Line-Based Relative Pose Estimation

Ali Elqursh Ahmed Elgammal
Department of Computer Science, Rutgers University, USA
{elqursh,elgammal}@cs.rutgers.edu

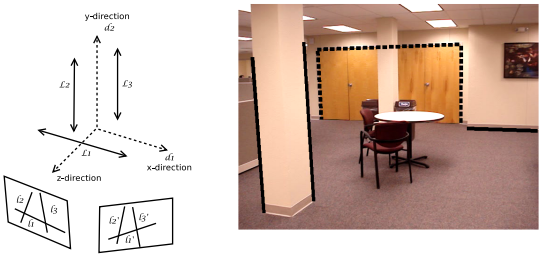
Abstract

We present an algorithm for calibrated camera relative pose estimation from lines. Given three lines with two of the lines parallel and orthogonal to the third we can compute the relative rotation between two images. We can also compute the relative translation from two intersection points. We also present a framework in which such lines can be detected. We evaluate the performance of the algorithm using synthetic and real data. The intended use of the algorithm is with robust hypothesize-and-test frameworks such as RANSAC. Our approach is suitable for urban and indoor environments where most lines are either parallel or orthogonal to each other.

1. Introduction

Structure from motion research in computer vision has reached the point where a fully automated reconstruction of the trajectory of a video camera moving through an unknown scene is becoming a routine practice. Having the ability to accurately localize a moving camera is an essential building block for robotic navigation and simultaneous localization and mapping (SLAM). Many achievements such as obtaining sparse reconstructions of cities, visual odometry and auto-calibration, to name a few, already exists.

Existing approaches were designed with the assumption that many point features can be accurately tracked. Therefore, the majority of the SfM literature uses point rather than line features. However, indoor environments consists mainly of planar surfaces with little texture and it is frequently the case that few point features can be localized. On the other hand, such environments are abundant in lines which can be more accurately localized and tracked due to their multi-pixel support. Additionally, indoor environments exhibit structures that can be exploited to achieve robust structure from motion even with few features. Existing approaches try to enforce structural constraints after computing the pose of the camera and thus do not benefit from the extra information available in the relative pose computation (e.g. [20]).



(a) (b)

Figure 1: 1a Primitive configuration. L_1, L_2, L_3 are three lines with $L_2 \parallel L_3$ and $\perp L_1$. 1b Two examples of primitive configurations. The dashed black configuration is an example of a trivial configuration that occurs in any rectangular structure such as windows and doors. The Solid Black configuration is a nontrivial example where the lines are not coplanar.

Relative pose estimation is the problem of computing the relative motion between two images and is a key ingredient for any SfM system. For relative pose estimation to be robust many primitives must be used. Typically this is done by using hypothesize-and-test frameworks such as RANSAC [7]. These frameworks require the existence of an algorithm that estimates the pose from a small sample of primitives.

This paper proposes a framework that solves the calibrated relative pose estimation from lines. The rotation is first computed from a "primitive configuration" which we define as a configuration of three lines with two of the lines parallel and their direction orthogonal to the third line. This configuration is depicted in Fig. 1. Note that none of the lines need to intersect. The three lines do not need to be coplanar. The plane defined by the two parallel lines need not be orthogonal to the third line. The parallel lines do not need to be on the same planar surface. The solid black lines in Fig.1b shows an example of such configuration. Such primitive configuration frequently occurs in indoor and urban environments.

Once the relative rotation is computed, the translation is then computed from any two intersections of two pairs of lines. Such intersection points may not corresponds to real intersections. The algorithm is ideal for use with RANSAC since it requires a minimal sample of three line matches be-

tween two images to compute a general rotation and any two intersection points to compute the translation. We do not assume prior knowledge of primitive configurations, rather the detection of primitive configurations is done automatically by RANSAC. The combination of weak structural constraints required by our method and the low number of primitives required makes our framework suitable for pose estimation in indoor and urban environments.

One important characteristic of our algorithm is that it decouples the computation of the relative rotation and translation. A major advantage over other algorithms is therefore, that the rotation computation is not susceptible to degeneracies due to a small baseline. In fact, we can effectively compute the relative rotation with zero baseline without having to handle it as a special case.

Our main contributions are :

- A method to compute the relative rotation between two cameras given a primitive configuration. The translation is computed from any two line intersections once the relative rotation is known.

- A framework where this algorithm can be used to detect triplet lines compromising a primitive configuration. Lines fitting this criteria are considered inliers and are used for the estimation of the camera pose.

2. Related Work

From two images lines do not put any constraints on the camera pose [14]. The trifocal tensor [23], is thus the standard method to achieve structure from motion from lines [14, 9]. The trifocal tensor has 18 degrees of freedom and at least 13 lines are needed to compute it [15]. Besides the requirement of three images, the large number of line correspondences required discourages the use of trifocal tensor in hypothesize-and-test frameworks. In contrast, we need only 3 lines correspondences to compute the relative rotation and two line intersections to compute the relative translation. Recently, [3] used line matches in two stereo pairs (4 images), to compute the camera pose. The issue of 3D line representation, triangulation, and bundle adjustment was investigated in [2] but the motion computation was done using the trifocal tensor.

