Fast and Accurate Algorithms for Projective Multi–Image Structure from Motion

John Oliensis (oliensis@research.nj.nec.com)
NEC Research Institute
4 Independence Way
Princeton, N.J. 08540
and
Yacup Genc
Siemens Research
Princeton, N.J. 08540

Abstract

We describe algorithms for computing projective structure and motion from a multi–image sequence of tracked points. The algorithms are essentially linear, work for any motion of moderate size, and give accuracies similar to those of a maximum–likelihood estimate. They give better results than the Sturm/Triggs factorization approach and are equally fast, and they are much faster than bundle adjustment. Our experiments show that the (iterated) Sturm/Triggs approach often fails for linear camera motions. In addition, we study experimentally the common situation where the calibration is fixed and approximately known, comparing the projective versions of our algorithms to mixed projective/Euclidean strategies. We clarify the nature of dominant–plane compensation, showing that it can be considered a small–translation approximation rather than an approximation that the scene is planar. We show that projective algorithms accurately recover the (projected) inverse depths and homographies despite the possibility of transforming the structure and motion by a projective transformation.

1 Introduction

This paper extends our previous multi–image structure–from–motion (SFM) algorithms [19][26][17] [24] from the Euclidean to the projective context. Previously, we assumed that the camera calibration was known and fixed; the new versions of our algorithms handle sequences with varying and unknown calibrations. We also adapt our algorithms for the common situation where the calibration is measured up to moderate errors and is known to be fixed.

As in our previous work, we aim for an approach that is fast and accurate on sequences with small "signal," i.e., with small (translational) image displacements. The small signal makes the reconstruction problem difficult, so to be effective an algorithm must exploit *all* the information in the sequence: we require an *intrinsically* multi–image approach [14].[1] On the other hand, the small displacements simplify the problem of finding correspondence and limit the effects of correspondence outliers on the reconstruction, since the correspondence errors cannot be large. Motivated by these factors, as in [19][26][17], we disregard the correspondence problem and present an algorithm that reconstructs from pre–tracked point features over a large number of images.

See [21][20] for extensions of our approach to a "direct method" ([5][8]) that reconstructs directly from the image intensities as well as from pre–tracked point and line data.

For small displacements, most scene points are visible in all the images. This makes it possible to use fast factorization methods, as we do here [19][26][17][31][30]. It also means that one can use multiple images to eliminate correspondences outliers. We have found that a good way to get reliable feature tracks is to compute correspondences first for image pairs and then restrict to features that are tracked consistently over all images. Eliminating features lowers the signal, so an intrinsically multi–image approach, that effectively exploits the consistent feature tracks over the *entire* sequence, becomes even more crucial.

As in our past work, we focus on scenes with a large range of depths from the camera (large perspective effects), which factorization approaches previous to our work could not handle. For shallow–depth scenes, one can use algorithms such as [31] to complement our approach. Our assumptions of large depth range and small image displacements imply that the camera motions are small, and we have designed our approaches to exploit this.

We present two classes of algorithms. One works only for motions that are "truly" general, i.e., with camera positions that do not all lie on a single plane, but we expect it to be more effective for such motions than the other approach. In [17], and below, we describe how one can determine self–consistently whether or

[1] By an intrinsically multi–frame approach (which can be either batch or recursive), we mean one that reconstructs directly from many images, rather than first reconstructing from subsets of a few images and combining the results.

not the motion is general enough for this approach to work. The second approach works for any moderate–sized motions, but it is most effective when the camera moves roughly along a line, and we propose it mainly for this type of motion. However, our experiments show that it also works well when the camera moves on a plane or makes unrestricted 3D motions.

Unlike the projective factorization approach of Sturm and Triggs [30] and its iterative extension [30][33][1][7], which we refer to as Sturm/Triggs *iterative factorization* or **STIF**, our approach needs no initial guess for the projective depths. It succeeds in situations where **STIF** fails, and it usually gives better results, whether the camera is moving on a line, on a plane, or executing a general 3D motion. Our approach factorizes a smaller matrix than that used by Sturm and Triggs and, thus, should be faster on large problems. It is much faster than bundle adjustment.

Our discussion illustrates a number of interesting theoretical points. We stress the close analogy between recovering rotations in the Euclidean context and recovering 2D projective transforms (homographies) in the projective one. We show that compensating for the dominant plane as in [9][27][11] can be understood as a small–translation approximation rather than as an approximation that the scene is planar. We show that algorithms can recover the (projected) inverse depths and homographies accurately even in projective SFM, despite the possibility of transforming the structure and motion by a projective transform.

Some of our experiments focus on the common situation where the camera calibration stays fixed over the sequence and is known approximately. In experiments with general motion, our approach gives results which are nearly as accurate as the maximum–likelihood least–squares estimates. A mixed Euclidean/projective strategy does somewhat better than a completely projective one. In experiments with linear motions, we find that a Euclidean version of our approach assuming a calibration that is slightly wrong does worse than a projective one for sideways or forward translations, but otherwise does better. Our results again compare well with those of a projective maximum–likelihood estimate (MLE).

1.1 Summary

The next section describes notation and preliminary results that we need to derive our algorithms. Section 2.2 shows that one can accurately recover the (projected) inverse depths despite the freedom to alter the structure by a projective transform. Section 2.3 derives an expression for the image displacements caused by the camera's motion and calibration changes, and Section 2.4 derives an expression for the flows due to infinitesimal 2D projective transforms, i.e., infinitesimal homographies.

Section 3 presents our general–motion algorithm in fully projective and mixed Euclidean/projective versions. Section 3.4 describes results for this algorithm. Section 4 presents our algorithm specialized for the

linear–translation case. Section 4.3 gives the results of experiments with it, and Section 4.4 explains how to extend it to handle any translational motion.

