Moving in Stereo: Efficient Structure and Motion Using Lines

Manmohan Chandraker[*] mkchandraker@cs.ucsd.edu

Jongwoo Lim[†] JLim@hra.com

David Kriegman[*] kriegman@cs.ucsd.edu

[*] University of California, San Diego
La Jolla, CA

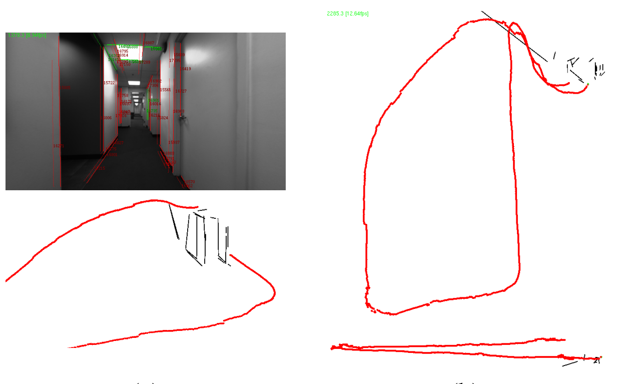
[†] Honda Research Institute, USA Inc.
Mountain View, CA

Abstract

We present a fast and robust system for estimating structure and motion using a stereo pair, with straight lines as features. Our first set of contributions are efficient algorithms to perform this estimation using a few (two or three) lines, which are well-suited for use in a hypothesize-and-test framework. Our second contribution is the design of an efficient structure from motion system that performs robustly in complex indoor environments.

By using infinite lines rather than line segments, our approach avoids the issues arising due to uncertain determination of end-points. Our cost function stems from a rank condition on planes backprojected from corresponding image lines. We propose a framework that imposes orthonormality constraints on the rigid body motion and can perform estimation using only two or three lines, through efficient solution of an overdetermined system of polynomials. This is in contrast to simple approaches which first reconstruct 3D lines and then align them, but perform poorly in real-world scenes with narrow baseline stereo.

Experiments using synthetic as well as real data demonstrate the speed, accuracy and reliability of our system.



(a) (b)

Figure 1. Camera motion estimated using the line-based structure-from-motion algorithms proposed in this paper, with narrow baseline stereo and *without* any bundle adjustment. The scene is an office environment and the length of the loop is about 50 meters. The camera starts inside a room, moves outside a door and returns to the room through the same door. Notice the accuracy in closing the loop (about 2% accumulated error after 1600 frames) and in recovering challenging purely forward motions. (a) An intermediate snapshot of line tracks and recovered structure and motion at that stage. (b) Top and side view of recovered motion for entire loop. Please see the text and the accompanying videos for details.

1. Introduction

Simultaneous estimation of the structure of a scene and the motion of a camera rig using only visual input data is a classic problem in computer vision. Interest points, such as corners, are the primary features of interest for sparse reconstruction, since they abound in natural scenes. Consequently, significant research efforts in recent times have been devoted to sparse 3D reconstructions from points [14].

However, it is often the case, especially in man-made scenes like office interiors or urban cityscapes, that not enough point features can be reliably detected, while an abundant number of lines are visible. While structure from motion (SFM) using lines has, in the past, received considerable attention, not much of it has been towards developing algorithms usable in a robust real-time system. On the other hand, important breakthroughs in solving minimal problems for points, such as [11, 13], have led to the emergence of efficient point-based reconstruction methods.

This paper proposes algorithms for estimating structure and motion using lines which are ideal for use in a hypothesize-and-test framework [4] crucial for designing a robust, real-time system. In deference to our intended application, the visual input for our algorithms stems from a calibrated stereo pair, rather than a monocular camera. This has the immediate advantage that only two stereo views (as opposed to three monocular views) are sufficient to recover the structure and motion using lines. The cost functions and representations we use in this paper are adapted to exploit the benefits of stereo input. Our entire system, unoptimized at the time of writing, captures stereo images, detects straight lines, tracks them and estimates the structure and motion in complex scenes at 10 frames per second.

Besides availability in man-made scenes, an advantage of using lines is accurate localization due to their multi-pixel support. However, several prior works rely on using finite line segments for structure and motion estimation, which has the drawback that the quality of reconstruction is limited by the accuracy of detecting end-points. In the worst case scenario, such as a cluttered environment where lines are often occluded, the end-points might be false corners or unstable T-junctions. Our approach largely alleviates these issues by using infinite lines as features and employing a problem formulation that depends on backprojected planes rather than reprojected end-points.

Commonly used multifocal tensor based approaches involve extensive book-keeping and use a larger number of line correspondences to produce a linear estimate, which can be arbitrarily far from a rigid-body motion in the presence of noise. In contrast, our solutions use only two or three lines and can account for orthonormality constraints on the recovered motion. Rather than resorting to a brute force nonlinear minimization, we reduce the problem to a small polynomial system of low degree, which can be solved expeditiously.

In theory, the use of calibrated stereo cameras also makes it possible to come up with simpler solutions that first reconstruct the 3D lines for each stereo pair and then compute a rotation and translation to align them. However, as we discuss in Section 3.1 and demonstrate in Section 5, relying on alignment of noisy 3D lines obtained from narrow baseline stereo is especially prone to errors in real-life scenarios.

In summary, the main contributions of this paper are:

- Efficient algorithms for stereo-based line SFM that are tailored for use in a hypothesize-and-test framework and are more robust than traditional approaches.
- A system to detect straight lines, track them and estimate structure and motion in a RANSAC framework (with optional nonlinear refinement) at very fast rates.

