Minimal Non-linear Camera Pose Estimation Method Using Lines for SLAM Applications

Yu Cao
Beihang University, China
cqcy1208@buaa.edu.cn

Haishu Tan
Foshan University, China
tanhs@163.com

Fuqiang Zhou
Beihang University, China
zfq@buaa.edu.cn

Abstract

In order to continuously estimate camera pose with known line features correspondences between 3D lines in the real world and 2D lines in the image plane, we present a novel non-linear optimization method utilizing Plücker coordinates and a minimal representation of rigid motion. Inspired by the bundle adjustment pose estimation method, we use a minimal 6 Degree of Freedom (DoF) vector to denote rigid motion based on the Lie Algebra and Lie group theory. For the first time, we deduct the Jacobian matrix of the line's Plücker coordinates over the motion vector. Thus we are able to optimize the reprojection error to the minimal to find the solution with all the orthogonormality contraints fully considered. Benefited from the use of non-redundant representation of 6-DoF motion, our method requires only at least 3 lines correspondences, which makes our method applicable with limited matching pairs. Experiments in both simulation and real world images show that our method is fast, accurate, robust and suitable for motion-only Bundle Adjustment pose estimation in SLAM applications.

1. Introduction

Camera pose estimation is the key task in many computer vision applications such as Structure from Motion(SfM) [1–3], Robot Localization [4] and Visual Simultaneously Locating And Mapping (VSLAM) [5–7]. To estimate the camera pose, the known correspondences of the real world features with their projections into the image plane are ultized. Features can be in the form of points or lines. Pose estimation using point features are called PnP problem. It is well solved by the Effective Perspective-n-Points (EPnP) method proposed by Lepetit et al. [8]. Lines are significant features enclosing high-level information in an image, which occur frequently in real world images [9]. Nowadays, lines features are learned progressively to estimate camera pose, which is the so-called Perspective-n-Lines (PnL) problem shown in Fig.1 . 3D lines in world frame are first transformed to the camera frame. Then the lines are mapped to image's 2D lines. The solution of the PnL problem is the 6 DoF pose estimation which reprojects the lines in 3D space onto the 2D image plane properly.

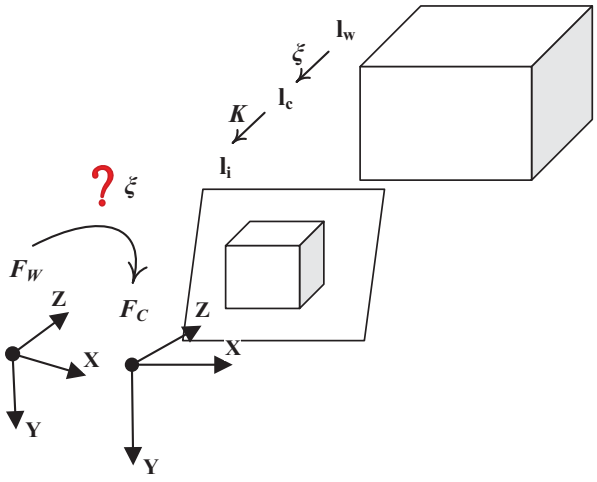


Figure 1. Perspective-n-Lines (PnL) problem finding a pose to validate 3D-2D line matches

Recently, remarkable progress has been made in the following 3 kinds of solutions to the PnL problem, namely Direct Linear Transform (DLT), polynomial solution and iterative solution.

Direct Linear Transform is the intuitive way to solve the problem as it directly solve the reprojection equations with Singular Value Decomposition (SVD). Then the solved variables are used to get an estimated pose. P?ibyl et al. has well implemented the DLT method based on lines' Plcker coordinates [10]. However, such method treats the 9 variables in the rotation matrix and the 3 variables in the translation vector as independent variables. This ignores the nonlinear orthonormal constraints of the variables. For example, the rotation matrix has 3 unit vectors orthogonal to each other. Therefore, a polynomial solution to the PnL problem merges the projection functions with the orthonormal-

978-1-5386-4886-5/18/$31.00 ©2018 IEEE
DOI 10.1109/WACV.2018.00109

ity constraints to form high-order equations. Usually, many more coefficients are introduced to help solve the high-order equations [11]. Xu et al. have made a complete analysis of a series of polynomial solutions [12]. Both DLT and polynomial methods parameterized the rigid motion with a redundant representation of at least 12 variables with multiple solutions. Additional validation step is compulsory to determine the final pose.

Another solution is to use non-linear method which iterates to minimize the reprojection error. Iterative method dates back to 1994 when Kumar proposed a non-linear optimization method optimizing the distances between the points and the reprojected image lines [13]. Nevertheless, the method treated a 3D line as 2 endpoints, which added the DoF since two 3D points have 6 DoF while a 3D line has 4 DoF. And the paper didn't figure out a way to strictly apply the gradient descendent method. What's more, to the best of our knowledge, existing iterative solutions still require a SVD [14]. And they don't fully utilize the orthonormality constraints lay in the Lie Group. Thereafter, we develop a complete iterative method that parameterizes rigid motion representation with a minimal 6 DoF vector in Lie Algebra to fully meet the orthonormality constraints [15]. The difficulty lies in how to calculate the incremental update and how to update the estimation. Motion-only Bundle Adjustment method coping with Perspective n Points (PnP) problem has been applied widely and successfully in the SLAM field [16]. The most successful open-source ORB-SLAM system [17] also applies this method to achieve a good pose estimation. It inspired us to model the whole system in the Lie Group instead of Euclidean space to solve the iterative problem.