2 Preliminaries

2.1 Notation and Definitions.

(This section is intended mainly for reference; the reader may wish to skip it on a first reading.) We use MATLAB notation: a semi-colon separates entries in a column vector, a comma or space separates entries in a row vector, and a colon indicates a range of indices. We denote the average value of a quantity X by $\langle X \rangle$. If M is a matrix (of any size) with some singular values much bigger than the rest, we refer to the large singular values as the *leading* singular values, and to the corresponding singular vectors as the *leading singular vectors.*

Given a vector $\mathbf{V}$, define $[\mathbf{V}]_2$ as the length–2 vector consisting of the first two components of $\mathbf{V}$. If $\mathbf{V}$ is a 3D point, define the *ideal image point* corresponding to $\mathbf{V}$ by $\underline{\mathbf{V}} \equiv [\mathbf{V}]_2 / V_z$. Note that we take the "ideal image" to have focal length 1. If $\mathbf{v}_1$ is a 2D image point, define the 3D image point $\overline{\mathbf{v}_1}$ corresponding to it by $\overline{\mathbf{v}_1} \equiv \left[\begin{array}{cc} \mathbf{v}_1; & 1 \end{array} \right]$. If M is a 3×3 matrix and $\mathbf{v}_1$ is a 2D image point, let $M * \mathbf{v}_1$ denote the image point obtained from $\mathbf{v}_1$ after multiplying by M: $M * \mathbf{v}_1 \equiv \underline{(M\overline{\mathbf{v}_1})}$. For 2D vectors $\mathbf{v}_1$, $\mathbf{v}_2$, we use the notation $\mathbf{v}_1 \times \mathbf{v}_2$ to mean $v_{1x}v_{2y} - v_{1y}v_{2x}$.

If $\mathbf{V}$ is a 3D vector, we have the identity $M * \underline{\mathbf{V}} = \underline{(M\mathbf{V})}$, since

$$\begin{aligned} M * \underline{\mathbf{V}} &= \frac{\left[M \left(\begin{array}{cc} \underline{\mathbf{V}}; & 1 \end{array} \right)\right]_2}{\left[M \left(\begin{array}{cc} \underline{\mathbf{V}}; & 1 \end{array} \right)\right]_z} = \left(\frac{V_z}{V_z}\right) \frac{\left[M \left(\begin{array}{cc} \underline{\mathbf{V}}; & 1 \end{array} \right)\right]_2}{\left[M \left(\begin{array}{cc} \underline{\mathbf{V}}; & 1 \end{array} \right)\right]_z} \\ &= \frac{[M\mathbf{V}]_2}{[M\mathbf{V}]_z} = \underline{M\mathbf{V}}. \end{aligned} \tag{1}$$

Note that in general $\underline{\mathbf{V}_1 - \mathbf{V}_2} \neq \underline{\mathbf{V}_1} - \underline{\mathbf{V}_2}$.

Assume we have N quantities ζ_a indexed by a. We define $\{\zeta\}$ to denote the length–N column vector whose a–th element is ζ_a. We also use the notation $\ddot{\Upsilon}$ to indicate a column vector, typically a "long" vector with length greater than 3. Similarly, if we have quantities ξ_{ab} indexed by a and b, we let $[[\xi]]$ denote the matrix whose (a,b)–th entry is ξ_{ab}.

Let there be N_p points tracked over N_I images, where we label the images by $i = \{0, 1, ...N_I - 1\}$. We take the zeroth image as the *reference image.* Define $\mathbf{P}_m \equiv \left(\begin{array}{ccc} X_m; & Y_m; & Z_m \end{array} \right)$ to be the m–th 3D point

in the coordinate system of the zeroth image. Let $\mathbf{p}_m^i \equiv (x_m^i; y_m^i)$ denote the m–th image point in the i–th image and let $(x_m; y_m) \equiv \mathbf{p}_m \equiv \mathbf{p}_m^0$.

Let K^i denote the calibration matrix for the i–th image, where we define these with respect to the ideal image, e.g.,

$$\mathbf{p}_m = K * \underline{\mathbf{P}_m} = \underline{K\mathbf{P}_m} = [K\mathbf{P}_m]_2 / Z_m$$

(neglecting noise). We take the calibration matrices to have the standard upper–diagonal form

$$K^i = \begin{bmatrix} K_{11}^i & K_{12}^i & K_{13}^i \\ 0 & K_{22}^i & K_{23}^i \\ 0 & 0 & 1 \end{bmatrix}.$$

Define $K \equiv K^0$.

Let $\mathbf{T}^i$ and R^i represent the translation and rotation from the reference image 0 to the i–th image, and let $R \equiv R^0$ and $\mathbf{T} \equiv \mathbf{T}^0$. We define the motion of the 3D point $\mathbf{P}$ under R^i and $\mathbf{T}^i$ by $\mathbf{P}' = R^i(\mathbf{P} - \mathbf{T}^i)$. Define $\mathbf{T}'^i = K\mathbf{T}^i$ (note that $T_z'^i = T_z^i$). We refer to the $\mathbf{T}'^i$ as the *3D epipoles*, since the epipoles in the reference image are given by $\mathbf{e}^i = \underline{\mathbf{T}'^i}$.

2.2 Projective Inverse Depths

In our experiments, we report results for the (projected) inverse depths. We show that one can recover these accurately in projective SFM.

In the projective context, one can recover the structure only up to a projective transform Π, where Π is a 4×4 matrix. The structure changes under Π by:

$$\Delta_m \begin{pmatrix} \mathbf{P}'_m; & 1 \end{pmatrix} = \Pi \begin{pmatrix} \mathbf{P}_m; & 1 \end{pmatrix},$$

where the Δ_m are scalars. Dividing by Z'_m and Z_m, we rewrite this in terms of modified constants $\tilde{\Delta}_m$ as

$$\tilde{\Delta}_m \begin{pmatrix} \underline{\mathbf{P}'_m}; & 1; & Z_m'^{-1}; \end{pmatrix} = \Pi \begin{pmatrix} \underline{\mathbf{P}_m}; & 1; & Z_m^{-1} \end{pmatrix},$$

where the $\underline{\mathbf{P}_m}$ are the ideal image coordinates.

Since we adopt the coordinate system of the zeroth image, we only need to consider Π that leave the image points fixed in this image. Then, for generic scenes, the first three rows of Π must have the form $\Pi_{1:3} \sim \begin{bmatrix} \mathbf{1}_3, & \mathbf{0}_3 \end{bmatrix}$, where $\mathbf{1}_3$ is the 3×3 identity matrix and $\mathbf{0}_3$ is a length–3 column of zeros. This implies

$$Z_m'^{-1} = \left(\Pi_{41} x_{0m} + \Pi_{42} y_{0m} + \Pi_{43} + \Pi_{44} Z_m^{-1}\right) / \Pi_{33}. \tag{2}$$

Thus, up to the small effects of noise in x_{0m}, y_{0m}, one can accurately recover the inverse depths up to an additive plane and a multiplicative scale.