We begin with a brief overview of relevant prior work in Section 2. Our algorithms for estimating structure and motion are described in Section 3, while the system details such as line detection and tracking are provided in Section 4. Synthetic and real data experiments are presented in Section 5 and we finish with discussions in Section 6.

2. Related Work

It can be argued that points provide "stronger" constraints on a reconstruction than lines [5]. For instance, an analog of the eight-point algorithm for two-view reconstruction using lines requires three views and thirteen correspondences [9]. Yet, using lines as primary features is sometimes beneficial, so there has been tremendous interest in the problem.

Multifocal tensors are a common approach to estimate the structure of a scene from several line correspondences. A linear algorithm for estimating the trifocal tensor was presented in [5], while the properties of the quadrifocal tensor are elucidated in [18]. It is apparent from these works that the book-keeping required to enforce non-linear dependencies within the tensor indices can be substantial [6][8]. Consequently, approaches such as [3] which compute the quadrifocal tensor between two calibrated stereo pairs, are too cumbersome for estimating just a 6-dof motion.

An algebraic construct is used to align the Plücker coordinates of two line reconstructions in [1], however, rigid body constraints can be accounted for only through nonlinear minimization. Extended Kalman Filter based techniques for estimating incremental displacements between stereo frames have also been proposed [21].

For various reasons, it is not reliable to use end-points for designing a structure and motion algorithm with lines. The issue of representing 3D lines is addressed in detail in [2]. The method in [20] minimizes a notion of reprojection error in the image plane which is robust to variable end-points across views. However, the cost function can be optimized only by iterative local minimization which requires initialization and is not amenable to a real-time implementation. Further, a minimum of six line correspondences in three views are required, which can be burdensome for a hypothesize-and-test framework. A similar procedure for 3D reconstruction of urban environments is presented in [17].

In recent years, there has been significant progress in the development of visual odometry systems [12], which rely on real-time solutions to minimal problems for point-based SFM such as 5-point relative orientation [11, 19, 13]. Similar to those works, our focus is to develop an efficient solution with lines that can be used in a RANSAC framework, but this leads to an over-determined system of polynomials. An over-constrained system, albeit an easier one, is solved in [16] for the pose estimation problem.

3. Structure and Motion Using Lines

Unless stated otherwise, 3D points are denoted as homogeneous column vectors $\mathbf{X} \in \mathbb{R}^4$ and 2D image points by homogeneous column vectors $\mathbf{x} \in \mathbb{R}^3$. The perspective camera matrix is represented by a 3×4 matrix $\mathtt{P} = \mathtt{K}[\mathtt{R}|\mathbf{t}]$, where $\mathtt{K}$ is the 3×3 upper triangular matrix that encodes the internal parameters of the camera and $(\mathtt{R}, \mathbf{t})$ represents the exterior orientation. We refer the reader to [7] for the relevant details of projective geometry and to [15] for an excellent treatment of 3D line geometry.

3.1. A Simple Solution?

Since we use calibrated stereo cameras, a first approach to the problem would naturally be to reconstruct the 3D lines and align them. There are several different ways to achieve this, one possibility is the following. First, reconstruct the 3D lines in the frame of each stereo pair - this is easily achieved by intersecting the planes obtained by backprojection of each corresponding pair of 2D lines. Next, the rotation that aligns the 3D lines is computed, for instance, by first aligning the mean directions of the lines and then computing the residual in-plane rotation. Finally, the translation can be computed by aligning the points that lie midway between the 3D lines in the two rotationally aligned frames.

The attractions of such approaches are that they are simple and geometrically intuitive. However, the quality of reconstruction obtained from such methods is critically dependent on the accuracy of reconstructed 3D lines, which is known to be quite poor for the practically important case of narrow baseline stereo. Moreover, if some scene lines lie close to the epipolar plane, then such strategies are liable to break down.

The algorithms that we describe next perform robustly in real-world scenes where situations such as above commonly arise. The need for stereo-based line reconstruction algorithms that look beyond the simple multistep reconstruction and alignment approach is also empirically borne out by the experiments in Section 5.

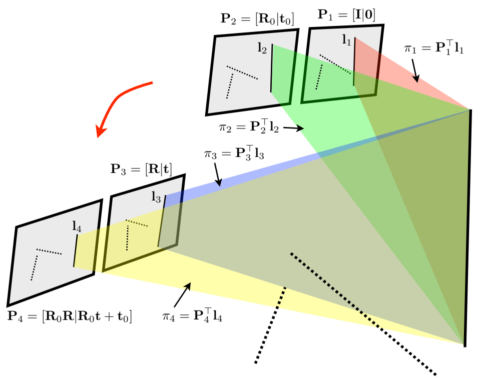
3.2. Geometry of the Problem

We will assume that the stereo pair is calibrated, so the intrinsic parameter matrices, $\mathtt{K}$, can be assumed to be identity. The left and right stereo cameras can be parametrized as $\mathtt{P}_1 = [\mathtt{I}|\mathbf{0}]$ and $\mathtt{P}_2 = [\mathtt{R}_0|\mathbf{t}_0]$, where $\mathtt{R}_0$ is a known rotation matrix and $\mathbf{t}_0$ is a known translation vector. Then, the camera matrices in a stereo pair related by a rigid body motion $(\mathtt{R}, \mathbf{t})$ are given by $\mathtt{P}_3 = [\mathtt{R}|\mathbf{t}]$ and $\mathtt{P}_4 = [\mathtt{R}_0\mathtt{R}|\mathtt{R}_0\mathbf{t} + \mathbf{t}_0]$.