In general, the main contribution of this paper is as follows.

1) We analyze the partial derivative (Jacobian matrix) of the lines' reprojection error over the 6-DoF motion vector in the tangent space of Lie Group. This creates the foundation for iterative method to update the estimation.

2) A new non-linear pose estimation method based on line features is proposed, which works with even only 3 pairs of lines' correspondences. The framework is very suitable for continuously updating the pose of a camera, which is usually the case in the Visual SLAM applications. Experiments on synthetic data and real world images have proven the accuracy and robustness of our algorithm.

The remainder of this paper is organized as follows: Section 2 presents the notations and the coordinates in our work. Section 3 introduces the math model and our iterative method in detail. Section 4 shows the experiments and the analysis of the results. At last, Section 5 concludes the paper.

2. Notation and Coordinates Systems

Lines in 3D space are represented by the homogeneous plücker coordinates, as the way introduced by P?ibyl et al. [18]. As Fig.2 depicts, two endpoints in homogeneous $\boldsymbol{A}(a_x, a_y, a_z, a_w)$ and $\boldsymbol{B}(a_x, a_y, a_z, a_w)$ makes a line $\boldsymbol{l}$. The plücker coordinates consist of two 3-by-1 vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. Fig.2 shows that $\boldsymbol{u}$ is the normal vector of the plane formed by the origin of the frame and the line. And $\boldsymbol{v}$ is the vector indicating the lines direction. They are calculated as equation (1) accordingly. Also constraint (2) that normal vector and direction vector are perpendicular must be satisfied, making the line of 4 DoF.

$$\boldsymbol{l} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} (a_x, a_y, a_z)^\top \times (b_x, b_y, b_z)^\top \\ a_w(b_x, b_y, b_z)^\top - b_w(a_x, a_y, a_z)^\top \end{bmatrix} \tag{1}$$

$$\boldsymbol{u}^\top \boldsymbol{v} = 0 \tag{2}$$

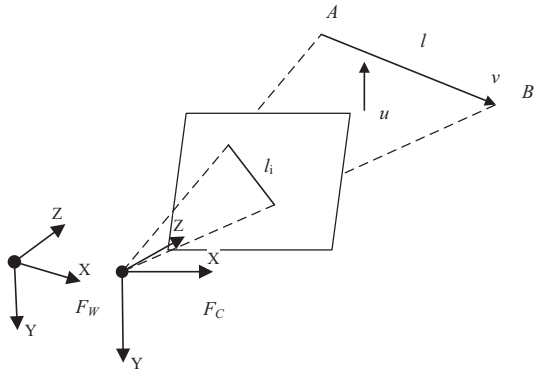


Figure 2. 3D line representation in Plücker coordinates

Then, the minimal representation of the cameras rigid motion is introduced. The camera pose is the rigid motion from the reference world frame $\boldsymbol{F_w}$ to the camera frame $\boldsymbol{F_c}$. The rigid motion comprises of 3 DoF rotation and 3 DoF translation. Equation (3) shows the coordinates transformation by the motion matrix $\boldsymbol{T_{wc}}$. $\boldsymbol{R}$ is the 3-by-3 rotation matrix indicating the camera's orientation. And $\boldsymbol{t}$ is the 3-by-1 translation vector showing the 3D position of the camera. The tilde symbolizes homogeneous cooridnate in the form of $(x, y, z, 1)^\top$.

$$\tilde{\boldsymbol{P_c}} = \boldsymbol{T_{wc}}\tilde{\boldsymbol{P_w}}, \boldsymbol{T_{wc}} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{bmatrix} \tag{3}$$

However, the rigid motion has only 6 DoF, meaning the 12 parameters of matrix $\boldsymbol{T}$ has 6 redundant degrees of freedom. So a Lie Algebra $se3$ representation of the rigid motion in exact 6 DoF as a vector $\boldsymbol{\xi}$ is used. In equation (4), $\boldsymbol{\rho}$ denotes the 3-by-1 vector indicating translation while $\boldsymbol{\theta}$ is

the 3-by-1 vector concerning rotation. The 6D vector is projected to the matrix T through 2 steps. First, the hat function $(\cdot^\wedge)$ transforms the 6 DoF vector into a 4-by-4 matrix. The $[\boldsymbol{\theta}]_\times$ is the 3-by-3 skew-symmetric matrix constructed from vector $\boldsymbol{\theta}$. Then the result is projected to the rigid motion matrix using an exponential function shown in equation (5). In the projection equations, $\boldsymbol{G_i}$ is the i-th generator matrix in the tangent space [16].

$$[\boldsymbol{\xi}]^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\theta} \end{bmatrix}^\wedge = \begin{bmatrix} [\boldsymbol{\theta}]_\times & \boldsymbol{\rho} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \sum_i \xi_i \boldsymbol{G_i} \tag{4}$$

$$\boldsymbol{T}(\boldsymbol{\xi}) = \boldsymbol{exp}([\boldsymbol{\xi}]^\wedge) = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\xi}) & \boldsymbol{t}(\boldsymbol{\xi}) \\ \mathbf{0}^\top & 0 \end{bmatrix} \tag{5}$$