Another category of SfM from lines, exploits common assumptions such as the existence of three orthogonal dominant directions. In [4], single view camera pose estimation from scene constraints is proposed where it is done using a vertical vanishing point and world to image homography. To compute the homography a reference plane had to be identified. In [27, 17], the three dominant directions are computed and then used to obtain reconstructions. Since finding the dominant directions involves using all lines in the image these methods fail if dominant direction can not be detected in an image. In contrast, lines in our primitive configurations do not need to be aligned to dominant directions. For the computation of the rotation they do not even have to be coplanar. Instead all directions present in the scene can be used as long as they are part of a primitive configuration. In [8], the constrained motion of the camera for turntable sequences is used in the SfM computation. Although, we show results on such sequences, we use these as a benchmarking tool rather than being restricted to them.

Using line segments instead of lines has been also explored, however these methods are faced with the difficulty of reliably matching end-points. In [25] they formulate the problem of reconstructing line segments in terms of an objective function which measures the re-projection error in the image plane. Since they use nonlinear minimization, the method is prone to not converge to the correct solution. Additionally, they require at least 6 correspondences in 3 images which is a problem for hypothesize-and-test frameworks. In contrast, our method uses lines (not line segments) and is thus not susceptible to reliability issues of endpoints. Our algorithm is also used in a RANSAC framework and is therefore more robust.

Recently, large progress was made in visual odometry [21, 19, 1], and urban reconstruction systems [20, 10]. Almost all such systems uses stereo cameras or camera rigs. These systems rely on solutions to the problem of relative pose estimation from a minimal set of 5 points [18, 24] inside a RANSAC framework. We focus on the relative pose estimation problem from lines. Using lines is complementary to using points and thus our approach can be used in any of these systems to compute the relative pose. Similar to the state of the art for points[18] which produces up to 10 solutions we produce 4 solutions. Relevant to these systems, is PTAM[16] which performs SFM using a hand-held camera but on small augmented reality workspaces.

A related field of research in the robotics community is that of Simultaneous Localization and Mapping (SLAM). In [5], SLAM from a single camera is performed. Point features are tracked and extended kalman filtering is used for updating the location of the camera and the 3D structure. A small set of known 3D world points are used in initialization. In [22], this system is extended to incorporate lines but still requires the initialization given some point and line features. In contrast, our approach does not require initialization.

3. Structure from motion using lines

3.1. Problem statement

Lines in two images do not provide any constraints on the camera pose [26]. However, in indoor and urban environments many lines are parallel and/or orthogonal. For example, most furniture have edges that satisfy these assumptions. This motivates the need to develop algorithms

that exploit such constraints.

Before delving into the details of our method, we introduce some notation. Let L_1, L_2, L_3 be three world lines with $L_2 \parallel L_3$ and $L_2, L_3 \perp L_1$. We call these a primitive configuration; see Fig. 1a. Let $\mathbf{l_1}$, $\mathbf{l_2}$, $\mathbf{l_3}$ be vectors of size 3 representing the homogeneous image coordinates of L_1, L_2, L_3 respectively. Also, let $\mathbf{d_2}$ represents the direction of lines L_2, L_3 and $\mathbf{d_1}$ represents the direction of L_1 in 3D. $\mathbf{d_1}, \mathbf{d_2}$ are represented as 3 dimensional vectors but have only 2 degrees of freedom. We will choose the world coordinate system such that the x-axis direction corresponds to $\mathbf{d_1}$ and the y-axis direction corresponds to $\mathbf{d_2}$. The z-axis, using a right hand coordinate system, is the direction orthogonal to $\mathbf{d_1}$ and $\mathbf{d_2}$ with +ve Z towards the camera. This is illustrated in Fig. 1a. The projection matrix for camera 1 and 2 is thus represented using $\mathbf{P_1} = \mathbf{K}[\mathbf{R_1}|\mathbf{t_1}]$ and $\mathbf{P_2} = \mathbf{K}[\mathbf{R_2}|\mathbf{t_2}]$. Where $\mathbf{K}$ is the calibration matrix, $\mathbf{R_i}$ is a rotation matrix, and $\mathbf{t_i}$ is a translation vector.

3.2. Relative rotation from three lines

Instead of computing the relative rotation directly, we first compute the rotation between the coordinate system defined by the primitive configuration and each camera denoted by $\mathbf{R_1}, \mathbf{R_2}$. Then we can compute the relative rotation between the two cameras, $\mathbf{R_{rel}}$, by

$$\mathbf{R_{rel}} = \mathbf{R_2}\mathbf{R_1^T}. \tag{1}$$

In the rest of the derivation we will be dealing with a single camera with $\mathbf{P}, \mathbf{R}$ denoting projection and rotation matrix for that camera. The relation between the direction of a line and its vanishing point can be written as [13]

$$\mathbf{v} = \mathbf{KRd}. \tag{2}$$

Since we choose $\mathbf{d_1}$ to coincide with the x-direction and $\mathbf{d_2}$ to coincide with the y-direction we have $\mathbf{d_1} = \begin{pmatrix}1 & 0 & 0\end{pmatrix}^T$, and $\mathbf{d_2} = \begin{pmatrix}0 & 1 & 0\end{pmatrix}^T$.