The back-projected plane through the camera center and containing the 3D line $\mathbf{L}$, imaged as a 2D line $\mathbf{l}$ by a camera $\mathtt{P}$, is given by $\boldsymbol{\pi} = \mathtt{P}^\top\mathbf{l}$. Suppose $\mathbf{l}_1$, $\mathbf{l}_2$, $\mathbf{l}_3$ and $\mathbf{l}_4$ are images of the same 3D line in the four cameras $\mathtt{P}_1, \cdots, \mathtt{P}_4$. See Figure 2. Then, the four back-projected planes, stacked row-wise, form a 4×4 matrix which is the primary geometric object of interest for our algorithms:

$$\mathtt{W} = \begin{bmatrix} \boldsymbol{\pi}_1^\top \\ \boldsymbol{\pi}_2^\top \\ \boldsymbol{\pi}_3^\top \\ \boldsymbol{\pi}_4^\top \end{bmatrix} = \begin{bmatrix} \mathbf{l}_1^\top & 0 \\ \mathbf{l}_2^\top\mathtt{R}_0 & \mathbf{l}_2^\top\mathbf{t}_0 \\ \mathbf{l}_3^\top\mathtt{R} & \mathbf{l}_3^\top\mathbf{t} \\ \mathbf{l}_4^\top(\mathtt{R}_0\mathtt{R}) & \mathbf{l}_4^\top(\mathtt{R}_0\mathbf{t} + \mathbf{t}_0) \end{bmatrix} \tag{1}$$

Since the four backprojected planes in Figure 2 intersect in a straight line, there are two independent points $\mathbf{X}_1$ and $\mathbf{X}_2$ that satisfy $\boldsymbol{\pi}_i^\top\mathbf{X}_j = 0$, for $i = 1, 2, 3, 4$ and $j = 1, 2$. Consequently, the matrix $\mathtt{W}$ has a 2D null-space, or has rank 2, when the lines in the 4 images correspond.



$\mathbf{P}_2 = [\mathbf{R}_0|\mathbf{t}_0]$
$\mathbf{P}_1 = [\mathbf{I}|\mathbf{0}]$
$\pi_1 = \mathbf{P}_1^\top\mathbf{l}_1$
$\mathbf{l}_2$
$\mathbf{l}_1$
$\pi_2 = \mathbf{P}_2^\top\mathbf{l}_2$
$\pi_3 = \mathbf{P}_3^\top\mathbf{l}_3$
$\mathbf{P}_3 = [\mathbf{R}|\mathbf{t}]$
$\mathbf{l}_3$
$\mathbf{l}_4$
$\mathbf{P}_4 = [\mathbf{R}_0\mathbf{R}|\mathbf{R}_0\mathbf{t} + \mathbf{t}_0]$
$\pi_4 = \mathbf{P}_4^\top\mathbf{l}_4$

Figure 2. Geometry of image formation. If the 4 image lines in each of the 4 cameras of the 2 stereo pairs actually correspond, the back-projected planes joining the respective camera centers to these straight lines would intersect in a straight line (the 3D line which was imaged by the 2 stereo pairs).

3.3. Linear Solution

Asserting that $\mathtt{W}$ has rank 2 is equivalent to demanding that each of its 3×3 sub-determinants have rank 2, which amounts to four independent constraints. One way to extract these constraints is to perform two steps of Gauss-Jordan elimination (using Househölder rotations) on the matrix $\mathtt{W}^\top$, to get a system of the form

$$\mathtt{W}_{gj} = \begin{bmatrix} \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 \\ \times & \times & f_1(\mathtt{R}, \mathbf{t}) & f_2(\mathtt{R}, \mathbf{t}) \\ \times & \times & f_3(\mathtt{R}, \mathbf{t}) & f_4(\mathtt{R}, \mathbf{t}) \end{bmatrix} \tag{2}$$

where $f_i(\mathtt{R}, \mathbf{t})$ are affine functions of $\mathtt{R}$ and $\mathbf{t}$. Since rank of a matrix is preserved by elementary operations, the matrix $\mathtt{W}_{gj}$ must also be rank 2 and thus, its third and fourth rows are linear combinations of the first two. It easily follows that

$$f_i(\mathtt{R}, \mathbf{t}) = 0\,, \quad i = 1, 2, 3, 4 \tag{3}$$

Thus, for $n \geq 3$, a linear estimate for the motion can then be obtained as a solution to a $4n \times 12$ system of the form $\mathtt{A}\mathbf{v} = \mathbf{b}$, formed by stacking up the linear constraints f_i, where $\mathbf{v} = [\mathbf{r}_1^\top\mathbf{r}_2^\top\mathbf{r}_3^\top\mathbf{t}^\top]^\top$ and $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ are columns of the rotation matrix.

The drawbacks that such a linear procedure suffers from in the presence of noise are similar to those of, say, the DLT algorithm for estimating the fundamental matrix using eight points. In particular, since orthonormality constraints have been ignored, the solution can be arbitrarily far from a rigid body motion.

3.4. Efficient Solutions for Orthonormality

In the frequently encountered case of narrow baseline stereo with nearly parallel camera principal axes, an additional drawback is that the matrix $\mathtt{A}$ is close to rank-deficient, which makes its inversion unstable. A more practical approach is to work with a low-rank projection of $\mathtt{A}$, that is, assume the solution lies in the space spanned by the last k singular vectors of $[\mathtt{A}, -\mathbf{b}]$. While more complex than the prior methods, the following method has the advantage of being able to impose orthonormality for general camera motions.