As Fig.2 shows, the 3D lines can be transformed from the world frame to the camera frame through the line motion projection matrix defined as equations (6,7). In the equations, $\boldsymbol{\xi}$ is the 6D representation of the rigid camera pose. $\boldsymbol{R}$ and $\boldsymbol{t}$ are its associated rotation matrix and translation vector. The upper 3-by-6 matrix $\boldsymbol{P_u}$ projects the line coordinates to the normal vector $\boldsymbol{u_c}$ while the $\boldsymbol{P_v}$ brings about the new $\boldsymbol{v_c}$ in the camera coordinates.

$$\boldsymbol{l}_c = \begin{bmatrix} \boldsymbol{u}_c \\ \boldsymbol{v}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & [\boldsymbol{t}]_\times \boldsymbol{R} \\ \mathbf{0}^\top & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_w \\ \boldsymbol{v}_w \end{bmatrix} \tag{6}$$

$$\boldsymbol{l}_c = \boldsymbol{P}(\boldsymbol{\xi}) \boldsymbol{l}_w = \begin{bmatrix} \boldsymbol{P_u} \\ \boldsymbol{P_v} \end{bmatrix} \boldsymbol{l}_w \tag{7}$$

At last, the pinhole camera model is used to map the 3D lines onto the 2D image plane. From the geometry of computer vision [19] we know that the image line is determined by the normal vector $\boldsymbol{u_c}$ and the camera intrinsics as equations(8). $\boldsymbol{K}$ is the camera intrinsic parameters matrix. Since the homogeneous coordinates equation is up to scale, we use projection function (9) to deal with equation free of scale factor. The non-tilde $\boldsymbol{l}$ is in the projected non-homogeneous coordinates.

$$\tilde{\boldsymbol{l}}_{\boldsymbol{i}} \cong \boldsymbol{K}^{-\top} \boldsymbol{u_c} \tag{8}$$

$$\boldsymbol{l}_i = proj(\tilde{\boldsymbol{l}}_{\boldsymbol{i}}) = \frac{1}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \begin{bmatrix} \tilde{\boldsymbol{l}}_{\boldsymbol{i}}(1) \\ \tilde{\boldsymbol{l}}_{\boldsymbol{i}}(2) \end{bmatrix} \tag{9}$$

3. Non-linear optimization method

Now we can state our optimization method as follows. Given a set of 2D lines $\boldsymbol{l_i}$ on the image plane and their corresponding 3D line $\boldsymbol{l_w}$ in world coordinates, we find a pose with which the reprojection error is minimum. A pose is denoted by a 6 DoF vector $\boldsymbol{\xi}$, and its line motion projection matrix is used to propose the prediction of the lines on the image plane. In our method, we estimate the pose by iteratively minimizing the distance, which is the residual error between the calculated line prediction $\check{\boldsymbol{l}}_{\boldsymbol{i}}$ and its measurement $\boldsymbol{l_i}$ in the real image as shown in equation (10).

$$\arg\min_{\xi} \sum \|\boldsymbol{l_i} - \check{\boldsymbol{l}}_{\boldsymbol{i}}(\boldsymbol{\xi})\|^2 \tag{10}$$

$$\check{\boldsymbol{l}}_{\boldsymbol{i}}(\boldsymbol{\xi}) = proj(\boldsymbol{K}^{-\top} \boldsymbol{P_u} \boldsymbol{l_w}) \tag{11}$$

The iteration is done by a Gauss-Newton method. The system is approximated by the equation conducted by the residual error and the Jacobian matrix of the projection function to the variable $\boldsymbol{\xi}$. The Gauss-Newton update function is shown in equation (12). The iteration update of the 6 DoF trivial vector is denoted as $\boldsymbol{\varepsilon}$. And the partial derivative matrix of the reprojected 2D line over the motion vector is represented by $\boldsymbol{J}$. The reprojection of the 3D lines can be separated in 2 steps. First, the 3D lines represented in Plücker coordinates in the world frame are transformed to the camera frame. Then the transformed lines are mapped to the 2D lines on the image according to the pinhole camera imaging model. Due to the fact that the projected image lines only concerns the line's u vector, the chain rule of the partial derivative is applied to get the final Jacobian matrix as equation (13). As can be seen clearly, the Jacobian matrix is divided into 2 parts considering the camera model and the coordinates transformation respectively.

$$\boldsymbol{J}^\top \boldsymbol{J} \boldsymbol{\varepsilon} = \boldsymbol{J}^\top (\boldsymbol{l_i} - \check{\boldsymbol{l}}_{\boldsymbol{i}}(\boldsymbol{\xi})) \tag{12}$$

$$\boldsymbol{J} = \frac{\boldsymbol{\partial} \check{\boldsymbol{l}}_{\boldsymbol{i}}}{\boldsymbol{\partial \varepsilon}} = \frac{\boldsymbol{\partial} \check{\boldsymbol{l}}_{\boldsymbol{i}}}{\boldsymbol{\partial u_c}} \cdot \frac{\boldsymbol{\partial u_c}}{\boldsymbol{\partial \varepsilon}} \tag{13}$$