Let us denote by $\mathbf{v_1}$ and $\mathbf{v_2}$ the vanishing points in the direction of the orthogonal line L_1 and the parallel lines L_2, L_3 respectively. Using (2), they can be expressed by

$$\mathbf{v_1} = \mathbf{KRd_1} = \mathbf{Kr_1}, \quad \mathbf{v_2} = \mathbf{KRd_2} = \mathbf{Kr_2}, \tag{3}$$

where $\mathbf{r_i}$ represents the ith column of $\mathbf{R}$. On the other hand, we know that $\mathbf{v_2}$ lies on the intersection of the lines $\mathbf{l_2}$,and $\mathbf{l_3}$, i.e.,

$$\mathbf{v_2} = \mathbf{Kr_2} = \mathbf{l_2} \times \mathbf{l_3}. \tag{4}$$

We also know that $\mathbf{d_1}$ is orthogonal to $\mathbf{d_2}$, i.e. $\mathbf{d_1} \cdot \mathbf{d_2} = 0$. Substituting from (2)

$$(\mathbf{R^T K^{-1} v_1}) \cdot (\mathbf{R^T K^{-1} v_2}) = 0 \tag{5}$$

$$\mathbf{v_1^T (K^{-T} K^{-1}) v_2} = 0 \tag{6}$$

Since $\mathbf{v_1}$ must satisfy the orthogonality constraint[1] (6) and also must lie on $\mathbf{l_1}$ (i.e., $\mathbf{v_1} \cdot \mathbf{l_1} = 0$) we get

$$\mathbf{v_1} = null\left(\begin{bmatrix} \mathbf{v_2^T (K^{-T} K^{-1})} \\ \mathbf{l_1} \end{bmatrix}\right) \tag{7}$$

[1]The matrix $\mathbf{K^{-T} K^{-1}}$ is the image of absolute conic (IAC) [6].

Thus algorithm for computing the rotation $\mathbf{R}$ of a single camera from a primitive configuration can be summarized as : Input: Given $\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}$

1. Compute $\mathbf{v_2}$ from equation (4)
2. Compute $\mathbf{v_1}$ from equation (7)
3. Compute $\mathbf{r_1}, \mathbf{r_2}$ from equations (3)
4. Compute $\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}$

Once we have computed $\mathbf{R_i}$ for each camera, the relative rotation, $\mathbf{R_{rel}}$, can be obtained from (1). Note that for the computation of $\mathbf{R_{rel}}$ we did not need to assume that L_1, L_2, and L_3 are coplanar.

A note about relative rotation in the case of zero baseline. In this case a point in the first image and its corresponding point in the second image are related by the homography $\mathbf{H} = \mathbf{K R_{rel} K^{-1}}$ [13]. Taking any three lines, not necessarily in a primitive configuration, and applying our algorithm we find that

$$\mathbf{R_2} = \begin{bmatrix}\mathbf{K^{-1} v_1'} & \mathbf{K^{-1} v_2'} & \mathbf{r_1 \times r_2}\end{bmatrix} \tag{8}$$

$$= \begin{bmatrix}\mathbf{K^{-1} H v_1} & \mathbf{K^{-1} H v_2} & \mathbf{r_1 \times r_2}\end{bmatrix} \tag{9}$$

$$= \begin{bmatrix}\mathbf{R_{rel} K^{-1} v_1} & \mathbf{R_{rel} K^{-1} v_2} & \mathbf{r_1 \times r_2}\end{bmatrix} \tag{10}$$

$$= \mathbf{R_{rel} R_1}, \tag{11}$$

where $\mathbf{v_i}, \mathbf{v_i'}$ are the vanishing points computed by our algorithm in the two images respectively. Therefore, $\mathbf{R_2 R_1^T} = \mathbf{R_{rel} R_1 R_1^T} = \mathbf{R_{rel}}$ and we get the correct relative rotation regardless of whether the lines are in a primitive configuration or not.

3.3. Degenerate Cases for Relative Rotation

From the algorithm described above we notice that the only step where degenerate cases can occur is when we compute the null space in equation (7). The dimension of null space is at least 1 since this is a 2×3 matrix. For the non-degenerate case the null space has dimension 1 and for degenerate cases the null space can have dimensions 2 or 3.

For the null space to have dimension 3 both $(\mathbf{K^{-T} K^{-1}})\mathbf{v_2}$ and $\mathbf{l_1}$ must be equal to 0 . However, it is not possible for $\mathbf{l_1}$ to be a zero vector because this does not correspond to a defined line. We are left with the case where the null space has dimension 2. This means the $\mathbf{l_1}$ is a linear combination of $(\mathbf{K^{-T} K^{-1}})\mathbf{v_2}$. We can write the condition where this will occur as

$$\mathbf{l_1} = (\mathbf{K^{-T} K^{-1}})\mathbf{v_2} \tag{12}$$

which is equivalent to

$$\mathbf{K^T l_1} = \mathbf{K^{-1} v_2} \tag{13}$$

Geometric interpretation : Let $\hat{\mathbf{d}}_\mathbf{i}$ be the directions of the lines in the coordinate frame of the camera. From equation (2) this implies $\hat{\mathbf{d}}_\mathbf{i} = \mathbf{K^{-1} v_i}$. We re-write equation (7) as

$$\hat{\mathbf{d}}_\mathbf{1} = \mathbf{K^{-1} v_1} = null\left(\begin{bmatrix} \mathbf{(K^{-1} v_2)^T} \\ \mathbf{(K^T l_1)^T} \end{bmatrix}\right) \tag{14}$$

It is known that $\mathbf{K^T l_1} = \mathbf{n}$, where $\mathbf{n}$ is the normal of the back projected plane from $\mathbf{l_1}$ in the coordinate frame of the camera [13]. From this we conclude that our algorithm computes the direction of the orthogonal line as the direction that is orthogonal to both $\mathbf{n}$ and $\hat{\mathbf{d}}_\mathbf{2}$. The degeneracy occurs when both $\mathbf{n}$ and $\hat{\mathbf{d}}_\mathbf{2}$ coincide.