We express the desired solution as a linear combination of the singular vectors of $\mathtt{A}$, call them $\mathbf{v}_i$:

$$\begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{t} \\ 1 \end{bmatrix} = x_1 \begin{bmatrix} | \\ | \\ \mathbf{v}_1 \\ | \\ | \end{bmatrix} + \cdots + x_k \begin{bmatrix} | \\ | \\ \mathbf{v}_k \\ | \\ | \end{bmatrix} \tag{4}$$

and the problem reduces to determining the coefficients $x_1, \cdots, x_k$ of the above linear combination, subject to orthonormality conditions:

$$\|\mathbf{r}_1\|^2 = 1\,, \qquad \|\mathbf{r}_2\|^2 = 1\,, \qquad \|\mathbf{r}_3\|^2 = 1 \tag{5}$$

$$\mathbf{r}_1^\top \mathbf{r}_2 = 0\,, \qquad \mathbf{r}_2^\top \mathbf{r}_3 = 0\,, \qquad \mathbf{r}_3^\top \mathbf{r}_1 = 0. \tag{6}$$

Substituting for $(\mathtt{R}, \mathbf{t})$ from (4) in (5) and (6), we get six polynomials of degree 2 in the k variables $x_1, \cdots, x_k$. This system of polynomial equations will have no solution in the general, noisy case and we must instead resort to a principled "least-squares" approach to extract the solution.

In general, greater numerical precision and speed can be obtained by reducing the degree of the system and the number of variables. One way to do so is to drop the scale, that is, divide each equation in (4) by x_k and replace the unit norm constraints in (5) by equal norm constraints:

$$\|\mathbf{r}_1\|^2 - \|\mathbf{r}_2\|^2 = 0\,, \qquad \|\mathbf{r}_2\|^2 - \|\mathbf{r}_3\|^2 = 0. \tag{7}$$

Now, the five polynomials in (7) and (6) have $k-1$ variables each, call them $q_i(x_1, \cdots, x_{k-1})$, for $i = 1, \cdots, 5$. Then, the corresponding reduced least squares problem is to minimize $q(x_1, \cdots, x_{k-1}) = \sum_{i=1}^{5} q_i^2(x_1, \cdots, x_{k-1})$. The case of fixed scale is a straightforward extension.

For ease of exposition, we will assume $k = 4$, but the following can be easily extended for other number of variables too. Then, each of the five polynomial equations consists of the ten monomials $[x_1^2, x_2^2, x_1x_2, x_1, x_2, x_1x_3, x_2x_3, x_3^2, x_3, 1]$. Since relatively low degree polynomials in a single variable can be solved very fast using methods like Sturm sequences, we will attempt to solve for a single variable first, say x_3. Then, each of the polynomials $q_i(x_1, x_2, x_3)$ can be rewritten as

$$c_1 x_1^2 + c_2 x_2^2 + c_3 x_1 x_2 + [1] x_1 + [1] x_2 + [2] = 0 \tag{8}$$

where we use the notation $[n]$ to denote some polynomial of degree n in the single variable x_3. Our system of polynomials, written out in a matrix format, now has the form

$$\begin{bmatrix} c_1 & c_2 & c_3 & [1] & [1] & [2] \\ c_1' & c_2' & c_3' & [1] & [1] & [2] \\ c_1'' & c_2'' & c_3'' & [1] & [1] & [2] \\ c_1''' & c_2''' & c_3''' & [1] & [1] & [2] \\ c_1'''' & c_2'''' & c_3'''' & [1] & [1] & [2] \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1x_2 \\ x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9}$$

Let the 5×6 matrix above be denoted as $\mathtt{G}$. Then, the i-th component of its null-vector, denoted as $\mathbf{u} = [x_1^2, x_2^2, x_1x_2, x_1, x_2, 1]^\top$ can be obtained as

$$u_i = (-1)^{i-1} \det(\widehat{\mathtt{G}}_i) \tag{10}$$

where $\widehat{\mathtt{G}}_i$ stands for the 5×5 matrix obtained by deleting the i-th column of $\mathtt{G}$. Thus, the vector $\mathbf{u}$ can be obtained, *up to scale*, with each of its components a polynomial of degree 4 (x_1^2, x_2^2, x_1x_2), 3 (x_1, x_2) or 2 (1) in the single variable x_3.

Now, we note that all the components of the vector $\mathbf{u}$ are not independent. In fact, in the noiseless case, they must satisfy three constraints

$$\begin{aligned} [u_4 u_5 = u_3 u_6] &: \quad (x_1) \times (x_2) = (x_1 x_2) \times (1) \\ [u_4^2 = u_1 u_6] &: \quad (x_1) \times (x_1) = (x_1^2) \times (1) \\ [u_5^2 = u_2 u_6] &: \quad (x_2) \times (x_2) = (x_2^2) \times (1) \end{aligned} \tag{11}$$

Substituting the expressions for $\mathbf{u}$ obtained from (10), we obtain three polynomials of degree 6 in the single variable x_3. Let us call these polynomials $p_i(x_3),\ i = 1, 2, 3$. Then, we have a system of three univariate polynomials that must be solved in a "least-squares" sense. We approach this as an unconstrained optimization problem

$$\min_{x_3} \ p_1^2(x_3) + p_2^2(x_3) + p_3^2(x_3). \tag{12}$$

At the minimum, the first-order derivative of the above objective function must vanish, so the optimal x_3 is a root of the polynomial

$$p(x_3) = p_1(x_3)p_1'(x_3) + p_2(x_3)p_2'(x_3) + p_3(x_3)p_3'(x_3). \tag{13}$$

Note that $p(x_3)$ is a degree 11 univariate polynomial, whose roots can be determined very fast in practice. There can be up to 11 real roots of $p(x_3)$, all of which must be tested as a candidate solution. Also worth noticing is that the odd degree of $p(x_3)$ guarantees at least one real solution.