To begin with, the partial derivative matrix of the camera model mapping can be easily determined through the linear line projection function (8,9). The partial derivative of the non-homogeneous 2D line with respect to the normal vector is shown in equation (13). $\tilde{\boldsymbol{l}}_{\boldsymbol{i}}$ is the homogeneous coordinates of the calculated 2D line.

$$\frac{\boldsymbol{\partial} \check{\boldsymbol{l}}_{\boldsymbol{i}}}{\boldsymbol{\partial u_c}} = \frac{1}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \begin{bmatrix} 1 & 0 & -\frac{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(1)}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \\ 0 & 1 & -\frac{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(2)}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \end{bmatrix} \boldsymbol{K}^{-\top} \tag{14}$$

Then the more tricky part is to calculate the Jacobian of the line coordinates transformation with respect to $\boldsymbol{\varepsilon}$. To fully consider the orthonormal constraints, the partial derivative should be within the $SE3$ space. So simply expressing the Jacobian by analyzing every element in u_c with respect to every element in $\boldsymbol{\varepsilon}$ is definitely not what we want. Therefore, similar to the expression in the pose estimation method using points correspondences [16], we define the Jacobian matrix as equation (15). The derivative is the sensitivity to change of the projected normal vector with respect to the small change in rigid motion vector $\boldsymbol{\varepsilon}$.

$$\frac{\boldsymbol{\partial u_c}}{\boldsymbol{\partial \varepsilon}} = \left. \frac{\boldsymbol{\partial} \boldsymbol{P_u}(\boldsymbol{\varepsilon}) \boldsymbol{P}(\boldsymbol{\xi}) \boldsymbol{l_w}}{\boldsymbol{\partial \varepsilon}} \right|_{\boldsymbol{\varepsilon}=\mathbf{0}} \tag{15}$$

In order to calculate the Jacobian matrix, we need to use the adjoint map of the Lie Group [15], which is defined as equations (16). The adjoint map of a vector in $se3$ has the identical form with the line motion projection matrix $\boldsymbol{P}$. First, we consider the partial derivative with respect to one element indexed with i in $\boldsymbol{\varepsilon}$, i.e. a small change in the i-th dimension. In the equation, e_i is the i-th basis vector in the 6 DoF Lie Algebra space, whose hat function corresponds to the i-th basis matrix $\boldsymbol{G_i}$ in the tangent space of Lie Group. τ is a small number. Then the $\boldsymbol{\varepsilon_i}$ is the small change in the i-th coordinate of the rigid motion vector. In order to simplify, we use $\boldsymbol{z}$ to denote the adjoint function of the 3D line in the camera coordinates $\boldsymbol{l_c}$, which is the line's coordinates after a projection with a small change of $\boldsymbol{\varepsilon_i}$. It is worthy to mention that the $\boldsymbol{R}$ and $\boldsymbol{t}$ in equation (16) comes form the exponential map of $\boldsymbol{\varepsilon_i}$ defined in Lie Group.

$$\boldsymbol{Ad}_{\boldsymbol{\varepsilon}_i}\Big|_{\boldsymbol{\varepsilon}_i=\tau \boldsymbol{e}_i} = \begin{bmatrix} \boldsymbol{R} & [\boldsymbol{t}]_\times \boldsymbol{R} \\ \boldsymbol{0}^\top & \boldsymbol{R} \end{bmatrix} = \boldsymbol{P}(\boldsymbol{\varepsilon_i}) \tag{16}$$

$$\boldsymbol{z} = \boldsymbol{Ad}_{\boldsymbol{\varepsilon_i}}(\boldsymbol{l_c}) = \boldsymbol{P}(\boldsymbol{\varepsilon_i})\boldsymbol{l_c} \tag{17}$$

According to the Lie Algebra theory, the hat function of the adjoint map matrix has a derivative in the form of Lie Bracket, which is specified in the equation as below. In the equation, vector $\boldsymbol{m}$ and $\boldsymbol{n}$ can be arbitrary 6D vectors.

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tau}(\boldsymbol{Ad}_{\tau \boldsymbol{m}}(\boldsymbol{n}))^\wedge\Big|_{\tau=0} = \boldsymbol{m}^\wedge \boldsymbol{n}^\wedge - \boldsymbol{n}^\wedge \boldsymbol{m}^\wedge \tag{18}$$

Replacing vector $\boldsymbol{m}$ and $\boldsymbol{n}$ with $\boldsymbol{\varepsilon_i}$ and $\boldsymbol{l_c}$, we can write the derivative of the hat matrix of z with respect to one element of $\boldsymbol{\varepsilon}$ in equation (19). Specifically, the equation is the changed line coordinates' derivative over the small change in one dimension as equation (20) suggests.