3.4. Relative translation

Once the relative rotation is computed we can now compute the relative translation $\mathbf{t_{rel}}$. We can compute $\mathbf{t_{rel}}$ from two intersection points. Since our primitive does not need the lines to be coplanar, they do not necessarily intersect in 3D. However, naturally there are many lines which are coplanar in any given scene, and their intersection points can be used (As we will see later, we leave the task of detecting if the lines are in fact planar to RANSAC).

Obviously, the relative translation can be only obtained up to scale. Let $\mathbf{p_1}, \mathbf{p_2}$ be any two intersection points of two pairs of lines and $\mathbf{p'_1}, \mathbf{p'_2}$ their corresponding points in the second image. Note that these intersection points are computed without any point feature detection. Without loss of generality we can assume that these points are normalized by pre-multiplying with $\mathbf{K^{-1}}$. Thus the epipolar constraint can be written as

$$\mathbf{p_i'^T}[\mathbf{t_{rel}}]_\times \mathbf{R_{rel}} \mathbf{p_i} = 0. \tag{15}$$

Since $\mathbf{R_{rel}}$ is known, we have 2 linear constraints on the elements of $\mathbf{t_{rel}}$, one for each intersection point, the relative translation can be linearly computed up to scale.

3.5. Number of Solutions

Typically, relative pose algorithms compute several solutions for each set of primitives and relies on other methods to disambiguate between the solutions. For example, the 7-point algorithm[13] produces up to 4 solutions, while the 5 point algorithm[18] computes up to 10 solutions given a set of 5 point correspondences. Similarly, here we show that there is up to 4 solutions for our algorithm and propose two methods to disambiguated between them.

First, we analyze the relation between line directions and vanishing points. Directions in $\mathbb{R}^3$ can be represented by unit vectors and therefore has 2 degrees of freedom. On the other hand, vanishing points are represented as point in $\mathbb{P}^2$. Points in $\mathbb{P}^2$ can be represented by a unit sphere where antipodal points are joined together. This implies that the mapping from directions to vanishing points is 2 to 1. To see this algebraically, we substitute into equation (2) with $\mathbf{d}$ and $-\mathbf{d}$ and get the same vanishing point.

$$\mathbf{v} = \mathbf{KRd} = \mathbf{KR}(-\mathbf{d}) \tag{16}$$

Where equality is up to scale. This is consistent with our intuition that there is an inherent ambiguity in determining line directions. For our algorithms for exterior orientation, this means we have up to 4 different solutions depending on the freedom of choosing the x-axis and y-axis directions. These solutions can be obtained by taking the 4 different combinations of signs for r_1 and r_2. These solutions are related by a 180 degrees rotations around the x-axis, the y-axis, or both.

For the case of relative rotation, a quick combinatorial analysis shows that there are 16 possible solutions based on the 4 solutions for each of $\mathbf{R_1}$ and $\mathbf{R_2}$. However, it turns out that this is not the case and that there are only 4 unique solutions. We can show this algebraically, by writing down the 16 possible sign changes in $\mathbf{R_1}$ and $\mathbf{R_2}$ and computing $\mathbf{R_{rel}} = \mathbf{R_2 R_1^T}$. The intuition is that there are only 4 possible choices for the relative directions of axes in the two images. One method to disambiguate between the solutions, that is suitable for consecutive video frames, is simply to assume that the rotation around any axis is not more than 90 degrees between frames. Another method, that is suitable for still images, is to use the appearance on the sides of the line to determine if the line was flipped between the two images.

4. Robust estimation

4.1. Generating candidate triplets

Unfortunately we can not use line segment matches directly in a hypothesize and test architecture. Instead, line segment matches must be grouped into candidate triplets. The reason for this is that when evaluating the error of a candidate solution, line matches do not put any constraints on the rotation. This follows from the fact that they do not put any constraints on the epipolar geometry. Thus to evaluate the error we measure the error in the rotation generated by other triplets.

Since the number of lines are typically in the order of ≈ 100, we accomplish this in a brute force manner. We first generate all possible triplets of lines and then prune them using a constraint that has a theoretical basis: the vanishing point must lie after the intersection point with orthogonal line. Our experiments show that this simple strategy is in fact effective.

4.2. RANSAC for relative rotation

Given many candidate triplets of lines, we would like to robustly estimate the rotation. To accomplish this we use RANSAC. Since we can estimate the relative rotation from a single triplet of lines in a primitive configuration, the number of iterations is greatly reduced [2].

The output of the relative rotation algorithm is a rotation matrix $\mathbf{R_{rel}}$ and two vanishing points for each camera $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v'_1}, \mathbf{v'_2}$. For RANSAC to work we need to define a distance function that tells us how well a rotation matrix

[2] In our experiments, when the number of candidate triplets is less than M=400 we simply evaluated the error for all triplets instead of random sampling.

