Structure from Motion using Points, Lines, and Intensities

John Oliensis[1] Michael Werman[2]
[1]NEC Research Institute, 4 Independence Way, Princeton, N.J. 08540, USA
[2]Institute of Computer Science, Hebrew University, 91904 Jerusalem, Israel
oliensis@research.nj.nec.com, werman@cs.huji.ac.il

Abstract

We present a fast factorization algorithm for estimating structure and motion simultaneously *from points, lines, and/or directly from the image intensities under full perspective. It generalizes the Oliensis method for points [26] to include lines and intensities as well.*

1 Introduction

Structure from motion (SFM), the problem of reconstructing an unknown 3D scene from multiple images of it, is one of the most studied problems in computer vision. Most SFM algorithms reconstruct the scene from previously computed feature correspondences, usually tracked points [12][35][34][39][21][42][2][31][13][37][14][24][40][33][1][26]. The alternative direct–method approaches reconstruct from the images intensities without a separate stage of correspondence computation [17][11][8][7][9][36][5][6][45].

These approaches have complementary advantages and disadvantages. Usually some fraction of the image data is of such low quality that it cannot be used to determine correspondence. Feature–based methods address this problem by pre-selecting a few distinctive point or line features that are relatively easy to track, while direct methods attempt to compensate for the low quality of some of the data by exploiting the redundancy of the total data. Feature–based methods have the advantage that their input data is relatively reliable, but they neglect most of the available image information and only give sparse reconstructions of the 3D scene. Direct methods have the potential to give dense and accurate 3D reconstructions, due to their input data's redundancy, but they can be unduly affected by large errors in a fraction of the data.

This paper describes an essentially linear algorithm that combines feature–based reconstruction and direct methods. It can use feature correspondences, if these are available, and/or the image intensities directly. Unlike optical–flow approaches [18], the algorithm exploits the rigidity constraint and needs no smoothing constraint (in principle[1]) even when it uses just intensity data.

Also, the algorithm can reconstruct from both point and line features. This is important, since straight lines are abundant, especially in human–made environments, and it is often possible to track both features types. Lines can be localized very accurately due to the possibility of finding many sample points on a single line, and they do not suffer from some of the problems of point tracking such as spurious T-junctions and the aperture effect. The algorithm described here is the first that can reconstruct from all three types of input data (in any combination or separately) over a sequence of arbitrarily many images under full perspective.

Algorithms based on tracked lines include [34][35], where the original linear algorithm for 13 lines in 3 images was presented, and the optimization approaches of [38] and [32][15][16][23][44]. [1][31][24], and references therein, describe work on lines in an affine framework. [41] describes a projective method for lines and points which involves computing the projective depths from a small number of frames. Hartley [13] presented a full perspective approach that reconstructs from points and lines tracked over three images.

Due to space limitations, we only report experimental results based on point and line correspondences. Results obtained using intensity data are reported in [25].

Algorithm Overview. A preliminary stage of the algorithm recovers and compensates for the rotations. The main algorithm constructs a shift data matrix that incorporates all the data and gives the change in feature position or intensity for each image with respect to a reference image. It factors this into two rank–3 matrices using the SVD and recovers the structure and motions from these factors by exploiting the known forms of the translational and rotational displacements. Finally, it improves the initial results by iterating modified versions of the earlier steps.

2 Definitions

We first assume that the calibration is known and take the focal length to be 1 (wlog). Section 4 describes how to modify the algorithm for uncalibrated sequences.

Assume a sequence of N_I images. Choose the zeroth image as the reference image. Let R^i, $\mathbf{T}^i$ denote the rotation and translation between this image and image i. We parameterize small rotations by the rotational velocity $\omega^i \equiv \left(\omega^i_x, \omega^i_y, \omega^i_z\right)^T$. Let $\left[\mathbf{T}^i\right]_2 \equiv \left(\begin{array}{cc} T^i_x & T^i_y \end{array}\right)^T$. Let a 3D point $\mathbf{P}$ transform as $\mathbf{P}' = R\left(\mathbf{P} - \mathbf{T}\right)$.

[1]This is true assuming the motion is small and the brightness constancy equation is valid. In practice, we often do impose smoothness in the algorithm.

Assume N_p points and N_{L} lines tracked over the sequence. For clarity, we use a different notation for the the tracked points than for the pixel positions in the intensity images. Let $\mathbf{q}_m^i \equiv (q_x, q_y)_m^i$ denote the m-th tracked point and $\mathbf{A}_l^i \equiv (\alpha, \beta, \gamma)_l^{Ti}$ denote the l-th line in the i-th image. Let $\bar{\mathbf{q}} \equiv (\mathbf{q}; \mathbf{1})^T$. Let $R * \mathbf{q}$ denotes the image point obtained from $\mathbf{q}$ after a rotation: $R * \mathbf{q} \equiv \left[\begin{array}{cc} (R\bar{\mathbf{q}})_x, & (R\bar{\mathbf{q}})_y \end{array} \right]^T / (R\bar{\mathbf{q}})_z$. Define the three point rotational flows $\mathbf{r}^{(1)}(x,y)$, $\mathbf{r}^{(2)}(x,y)$, $\mathbf{r}^{(3)}(x,y)$ by

$$\begin{array}{l} \left[\begin{array}{ccc} \mathbf{r}^{(1)}, & \mathbf{r}^{(2)}, & \mathbf{r}^{(3)} \end{array} \right] \\ \equiv \left[\left(\begin{array}{c} -xy \\ -(1+y^2) \end{array} \right), \left(\begin{array}{c} 1+x^2 \\ xy \end{array} \right), \left(\begin{array}{c} -y \\ x \end{array} \right) \right]. \end{array}$$

$\hat{\mathbf{q}}_m^i$ denotes the 3D unit vector $\bar{\mathbf{q}} / |\bar{\mathbf{q}}| = (\mathbf{q}; \mathbf{1})^T / |(\mathbf{q}; \mathbf{1})|$. Similarly, $\hat{\mathbf{A}} = \mathbf{A} / |\mathbf{A}|$. Let $\mathbf{s}_m^i \equiv \mathbf{q}_m^i - \mathbf{q}_m^0$ denote the image displacement for the m–th tracked point and $\mathbf{Q}_m = \left(\begin{array}{ccc} Q_x & Q_y & Q_z \end{array} \right)_m^T$ denote the 3D scene point corresponding to the tracked point q_m.