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tau}(\boldsymbol{z})^\wedge\Big|_{\tau=0} = \boldsymbol{\varepsilon_i^\wedge l_c^\wedge} - \boldsymbol{l_c^\wedge \varepsilon_i^\wedge}, \boldsymbol{\varepsilon_i^\wedge} = \boldsymbol{G_i} \tag{19}$$

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tau}(\boldsymbol{z})^\wedge\Big|_{\tau=0} = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tau}(\boldsymbol{P}(\tau(e_i))\boldsymbol{l_c})^\wedge\Big|_{\tau=0} \tag{20}$$

Combining the equations (4,6,20),the derivative of the hat matrix of z is written in equation (21). Since the projection of the 2D lines on the image plane is the linear function of the normal vector, we only notice the part where the normal vector is related. Therefore, the upper right 3-by-1 vector is the partial derivative of the projected normal vector with respect to one element i in the small increment $\boldsymbol{\varepsilon_i}$. Repeatedly finding the partial derivative over all the dimensions in the motion vector space leads to the Jacobian matrix of the normal vector over the motion vector.

$$\frac{\boldsymbol{\partial}}{\boldsymbol{\partial}\tau}(\boldsymbol{z})^\wedge\Big|_{\tau=0} = \begin{bmatrix} * & \frac{\boldsymbol{\partial P_u(\varepsilon_i) l_c}}{\boldsymbol{\partial \varepsilon_i}}\Big|_{\boldsymbol{\varepsilon_i}=\boldsymbol{0}} \\ \boldsymbol{0}^\top & 0 \end{bmatrix} \tag{21}$$

Now we can write down the 3-by-6 Jacobian matrix for the normal vector in equation (22). Applying the chain rule of the partial derivative, we get the Jacobian matrix of the whole system combining equations (13,14,22). The partial derivative of the reprojected line over the 6 DoF motion vector is shown in equation (23). Noticing that the rigid motion has 6 degree of freedom while each line correspondence gives 2 constraint equation, we conclude that our method needs at least 3 lines to calculate the rigid motion, which is the minimal because there is no redundant parameter.

$$\frac{\boldsymbol{\partial u_c}}{\boldsymbol{\partial \varepsilon}} = \frac{\boldsymbol{\partial P_u(\varepsilon) l_c}}{\boldsymbol{\partial \varepsilon}}\Big|_{\boldsymbol{\varepsilon}=\boldsymbol{0}} = \begin{bmatrix} -[\boldsymbol{v_c}]_\times & -[\boldsymbol{u_c}]_\times \end{bmatrix} \tag{22}$$

$$\boldsymbol{J} = \frac{1}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \begin{bmatrix} 1 & 0 & -\frac{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(1)}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \\ 0 & 1 & -\frac{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(2)}{\tilde{\boldsymbol{l}}_{\boldsymbol{i}}(3)} \end{bmatrix} \boldsymbol{K}^{-\top} \begin{bmatrix} -[\boldsymbol{v_c}]_\times & -[\boldsymbol{u_c}]_\times \end{bmatrix} \tag{23}$$

From the residual error and the Jacobian matrix, we can calculate the small increment of the rigid motion vector $\boldsymbol{\varepsilon}$ according to the Gauss-Newton incremental update calculation equation(12). Then the motion vector is updated with the exponential function in $SE3$ as equation (24). It shows that we actually update the motion in the Euclidean space in order to simplify computation.

$$\boldsymbol{T}(\boldsymbol{\xi_{k+1}}) = \boldsymbol{exp}(\boldsymbol{\varepsilon})\boldsymbol{T}(\boldsymbol{\xi_k}) \tag{24}$$

Our method aims at applications of SLAM, which assumes successive frames don't have much change in position. So we set the first initial position as the same of the world frame, a zero vector in the 6D space. The algorithm will search by the gradient-descending direction until the gradient is small enough. Usually it converges to the result with less than 10 iterations.

4. Experimental evaluation

4.1. Synthetic lines

Experiments on simulated lines are conducted to test our methods accuracy, speed and robustness. Monte Carlo method is adopted to generate random lines in the 3D space. Then we generate a random rigid motion to transform those lines to the camera coordinates. At last, we use the camera intrinsic matrix of an industrial camera in our lab to serve as the virtual camera to project the 3D lines onto the image plane. We set the camera resolution as VGA(640*480) and the focal length as 800 pixels. The distance between the camera and the center of the random lines is set to 2 meters. To test the robustness of our method, Gaussian distributed image noises in different standard deviations were tried.

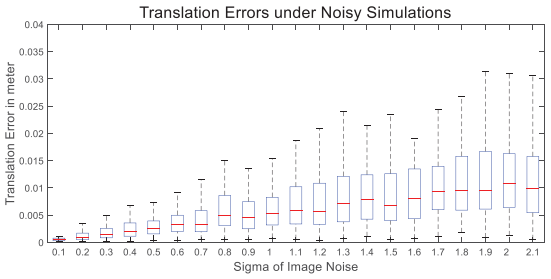
4.1.1 Test under noisy simulation

In order to test the accuracy and the robustness of our algorithm, we have repeated the random test 100 times under each image noise level in the interval of 0.1 pixel within 2.1 pixels respectively. The sigma is the standard deviation of the image noise, which served to perturb the endpoints of the lines. The translation error ΔT is calculated as the distance from the gorundtruth position $\boldsymbol{T}$ to the estimated position $\boldsymbol{T'}$ as equation (25). To estimate the rotation error, we first get the rotation difference matrix $\boldsymbol{\Delta R}$ calculated as equation (26). And then the rotation difference matrix is converted into an rotation error angle according to Rodrigues' rotation formula [20].