$\mathbf{R_{rel}}$, computed from a sample, fits other samples. Before deriving this function, we note that from (2) we can write the relation between vanishing points in two images and the line direction as

$$\mathbf{v_i} = \mathbf{K}\mathbf{d_i}, \quad \mathbf{v'_i} = \mathbf{K}\mathbf{R_{rel}}\mathbf{d_i} \quad i = 1, 2 \tag{17}$$

Re-arranging we get two equations for $\mathbf{d}$

$$\mathbf{d} = \mathbf{K^{-1}}\mathbf{v} \tag{18}$$

$$\mathbf{d} = \mathbf{R_{rel}^T}\mathbf{K^{-1}}\mathbf{v'} \tag{19}$$

If a rotation matrix fits a primitive configuration perfectly then both equations (18),(19) should give the same direction $\mathbf{d}$. However, in general this is not the case and, therefore, the directions computed from the two equations will be different. A suitable distance function would be the sum of angles between each two directions computed from (18),(19).

$$dist(\mathbf{R_{rel}}, \{\mathbf{v_i}, \mathbf{v'_i}\}_{\mathbf{i=1,2}}) = \sum_{j=1}^{2} \left| cos^{-1}\left(\frac{\mathbf{K^{-1}}\mathbf{v_j} \cdot \mathbf{R_{rel}^T}\mathbf{K^{-1}}\mathbf{v'_j}}{\|\mathbf{K^{-1}}\mathbf{v_j}\| \cdot \left\|\mathbf{K^{-1}}\mathbf{v'_j}\right\|} \right) \right|. \tag{20}$$

As a result of RANSAC we get a set of inliers and a rotation matrix that produced this set of inliers. This rotation matrix was computed from one primitive configuration and fits the set of N inliers within the specified threshold. The final step of RANSAC is to re-estimate the rotation matrix from all inliers. Formally, we want to compute a rotation $\mathbf{R_{rel}}$ that minimizes

$$\sum_{i=1}^{N}\sum_{j=1}^{2} \mid angle(\mathbf{K^{-1}}\mathbf{v_{ij}}, \mathbf{R_{rel}^T}\mathbf{K^{-1}}\mathbf{v'_{ij}}) \mid \tag{21}$$

where $\mathbf{v_{ij}}, \mathbf{v'_{ij}}$ are the vanishing point corresponding to primitive configuration i and direction j in cameras 1 and 2 respectively. This is a nonlinear optimization in $\mathbf{R_{rel}}$. Equivalently we try to minimize the following

$$\sum_{i=1}^{N}\sum_{j=1}^{2} \left\| \frac{\mathbf{K^{-1}}\mathbf{v_{ij}}}{\|\mathbf{K^{-1}}\mathbf{v_{ij}}\|} - \mathbf{R_{rel}^T}\frac{\mathbf{K^{-1}}\mathbf{v'_{ij}}}{\left\|\mathbf{K^{-1}}\mathbf{v'_{ij}}\right\|} \right\|_{\mathbf{2}}^{\mathbf{2}} \tag{22}$$

In matrix form this equivalent to minimizing the Frobenius norm :

$$\mathbf{R_{rel}} = \mathbf{argmin} \left\|\mathbf{D} - \mathbf{R^T}\mathbf{D'}\right\|_{\mathbf{F}}^{\mathbf{2}}, \tag{23}$$

where $\mathbf{D}, \mathbf{D'}$ are two $3 \times 2N$ matrices formed by the concatenation of unit directions in the first and second camera respectively. This is known as the "orthogonal Procrustes problem", whose solution is given by [11] as $\mathbf{R_{rel}} = \mathbf{UV^T}$, where $\mathbf{USV^T}$ is the SVD of $\mathbf{D'D^T}$. It is important to note that unlike previous approaches we do not need many lines to have equal directions. Every pair of triplets, one triplet in each image, votes for a certain relative rotation. If these triplets satisfies our primitive configuration assumptions, even if not aligned to dominant directions, then the estimated rotation is equal to the true rotation and they give support to the same solution. Other triplets that are not primitive configurations will vote for random rotations and will have little support. It is the combination of RANSAC and our algorithm that enables the automatic detection of primitive configurations from two images. As noted earlier in the case of zero baseline, all line triplets will produce the correct relative rotation and we will not be able to detect primitive configurations in this case.

4.3. RANSAC for relative translation

Once the relative rotation is computed robustly, we can then compute the relative translation from intersection points of lines. Although they may seem similar, intersection points are different from point features. The reason is that intersection points occur between any two lines regardless of weather there is a corresponding image feature at that position. In fact, the intersection point may exist outside the image.

We proceed by computing all intersection points between all possible pairs of N lines in the image; there are $N(N-1)/2$ such pairs. Of these, intersection points satisfying the epipolar constraint (15) are considered inliers. RANSAC is then used to detect inliers using the symmetric epipolar error [28]

$$d(\mathbf{p}, \mathbf{p'}) = d_\perp(\mathbf{p}, \mathbf{F^T}\mathbf{p'}) + d_\perp(\mathbf{p'}, \mathbf{F}\mathbf{p}) \tag{24}$$

where $\mathbf{F}$ is computed as $\mathbf{K^{-T}}[\mathbf{t_{rel}}]_\times\mathbf{R_{rel}}\mathbf{K^{-1}}$ and $\mathbf{t_{rel}}$ is computed from two intersection point samples. Next, similar to the relative orientation, we re-estimate $\mathbf{t_{rel}}$ from all inlier intersection points. This is done by using least squares on the linear constraints (15) generated from all inlier points.

An obvious concern is the number of iteration needed by RANSAC to achieve an outlier free sample given that we use all intersection points. If we use a pessimistic estimate that 90% of the intersection points are outliers, and we want to achieve an outlier free sample with probability of 0.99, then a simple computation [7] tells us that we need 458 iterations. We are able to tolerate a high proportion of outliers since we use only two intersection points to estimate relative translation.

