the camera moves fast to some new place with too few reconstructed line landmarks being observed, SLAM failure is detected. The system automatically recovers when an old view reappears in the visual field. Fig. 9 shows the process from normal SLAM to failure and then to recovery.

To illustrate that the system is robust to partial occlusion and illumination change, we move some objects in front of the camera during the SLAM procedure as shown in Fig. 10. Actually, since a camera motion model is employed, the effect of occlusion is reduced when the error of estimated camera velocity and angular velocity is relatively small. And the nature of the line landmark further improves the robustness of our approach to aforementioned image changes.

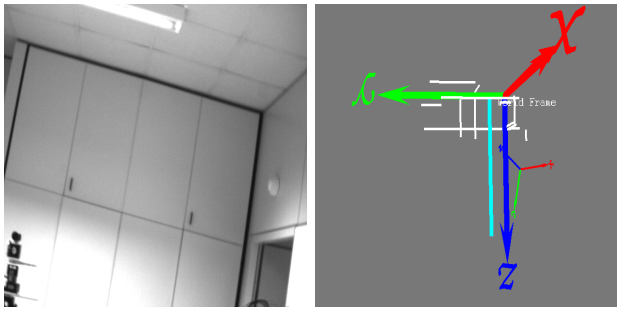

Fig. 8. One image of room environment and generated map. The final map includes 18 line landmarks generated from 10306 frames. The processing rate is more than 27Hz.

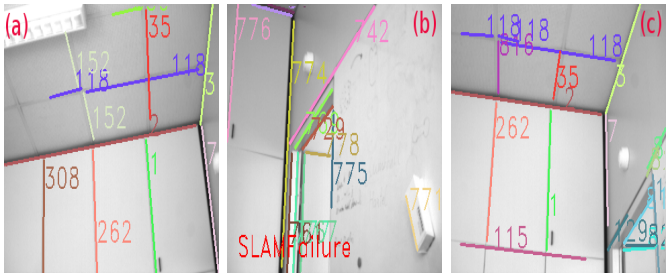

Fig. 9. Illustration of failure recovery. (a) shows normal tracking. (b) shows SLAM failure is detected and then system is recovered as shown in (c). Notice that lines 1, 2, 3, 7, 35, 118 and 262 are found again after recovery.

VI. Conclusion

We present a hand-held monocular SLAM system based on line segments. Due to the nature of line features, the generated map gives higher level of geometrical information of environment than a map with point features, and the system is robust to illumination change, partial occlusion and fast camera motion. One of the important contributions of our work is the derivation of a novel line projection function which is compact, fast, and allows to easily compute the Jacobian matrix for EKF under the perspective camera model. Another contribution is the failure recovery method based on the angle histogram of keyframes which is quick to recover and also saves some memory. In addition, we also benefit from the form of Plücker line to represent spatial lines which splits a line into direction and moment. This makes it reasonable and feasible to only update the line moment in the EKF and to cut in half the dimension of the state vector for lines which are parallel with the world coordinate axes.

Nevertheless, due to the complexity of the EKF method, the size of the map is bounded. Future work will attempt to combine some efficient map management algorithms in order to expand its application. There is another open question: for non-perspective cameras which suffer from severe radial distortion, such as a fisheye camera, the line projection function is still unclear because generally, the projection of a spatial line is a curve in the image. Undistorting the image first is the current way to tackle this problem. It worth to find the line projection function under more general camera model.