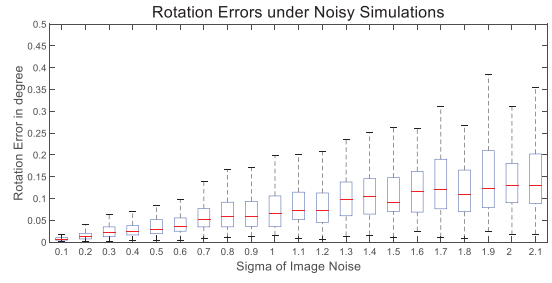
$$\Delta T^2 = \|\boldsymbol{T} - \boldsymbol{T'}\|^2 \tag{25}$$

$$\boldsymbol{\Delta R} = \boldsymbol{R}^\top \boldsymbol{R'} \tag{26}$$

The test results on random line pairs are shown in the Fig. 3. From the figure we can see that the error grows with the image noise. The median translation error is about 1 cm under the noise sigma of 2 pixels. And the largest translation error is smaller than 4 cm. Our methods position error under sub-pixel image noise is less than 2 cm. The rotation error is always smaller than 0.4°, showing a high accuracy of orientation estimation. In general, our method has a good accuracy dealing with the noisy images. However, under the noise-free case, our algorithm comes up with the accurate result even with a minimal set of 3 matching pairs of lines.



(a)



(b)

Figure 3. Translation(a) and rotation(b) errors under noisy simulations

4.1.2 Comparison with the state-of-the-arts

To further compare the accuracy and robustness of our method with the state-of-the-arts, we repeated random experiments in the same noise level of 2 pixels. As far as we are concerned, there isn't a strict non-linear optimization method for pose estimation using lines. So we applied 5 other non-iterative methods to compare. The boxplot of the error distribution were drawn to show the performance of the algorithms under the same data set. Here are 5 well-developed algebraic methods which estimate the camera pose by finding the solution of Direct Linear Transform (DLT) or the polynomials from the constraints.

Ansar developed a method dealing with no less than 4 lines with a single solution by using lifting method [21]. In his paper, he applied a para-perspective method to solving the pose problem with only one solution. And the nonlinear method is more robust than the linear methods when noise exists both theoretically and experimentally. However, it has a high computational complexity of $O(n^2)$, where n is the number of lines.

To achieve a more computationally efficient $O(n)$ method, Mirzaei proposed a method able to handle with minimum 3 lines. The result comes from 27 candidate solutions of polynomials [22]. Slow construction of the polynomials makes the method running at a high computational time. For example, it spends 78 ms dealing with 10 lines while our method finishes within 8 ms.

Also there are methods exploiting the linear formulation of the problem, called LPnL. LPnL_Bar_LS [12] uses barycentric coordinates depending on 4 arbitrarily distributed control points. Similar to the EPnP [8] method, such method aligns the control points in camera and world frame to estimate the camera pose. The linear-formulation based method also aims at large line correspondences.

Pribyl proposed a DLT method [10] using plücker coordinates coping with large line sets. It solves the same equataion (5) as in our method. However, the DLT-Plucker method treats the line projection matrix $\boldsymbol{P_u}$ as 18 variables to solve independently. Then a SVD is used to decompose the matrix to get 4 possible solutions as the decomposition of the Essential matrix in multiple view problem. Also additional validation step is needed to determine the solution. This method requires at least 9 lines as each line provides 2 equations.

Recently, Pribyl improved the DLT line method with a novel DLT-Combined Lines method [18]. It utilizes the 2D line structure and both structures in 3D lines and points. The minimum required line pairs is reduced to 5. However, the method is more suitable for large line sets. All of the above mentioned method were tested along with our method under the same noisy level(sigma = 2 pixels) in a small rigid motion to ensure a converged result of our method. The comparative results is shown in the Fig. 4. As

shown in the Fig.4, under our test random line sets and rigid transforms, the recent developed methods of LPnL_Bar_LS, DLT-plücker, DLT-combined and ours have more robust and accurate performances in both orientation and position estimation. The Ansar's and Mirzaei's methods are too sensitive to noises, leading to an unstable output. In addition, among the above methods, Mirzaei's is the only one that works with 3 line pairs. But it doesn't have a converged result with noise existing. In conclusion, our method has a performance on par with the state-of-the-arts in both rotation and tranlation estimation while only our algorithm can work with only 3 noisy matching line pairs.

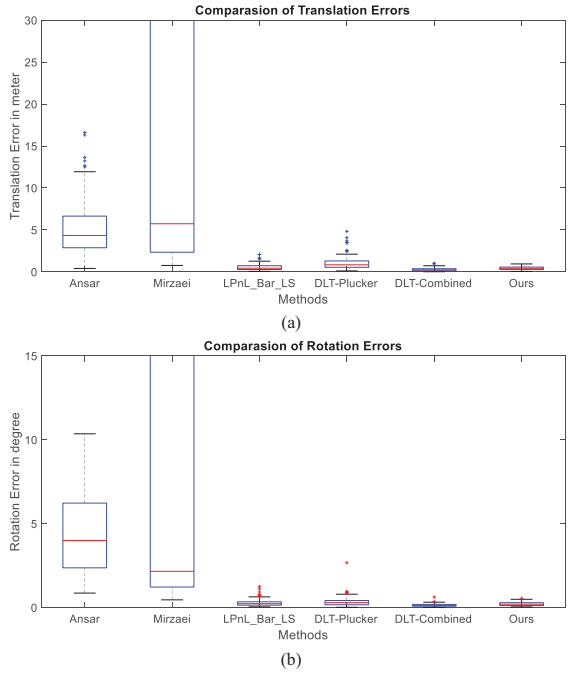


Figure 4. Boxplot of translation(a) and rotation(b) errorsrotation with different methods