Define a $N_p \times N_p$ projection matrix Q_L annihilating the three length–N_p vectors $\{1\}$, $\{x\}$, $\{y\}$ (refer to Section 2.1 for our use of the curly brackets). Then (2) implies one can accurately recover the projection $Q_L\left\{Z^{-1}\right\}$ up to scale and the small effects of noise. In our experiments, we compare the accuracy of our various algorithms in recovering this projection.

2.3 Image Displacements due to Varying Calibration

Consider two images, e.g., images 0 and i, and neglect the noise. We have

$$\mathbf{p}_m = K * \underline{\mathbf{P}_m}, \;\; \mathbf{p}_m^i = K^i * \underline{R^i\left(\mathbf{P}_m - \mathbf{T}^i\right)}.$$

Note that

$$K^i * \underline{R^i\left(\mathbf{P}_m - \mathbf{T}^i\right)} = \underline{K^i R^i\left(\mathbf{P}_m - \mathbf{T}^i\right)} = \frac{\left[K^i R^i\left(\mathbf{P}_m - \mathbf{T}^i\right)\right]_2}{\left[R^i\left(\mathbf{P}_m - \mathbf{T}^i\right)\right]_z}$$

(one can eliminate K^i from the denominator on the right since $\left[K^i\right]_{33} = 1$). Let

$$\mathbf{p}_{Tm}^i \equiv K * \underline{\left(\mathbf{P}_m - \mathbf{T}^i\right)}$$

represent the image point one would get by a pure translation without changing the *original* calibration $K = K^0$. Write the image displacements as the sum of a pure translational piece and a remaining piece that contains the rotational effects:

$$\begin{aligned} \mathbf{d}_m^i &= K^i * \underline{R^i\left(\mathbf{P}_m - \mathbf{T}^i\right)} - \mathbf{p}_{Tm}^i + \mathbf{p}_{Tm}^i - \mathbf{p}_m \\ &\equiv \mathbf{d}_{Tm}^i + \mathbf{d}_{Rm}^i, \end{aligned}$$

where

$$\begin{aligned} \mathbf{d}_{Tm}^i &\equiv \mathbf{p}_{Tm}^i - \mathbf{p}_m, \\ \mathbf{d}_{Rm}^i &\equiv K^i * \underline{R\left(\mathbf{P}_m - \mathbf{T}^i\right)} - \mathbf{p}_{Tm}^i. \end{aligned}$$

Recall that $\mathbf{T}'^i = K\mathbf{T}^i$, and $T_z'^i = T_z$, $\left[K\mathbf{P}_m\right]_z = Z_m$. Then

$$\begin{aligned} \mathbf{d}_{Tm}^i &= K * \underline{\left(\mathbf{P}_m - \mathbf{T}^i\right)} - \mathbf{p}_m \\ &= \frac{\left[K\mathbf{P}_m - \mathbf{T}'^i\right]_2}{\left[K\mathbf{P}_m - \mathbf{T}'^i\right]_z} - \mathbf{p}_m = \frac{\left[K\mathbf{P}_m\right]_2 - \left[\mathbf{T}'^i\right]_2}{Z_m - T_z^i} - \mathbf{p}_m = \frac{Z_m\mathbf{p}_m - \left[\mathbf{T}'^i\right]_2}{Z_m - T_z^i} - \mathbf{p}_m \\ &= \frac{T_z^i\mathbf{p}_m - \left[\mathbf{T}'^i\right]_2}{Z_m - T_z^i} = Z_m^{-1}\frac{T_z^i\mathbf{p}_m - \left[\mathbf{T}'^i\right]_2}{1 - Z_m^{-1}T_z^i}. \end{aligned}$$

Thus,

$$\mathbf{d}_m^i = \frac{Z_m^{-1}\left(T_z^i \mathbf{p}_m - \left[\mathbf{T}'^i\right]_2\right)}{1 - Z_m^{-1} T_z^i} + \mathbf{d}_{Rm}^i, \tag{3}$$

where one can rewrite $\mathbf{d}_{Rm}^i$ as

$$\begin{aligned}
\mathbf{d}_{Rm}^i &= \underline{K^i R \left(\mathbf{P}_m - \mathbf{T}^i\right)} - \mathbf{p}_{Tm}^i = \underline{\left(K^i R (K)^{-1}\right) K \left(\mathbf{P}_m - \mathbf{T}^i\right)} - \mathbf{p}_{Tm}^i \\
&= \left(K^i R (K)^{-1}\right) * \mathbf{p}_{Tm}^i - \mathbf{p}_{Tm}^i \\
&= \mathbf{p}_m^i - \left(K^i R (K)^{-1}\right)^{-1} * \mathbf{p}_m^i.
\end{aligned}$$

The $\mathbf{d}_{Rm}^i$ represent the displacements due to a pure 2D projective transform $K^i R \left(K^0\right)^{-1}$. The form of (3) leads naturally to our algorithms described below.

2.4 Infinitesimal 2D Projective Transforms

We derive the form of an infinitesimal 2D projective transform or homography. Suppose that a set of image points $\mathbf{p}'_m$ is given by a small 2D projective transform M of image points $\mathbf{p}_m$. Then $\mathbf{p}'_m = M * \mathbf{p}_m$. Write the 3×3 matrix M as

$$M \equiv \mathbf{1}_3 + \boldsymbol{\mu}, \;\; \boldsymbol{\mu} \equiv \begin{bmatrix} F & I \\ J^T & 0 \end{bmatrix}, \tag{4}$$

where $\mathbf{1}_3$ is the 3×3 identity matrix, $\boldsymbol{\mu}$ is small, F is a 2×2 matrix, and I and J are 2×1 vectors. For the 2D image displacements,

$$\mathbf{p}'_m - \mathbf{p}_m = \frac{\mathbf{p}_m + F\mathbf{p}_m + I}{1 + J^T \mathbf{p}_m} - \mathbf{p}_m = I + F\mathbf{p}_m - \mathbf{p}_m J^T \mathbf{p}_m + O\left(\boldsymbol{\mu}^2\right). \tag{5}$$

For each of the 8 parameters in F, I, J, we define a length–2 vector $\mathbf{h}_m^{(b)}$ giving the corresponding first–order displacement due to the infinitesimal homography. Thus

$$\begin{aligned}
\mathbf{h}_m^{(I1,I2,F11,F12)} &\equiv \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} x_m \\ 0 \end{pmatrix}, \begin{pmatrix} y_m \\ 0 \end{pmatrix}\right), \qquad (6) \\
\mathbf{h}_m^{(F21,F22,J1,J2)} &\equiv \left(\begin{pmatrix} 0 \\ x_m \end{pmatrix}, \begin{pmatrix} 0 \\ y_m \end{pmatrix}, \begin{pmatrix} x_m^2 \\ x_m y_m \end{pmatrix}, \begin{pmatrix} x_m y_m \\ y_m^2 \end{pmatrix}\right).
\end{aligned}$$

Define the 2×8 matrix $\mathbf{h}_m$ such that the b–th column of $\mathbf{h}$ equals $\mathbf{h}_m^{(b)}$.