We parameterize a 3D line $\mathbf{L}$ by two planes that it lies in. The first plane, described by $\mathbf{A}$, passes through the center of projection and the imaged line in the reference image. The normal to the second plane, $\mathbf{B}$, is defined by requiring $\mathbf{B} \cdot \mathbf{A} = 0$ and $\mathbf{B} \cdot \mathbf{Q} = -1$, for any point $\mathbf{Q}$ on $\mathbf{L}$.

Let $\mathbf{p}_n \equiv (x_n, y_n)^T$ be the image coordinates of the n–th pixel position. Order the pixels from 1 to N_X, the total number of pixels. Let I^i denote the i–th intensity image and let $I_n^i = I^i(\mathbf{p}_n)$ denote the image intensity at the n–th pixel position in I^i. Let $\mathbf{P}_n$ denote the 3D point imaged at $\mathbf{p}_n$ in the reference image, with $\mathbf{P}_n \equiv (X_n, Y_n, Z_n)^T$ in the coordinate system of I^0. Let $\mathbf{d}_n^i$ denote the shift in image position from I^0 to I^i of the 3D point $\mathbf{P}_n$.

Suppose V^a is a set of quantities indexed by the integer a. We use the notation $\{V\}$ to denote the vector with elements given by the V^a.

3 Algorithm Description

3.1 Preliminary Processing

Like the original approach of [26], the algorithm requires that the translational motion not be too large (e.g., with $|T/Z_{\min}| \leq 1/3$) and that the camera positions do not lie in a plane. (One can automatically detect planar configurations and use other, related algorithms for these situations [29][28].)

Given tracked points and lines, we approximately recover the rotations between the reference image and each following image as in [26] by minimizing:

$$\sum_m \left(\hat{\mathbf{q}}_n^i - R^i \mathbf{q}_m^0 \right)^2 + \mu \sum_l \left(\hat{\mathbf{A}}_l^i - R^i \hat{\mathbf{A}}_l \right)^2 \qquad (1)$$

with respect to the rotations R^i, where one should adjust the constant μ according to the relative accuracy of the point and line measurements.[2] [30] shows that minimizing (1) gives accurate rotations as long as the translations are not large. After recovering the rotations, we compensate for them and can henceforth treat them as small.

3.2 Points

We follow the description in [26]. The point displacements are

$$\mathbf{s}_m^i = \frac{Q_{z,m}^{-1}(\mathbf{q}_m T_z^i - [\mathbf{T}^i]_2)}{1 - Q_{z,m}^{-1} T_z^i} + \mathbf{f}(R^i, \mathbf{q}_m^i),$$

where the rotational flow $\mathbf{f}(R^i, \mathbf{q}_m^i) \equiv \mathbf{q}_m^i - (R^i)^{-1} * \mathbf{q}_m^i$. Under the assumptions, we can approximate

$$\begin{array}{l} \mathbf{s}_m^i \approx Q_{z,m}^{-1}(\mathbf{q}_m T_z^i - [\mathbf{T}^i]_2) \\ \quad + \omega_x^i \mathbf{r}^{(1)}(\mathbf{q}_m) + \omega_y^i \mathbf{r}^{(2)}(\mathbf{q}_m) + \omega_z^i \mathbf{r}^{(3)}(\mathbf{q}_m), \end{array}$$

where the last three terms give the first–order rotational flow. Correspondingly, define the three length–$2N_p$ translational flow vectors

$$\begin{array}{l} \Phi_x \equiv - \left[\begin{array}{c} \{Q_z^{-1}\} \\ \{0\} \end{array} \right], \Phi_y \equiv - \left[\begin{array}{c} \{0\} \\ \{Q_z^{-1}\} \end{array} \right], \\ \Phi_z \equiv \left[\begin{array}{c} \{q_x Q_z^{-1}\} \\ \{q_y Q_z^{-1}\} \end{array} \right], \end{array}$$

and the length–$2N_p$ rotational flows

$$\begin{array}{l} \Psi_x \equiv \left[\begin{array}{c} \left\{ r_x^{(1)}(\mathbf{q}) \right\} \\ \left\{ r_y^{(1)}(\mathbf{q}) \right\} \end{array} \right], \Psi_y \equiv \left[\begin{array}{c} \left\{ r_x^{(2)}(\mathbf{q}) \right\} \\ \left\{ r_y^{(2)}(\mathbf{q}) \right\} \end{array} \right], \\ \Psi_z \equiv \left[\begin{array}{c} \left\{ r_x^{(3)}(\mathbf{q}) \right\} \\ \left\{ r_y^{(3)}(\mathbf{q}) \right\} \end{array} \right]. \end{array}$$

Collect the shifts $\mathbf{s}_m^i$ into a $N_i \times 2N_p$ matrix $\mathcal{S}$, where each row corresponds to a different image i and equals $\left[\begin{array}{cc} \{s_x^i\}^T & \{s_y^i\}^T \end{array} \right]$. Then

$$\begin{array}{l} \mathcal{S} \approx \{T_x\} \Phi_x^T + \{T_y\} \Phi_y^T + \{T_z\} \Phi_z^T \\ \quad + \{\omega_z\} \Psi_x^T + \{\omega_y\} \Psi_y^T + \{\omega_z\} \Psi_z^T. \end{array} \qquad (2)$$

3.3 Lines

Measurement model. One can reasonably assume that the noise of a measured line is proportional to the integrated image distance between the measured and true positions of the line. Let θ_{FOV} denote the field of view (FOV) in radians. Typically, $\theta_{\mathrm{FOV}} < 1$. If the measured line differs from the true one by a z–rotation, this gives a noise of $O\left((\theta_{\mathrm{FOV}}/2)^2 \right)$. A rotation around an axis in the x–y plane gives a noise of $O(\theta_{\mathrm{FOV}})$, which is typically larger. (These estimates reflect the fact that

[2] (1) assumes a good noise model for points but a less good one for lines. (See the discussion of the line measurement model in the next section.) The advantage of using (1) is that one can compute the minimizing R^i exactly.

the displacement region bounded by the true and measured lines looks like 2 triangles in the first case and a quadrilateral in the second.) For our representation, this implies that line measurements determine A_z more accurately than A_x, A_y, by a factor roughly of $1/\theta_{\mathrm{FOV}}$.

Image flow for lines. Let $A \equiv (\mathbf{A};0)$ and $B \equiv (\mathbf{B};-1)$ be the homogeneous length–4 vectors corresponding to the plane normals $\mathbf{A}$ and $\mathbf{B}$ for the line $\mathbf{L}$. Let $Q \equiv (\mathbf{Q};1)$ be the homogeneous vector corresponding to a point on $\mathbf{L}$. Then $A \cdot B = A \cdot Q = B \cdot Q = 0$.