4.2. Experiments on real-world images

Besides simulation experiments, we also test our method on real images of indoor scenes. The VGG corridor dataset is tried with our algorithm to do pose estimation. The dataset includes the coordinates of the starting and ending points of the 3D lines in the world coordinates and the corresponding image points of the 2D lines on the image plane. Also the calibrated intrinsics parameters are given in the matrix $\boldsymbol{K}$. Each image has an associated camera projection matrix P_c which encodes the cameras pinhole model and its estimated pose[13] as equation (27). The decomposed rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ are used as the groundtruth.

$$\boldsymbol{P_c} = \boldsymbol{K}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix} \tag{27}$$

To test our methods accuracy intuitively, we draw the 3D lines reprojections onto the 2D image plane from both the groundtruth camera pose and the estimated pose of our method respectively. The result is shown in Fig.5 where our estimated pose has a 4 cm position error and 0.04° rotation error. From the figure we can see that the estimated lines recovers very well despite some endpoints don't match strictly. However, as the unmatched endpoints still lay at the same line, they still make a good reconstruction of the 2D lines on the image plane. In addition, as the 2D lines are represented by non-homogeneous coordinates, points on the same line will make an identical line expression regardless of the relative position of the line.

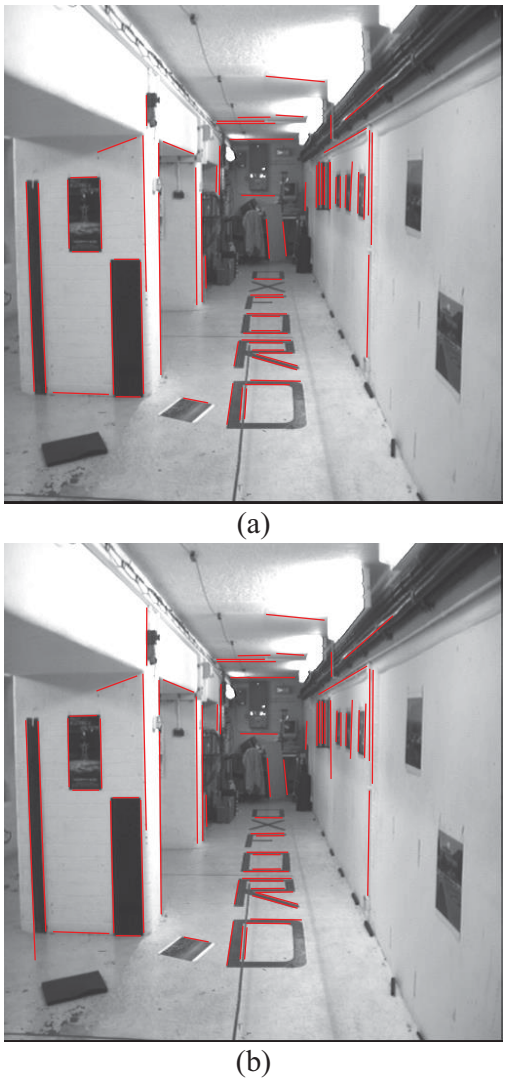
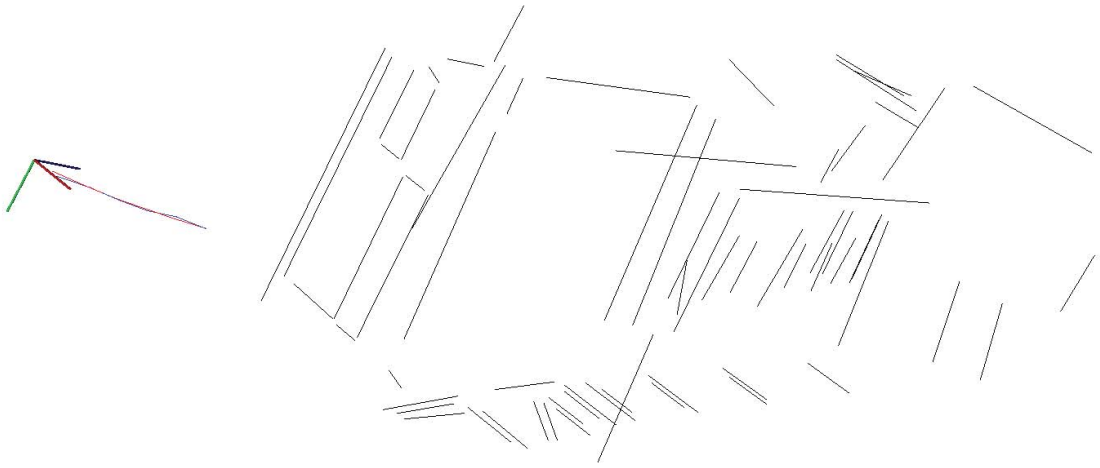