Finally, once the candidates for x_3 have been obtained, the corresponding candidates for x_1 and x_2 are obtained by substituting the value of x_3 in the expression for $\mathbf{u}$ and reading off the values of $x_1 = \dfrac{u_4}{u_6}$ and $x_2 = \dfrac{u_5}{u_6}$. The rotation and translation can now be recovered, up to a common scale factor, by substituting in (4). We can fix scale by, for instance, fixing the determinant of $\mathtt{R}$ to 1.

3.5. Solution for Incremental Motion

For our structure from motion application, the input is a high frame-rate video sequence. So, it is reasonable to assume that the motion between subsequent frames is very small. Approximating the incremental displacement along the manifold of rotation matrices by that along its tangent space (the $so(3)$ manifold of 3×3 skew-symmetric matrices), the incremental rotation can be parametrized as $\mathtt{R} \approx \mathtt{I} + [\mathbf{s}]_\times$, where $\mathbf{s} \in \mathbb{R}^3$. Now, n lines give $8n$ linear constraints of the form (3) on the 6 unknowns $(s_1, s_2, s_3, \mathbf{t})$, which can be estimated using a singular value decomposition (SVD).

3.6. A Note on Number of Lines

It might be noticed by the careful reader, that two lines are the minimum required to fix the six degrees of freedom of the motion of the stereo pair. Indeed, all the solutions proposed in the preceding sections (except, of course, the linear solution of Section 3.3) can work with just two lines.

For a hypothesize-and-test framework, a solution that relies on the least possible amount of data is indispensable to minimize computational complexity. However, given the narrow baseline configuration we use in practice, it is desirable to use more than the minimum number of lines to enhance robustness. While we demonstrate the possibility of using only two lines in synthetic experiments, the real data experiments that we report use three lines within the RANSAC implementation. Although using even more lines will increase robustness, the trade-off will be a more expensive RANSAC procedure.

4. System Details

4.1. Line Detection, Matching and Tracking

The success of any SFM system critically depends on the quality of feature detection and tracking. To detect lines, we first construct Sobel edge images and then apply non-maximal suppression to accurately locate the peak/ridge location of both strong and weak edges. The following non-maximal suppression method is used to build the edge map:

$$I_{\mathbf{x}} = \max_{\mathbf{dx}} \ \sigma\left(\frac{S_{\mathbf{x}} - (S_{\mathbf{x}+\mathbf{dx}} + S_{\mathbf{x}-\mathbf{dx}})/2}{\max((S_{\mathbf{x}+\mathbf{dx}} + S_{\mathbf{x}-\mathbf{dx}})/2, \theta_{edge})}; s, m\right)$$

where $S_{\mathbf{x}}$ is the Sobel edge response at a pixel $\mathbf{x} = (x,y)^\top$, $\sigma(x;s,m) = \frac{1}{1+\exp(-(x-m)/s)}$ is a sigmoid function, θ_{edge} is a free parameter and $\mathbf{dx} \in \{(1,0)^\top, (0,1)^\top, (1,1)^\top, (1,-1)^\top\}$ is the testing direction (Fig. 3 a).

Well-known straight line detection algorithms like Hough transform or its variants do not perform well in complex indoor scenes. Instead, motivated by divide-and-conquer schemes, we divide the edge map into small 8×8-pixel cells (Fig. 3 b) and detect edge blobs in each cell. To decide if an edge blob is likely to be a line segment, we build the convex hull of the points and check its thickness, which is approximated by its area divided by the distance between two farthest points in the blob. We maintain the convex hull and the mean and covariance of the point coordinates for each line segment. When two line segments are merged, these pieces of information can be combined without having to refer to the original points. At the next level, each cell contains 4 sub-cells of the previous level. In each cell, pairs of line segments with similar directions are merged if the thickness of the combined convex hull is within a threshold, and this process is repeated until the cell covers the entire image.

To establish stereo correspondence, we use dense stereo matching with shrunk frames and a sum-of-absolute-differences measure. Lines are tracked from one frame to the next using multi-level Lucas-Kanade optical flow [10].

Low-level image processing, including image rectification, Sobel edge computation and non-maximal suppression is implemented using the NVidia CUDA framework and runs very fast on a GPU. Our line segment detection is on CPU and runs fast, but since it is inherently parallel, a further speed-up can be obtained by implementing it on GPU. The stereo matching and multi-level optical flow computation are also implemented in CUDA.