2.5 Additional Definitions for the Algorithms

The homography flow vectors $\ddot{\Psi}^{(b)}$. Assuming N_p image points, we define eight length–$2N_p$ "homography flow" vectors $\ddot{\Psi}^{(b)}$ corresponding to the $\mathbf{h}_m^{(b)}$ defined above, with $b = \{I1, I2, F11, F12, F21, F22, J1, J2\}$, by

$$\ddot{\Psi}^{(I1)} \equiv \begin{bmatrix} \{1\} \\ \{0\} \end{bmatrix}, \ddot{\Psi}^{(I2)} \equiv \begin{bmatrix} \{0\} \\ \{1\} \end{bmatrix}, \ldots \ddot{\Psi}^{(J2)} \equiv \begin{bmatrix} \{xy\} \\ \{y^2\} \end{bmatrix}, \tag{7}$$

where each quantity in curly brackets represents a length–N_p column vector (see Section 2.1). Define the $2N_p \times 8$ matrix Ψ such that its b–th column equals $\ddot{\Psi}^{(b)}$.

Translational flow vectors $\ddot{\Phi}^{(b)}$. Define the three length–$2N_p$ translational–flow vectors $\ddot{\Phi}_x$, $\ddot{\Phi}_y$, $\ddot{\Phi}_z$, corresponding to the translational displacements in (3), by

$$\ddot{\Phi}_x \equiv -\begin{bmatrix} \{Z^{-1}\} \\ \{0\} \end{bmatrix}, \quad \ddot{\Phi}_x \equiv -\begin{bmatrix} \{0\} \\ \{Z^{-1}\} \end{bmatrix}, \quad \ddot{\Phi}_z \equiv \begin{bmatrix} \{xZ^{-1}\} \\ \{yZ^{-1}\} \end{bmatrix},$$

where each quantity in curly brackets represents a length–N_p column vector (see Section 2.1). Let $\ddot{\Phi}\left(\mathbf{T}, Z^{-1}\right) \equiv \ddot{\Phi}_x T_x + \ddot{\Phi}_y T_y + \ddot{\Phi}_z T_z$. It encapsulates the translational flow for a translation $\mathbf{T}$.

Image displacement matrix. We organize the observed image displacements $\mathbf{d}_m^i$ into a $(N_I - 1) \times 2N_p$ matrix D, by putting all the x and then the y coordinates for a given image on a single row.

Bias Compensation Matrix C. Define a $(N_F - 1) \times (N_F - 1)$ matrix C by

$$C_{ii'} \equiv \delta_{ii'} + 1.$$

We use

$$[C^{-1/2}]_{ii'} = \delta_{ii'} - \frac{1 + N_F^{-1/2}}{N_F - 1} 1_{ii'}.$$

Note that one can compute products of $C^{-1/2}$ and C with $O(N)$ computation.

Modified translation vectors $\ddot{T}'_{Ca}$. For $a \in \{x, y, z\}$, let $\{T'_a\}$ be the length–$(N_I - 1)$ vector whose elements are the T'^i_a. Define

$$\ddot{T}'_{Ca} \equiv C^{-1/2} \{T'_a\}.$$

3 General–Motion Algorithm

Assumptions. As in the approach of [17], the algorithm requires that the translational motion not be too large (e.g., with $|\mathbf{T}'|/Z \leq 1/3$) and that the camera positions do not lie in a plane. One can automatically detect when the camera is moving on a plane or line and use the approach of **Algorithm II** below [26].

3.1 Algorithm (Proj)

Step P1: Homography compensation. Assuming that the translations are *zero*, we recover the homographies $M^i \sim K^i R^i (K)^{-1}$ separately between the reference image and each subsequent image i. We compensate for these homographies, defining the compensated image i by $\mathbf{p}^i_{cm} \equiv \left(M^i\right)^{-1} * \mathbf{p}^i_m$. Let the image displacements $\mathbf{d}^i_m$ and displacement matrix D now refer to the compensated image points $\mathbf{p}^i_{cm}$.

Step P2a: Homography elimination. Using Householder matrices [25][4], we compute a $(2N_p - 8) \times 2N_p$ matrix H such that: 1) H annihilates the subspace generated by the 8 vectors $\ddot{\Psi}^{(b)}$; 2) the rows of H are unit norm and orthogonal (i.e., $H^T H$ is a projection).

Step P2b: Singular value decomposition. We define the modified displacement matrix

$$D_{CH} \equiv C^{-1/2} D H^T$$

and compute its singular–value decomposition (SVD). Let the length–$(2N_p - 8)$ column vectors $\ddot{A}^{(1)}, \ddot{A}^{(2)}, \ddot{A}^{(3)}$ denote the three leading right singular vectors of D_{CH}, and let $A \equiv \left[\begin{array}{ccc} \ddot{A}^{(1)}, & \ddot{A}^{(2)}, & \ddot{A}^{(3)} \end{array} \right]$.

Step P3a: Depth recovery. We recover the depths by solving the linear system of $3\,(2N_p - 8)$ equations

$$H \left[\begin{array}{ccc} \ddot{\Phi}_x \left(\{Z^{-1}\}\right), & \ddot{\Phi}_y \left(\{Z^{-1}\}\right), & \ddot{\Phi}_z \left(\{Z^{-1}\}\right) \end{array} \right] = AU \tag{8}$$

for the Z_m^{-1} and the unknown 3×3 matrix U.