After a rotation R and translation T, we determine the transformed A', B' from the requirement that $A' \cdot Q' = B' \cdot Q' = 0$. We can satisfy this requirement using

$$A^* = \left[\begin{array}{c} R\mathbf{A} \\ T \cdot \mathbf{A} \end{array} \right], B^* = \left[\begin{array}{c} R\mathbf{B} \\ \mathbf{T} \cdot B + B_4 \end{array} \right].$$

But A^* doesn't necessarily have $A'_4 = 0$, as the A' for the new image of the line must. So we set $A' \equiv A^* - A^*_4 B^* / B^*_4$, which implies that the new image line is given by

$$\mathbf{A}' = R\left(\mathbf{A} - \mathbf{B}\frac{\mathbf{T}\cdot\mathbf{A}}{\mathbf{T}\cdot\mathbf{B} + B_4}\right). \tag{3}$$

Now consider the line flow $\delta\mathbf{A}^i_l = \mathbf{A}^i_l - \mathbf{A}^0_l$. Define $\delta\mathbf{A}_R \equiv \mathbf{A}^i - R^{-1}\mathbf{A}^i$ and

$$\delta\mathbf{A}_T \equiv R^{-1}\mathbf{A}^i - \mathbf{A} = \frac{(\mathbf{T}\cdot\mathbf{A})}{1 - \mathbf{T}\cdot\mathbf{B}}\mathbf{B},$$

so $\delta\mathbf{A} = \delta\mathbf{A}_T + \delta\mathbf{A}_R$. (Recall $B_4 = -1$.) For small rotations and translations, with $|\mathbf{T}\cdot\mathbf{B}| \ll 1$,

$$\delta\mathbf{A}^i \approx (\mathbf{T}\cdot\mathbf{A})\,\mathbf{B} + \omega \times \mathbf{A}. \tag{4}$$

$\mathbf{B}\cdot\mathbf{A} = \mathbf{0}$ implies that $\delta\mathbf{A}\cdot\mathbf{A} \approx 0$. To eliminate the scale ambiguity in defining each $\mathbf{A}$, we take $\left|\mathbf{A}^0_l\right| = 1$ in the reference image and require that the line flow satisfies $0 = \delta\mathbf{A}^i_l \cdot \mathbf{A}^0_l \equiv \left(\mathbf{A}^i_l - \mathbf{A}^0_l\right)\cdot\mathbf{A}^0_l$ *exactly.* We normalize all $\mathbf{A}^0$ to the same magnitude in the reference image to reflect the fact that the measurement errors should be similar for all lines. The requirement that $0 = \delta\mathbf{A}^i_l \cdot \mathbf{A}^0_l$ "fixes the gauge" in a way that is consistent with our line representation and small–motion assumption.

After "gauge fixing," each $\delta\mathbf{A}^i_l$ incorporates just two degrees of freedom. We represent $\delta\mathbf{A}^i_l$ by its projection along two directions, $\mathbf{A}_l \times (\hat{\mathbf{z}} \times \mathbf{A}_l)$ and $\hat{\mathbf{z}} \times \mathbf{A}_l$, which we refer to respectively as the *upper* and *lower* directions. Let the unit 3–vectors $\mathbf{P}^l_U$ and $\mathbf{P}^l_L$ project onto these two directions. For typical $\theta_{\mathrm{FOV}} < 1$, $|\mathbf{A}_l \cdot \hat{\mathbf{z}}| \ll 1$ and the upper component of $\mathbf{A}_l$ roughly equals $A_{l,z}$. Thus, image measurements determine the upper component more accurately than the lower, by roughly $1/\theta_{\mathrm{FOV}}$.

In analogy to the point definitions, define the line translational flows

$$\Phi_{\mathbf{L}x} \equiv \left[\begin{array}{c} \{A_x B_U\} \\ \{A_x B_L\} \end{array} \right], \Phi_{\mathbf{L}y} \equiv \left[\begin{array}{c} \{A_y B_U\} \\ \{A_y B_L\} \end{array} \right],$$

$$\Phi_{\mathbf{L}z} \equiv \left[\begin{array}{c} \{A_z B_U\} \\ \{A_z B_L\} \end{array} \right],$$

where these are length–$2N_L$ vectors. $B_U \equiv \mathbf{B}\cdot\mathbf{P}_U$ and $B_L \equiv \mathbf{B}\cdot\mathbf{P}_L$ are the upper and lower components of $\mathbf{B}$. Similarly, define the rotational flows

$$\Psi_{\mathbf{L}x} \equiv \left[\begin{array}{c} \{\mathbf{P}_U \cdot (\hat{\mathbf{x}} \times \mathbf{A})\} \\ \{\mathbf{P}_L \cdot (\hat{\mathbf{x}} \times \mathbf{A})\} \end{array} \right], \Psi_{\mathbf{L}y} \equiv \left[\begin{array}{c} \{\mathbf{P}_U \cdot (\hat{y} \times \mathbf{A})\} \\ \{\mathbf{P}_L \cdot (\hat{y} \times \mathbf{A})\} \end{array} \right], \tag{5}$$

$$\Psi_{\mathbf{L}z} \equiv \left[\begin{array}{c} \{\mathbf{P}_U \cdot (\hat{\mathbf{z}} \times \mathbf{A})\} \\ \{\mathbf{P}_L \cdot (\hat{z} \times \mathbf{A})\} \end{array} \right].$$

Let Λ be the $N_I \times 2N_{\mathbf{L}}$ matrix where each row corresponds to a different image i and equals $\left[\begin{array}{cc} \{\mathbf{P}_U \cdot \delta\mathbf{A}^i\}^T & \{\mathbf{P}_U \cdot \delta\mathbf{A}^i\}^T \end{array} \right]$. Then

$$\begin{aligned} \Lambda \approx & \{T_x\}\,\Phi^T_{\mathbf{L}x} + \{T_y\}\,\Phi^T_{\mathbf{L}y} + \{T_z\}\,\Phi^T_{\mathbf{L}z} \\ & + \{\omega_z\}\,\Psi_{\mathbf{L}x}^T + \{\omega_y\}\,\Psi_{\mathbf{L}y}^T + \{\omega_z\}\,\Psi_{\mathbf{L}z}^T. \end{aligned} \tag{6}$$