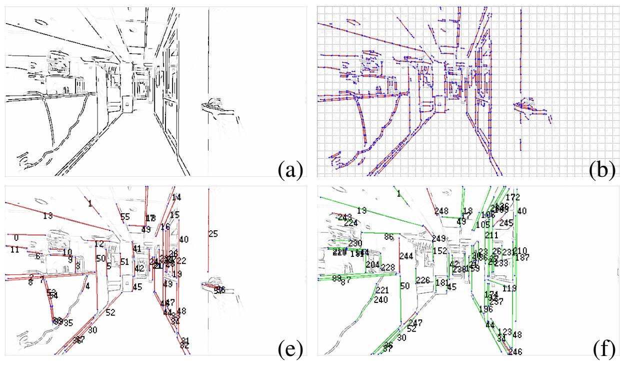


Figure 3. Line detection and tracking. (a) after non-maximal suppression. (b) line segments in initial cells (level-0). (c) detected lines after merging cells and filtering short line segments (shown with line id's). (d) tracked (green) and newly detected (red) line segments after 15 frames.

4.2. Efficiently Computing Determinants

A critical step of the algorithm in Section 3.4 is to compute the determinants of the 5×5 sub-matrices of $\mathtt{G}$. It is important to carefully design the computation of these determinants, since the number of arithmetic operations can explode very quickly, which might adversely affect the numerical behavior of the algorithm.

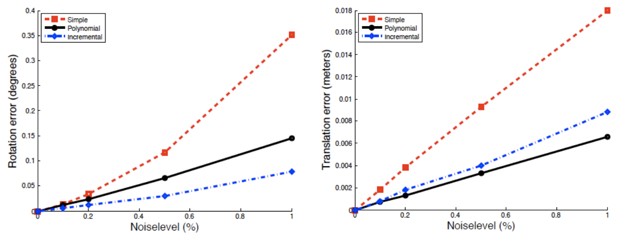

(a) Rotation error (b) Translation error

Figure 4. Rotation and translation errors with small camera motion, for simple solution (red curve), polynomial solution (black curve) and the incremental solution (blue curve), each using two lines.

As an illustration, to compute the polynomial corresponding to u_1 in (10), consider the matrix $\mathtt{G}'_1$, which is $\widehat{\mathtt{G}_1}$ with the column order reversed. Then, $\det({\mathtt{G}'_1}^\top) = \det(\widehat{\mathtt{G}_1})$. Each of the lower 4×4 sub-determinants of ${\mathtt{G}'_1}^\top$ is a polynomial of degree 2. It now requires relatively little book-keeping to determine the coefficients of $x_3^4, \cdots, x_3^0$ in u_1 by appropriately convolving the coefficients of degree 2 polynomials in the first row of ${\mathtt{G}'_1}^\top$ with those from its lower 4×4 sub-determinants. The symbolic interactions of the coefficients are pre-computed and only need to be evaluated for values of $\mathbf{v}_1, \cdots, \mathbf{v}_4$. Similar steps can be taken to compute the coefficients of various powers of x_3 in $u_2, \cdots, u_6$.

5. Experiments

We report experimental results for synthetic and real data, comparing the results obtained using the simple solution (Section 3.1), the efficient polynomial system based solution (Section 3.4) and the incremental solution (Section 3.5).

5.1. Synthetic Data

Our synthetic data experiments are designed to quantify the performance of the various algorithms discussed in this paper across various noise levels, for small and large camera motions. The stereo baseline is fixed at 0.1 units, while lines are randomly generated in the cube $[-1, 1]^3$. The first stereo pair is randomly generated and the second one is displaced with a random motion that depends on the type of experiment. Zero mean, Gaussian noise of varying standard deviations is added to the coordinates of the image lines in each view. All the experiments are based on 1000 trials.

For the first set of experiments, the motion is kept small. The performances of the simple solver, the incremental solver and the polynomial system based solver, each using the minimum two lines, are shown in Figure 4. Similar to [11], the lower quartile of error distributions are used for all the algorithms, since their targeted use is in a RANSAC framework where finding a fraction of good hypotheses is more important than consistency. It can be seen that the incremental and polynomial solver yield lower errors than the simple solver, although all three solvers perform well.

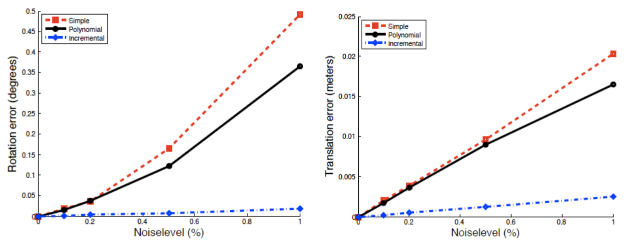

(a) Rotation error (b) Translation error

Figure 5. Rotation and translation errors with small camera motion, for simple solution (red curve), polynomial solution (black curve) and incremental solution (blue curve), each using three lines.

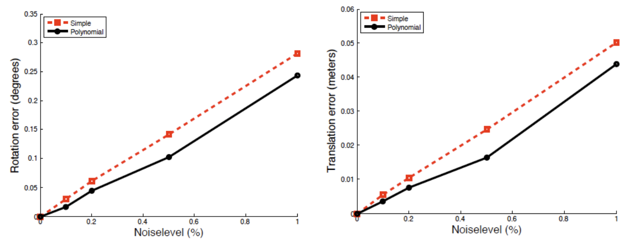

(a) Rotation error (b) Translation error

Figure 6. Rotation and translation errors with large camera motion, for the simple solution (red curve) and the polynomial solution (black curve), each using three lines.

In the second set of experiments, again for small camera motion, the performances of each of the solvers are evaluated using three lines. As discussed in Section 3, while sampling three lines is still relatively inexpensive in RANSAC, the added robustness can be valuable in a real-world application. As Figure 5 shows, the incremental solver gives much lower error rates than the other two solvers in this case.