Step P3b: Translation recovery. Using the recovered values for the $\ddot{\Phi}_a$, we solve the linear system of $(N_I - 1)\,(2N_p - 8)$ equations

$$D_{CH} = \left(\begin{array}{ccc} \ddot{T}'_{Cx} & \ddot{T}'_{Cy} & \ddot{T}'_{Cz} \end{array} \right) \left(\begin{array}{ccc} \ddot{\Phi}_x & \ddot{\Phi}_y & \ddot{\Phi}_z \end{array} \right)^T H^T. \tag{9}$$

for the $\ddot{T}'^i_{Ca}$. The $\{T'_a\}$ are given by $\{T'_a\} = C^{1/2} \ddot{T}'_{Ca}$.

Step P4: Improved homography recovery. Let the $\mathbf{T}'^i_E$ represent the current estimates of the $\mathbf{T}'^i$. For each i, we solve the linear system of N_p equations

$$\frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times \mathbf{d}^i_m = \frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times I^i + \frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times F^i \mathbf{p}_m - \left(\frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times \mathbf{p}_m \right) J^{iT} \mathbf{p}_m, \tag{10}$$

for I^i, F^i, J^i. Our estimate of the residual homography from the reference image to image i is

$$\mathbf{1}_3 + \begin{bmatrix} F^i & I^i \\ J^{iT} & 0 \end{bmatrix}.$$

Step P5: Iteration (optional). We iteratively repeat until convergence the following: 1) Compensate for the residual homographies newly computed in Step P4; 2) Repeat Steps P2b–P4.

3.2 Discussion

Step P1: Homography compensation. This is a preprocessing step and sometimes unnecessary. Its purpose is to make the effective motion smaller before applying the factorization. One could also apply this step in the Sturm/Triggs approaches of [30][33][1][7], since these, like ours, work best for small motions.

The errors in recovering the M^i scale with the size of the translational image displacements,

$$\delta M^i \equiv \left(M^i\right)^{-1}\left(K^i R^i \left(K\right)^{-1}\right) \sim O\left(\left|\mathbf{p}_T^i - \mathbf{p}\right|\right) \sim O\left(Z^{-1}\left|\mathbf{T}'^i\right|, \boldsymbol{\eta}\right),$$

where $\boldsymbol{\eta}, Z^{-1}\left|\mathbf{T}'^i\right|$ in the correction term denote the average sizes of the image noise, inverse depths, and translations. Under our moderate–translation assumption, the $\left|\delta M^i\right|$ are small.

Compensating for the M^i is equivalent to compensating the dominant plane[2] [9][27][11]. Our formulation makes it clear that one can understand this compensation as a small–translation approximation rather than as an approximation that the scene is planar.

Steps P2a and b: Homography elimination and SVD. One can compute H and its products with $O\left(N_p\right)$ computation [17]. Our current implementation computes the products of H in $O(N_p^2)$, since MATLAB's overhead makes our $O\left(N_p\right)$ implementation quite slow.

After the homography compensation in Step P1, we have

$$\mathbf{d}_m^i = \frac{Z_m^{-1}\left(T_z^i \mathbf{p}_m - \left[\mathbf{T}'^i\right]_2\right)}{1 - Z_m^{-1} T_z^i} + \mathbf{p}_m^i - \left(\delta M^i\right)^{-1} * \mathbf{p}_m^i, \tag{11}$$

where

$$\mathbf{p}_m^i - \left(\delta M^i\right)^{-1} * \mathbf{p}_m^i \sim O\left(Z^{-1}\left|\mathbf{T}'\right|, \boldsymbol{\eta}\right).$$

We neglect the denominator in (11) and D, since it causes a second order correction $O\left(Z^{-2}\left|\mathbf{T}'\right|^2\right)$. Multiplying D by H eliminates the first–order corrections due to δM^i, so up to second order we get a bilinear

[2] Though the initial step of our method is the same as in [9], our subsequent algorithm differs in two important respects: it is linear, and it corrects for the first–order errors in the compensation of the dominant plane.

expression for D_{CH}:

$$D_{CH} \approx \left(\ddot{T}'_{Cx} \ddot{\mathbf{\Phi}}_x^T + \ddot{T}'_{Cy} \ddot{\mathbf{\Phi}}_y^T + \ddot{T}'_{Cz} \ddot{\mathbf{\Phi}}_z^T \right) H^T + O\left(\boldsymbol{\eta}, Z^{-2} \left| \mathbf{T}' \right|^2 \right). \tag{12a}$$

Step 2b is based on approximating D_{CH} as bilinear in the structure and motion.

Our use of H to annihilate the residual homographies derives from the optical–flow technique of Jepson and Heeger [10], as generalized by [25] to apply to arbitrary, rather than regular, image–point configurations.

For sideways translations, the denominator in (11) exactly equals 1. If we can succeed in making the residual homography δM^i small, for example by means of the iteration in Step P5, then D_{CH} becomes *exactly* bilinear for sideways translations, no matter how big they are. Thus, our algorithm is capable of giving good results for sideways translations even when they are very large.[3] Our experiments with the Euclidean version of our approach support this conclusion: we showed in [15] that the equivalent of Step P5 makes the Euclidean equivalent of the homographies (the rotations) small even when the translations are large. We directly confirmed on a few test sequences that Step P5 makes the residual homographies small even for large sideways translations, see below. However, our approach is intended mainly for small or moderate translations. One can easily handle sequences with large sideways translations by few–frame methods [14].

As discussed in [17], multiplying by $C^{-1/2}$ reduces the bias due to singling out the reference image for special treatment.

This algorithm **Proj** requires that D_{CH} has 3 singular values that are much larger than the rest. This is the precise form in which we impose our general–motion assumption. If D_{CH} does not have 3 large singular values for a given sequence, one should use **Algorithm II** below rather than **Proj**.

Step P3a: Depth recovery. (8) reflects the fact that the recovered right singular vectors $\ddot{A}^{(b)}$ must generate approximately the same subspace as the three translational flow vectors $\ddot{\mathbf{\Phi}}_{x,y,z}$.

As discussed in Section 2.2, one can recover the Z_m^{-1} only up to an additive plane and multiplicative scale. One can see this explicitly from the fact that H annihilates all contributions to the $\ddot{\mathbf{\Phi}}_{x,y,z}$ from the $\{1\}$, $\{x\}$, $\{y\}$ components of $\left\{Z^{-1}\right\}$, where $\{1\} \ldots \left\{Z^{-1}\right\}$ are length–N_p vectors.

One can solve (8) with $O\left(N_p\right)$ computation, as discussed in [17]. In our current implementation, we use a simple $O(N_p^3)$ method.