4.4. Nonlinear refinement

Since we compute $\mathbf{t_{rel}}$ after the computation of $\mathbf{R_{rel}}$, any error in $\mathbf{R_{rel}}$ will affect the computation of $\mathbf{t_{rel}}$. In addition, although we used line directions to compute $\mathbf{R_{rel}}$ we did not use the evidence from intersection points in its computation. We, therefore, devised a nonlinear refinement step improve the relative orientation. To accomplish this we use Levenberg-Marquardt algorithm to minimize the symmetric transfer error over all inlier intersection points $\mathbf{p_i}$, $\mathbf{p'_i}$ for the RANSAC of $\mathbf{t_{rel}}$

$$\mathbf{R_{rel}}, \mathbf{t_{rel}} = \underset{\mathbf{R},\mathbf{t}}{\arg\min} \sum_{\mathbf{i}} \mathbf{d}(\mathbf{p_i}, \mathbf{p'_i})^{\mathbf{2}} \tag{25}$$

Where $d(\mathbf{p_i}, \mathbf{p_i'})$ is the symmetric epipolar distance defined in (24). A careful reader may wonder how does this optimization take into account the properties of our primitive configuration. To see how this is handled, we show first that all vanishing points corresponding to intersection of parallel lines are inliers to the epipolar geometry.

$$\mathbf{v'Fv} = \mathbf{d^T R^T K^T K^{-T} [t]_\times R K^{-1} K d} \quad (26)$$

$$= (\mathbf{Rd})^\mathbf{T} [\mathbf{t}]_\times (\mathbf{Rd}) = 0 \quad (27)$$

Since vanishing points are already part of our inlier set, we are in fact adding soft constraints on $\mathbf{R_{rel}}$ to our optimization in equation (25).

5. Experiments

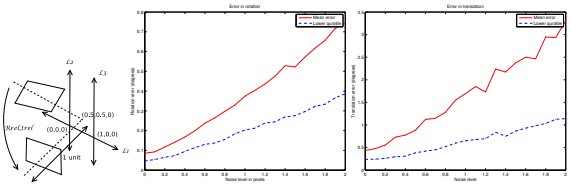
Here we show the results of our framework using synthetic and real datasets. Lines were automatically detected using LSD detector[12] and tracked to obtain line matches. In all real data experiments the camera was calibrated using Zhang method [29] and *No bundle adjustment was used. More results are available in the supplemental materials.*

Synthetic data Our synthetic experiment is designed to quantify the performance of our algorithm. In the synthetic experiment we create three coplanar lines; two of them parallel to each other and orthogonal to the third. The setup is depicted in Fig. 2a. The distance between the two parallel line is one unit. Two cameras are then randomly generated on a unit sphere centered around the (0.5,0.5,0) with the principal axis pointing toward that point. For image lines, we estimate each line from a set of points that lie on the true image line. 20 points were used to estimate each line. We add a zero mean Gaussian noise of varying standard deviations to the image coordinates of these points. The noise standard deviation σ is in pixels and we perform experiments for $\sigma \in (0, 2)$.

The performance of the algorithm is shown in Fig. 2. Each data point is based on 1000 trials. Similar to [3, 18] the lower quartile of the error distributions is displayed since the targeted use of our algorithm is in hypothesize-and-test frameworks such as RANSAC. As can be seen the rotation error is at most 0.7° (0.35° for the lower quartile) while the translation error is at most 3.5° (1° for the lower quartile). Since we re-estimate the relative pose from all inliers after RANSAC, these accuracy rates are sufficient for most applications. The higher translation error can be explained by the cascading of error due to the use of the relative rotation in the computation of the relative translation.

PTZ Camera: The purpose of this experiment is to evaluate the accuracy and robustness of our method in the case of small baseline. For that purpose we used a Pan Tilt Zoom (PTZ) Camera[3]. The camera pans 340° only, with a 20° gap. The sequence consists of 46 frames with the angle between each pair of frames $= 7.55^\circ$. To close the loop, the first frame is also used as the last frame in the sequence.

[3]The camera used is a Sony SNCRZ30N PTZ Camera.



(a) Setup (b) Rotation Error (c) Translation Error

Figure 2: Setup and error plots for the synthetic experiment. (a) Setup is a single triplet viewed from different positions. Each data point is the mean of 1000 runs. (b) Error in rotation. (c) Error in the direction of the translation vector. Both errors are in degrees.

Fig. 3 shows the computed total rotation as a function of the frame number and the error per frame using both our algorithm and the 5 point algorithm[18] for comparison[4]. The high error for the 5 point algorithm is not surprising, since the baseline is almost zero making it a near degenerate case for the 5 point algorithm. On the other hand, our algorithm can compute the correct rotation even with a zero baseline with no special handling as was shown before. The mean error in the rotation is 0.06° for our algorithm compared to 1.9° using the 5 point algorithm. The cumulative error in closing the loop is $< 0.7^\circ$.

Lab sequence: The purpose of this sequence is to evaluate our algorithm under realistic conditions and compare it with point based methods (See Fig. 4). This sequence consists of 815 frames captured using a hand-held camera[5]. Starting from one point a person holding the camera moves around a group of cubicles and returns back to the same point (See Fig.4b). Total distance traversed is around 40 meters. Lines were automatically detected and tracked. Besides our algorithm we tested the 5 point algorithm [18] using SIFT features and an available camera tracking software called Voodoo[6].