Figure 5. Groundtruth (a) of the lines projection and the estimated lines (b) of our method on the image

(a)



(b)

Figure 6. Reconstructed trajectory of the VGG datasets:(a)Image sequence of the corridor,(b) Reconstructed 3D trajectory in blue and groundtruth trajectory in red

From the VGG dataset we can also estimate the cameras trajectory from the image sequence as shown in Fig.6. Motion-only BA method is applied to estimate the pose continuously. It is assumed that successive frames won't have a great pose change. So the pose of last framethe is set as the initial pose guess of the consecutive frame. The average runtime spent on our algorithm is 10ms on the Matlab platform. Implementation on other platform can well accelerate the method.

As illustrated in the Fig.6, the red line is the decomposed trajectory while the blue one is our estimation. The world coordinates are also shown in the left. We have an estimation very close to the groundtruth. After the camera has a motion of 9.19m, we have only got the position error of 0.066m. Considering the rotation error, we get an average rotation error of 0.0045°, showing an accurate direction determination.

5. Conclusion and Futurework

In general, this article presents a new iterative non-linear method making fully consideration of the orthonormality constraints to continuously estimate the pose of the camera for SLAM applications. For the first time the minimal representation of the rigid motion in a 6 DoF vector is used in the optimized problem using lines, which makes the solution unique and in the orthonormal space. Also we have fully utilized the constraints from known lines correspondences, which makes our method work with at least 3 lines. To apply the non-linear optimized method, we deduct the partial derivatives from the adjoint map of the Lie Group and build up a novel complete Gaussian-Newton optimizing system. Experiments have validated the method to be accurate, fast and robust. Aiming at the SLAM applications, our method deals well with the indoor situation to continously estimate pose based on small sets of line correspondences.

Furthermore, there are still works to do in the future. First, as a non-linear optimization method, the proposed algorithm needs a good initial guess. A bad initialization will certainly lead to diverging iterations. The converging conditions should be further learned. How the initial pose estimation will affect the final result should be researched carefully. In addition, in the real-world SLAM applications, line pairs are never perfectly matched. It is necessary to add robust methods like RANSAC to tackle the outliers' problem.

6. Acknowledgement

This work was supported by the National Natural Science Foundation of China (No. 61471123) and the Natural Science Foundation of Guangdong Province (No. 2015A030313639).

References

[1] M. Chandraker, J. Lim, and D. Kriegman, "Moving in stereo: Efficient structure and motion using lines," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1741–1748, IEEE, 2009.

[2] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer Vision Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[3] C. J. Taylor and D. Kriegman, "Structure and motion from line segments in multiple images," *Pattern Analysis Machine Intelligence IEEE Transactions on*, vol. 17, no. 11, pp. 1021–1032, 1995.

[4] L. Zhuang, Y. Han, Y. Fan, Y. Cao, B. Wang, and Q. Zhang, "Method of pose estimation for uav landing," *Chin. Opt. Lett*, vol. 10, p. S20401, 2012.

[5] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-d line-based map using stereo slam," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1364–1377, 2015.

[6] T. Lemaire and S. Lacroix, "Monocular-vision based slam using line segments," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 2791–2796, April 2007.

[7] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "Structslam: Visual slam with building structure lines," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 1364–1375, April 2015.

[8] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[9] F. Zhou, Y. Cao, and X. Wang, "Fast and resource-efficient hardware implementation of modified line segment detector," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[10] B. Pibyl, P. Zemk, and M. adk, "Camera pose estimation from lines using plcker coordinates," in *British Machine Vision Conference*, 2015.

[11] L. Zhang, C. Xu, K. M. Lee, and R. Koch, "Robust and efficient pose estimation from line correspondences," in *ACCV*, pp. 217–230, 2012.

[12] C. Xu, L. Zhang, L. Cheng, and R. Koch, "Pose estimation from line correspondences: A complete analysis and a series of solutions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1209–1222, June 2017.

[13] A. Hanson, "Robust methods for estimating pose and a sensitivity analysis," vol. 60, 12 1994.

[14] X. Zhang, X. Sun, Y. Yuan, Z. Zhu, and Q. Yu, "Iterative determination of camera pose from line features," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 81–86, 2012.

[15] J. Hilgert and K. H. Neeb, *Structure and Geometry of Lie Groups*. Springer New York, 2012.

[16] H. Strasdat, "Local accuracy and global consistency for efficient slam," 2012.

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[18] B. Pibyl, P. Zemk, and M. adk, "Absolute pose estimation from line correspondences using direct linear transformation," *Computer Vision and Image Understanding*, vol. 161, pp. 130 – 144, 2017.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] R. M. Murray, S. S. Sastry, and Z. Li, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1994.

[21] A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *Pattern Analysis Machine Intelligence IEEE Transactions on*, vol. 25, no. 5, pp. 578–589, 2002.

[22] F. M. Mirzaei and S. I. Roumeliotis, "Globally optimal pose estimation from line correspondences," in *IEEE International Conference on Robotics and Automation*, pp. 5581–5588, 2011.