3.4 Intensities

One can apply the same arguments to the $\mathbf{d}^i_n$, the image shifts corresponding to the 3D point imaged at the pixel position $\mathbf{p}_n$, as to the $\mathbf{s}^i_m$ for the tracked feature points. Thus

$$\begin{aligned} \mathbf{d}^i_n \approx & Z^{-1}_n(\mathbf{p}_n T^i_z - \left[\mathbf{T}^i\right]_2) \\ & + \omega^i_x \mathbf{r}^{(1)}(\mathbf{p}_n) + \omega^i_y \mathbf{r}^{(2)}(\mathbf{p}_n) + \omega^i_y \mathbf{r}^{(3)}(\mathbf{p}_n). \end{aligned}$$

Let $\nabla I_n \equiv \nabla I(\mathbf{p}_n) \equiv (I_x, I_y)^T_n$ represent the gradient of the image intensity for I^0, with some appropriate definition for a discrete grid of pixels. Let ΔI^i_n define the change in intensity with respect to the reference image. Its simplest definition is $\Delta I^i_n = I^i_n - I^0_n$. The brightness constancy equation is

$$\Delta I^i_n + \nabla I_n \cdot \mathbf{d}^i_n = 0. \tag{7}$$

This and the previous equation imply that

$$\begin{aligned} -\Delta I^i_n = \nabla I_n \cdot \mathbf{d}^i_n \approx & Z^{-1}_n(\nabla I_n \cdot \mathbf{p}_n T^i_z - \nabla I_n \cdot \left[\mathbf{T}^i\right]_2) \\ & + \nabla I_n \cdot \left(\omega^i_x \mathbf{r}^{(1)}_n + \omega^i_y \mathbf{r}^{(2)}_n + \omega^i_y \mathbf{r}^{(3)}_n\right). \end{aligned}$$

Define the three length–N_X translational flow vectors

$$\begin{aligned} & \Phi_{\mathbf{I}x} \equiv -\left\{Z^{-1} I_x\right\}, \Phi_{\mathbf{I}y} \equiv -\left\{Z^{-1} I_y\right\}, \\ & \Phi_{\mathbf{I}z} \equiv \left\{Z^{-1}\left(\nabla I \cdot \mathbf{p}\right)\right\}, \end{aligned}$$

and the length–N_X rotational flows

$$\begin{aligned} & \Psi_{\mathbf{I}x} \equiv \left\{\nabla I \cdot \mathbf{r}^{(1)}(\mathbf{p})\right\}, \Psi_{\mathbf{I}y} \equiv \left\{\nabla I \cdot \mathbf{r}^{(2)}(\mathbf{p})\right\}, \\ & \Psi_{\mathbf{I}z} \equiv \left\{\nabla I \cdot \mathbf{r}^{(3)}(\mathbf{p})\right\}. \end{aligned}$$

Let Δ be a $N_I \times N_X$ matrix, where each row corresponds to an image i and equals $\left\{\Delta I^i\right\}^T$. Then

$$\begin{aligned} \Delta \approx & \{T_x\}\,\Phi^T_{\mathbf{I}x} + \{T_y\}\,\Phi^T_{\mathbf{I}y} + \{T_z\}\,\Phi^T_{\mathbf{I}z} \\ & + \{\omega_z\}\,\Psi_{\mathbf{I}x}^T + \{\omega_y\}\,\Psi_{\mathbf{I}y}^T + \{\omega_z\}\,\Psi_{\mathbf{I}z}^T. \end{aligned} \tag{8}$$

3.5 Reconstruction

Define the *total* rotational flow vectors for points, lines, and image intensities by:

$$\bar{\Psi}_x^T \equiv \left[\begin{array}{ccc} \Psi_x^T & w_{\mathbf{L}} \Psi_{\mathbf{L}x}^T & w_{\mathbf{I}} \Psi_{\mathbf{Ix}}^T \end{array} \right],$$

and similarly for the y and z components. Here $w_{\mathbf{L}}$ and $w_{\mathbf{I}}$ are constant weights that the user should set according to the relative noisiness of the point, line, and intensity data. The $\bar{\Psi}_a$ have length $2N_p + 2N_{\mathbf{L}} + N_X \equiv N_{\text{tot}}$. One can verify from the definitions that the $\bar{\Psi}$ are computable from measured quantities and can be taken as known. Using Householder matrices, one can compute as in [26] a $(N_{\text{tot}} - 3) \times N_{\text{tot}}$ matrix $\mathbf{H}$ annihilating the three $\bar{\Psi}_{x,y,z}$. Computing and multiplying by $\mathbf{H}$ cost $O(N_{\text{tot}})$ computation.

Define the $N_I \times N_{\text{tot}}$ matrix

$$\bar{\mathbf{D}} \equiv \left[\begin{array}{ccc} \mathcal{S} & w_{\mathbf{L}} \Lambda & w_{\mathbf{I}} \Delta \end{array} \right].$$

Let C be a constant $(N_I - 1) \times (N_I - 1)$ matrix with $C_{ii'} \equiv \delta_{ii'} + 1$. (As discussed in [26], we include C to counter the bias due to singling out the reference image for special treatment.) Define

$$\bar{\mathbf{D}}_{CH} \equiv C^{-1/2} \bar{\mathbf{D}} \mathbf{H}^T.$$

Define the *total* translational flow vectors by

$$\bar{\Phi}_x \equiv \left[\begin{array}{c} \Phi_x \\ w_{\mathbf{L}} \Phi_{\mathbf{L}x} \\ w_{\mathbf{I}} \Phi_{\mathbf{Ix}} \end{array} \right],$$

and similarly for y and z. (2), (6), and (8) imply that

$$\bar{\mathbf{D}}_{CH} \approx C^{-1/2} \{T_x\} \bar{\Phi}^T \mathbf{H}^T + C^{-1/2} \{T_y\} \bar{\Phi}_y^T \mathbf{H}^T + C^{-1/2} \{T_z\} \bar{\Phi}_z^T \mathbf{H}^T. \quad (9)$$

$\bar{\mathbf{D}}_{CH}$ is approximately rank 3. Our basic algorithm is:

1. Define $\mathbf{H}$ and compute $\bar{\mathbf{D}}_{CH}$. Using the singular value decomposition (SVD), compute the best rank–3 factorization of $\bar{\mathbf{D}}_{CH} \approx M^{(3)} S^{(3)T}$, where $M^{(3)}$, $S^{(3)}$ are rank 3 matrices corresponding respectively to the motion and structure.