In the next set of experiments, the camera motion is allowed to be large. The incremental solver cannot be used in this case, while the polynomial solver performs slightly better than the simple solver (Figure 6).

5.2. Real Data

While all the solvers achieve reasonable error rates for the synthetic data experiments, the utility of the algorithms proposed in this paper is evident in real-world situations, where noise is large and stereo baseline can be quite narrow compared to scene depth.

Our system is extensively tested in indoor office environments where it is intended to be deployed. The image sequences for our experiments are acquired using a stereo pair with baseline 7.4 cm, with a 640×360-pixel resolution and a $110° \times 80°$ wide field-of-view for each camera. All the experiments are conducted in a RANSAC frame-

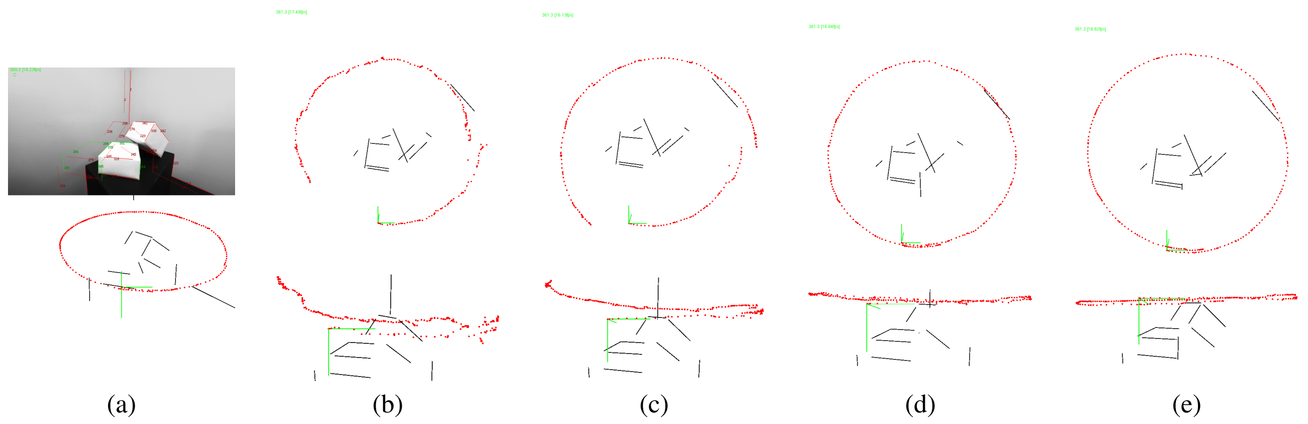


Figure 7. Structure and motion estimation for a turntable sequence. (a) The last frame overlaid with lines. Solid lines are left camera, dotted lines are correspondences in the right camera. Red lines indicate inliers of the estimated motion. (b) raw result of 3-line simple solver in RANSAC. (c) result of 3-line simple solver after (within-frame) refinement using inliers from RANSAC. (d) raw result of 3-line incremental solver in RANSAC. (e) result of 3-line incremental solver after RANSAC and within-frame refinement.

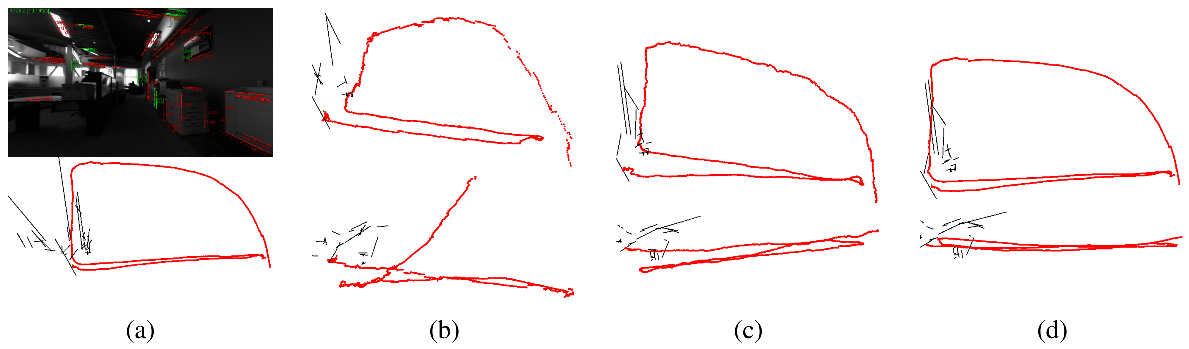


Figure 8. Structure and motion estimation for an indoor office sequence. (a) The last frame overlaid with lines. Red lines indicate inliers of the estimated motion, dotted lines are stereo correspondences. (b) Raw result from 3-line incremental solver after RANSAC. (c) Camera motion recovered from incremental solver after refinement. (d) Camera motion recovered from 3-line simple solver after nonlinear refinement using inliers and filtering out several bad estimates.

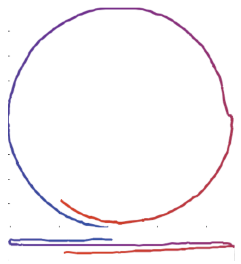
work. The RANSAC estimate can be optionally refined by a within-frame, nonlinear local refinement procedure using all the inliers. (Note that this is not bundle adjustment, as no information is aggregated over multiple frames.)

The first dataset is obtained by imaging a set of boxes rotating on a turntable. This is an interesting dataset because there are very few straight lines for estimation, with some outliers in the form of background lines that do not move with the turntable. The ground truth rotation of the turntable is slightly more than $360°$. Note that the stereo baseline here is wide relative to the scene depth.