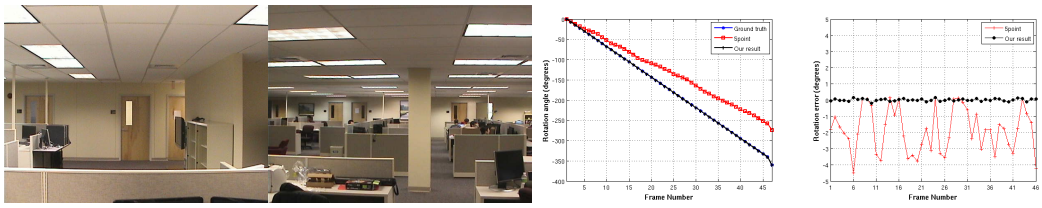
This sequence is challenging for several reasons. There is large portions of forward motion and the camera is a typical amateur camera that exhibits both limited field of view and radial distortion. Finally, the camera motion is jittery and exhibits motion blur.

As we hypothesized, Fig. 4 shows that our algorithm is superior to point based algorithms on this indoor sequence. The average number of lines per frame is 117 and for points is 500. Although there are fewer lines per frame, each line can be a member of many primitive configurations. Many of the point features are on the textured carpet and is poorly tracked. The side view shows a relatively small vertical drift even with such a large motion. The plan view shows that the

[4]5 point code provided by D.Nister at http://www.vis.uky.edu/~dnister/Executables/RelativeOrientation/

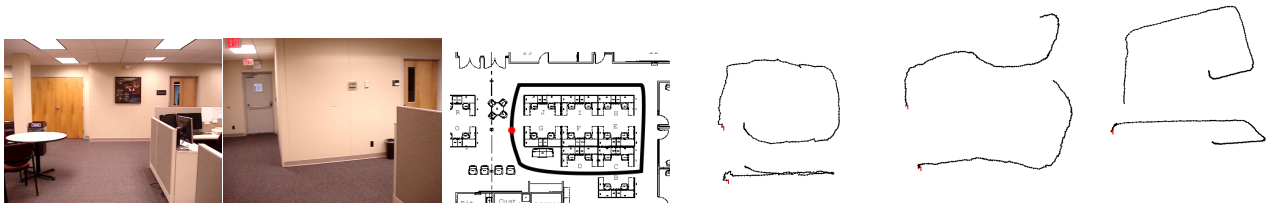
[5]An amateur SONY-DSC W55 camera.

[6]http://www.digilab.uni-hannover.de/docs/manual.html



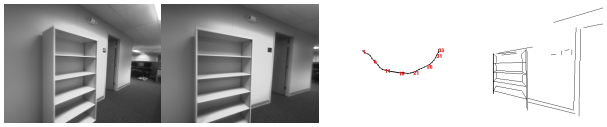
(a) (b) (c) (d)

Figure 3: Best seen in color. (a)(b) Two frames from the sequence. (c) Cumulative rotation for our algorithm and the 5 point algorithm with ground truth. Note that our result is almost identical to the ground truth. (d) Rotation error per frame. Mean error is only 0.06°. This is because the zero baseline is not a degenerate case for the rotation computation. The error in closing the loop is $< 0.7^\circ$.



(a) (b) (c) (d) (e)

Figure 4: Lab sequence consisting of 815 frames captured using a hand-held camera. Distance traversed is around 40 meters. (a) Two images from the sequence. (b) Floor plan showing approximate trajectory taken with Red dot indicating starting and ending point. (c)(d)(e) Plan and side views of the reconstructed trajectory using our algorithm, 5-point algorithm and voodoo camera tracker respectively.



(a) (b) (c) (d)

Figure 5: (a)(b) Two frames from the bookcase sequence. (c) Plan view of the reconstructed camera trajectory. (d) Reconstruction.

(a) (b) (c) (d) (e)

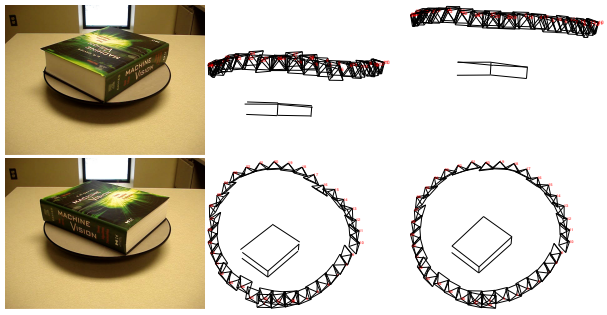
Figure 6: Corridor Sequence.(a)(b)(c) Three frames from the corridor sequence with example primitive configuration detected (shown in red). (d) Plan view of the recovered motion that shows the leftward motion. (e) Side view of the motion.

error in closing the loop is relatively small. This sequence shows that our framework may be used for visual odometry applications.

Bookcase sequence: In this sequence the hand-held camera was moved in half a circle while being targeted at a bookcase. Fig. 5 shows a plan view of the reconstructed trajectory and the reconstructed result. The reconstruction was obtained by triangulating the lines from two images. Note that most of the images consists of planar surfaces with few point features.

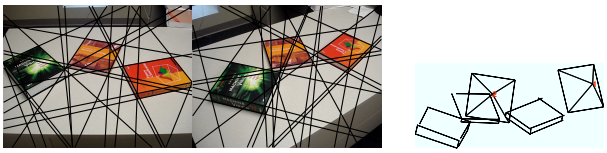
Corridor Sequence: This is a benchmark sequence of 8 frames obtained from a camera that is mounted on a mobile vehicle[7]. The vehicle moves along the floor slightly turning to the left. We perform line detection and matching automatically. Two views of the reconstructed sequence is shown in Fig. 6. Our result captures the general motion in the sequence. We take the results provided with the dataset as the ground truth. The mean error for rotation is 1.36°.