2. $S^{(3)}$ satisfies

   $$\left[\begin{array}{ccc} \bar{\Phi}_x & \bar{\Phi}_y & \bar{\Phi}_z \end{array} \right] = \mathbf{H}^T S^{(3)} U + \left[\begin{array}{ccc} \bar{\Psi}_x & \bar{\Psi}_y & \bar{\Psi}_z \end{array} \right] \Omega, \quad (10)$$

   where U and Ω are unknown 3×3 matrices. We eliminate the unknowns Q_z, $\mathbf{B}$, and Z^{-1} from the $\bar{\Phi}_a$ in this system of equations to get $3N_{\text{tot}}$ linear constraints on the 18 unknowns in U and Ω. We solve these constraints with $O(N_{\text{tot}})$ computation using the SVD.

3. Given U and Ω, we recover the structure unknowns Q_z, $\mathbf{B}$, and Z^{-1} from (10).

4. Given U, we use $S^{(3)} U \approx \left[\begin{array}{ccc} \bar{\Phi}_x & \bar{\Phi}_y & \bar{\Phi}_z \end{array} \right]$ and (9) to recover the translations.

5. We recover the rotations ω_x^i, ω_y^i, ω_z^i from

   $$\omega_x^i \bar{\Psi}_{xn} + \omega_y^i \bar{\Psi}_{yn} + \omega_z^i \bar{\Psi}_{zn} = C^{-1/2} \bar{\mathbf{D}}_n^i - \left(C^{-1/2} \left(\{T_x\} \bar{\Phi}^T + \{T_y\} \bar{\Phi}_y^T + \{T_z\} \bar{\Phi}_z^T \right) \right)_n^i$$

3.6 Comments.

The description above omits some steps which are important for bias correction. 1) We weight the upper line components in $\bar{\mathbf{D}}$ by a factor proportional to $1/\theta_{\text{FOV}}$, to account for the greater accuracy in the measurement of these components. 2) To correct for bias due to the fact that the FOV is typically small, in Steps 2–4 we weight T_z^i by a factor proportional to the FOV and weight $\bar{\Phi}_z$ by the inverse of this, while leaving the x and y components untouched. See [28] for a similar bias correction in a point–based algorithm. 3) As described in [26], one can iterate the steps of the algorithm to give better results and correct for the small motion assumptions. This involves multiplying the original feature point shifts by $1 - Z^{-1} T_z$ and the line shifts by $1 - \mathbf{B} \cdot \mathbf{T}$. The algorithm is guaranteed to converge to the correct reconstruction if the motion and noise are small and the camera positions do not lie on a plane [26].

Of course, the results for the intensity–based part of our algorithm depend crucially on the technique used for computing derivatives. To make this more robust, one should iteratively reduce the size of the displacements $\mathbf{d}_n^i$ by warping and recomputing the reconstruction in a coarse–to–fine mode [3][4][20]. Our current preliminary implementation simply computes the image derivatives at a single scale, using consistency between the spatial derivatives computed for different images to determine whether the assumption of small image motion holds for this current scale. Only those pixel sites where the assumption does hold are used for the motion computation.

We have sometimes found it useful to preprocess the image intensities prior to running the algorithm. We first compute a Laplacian image and then transform the intensities by a sigma function to enhance edgy regions and suppress textureless ones. This gives the intensity images a form that is intermediate between the unprocessed images and a selected set of tracked features.

Comparison to the Irani approach [19]. The part of our algorithm that reconstructs directly from the image intensities is related to [19]. [19] writes the brightness constancy equation (7) in matrix form as $\Delta = -\mathbf{DI}$,where $\mathbf{D}$ tabulates the shifts $\mathbf{d}_n^i$ and $\mathbf{I}$ contains the intensity gradients $\nabla I(\mathbf{p}_n)$. [19] notes that $\mathbf{D}$ has rank 6 (for a camera with known calibration), which implies that Δ must have rank 6. To reduce the effects of noise, [19] projects the observed Δ onto one of rank 6.

[19] then applies a multi–image form of the Lucas–Kanade approach to recovering optical flow [22], which yields a matrix equation $\mathbf{D}\mathbf{I}_2 = -\Delta_2$,where the entries of $\mathbf{I}_2$ are squared intensity gradients $I_a I_b$ summed over the "smoothing" windows, and the entries of Δ_2 have the form $I_a \Delta I$. Due to the added Lucas-Kanade smoothing constraint, the shifts $\mathbf{D}$ or $\mathbf{d}_n^i$ can be computed as $\mathbf{D} = -\Delta_2 / [\mathbf{I}_2]^+$,where $[\mathbf{I}_2]^+$ denotes the pseudo–inverse, except in smoothing windows where the image intensity is constant in at least one direction. Using the rank constraint on $\mathbf{D}$, [19] determines additional entries of $\mathbf{D}$ for the windows where the intensity is constant in one direction.

The method of [19] differs significantly from ours. The essential step for recovering correspondence is a multi–frame generalization of the optical–flow approach of [22], which relies on a smoothness constraint and not on the rigidity constraint. In particular, it does not use the known form of the rotational and translational flows. [19] uses the factorization of $\mathbf{D}$, which is crucial in our method, simply to fill out the entries of $\mathbf{D}$ that could not be computed initially.

Bas-relief for lines. Due to the bas–relief ambiguity, it is difficult to recover the constant component of the vector $\{Z^{-1}\}$ for point features, though typically one can recover all other components accurately [26]. We derive a similar result for lines.

The rotational flow for a line is $\omega \times \mathbf{A}$. For typical $\theta_{\mathrm{FOV}} \ll 1$, $|A_z| \ll |A_{x,y}|$. Thus, $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ rotations give flows roughly proportional to $\hat{y} \times A \sim -A_x \hat{z}$ and $(\hat{x} \times A) \sim A_y \hat{z}$. From (4), the effects of a z–translation T_z are suppressed by $|A_z|$, and the translational flows due to the constant component of $\{B_z\}$ are also given roughly by linear combinations of the $A_x\hat{\mathbf{z}}$ and $A_y\hat{\mathbf{z}}$. Therefore, it is difficult to untangle the effects of the $\{B_z\}$ constant component from rotational effects, which makes this component difficult to recover accurately.