Step P3b: Translation Recovery Because (9) is linear in the translations, its solution only requires inverting a 3×3 matrix.

[3] We presented an algorithm specialized for sideways motions in [26].

Step P4. Improved homography recovery. From (11), we get

$$\begin{aligned}\frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times \mathbf{d}^i_m &\approx \frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times \left(\mathbf{p}^i_m - \left(\delta M^i\right)^{-1} * \mathbf{p}^i_m\right) + O\left(\left|\delta\mathbf{T}'^i\right| Z^{-1} \left|\mathbf{T}'^i\right|, \boldsymbol{\eta}\right), \qquad (13)\\ &\approx \frac{\mathbf{p}_m - \underline{\mathbf{T}'^i_E}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^i_E}\right|} \times \left(\delta M^i * \mathbf{p}^i_m - \mathbf{p}^i_m\right) + O\left(\left|\delta\mathbf{T}'^i\right| Z^{-1} \left|\mathbf{T}'^i\right|, \boldsymbol{\eta}\right),\end{aligned}$$

where $\left|\delta\mathbf{T}'^i\right|$ denotes the size of the error in estimating the $\mathbf{T}'^i$ and $\left|\mathbf{T}'^i\right|$ denotes the average size of $\mathbf{T}'^i$. As before, $\left|Z^{-1}\right|$ and $\boldsymbol{\eta}$ denote the average sizes of the inverse depths and the noise. Define I^i, F^i, J^i as in (4) by

$$\delta M^i \approx \mathbf{1}_3 + \begin{bmatrix} F^i & I^i \\ J^{iT} & 0 \end{bmatrix}.$$

Then (13) and (5) give the linear system (10).

The linear system (10) does not determine the residual homographies δM^i unambiguously. This just reflects the projective covariance of the equations: one is free to change the projective basis, which changes the values of the δM^i, Z_m^{-1}, and the $\mathbf{T}'^i$. As discussed in Section 2.2, one can characterize the projective transforms that leave the reference image fixed by their effects on the Z_m^{-1}. We remove the ambiguity in recovering the δM^i by setting to zero the part of δM^i that would add a plane to the inverse depths.

Remark. One could also estimate improved values of the image points as in Step 4 of the algorithm of [17], but we have not implemented this.

3.3 Variations

The algorithm we have presented is fully projective: it can handle arbitrary changes in the linear calibrations K^i between images. But, most sequences are taken with a single camera, so that the calibration remains essentially constant over the sequence (with the possible exception of the focal length). We have considered a few simple modifications of Step P1 that exploit this. One variation, the **fixed** algorithm, recovers the homographies between the reference and subsequent images by a least–squares optimization under the assumption that the calibration K is fixed. (Like the original Step P1, it assumes that the translations are zero.) The other variations exploit the fact that the calibration error is typically small, so that one can neglect the homographies $K^i R(K)^{-1}$ or approximate them by pure rotations R^i. The **Proj-nocomp** variation of our algorithm does not compensate for the homographies at all, simply eliminating Step P1.

The version **Proj–Unrot** approximates $K^i R^i (K)^{-1}$ by a pure rotation. Instead of Step P1, it computes and compensates for the best rotations transforming the reference image to the subsequent images [17].

Finally, we have compared our projective algorithms to a Euclidean version of the same algorithm, which we described in [17].

3.4 General–Motion Experiments

In the following synthetic experiments, the motion, structure, and image noise varied randomly for each trial. The sequences consisted of 15 images of 30 points with a 60° field of view (FOV). The 3D depths varied from $20 \leq Z \leq 100$. In Experiments 1–4 we randomly chose each translation component (with respect to the zeroth camera position) such that $-T_{\max} \leq T_{x,y,z} \leq T_{\max}$ and added random rotations up to a maximum of about 20°. In Experiment 5, the motion consists of a *scene* rotation by up to about 30°. (This corresponds to very large camera translations in camera–centered coordinates and poses a serious challenge for our approach, since it is targeted for moderate translations. Other algorithms, e.g., the two–image or Tomasi/Kanade approach, are more appropriate for this situation [14][31].) The noise was one–pixel Gaussian, assuming a 512×512 image and the specified FOV.

We simulated calibration error by multiplying the images derived as above by a matrix K, with

$$K = \begin{bmatrix} 1.1 & 0 & .1 \\ 0 & 1.05 & .09 \\ 0 & 0 & 1 \end{bmatrix}.$$

This corresponds to a calibration error in the focal length of 5–10%.

Table 1 shows the results for several versions of our algorithm in comparison to the maximum–likelihood estimates. The entries give the mean error in degrees over all trials between the recovered and ground–truth values of $Q_L\left\{Z^{-1}\right\}$ (Section 2.2). **Proj** refers to our algorithm of Section 3.1, and **P–U** refers to **Proj–Unrot** described above. **MLE** gives the results for the maximum–likelihood least–squares estimate, computed by a standard Levenberg–Marquardt steepest–descent approach starting from the ground truth. **Euc** is our Euclidean algorithm [17].

Proj–Unrot gives the best of the tabulated results. The **fixed** results (not shown) are comparable, but not good enough to justify their extra computational cost. For small translations, with $T_{\max} = 2$, where our algorithm should be most accurate, **Proj–Unrot** does only 11% worse than the MLE. **Euc** also does well. For $T_{\max} = 8$, its results are only 12% worse than those of **Proj–Unrot**. One of the reasons for the good performance of **Proj–Unrot** and **Euc** is that the calibration "error," i.e., the difference between K and the identity matrix, is moderate.

Expt	#Seqs	**Proj**	**P–U**	**Euc**	**MLE**	$T_{\max}$
1	100	2.64	2.58	3.17	2.32	2
2	400	1.51	1.29	1.46		4
3	90	1.71	1.17	1.30	0.79	6
4	410	1.51	1.29	1.45		8
5*	100	6.09	6.60	17.11	0.26	*

Table 1: Experiments 1–5: general motion. Average errors in degrees in the projected inverse depths $Q_L\{Z^{-1}\}$.

As expected, our algorithm does relatively less well than the MLE in Experiment 5. However, because of the large baselines in this experiment, the resulting translational image displacements constrain the reconstruction so strongly that our algorithms still give good results, i.e., **Proj–Unrot** averages better than 7° error.