[7]The corridor sequence and computed reconstruction were obtained from http://www.robots.ox.ac.uk/~vgg/data1.html



(a) (b) (c)

Figure 7: (a) Two frames from the book turntable sequence. (b) side and plan 3D views of the reconstructed sequence using our linear algorithm only. (c) side and plan using our algorithm with inter-frame nonlinear refinement. No bundle adjustment was used. All frames have between 6 to 8 lines visible which is less than the 13 lines required for the trifocal tensor. The cumulative error in the rotation is only 3.7 degrees.



(a) (b) (c)

Figure 8: Three books in two views. The books are oriented in different directions and therefore there is no dominant directions. (a)(b) The two images that were used with back projected lines. (c) A perspective view of the three books.

Turntable Book Sequence: The purpose of this sequence is to evaluate the ability to compute the relative motion when the trifocal tensor can not be applied. All frames have between 6 to 8 lines which is less than the minimum number of lines required to estimate the trifocal tensor. Due to the clutter in the background that do not rotate with the turntable, we extracted the lines manually for this dataset but performed the matching automatically. Fig. 7b, 7c show two views of the reconstructed motion and structure using our algorithm with and without nonlinear refinement. Using our linear algorithm the error in the rotation for last camera is $3.7°$, and the average error over all relative rotations is $1.472°$.

Three books: The purpose of this test sequence is to demonstrate that our algorithm can be applied even when there are no dominant directions and that we can compute the relative pose from only two images. Three books were oriented at different angles and two images were captured. Fig. 8 shows the images with re-projected lines and a perspective view of the reconstruction.

6. Conclusion and Discussion

We have presented a framework for the computation of the relative motion between two images using a triplet of lines under minimal assumptions. We show how our algorithm can be used in a RANSAC framework to detect triplets of lines satisfying this property and compute the relative pose robustly. The performance of the algorithm was evaluated using synthetic and real datasets.

We focused in our approach on how to use the existing structural constraints to improve the relative pose estimation. We proved and demonstrated experimentally that the relative rotation can be computed in the case of zero baseline. In indoor environments, which are abundant in lines and scarce in points, we demonstrated that our algorithms outperforms point based methods. Finally, our algorithm can be used as part of any SfM pipeline to make it suitable for indoor environments.

References

[1] M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *ICPR*, volume 3, pages 1063 –1068, 2006.

[2] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *CVIU*, 100(3):416–441, Dec. 2005.

[3] M. Chandraker, J. Lim, and D. J. Kriegman. Moving in Stereo: Efficient Structure and Motion Using Lines. In *ICCV*, Kyoto, Japan, 2009.

[4] A. Criminisi, I. Reid, and A. Zisserman. Single View Metrology. *IJCV*, 40(2):123–148, 2000.

[5] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, volume 2, pages 1403–1410, 2003.

[6] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera Self-Calibration: Theory and Experiments. In *ECCV*, pages 321–334, London, UK, 1992. Springer-Verlag.

[7] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[8] A. Fitzgibbon, G. Cross, and A. Zisserman. Automatic 3D model construction for turn-table sequences. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, SMILE, pages 155–170, 1998.

[9] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *ECCV*, volume 1406, pages 311–326, 1998.

[10] Y. Furukawa, B. Curless, and S. Seitz. Manhattan-world stereo. In *CVPR*, pages 1422–1429, 2009.

[11] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[12] R. Grompone Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: a fast line segment detector with a false detection control. *PAMI*, 32(4):722–732, Apr. 2010.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[14] R. I. Hartley. A linear method for reconstruction from lines and points. In *ICCV*, pages 882–887, June 1995.

[15] R. I. Hartley. Lines and Points in Three Views and the Trifocal Tensor. *IJCV*, 22(2):125–140, 1997.

[16] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, Nov. 2007.

[17] J. Kosecka. Extraction, matching and pose recovery based on dominant rectangular structures. *CVIU*, 100(3):274–293, 2005.

[18] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *PAMI*, 26(6):756–777, 2004.

[19] D. Nistér, O. Naroditsky, and J. Bergen. Visual Odometry. In *CVPR*, volume 1, pages 652–659. IEEE, 2004.

[20] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed Real-Time Urban 3D Reconstruction from Video. *IJCV*, 78(2-3):143–167, 2008.

[21] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In *ICRA*, pages 4293–4299. IEEE, May 2009.

[22] P. Smith, I. Reid, and A. Davison. Real-time monocular SLAM with straight lines. In *BMVC*, 2006.

[23] M. Spetsakis and J. Y. Aloimonos. A multi-frame approach to visual motion perception. *IJCV*, 6(3):245–255, 1991.

[24] H. Stewenius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, June 2006.

[25] C. J. Taylor and D. J. Kriegman. Structure and Motion from Line Segments in Multiple Images. *PAMI*, 17(11):1021–1032, 1995.

[26] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *PAMI*, 14(3):318–336, Mar. 1992.

[27] T. Werner and A. Zisserman. New Techniques for Automated Architectural Reconstruction from Photographs. In *ECCV*, pages 541–555, London, UK, 2002. Springer-Verlag.

[28] Z. Zhang. Determining the Epipolar Geometry and its Uncertainty: A Review. *IJCV*, 27(2):161–195, 1998.

[29] Z. Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In *ICCV*, volume 1, page 666, Los Alamitos, CA, USA, 1999. IEEE Computer Society.