One can get a more quantitative analysis of this effect by arguments similar to those of [26]. Assuming that the input data consists of lines only, (10) implies that $\bar{\mathbf{P}}_S \mathbf{H} \left[\begin{array}{ccc} \bar{\Phi}_{\mathbf{L}x} & \bar{\Phi}_{\mathbf{L}y} & \bar{\Phi}_{\mathbf{L}z} \end{array} \right] = 0$, where $\bar{\mathbf{P}}_S$ is a projection matrix that annihilates $S^{(3)}$. Those $\{\mathbf{B}\}$ components that produce small overlaps $\sum_a \bar{\Phi}_{\mathbf{L}a}^T \mathbf{H}^T \mathbf{H} \bar{\Phi}_{\mathbf{L}a}$ will also produce small overlaps $\sum_a \bar{\Phi}_{\mathbf{L}a}^T \mathbf{H}^T \bar{\mathbf{P}}_S \mathbf{H} \bar{\Phi}_{\mathbf{L}a}$. It follows from standard perturbation theory [10][26] that noise will mix these components into the true solution for the $\mathbf{B}$, where the amount of contamination is inversely proportional to the size of the eigenvalue. Define $\mathcal{B} \equiv [\ \{B_x\};\ \{B_y\};\ \{B_z\}\ ]$. We computed for several sequences the eigenvalues of $\mathcal{H}_B$, defined such that $\mathcal{B}^T \mathcal{H}_B \mathcal{B} = \sum_a \bar{\Phi}_{\mathbf{L}a}^T \mathbf{H}^T \mathbf{H} \bar{\Phi}_{\mathbf{L}a}$, and found that, as expected, there was one small eigenvalue, whose eigenvector approximately equaled the constant component of $\{B_z\}$. Thus, one can use the input data to estimate the inaccuracy in recovering this component.

Our derivation suggests an improvement of our algorithm that is analogous to one of the steps of the point–based approach of [26]. [26] found that reconstructions improved significantly after recomputing separately the structure component affected by the bas–relief ambiguity. Similarly, it may be possible to improve our results for lines by recomputing the bas–relief–affected component of $\{\mathbf{B}\}$.

4 Projective Algorithm

The algorithm described here for calibrated sequences generalizes in a straightforward way to deal with uncalibrated sequences. In this projective case, instead of a preliminary stage of rotation computation, one computes planar homographies between the reference image and each subsequent image. It is worth noting that one can easily and accurately compute these homographies by a multi–frame generalization of the projective–reconstruction method of [37].

We briefly describe how this works for the example of tracked points. Assume the scene is planar. Then

$$\lambda_m^i \bar{\mathbf{q}}_m^i = M^i \mathbf{S}_m, \qquad (11)$$

where M^i is a 3×3 matrix (a homography), $\mathbf{S}_m$ is the structure 3–vector giving the position of the m–th point on the plane (in some arbitrary coordinate system), and λ_m^i is the projective depth. The steps of the homography recovery are:

1. Take $\lambda_m^i = 1$ initially.
2. For the current estimate of the λ_m^i, collect the $\lambda_m^i \bar{\mathbf{q}}_m^i$ into a single $3N_I \times N_p$ matrix Γ. Use the SVD to decompose this into the product of two rank 3 factors: $\Gamma \approx M^{(3)} S^{(3)}$.
3. For the current $M^{(3)}$, $S^{(3)}$, compute the λ_m^i minimizing $\left|\Gamma - M^{(3)} S^{(3)}\right|^2$. Return to Step 2.

After convergence, $M^{(3)}$ contains the M^i estimates. Our estimate of the homography taking the reference image to image i is $M^0 \backslash M^i$.

As shown in [26], this procedure is guaranteed to converge to a local minimum of $\left|\Gamma^{(4)}\right|^2 / \left|\Gamma\right|^2$, where $\Gamma^{(4)}$ is the difference between Γ and its best rank–3 approximation. Also, [41] shows that it gives nearly the optimal estimates of the M^i if the true motion is not too large (and the scene is truly planar).

[27] has already described the uncalibrated version of our point–based algorithm in [26]. Here we briefly describe the uncalibrated version of the line–based algorithm. Let K^i be the standard 3×3 matrix (with zero lower–diagonal entries) giving the calibration parameters for the i–th image. For the uncalibrated case, (3) still holds if one replaces $R \to (K')^{-T} RK \equiv M_H$ and $\mathbf{T} \to K\mathbf{T}$, where K and K' are the calibration matrices for the first and second image. M_H is a general homography and is the *inverse transpose* of the analogous homography for points.

The first–order approximation of the new version of (3) is (4) but with $\mathbf{T} \to K\mathbf{T}$ and with $\omega \times \mathbf{A}$ replaced by $\delta M_H \mathbf{A}$, where $M_H \equiv \mathbf{1}+ \delta M_H$.

We define $\mathbf{H}$ to annihilate the first–order homography flows due to δM_H, instead of the rotational flows as for the calibrated case. Since the first–order approximation of M_H^{-T} is $\mathbf{1}- \delta M_H^T$, one can easily define $\mathbf{H}$ to annihilate the first–order flows for both points and lines. Represent

$$\delta M_H \equiv \begin{bmatrix} F & G \\ J^T & 0 \end{bmatrix},$$

where F is 2×2 and G and J are 2×1. The first–order point and line displacements due to each of the 8 parameters in δM_H are given in the following table:

	$F_{1,1}$	$F_{1,2}$	$F_{2,1}$	$F_{2,2}$	G_1	G_2	J_1	J_2
A_x :	A_x	A_y	0	0	A_z	0	0	0
A_y :	0	0	A_x	A_y	0	A_z	0	0
A_z :	0	0	0	0	0	0	A_x	A_y
q_x :	$-x$	0	$-y$	0	$-x^2$	$-xy$	-1	0
q_y :	0	$-x$	0	$-y$	$-xy$	$-y^2$	0	-1

We define $\mathbf{H}$ to annihilate the 8 vectors associated with the columns of this table.

After the indicated replacements, the uncalibrated algorithm is the same as the calibrated one.

Projective covariance for lines. [27] showed that, if one fixes the point coordinates in some reference image, the remaining freedom under a projective transform is: $Z^{-1} \to ax+by+c+dZ^{-1}$, where x and y are the image coordinates. This corresponds to scaling and adding an arbitrary plane to the structure.

One can derive a similar relation for $\mathbf{B}$. Let $\mathbf{P}$ be a 3D point on the line determined by $\mathbf{B}$. Then $\mathbf{B}\cdot\mathbf{P} = -1$ implies

$$\mathbf{B} \cdot (x, y, 1) = -Z^{-1}.$$

A projective transform must preserve $\mathbf{B'}\cdot\mathbf{P'} = -1$. From this, and the transform of Z^{-1} given above, it follows that $\mathbf{B'} - \mathbf{B} = (a, b, c)$ up to a scaling, with the same constants a, b, c as for the point transform. This relation holds with the same constants for any line $\mathbf{B}$.