We also computed the results (not shown) for another variation of our algorithm, where instead of Step P1 we computed all the homographies between the reference and subsequent images in a single optimization. (One way of doing this is to use the 2D version of the Sturm/Triggs iterative factorization approach [30][16].) Since Step P1 computes the transform between image 0 and image i separately for each i, it overweights the reference image noise, and by computing all the transforms simultaneously we can avoid this. However, we found that the single–optimization approach did not improve our results.

Real–Image Sequence. We tested our algorithm on the Castle real image sequence available from CMU. Figure 1 shows the first image of this sequence. We generated the images for this experiment by multiplying the correctly calibrated images (for unit focal length) by 12.2 and then shifting by $(0.403; 0.093)$ (to center the images and scale them to unity). Thus the assumed focal length was incorrect by a factor of 12.2. Our pure projective approach **Proj** gave an error for the projected inverse depths of .40°, compared to .04° for the MLE. **Euc** gave an error of 28.2°. The largeness of the Euclidean error is due to the very large error in the assumed focal length.

Since the scene in this sequence has a very shallow depth, an affine approach such as the Tomasi/Kanade algorithm [31] is more appropriate than ours [17].

Comparison to Sturm/Triggs Iterative Factorization (STIF). The Sturm/Triggs iterative factorization method (**STIF**) [30][33][1][7] is an approach that, like ours, deals well with small motions and large perspective effects. We compared our approach to an implementation of **STIF** that we created previously for other purposes. We have optimized the code for **STIF** to about the same extent as for ours. In implementing **STIF**, we followed the advice of [33]: before applying the algorithm, we first centered and scaled

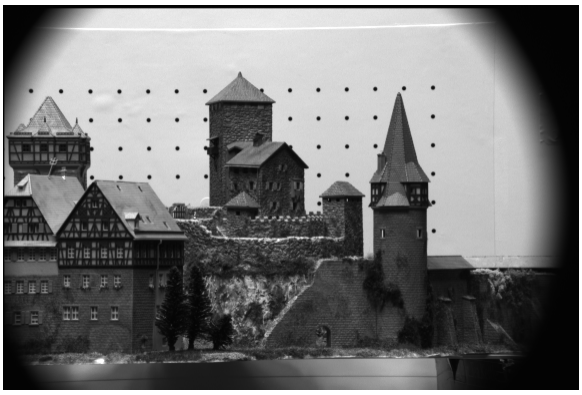

Figure 1: Castle image

each image to a unit box, and then normalized each homogenous image point to unit norm. We initialized the algorithm by setting all the unknown projective depths equal to one [33][1][7], as is appropriate for small motions. (Since we aim for a true multi–image technique that works for low signal–to–noise, we do not compute the projective depths from two or three images as in the original Sturm/Triggs algorithm [30].) The implementation of Steps P2 and P3 of our algorithm requires about 80 lines of MATLAB code, and Steps P4 and P5 require an additional 30. **STIF** requires about 80 lines.

We created synthetic sequences using the ground–truth structure from two real–image sequences: the UMASS/Martin–Marietta rocket–field sequence [3] and the UMASS PUMA sequence [12]. The points in the rocket–field sequence range from 17 to 67 in depth and cover an effective FOV of 37°, while the PUMA points range from 13 to 32 in depth and cover an effective FOV of 33°. Table 2 gives the parameters of the experiments, and Table 3 shows the errors in recovering the projected inverse depths, the epipoles in the zeroth image, and the homographies. As Table 2 indicates, in most experiments each image had a different, randomly chosen calibration. Define $r(a)$ to be a random number chosen uniformly in the interval $[a, -a]$. For each of the N_I images, we selected the calibration matrix via

$$\begin{aligned}
\left[K^i\right]_{11} &= 3.5 + r(2.5), \\
\left[K^i\right]_{12} &= 0, \\
\left[K^i\right]_{22} &= \left[K^i\right]_{11}(1 + r(0.1)), \\
\left[K^i\right]_{13} &= r(0.5), \\
\left[K^i\right]_{23} &= r(0.5).
\end{aligned}$$

(We chose r separately for each entry of the matrix.)

Table 2 gives the size of the added noise in pixels, where we define the size of a pixel by taking the

maximum magnitude of the image point coordinates to correspond to 256 pixels. Note that this is after applying the calibration matrix K. Since the shift in the camera center due to $\left[K^i\right]_{1:2,3}$ displaces the image region from the origin, this noise is usually larger than it would be for an image region centered on the origin.

We define the epipole error for a sequence as the average over images $i \in [1, 2, \ldots N_I - 1]$ of

$$\frac{\left|\mathbf{e}^i_{\text{calc}} - \mathbf{e}^i_{\text{true}}\right|}{\left|\mathbf{e}^i_{\text{true}}\right|},$$

where $\mathbf{e}^i_{\text{calc}}$ and $\mathbf{e}^i_{\text{true}}$ are the calculated and true epipoles in the reference image. One can show, as in Section 2.2, that projective transforms that leave the reference image fixed also leave these epipoles fixed.

We obtain the homography error for a sequence by averaging the homography error over all images with $i \geq 1$. By the "homography error," we mean the error in recovering the $K^i R^i \left(K\right)^{-1}$. If a projective transform leaves the reference image fixed, one can show that it changes this matrix by

$$K^i R^i \left(K\right)^{-1} \longrightarrow K^i R^i \left(K\right)^{-1} + \overline{\mathbf{e}^i_{\text{true}}}\, \mathbf{V}^{iT}$$

(up to noise), where the $\mathbf{V}^i$ are length–3 vectors. If G^i is a 3×3 homography matrix for the i–th image, define the corresponding *invariant homography matrix* by

$$\text{invar}\left(G^i\right) = G^i - \overline{\mathbf{e}^i_{\text{true}}}\, \left(\overline{\mathbf{e}^i_{\text{true}}} \setminus G^i\right),$$

where we use the backslash notation of MATLAB, i.e., $\overline{\mathbf{e}^i_{\text{true}}} \setminus G^i$ is the matrix division of $\overline{\mathbf{e}^i_{\text{true}}}$ into G^i. With this definition, $\text{invar}\left(G^i\right)$ is invariant (up to noise) to projective transforms that leave the reference image fixed.