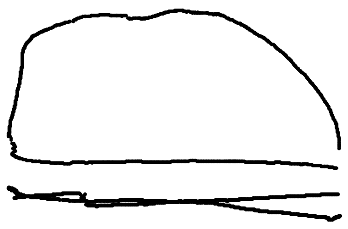
The results obtained using the various algorithms discussed in the paper are shown in Figures 7 and 9. First, it can be seen that the output of the simple solution is jerky and does not yield a complete circle, even after nonlinear refinement (Fig. 7 b, c). On the other hand, the incremental solver as well as the polynomial system solver yield fairly good reconstructions without any refinement (Fig. 7d), but almost perfect results with nonlinear refinement (Fig. 7e, Fig. 9a). These results are obtained at about 15 fps.

The next dataset involves a loop traversal in an office environment. This scene is challenging, since there are significant portions of purely forward motion and there are several false edges due to glass surfaces or short-lived tracks due to varying illumination. Most importantly, the stereo baseline is very narrow compared to the depth of scene lines (several meters). The results on one such sequence are shown in Figure 1, here, we describe the results on another sequence obtained in the same space.

The simple solution completely breaks down for this situation, which is expected as explicit reconstruction produces noisy 3D lines and aligning them results in bad motion estimates. By pruning away a large number of failed frames, it is possible to discern the trajectory, but the resulting shape is still inaccurate (Figure 8b). In comparison, the results obtained with the incremental solver closely mimic the ground



(a)



(b)

Figure 9. Top and side views of recovered camera trajectories for (a) turntable sequence (b) another corridor sequence acquired in the same work area.

truth, with no failure cases (Figure 8b,c). Note the accuracy of loop closure and the correctly recovered trajectory even in the forward motion segments of the traversal. The length of the loop is about 60 meters, with a 20 meter forward motion overlap. There are about 1750 frames in the sequence and the total error is about 1 meter. The entire system from acquisition to motion estimation runs at about 10 frames per second. RANSAC estimation of structure and motion forms only 5% of the time spent for each frame. A similar result using the polynomial solver for another loop traversal is shown in Figure 9b. Note that these results are obtained *without* any inter-frame bundle adjustment. [1]

6. Discussions

In this paper, we have reported developments in the construction of a robust, real-time stereo-based system for performing structure and motion estimation using infinite lines. One of our primary contributions is a set of fast algorithms that require only two or three lines, are more robust than simple approaches requiring explicit 3D reconstruction and are well-suited for use in a RANSAC framework. Our experiments demonstrate that the system, although not fully optimized yet, performs robustly at high frame rates in challenging indoor environments.

While bundle adjustment was not used for any of the sequences in this paper to better illustrate the properties of our algorithms, it can conceivably lead to further improvements in the motion estimates. To conclude, we note that the success of any real-time system depends on pragmatic design considerations, which we will describe in detail in an extended version of the paper.

Acknowledgments

The authors would like to thank Marc Pollefeys, Jan-Michael Frahm and Brian Clipp for several helpful discussions. Part of this work was completed when the first author was an intern at Honda Research Institute, Mountain View.

[1]Videos at `http://vision.ucsd.edu/~manu/linesfm`.

References

[1] A. Bartoli and P. Sturm. The 3D line motion matrix and alignment of line reconstructions. *IJCV*, 57(3):159–178, 2004.

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

[3] A. Comport, E. Malis, and P. Rives. Accurate quadrifocal tracking for robust 3D visual odometry. In *ICRA*, pages 40–45, 2007.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24:381–195, 1981.

[5] R. I. Hartley. Lines and points in three views and the trifocal tensor. *IJCV*, 22(2):125–140, 1997.

[6] R. I. Hartley. Computation of the trifocal tensor. In *ECCV*, pages 20–35, 1998.

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[8] A. Heyden. *Geometry and Algebra of Multiple Projective Transformations*. PhD thesis, Lund University, 1995.

[9] Y. Liu and T. S. Huang. A linear algorithm for motion estimation using straight line correspondences. *CVGIP*, 44(1):33–57, 1988.

[10] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages 121–130, 1981.

[11] D. Nistér. An efficient solution to the five-point relative pose problem. *PAMI*, 26(6):756–770, 2004.

[12] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *CVPR*, pages 652–659, 2004.

[13] D. Nistér and H. Stewénius. A minimal solution to the generalized 3-point pose problem. *JMIV*, 27(1):67–79, 2007.

[14] 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. N. 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.

[15] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer, 2001.

[16] L. Quan and Z. Lan. Linear N-point camera pose determination. *PAMI*, 21(8):774–780, 1999.

[17] G. Schindler, P. Krishnamurthy, and F. Dellaert. Line-based structure from motion for urban environments. In *3DPVT*, pages 846–853, 2006.

[18] A. Shashua and L. Wolf. On the structure and properties of the quadrifocal tensor. In *ECCV*, pages 710–724, 2000.

[19] H. Stewénius, C. Engels, , and D. Nistér. Recent developments on direct relative orientation. *ISPRS*, 60:2006, 284-294.

[20] C. J. Taylor and D. J. Kriegman. Structure and motion using line segments in multiple images. *PAMI*, 17(11):1995, 1021-1032.

[21] Z. Zhang and O. D. Faugeras. Estimation of displacements from two 3-D frames obtained from stereo. *PAMI*, 14(12):1141–1156, 1992.