The fact that $\mathbf{B}$ is recoverable only up to an overall shift is also clear from the uncalibrated version of our algorithm. When we use $\mathbf{H}$ to eliminate the first–order effects of small homographies, we also eliminate the translational image flows due to the three constant components of the $\mathbf{B}$. Thus, these components cannot be recovered.

5 Experiments

Figure 1 shows the results of applying our algorithm for points and lines to a real image sequence taken in our laboratory. The translational motion was rather large, with max $(|\mathbf{T}| / Z) = 0.36$ and max $(|\mathbf{T}|\,|\mathbf{B}|) = 0.70$. The images were 1536×1024 pixels, with a nominal FOV of $51° \times 38°$. We tracked 41 points automatically over 7 images and also tracked 8 lines manually. The first image in Figure 1 shows the selected points and lines in the reference image. The other images in show the reprojections of the reconstructed 3D points and lines. (The diagonal line from the top left corner in all images reflects a bug of the MATLAB plotting routine and was not tracked.) The average error between the tracked and reprojected point positions was 1.57 pixels. When we applied our algorithm to the point data alone, the average reprojection error was 1.86 pixels. The angular errors between the reprojected line normals $\mathbf{A}_l^i$ and their tracked positions averaged 0.48°. Note that this sequence has nonlinear distortions, which are evident in the fit of the baseboard line.

We also ran our algorithm on the Castle sequence.[3] This sequence consists of 28 points tracked over 11 images. We converted it to a sequence of 14 tracked lines and 14 points by defining lines between the first 14 successive points. Ground truth is available for this sequence. The average fractional error in $\mathbf{B}$ for the 14 lines was 0.012, while that for the translations was 0.031. The average rotation error was 0.05°. The average fractional error in the Z was 6.6×10^{-4}.

References

[1] K. Astrom, A. Heyden, F. Kahl, M. Oskarsson, "Structure and Motion from Lines under Affine Projections," *ICCV* 285-292, 1999.

[2] A. Azarbayejani and A. Pentland, "Recursive estimation of motion, structure and focal length," **PAMI** 562–575, 1995.

[3] J. R. Bergen, P. Anandan, K. J. Hanna, R. Hingorani, "Hierarchical Model-Based Motion Estimation," *ECCV* 237–252, 1992.

[4] P. J. Burt, E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," **IEEE Trans. Communications** 31:4, 532–540, 1983.

[5] C. Fermuller, Y. Aloimonos, "On the Geometry of Visual Correspondence," **IJCV** 21:3, 223–247, 1997.

[6] C. Fermuller, Y. Aloimonos, "Direct Motion Perception," in *Visual Navigation*, Y. Aloimonos, ed., Lawrence Erlbaum, 135–177, 1997.

[7] C. Fermüller and Y. Aloimonos, " Qualitative Egomotion," **IJCV** 15, 7–29, 1995.

[8] C. Fermuller, "Passive Navigation as a Pattern-Recognition Problem," **IJCV** 14:2, 147–158, 1995.

[9] C. Fermüller and Y. Aloimonos, " Direct Perception of Three-Dimensional Motion through Patterns of Visual Motion, " **Science** 270, 1973-1976, 1995.

[10] G. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins Press, Baltimore, Maryland,1983.

[3] http://www.cs.cmu.edu/~cil/cil-ster.html

[11] K. J. Hanna, "Direct Multi-Resolution Estimation of Ego-Motion and Structure from Motion," *Motion Workshop,* Princeton, N.J., 156–162, 1991.

[12] C.G. Harris and J.M. Pike, "3D positional integration from image sequences," **IVC** 6 87–90, 1988.

[13] R. I. Hartley, "Lines and Points in Three Views and the Trifocal Tensor," **IJCV** 22, 125-140, 1997.

[14] R. I. Hartley, "In Defense of the Eight–Point Algorithm," **PAMI** 19, 580–593, 1995.

[15] R.J. Holt and A. N. Netravali, "Number of Solutions for Motion and Structure from Multiple Frame Correspondence," **IJCV** 23:1, 5–15, 1997.

[16] R.J. Holt and A. N. Netravali, "Uniqueness of Solutions to Structure and Motion from Combinations of Point and Line Correspondences," **JVCIR** 7:2, 126–136, 1996.

[17] B.K.P. Horn, and E. J. Weldon, Jr. "Direct Methods for Recovering Motion," **IJCV** 2:1, 51–76, 1988.

[18] B.K.P. Horn and B. G. Schunck, "Determining Optical Flow, **AI** 17, 185–203, 1981.

[19] M. Irani, "Multi-Frame Optical Flow Estimation using Subspace Constraints," *ICCV* 626–633, 1999.

[20] M. Irani, B. Rousso, S. Peleg, "Recovery of Ego-Motion Using Region Alignment," **PAMI** 19:3, 268–272, 1997.

[21] A.D. Jepson and D.J. Heeger, "Linear subspace methods for recovering translational direction," in *Spatial Vision in Humans and Robots*, Cambridge University Press, 39–62, 1993.

[22] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *IJCAI* 674–679, 1981.

[23] A.Mitiche, O. Faugeras, J. K. Aggarwal, "Counting Straight Lines," **CVGIP** 47:3, 353–360, 1989..

[24] D. Morris and T. Kanade, "A unified factorization algorithm for points, line segments and planes with uncertainty models," *ICCV* 696–702, 1998.

[25] J. Oliensis, "Direct Multi–Frame Structure from Motion for Hand–Held Cameras" NECI TR, 1999.

[26] J. Oliensis, "A Multi–frame Structure from Motion Algorithm under Perspective Projection," **IJCV** 34:2/3 163–192, 1999.

[27] J. Oliensis and Y. Genc, "Fast Algorithms for Projective Multi–Frame Structure from Motion," *ICCV* 536–543, 1999.

[28] J. Oliensis, "Computing the Camera Heading from Multiple Frames," *CVPR* 203–210, 1998.

[29] J. Oliensis, "Structure from Linear and Planar Motions," *CVPR* 335–342, 1996.

[30] J. Oliensis, "Rigorous Bounds for Two–Frame Structure from Motion," *ECCV* 184–195, 1996.

[31] L. Quan and T. Kanade, "Affine Structure from Line Correspondences with Uncalibrated Affine Cameras," **PAMI** 19:8 834-845, 1997.