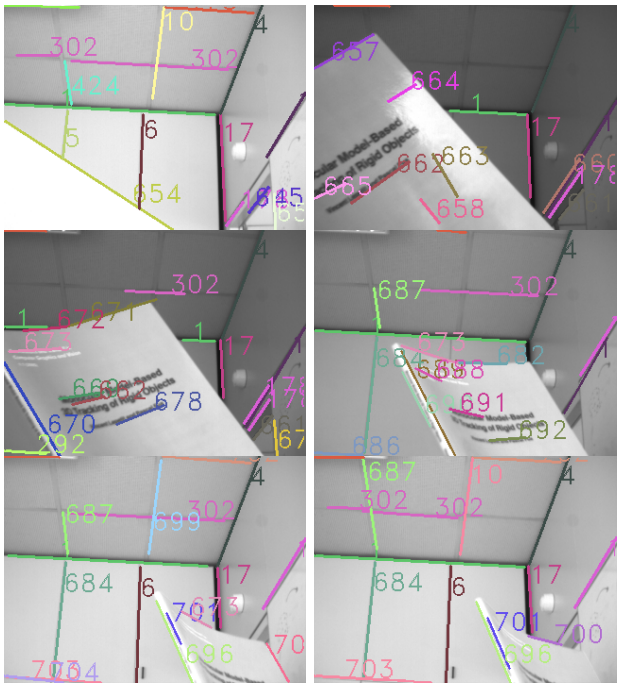

Fig. 10. Illustration of occlusion and illumination change. A book moves in front of the camera and line landmarks 1, 4, 6, 17, and 302 are kept tracking whenever they are viewed.

ACKNOWLEDGMENT

This work was supported by German Science Foundation DFG KO-2044/6-1 and China Scholarship Council (No.2009611008). Special thanks to the members of the Kiel Multimedia Information Processing group and Associate Prof. Yuanxin Wu, the author of National Excellent Doctoral Dissertation of PR China (FANEDD 200897), for many helpful discussions and software assistance.

REFERENCES

[1] G. Andrew and M. C. Walterio. Real-time model-based slam using line segments. *Advances in Visual Computing*, pages 354–363, 2006. 1

[2] G. Bleser. *Towards visual-inertial SLAM for mobile augmented reality*. PhD thesis, University of Kaiserslautern, 2009. 4

[3] A. Davison, I. Reid, N. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29:1052–1067, 2007. 1, 2, 5

[4] E. Eade and T. Drummond. Edge landmarks in monocular slam. In *2006 British Machine Vision Conference*, pages 469–476, 2006. 1

[5] L. A. Fernandes and M. M. Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern Recognition*, 41:299–314, 2008. 2

[6] J. S. Goddard. *Pose and motion estimation from vision using dual quaternion-based extended kalman filtering*. PhD thesis, University of Tennessee, 1997. 1, 4

[7] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision (Second Edition)*. Cambridge University Press, second edition, 2004. 1, 3, 4

[8] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007 IEEE and ACM International Symposium on*, pages 225–234, 2007. 1, 2, 5

[9] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *2008 European Conference on Computer Vision*, 2008. 1, 2, 6

[10] T. Lemaire and S. Lacroix. Monocular-vision based slam using line segments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2791–2796, 2007. 1, 4

[11] J. C. McGlone, W. Foerstner, and B. Wrobel. *Manual of Photogrammetry, chapter 2 Mathematical Concepts in Photogrammetry*. asprs, 2004. 4

[12] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Monocular vision based slam for mobile robots. In *Pattern Recognition, 2006 International Conference on*, pages 1027–1031, 2006. 1

[13] P. Smith, I. Reid, and A. Davison. Real-time monocular slam with straight lines. In *2006 British Machine Vision Conference*, volume 1, pages 17–26, 2006. 1

[14] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Real-time monocular slam: Why filter? In *Robotics and Automation, 2010 IEEE International Conference on*, 2010. 2

[15] C. J. Taylor and D. J. Kriegman. Structure and motion from line segments in multiple images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17:1021–1032, 1995. 1, 4

[16] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32:722 –732, 2010. 2

[17] Z. Wang, F. Wu, and Z. Hu. Msld: A robust descriptor for line matching. *Pattern Recognition*, 42:941–953, 2009. 1, 3

[18] B. Williams, G. Klein, and I. Reid. Real-time slam relocalisation. In *Computer Vision, 2007 IEEE International Conference on*, pages 1–8, 2007. 2

[19] Y. Wu, X. Hu, D. Hu, T. Li, and J. Lian. Strapdown inertial navigation system algorithms based on dual quaternions. *Aerospace and Electronic Systems, IEEE Transactions on*, 41:110–132, 2005. 3