[32] E. Salari and C. M. Jong, "A method to calculate the structure and motion parameters from line correspondences," **PR** 23 553–561, 1990.

[33] S. Soatto and P. Perona, "Reducing Structure–From–Motion A General Framework for Dynamic Vision Part 2: Implementation and Experimental Assessment," **PAMI** 20:9 943–960, 1998.

[34] M. Spetsakis "A Linear Algorithm for Point and Line-Based Structure from Motion", **CVGIP** 56:2 230-241, 1992.

[35] M. Spetsakis and Y. Aloimonos, "Structure from Motion Using Line Correspondences," **IJCV** 4:3 171–183, 1990.

[36] G. Stein, A. Shashua, "Direct Methods for Estimation of Structure and Motion from Three Views," CVPR 400–406, 1997.

[37] P. Sturm and B. Triggs, "A factorization based algorithm for multi–image projective structure and motion," *ECCV* 709–720, 1996.

[38] C.J. Taylor, D. Kriegman, "Structure and Motion from Line Segments in Multiple Images," **PAMI** 17:11 1021-1032, 1995.

[39] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: A factorization method," **IJCV** 9, 137-154, 1992.

[40] P. H. S. Torr, A. Fitzgibbon, A. Zisserman, "Maintaining Multiple Motion Model Hypotheses over Many Views to Recover Matching and Structure," *ICCV* 485–491, 1998.

[41] B. Triggs, "Factorization methods for projective structure and motion," *CVPR* 845–851, 1996.

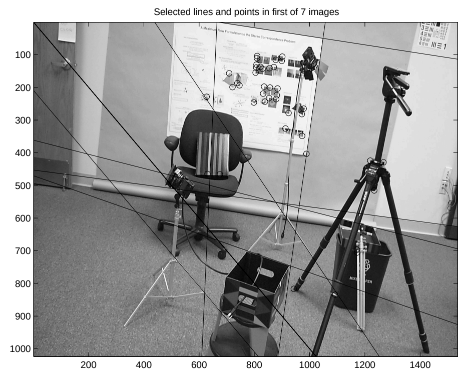
[42] Weng, J., Ahuja, N., and Huang, T.S., "Optimal Motion and Structure Estimation," **PAMI** 15:9 1993, 864-884.

[43] J. Weng, T. Huang, N. Ahuja, "Estimating Motion and Structure from Line Matches: Performance Obtained and Beyond," *ICPR*I, 168–172, 1990.

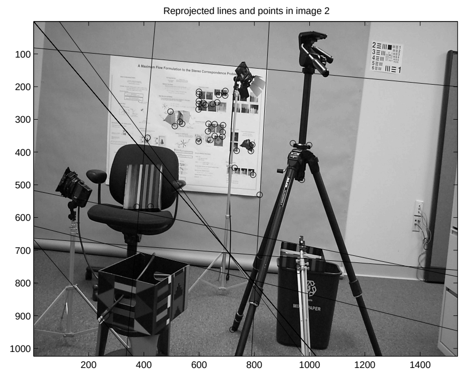
[44] Weng, J., Huang, T.S., and Ahuja, N., "Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation," **PAMI** 11:5, 1989, 451-476.

[45] L. Zelnik-Manor and M. Irani, "Multi-Frame Alignment of Planes," *CVPR* I:151–156, 1999.

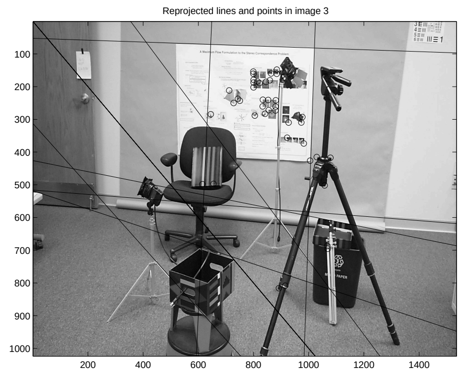
.



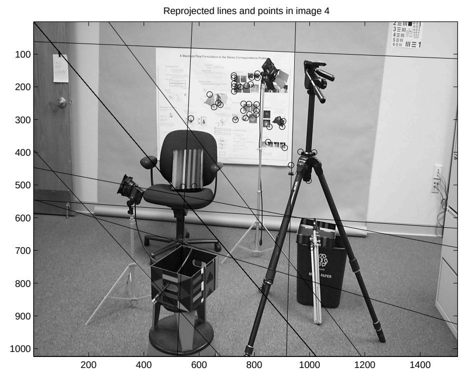
(1)



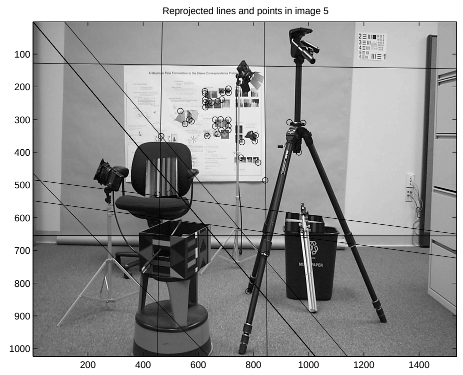
(2)



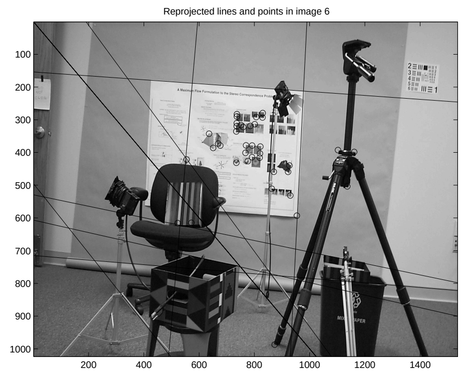
(3)



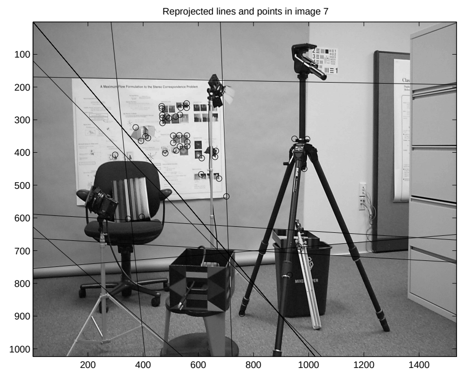
(4)



(5)



(6)



(7)

Figure 1: The seven images used are shown. The lines were marked in the first image and reprojected from the reconstructed motion and 3D structure in the other images. (The diagonal line from the top left corner is a bug of the MATLAB plotting routine.)