As before, we denote the recovered homographies by M^i. We define the homography error for image i by the angle in degrees between the length–9 vectors $\mathbf{V}_{\text{true}}$ and $\mathbf{V}_{\text{calc}}$, where these contain the entries, respectively, of $\text{invar}\left(K^i R^i \left(K\right)^{-1}\right)$ and $\text{invar}\left(M^i\right)$.

The results show that our algorithm **Proj** usually gives better results for the structure and motion than **STIF**. One iteration of **Proj** takes slightly longer than one iteration of Sturm/Triggs factorization (recall that our current implementation of **Proj** is slower than necessary). However, the preprocessing Step P1, that is, the initial homography recovery, accounts for most of the computation time. Table 3 shows that the factorization part of our algorithm, Steps P2–P3, is about four times *faster* than the Sturm/Triggs factorization. Also, the computation times of Experiment 7 suggest that our approach converges more quickly than **STIF**; see also Section 4.4.

We also checked the performance of our algorithm on a sequence with large sideways translations. With the PUMA structure, $\sigma_{\text{ROT}} = 5°$, noise= 0.1 pixels, $T_z = r\left(0.5\right)$ and $T_{x,y} = r\left(5\right)$ (refer to Table 2 and the

Expt.	Struct	Uncalib	σ_T	σ_{ROT} deg.	Noise pix.	**STIF** $\mathrm{iter}_{\max}$	**Proj** $\mathrm{iter}_{\max}$
6	Puma	1	.1	0	0	1	1
7	Puma	1	.1	0	0	100	100
8	Puma	1	.1	15	.1	1	1
9	Puma	1	2	15	.5	1	1
10	Puma	0	.5	15	1	1	1
11	Puma	1	2	15	1	1	1
12	Puma	1	.1	5	1	1	1
13	Puma	1	3	15	.5	1	1
14	Rand	0	1	5	.5	1	1
15	Rand	1	.5	15	.5	1	1
16	Rock	1	1	5	.3	1	1
17	Rock	1	.5	5	.3	1	1
18	Rock	0	1	5	.5	1	1
19	Rock	1	2	5	.3	1	1

Table 2: Parameters for the experiments comparing **Proj** and the Sturm/Triggs iterative factorization approach (**STIF**). The 'Struct' column indicates the structure used to generate the sequences. A 1 in the 'Uncalib' column indicates that we introduced different calibrations for each image (see text). We choose T_x, T_y, T_z independently as Gaussian variables. The columns labelled σ_T tabulate the standard deviations of the $T_{x,y,z}$. σ_{ROT} characterizes the typical size of the rotations. The 'noise' column indicates the standard deviation of the Gaussian noise added. The remaining two columns indicate the maximum number of iterations allowed, respectively, for the iterative factorization method and for Step P5 of the **Proj** algorithm.

Expt.	Time (sec)			Z^{-1}			Epis	Epis	Homog	Homog	Cycles
	NoComp	**Proj**	**STIF**	NC	**Proj**	**STIF**	**Proj**	**STIF**	**Proj**	**STIF**	**Proj**
6	0.043	0.19	0.15	0.31	0.063	3.2	0.11	2.4	0.0006	2.8	1
7	0.044	0.35	5.1	0.27	0.03	0.14	0.0076	0.12	0.0002	0.052	4.2
8	0.043	0.19	0.15	38	17	45	2.5	4.2	0.25	6.7	1
9	0.042	0.19	0.15	8.6	8.4	8.1	2.7	6.9	1.8	5.4	1
9(med)	0.04	0.19	0.15	6.8	6.7	7.1	1.1	2.8	1.1	4.9	1
10	0.043	0.19	0.15	22	17	31	3.5	4.3	3.2	14	1
11	0.043	0.19	0.15	10	12	13	4.9	4	3.5	5.8	1
11(med)	0.04	0.19	0.15	9.1	11	11	1.4	2.8	2.3	5.2	1
12	0.046	0.19	0.15	56	72	65	2.9	6.9	1.0	11	1
13	0.043	0.19	0.15	7.3	8	7.5	2	7.2	2.6	5.7	1
14	0.04	0.18	0.15	1.4	2	3.5	3.3	10	4.3	7.3	1
15	0.043	0.19	0.15	11	12	17	2.3	6.1	3.9	7.2	1
16	0.023	0.14	0.1	5.9	7.7	5.4	2.5	2.8	1.8	3.3	1
17	.025	0.14	0.1	11	12	14	2.4	3.8	1.5	5.5	1
18	0.025	0.14	0.1	3.3	3.8	5.1	1.7	4.6	2.6	5.7	1
19	0.025	0.14	0.1	4.2	13	4	2.4	5.3	6.3	4.2	1
19(med)	0.02	0.14	0.1	3.9	6.4	3.6	1.1	2.3	4.1	4.1	1

Table 3: Results for the experiments with parameters given in the previous table. 'NoComp' and 'NC' both refer to Steps P2 and P3 of the **Proj** algorithm, without any computation of or initial compensation for the homographies. The columns labelled '**STIF**' give results for the Sturm/Triggs iterative factorization approach. We present errors for the projected inverse depths ('Z^{-1}'), the epipoles ('Epis'), and the homographies ('Homog'). See text for an explanation of the error measures used. The 'Cycles' column indicates the number of cycles used by **Proj** in Step P5. All quantities shown represent the mean of the results over 100 random trials, except for those rows labelled by '(med)', where the quantities represent the median over 100 random trials.

definition of $r(a)$ above), our algorithm gave errors for the projected depths of 1.6°. For this sequence, the largest translation had size 13.4, which is comparable to 13.9, the distance of the closest 3D point to the camera's reference position.

3.4.1 Speed Comparison to Bundle Adjustment.

In [32], the authors find that bundle adjustment on 32 images and 32 points requires roughly 10^9 floating point operations. (This number depends on the difficulty of the optimization problem, on the starting point for the minimization, and on the convergence tolerance, but the authors of [32] do not give this information.) We created 32 images of the PUMA structure. As measured by the MATLAB flops functions, one iteration of **Proj** (which is usually enough for good results, as Table 3 indicates) took less than 10^7 floating point operations. Recall that our current implementation of **Proj** is unnecessarily slow, $O\left(N_p^3\right)$ rather than than $O\left(N_p\right)$. This is because Step P3a uses a straightforward $O\left(N_p^3\right)$ method to solve for the depths instead of a faster $O\left(N_p\right)$ method. Also, we currently compute products of H in $O\left(N_p^2\right)$, without exploiting the special structure of H which would reduce the computation to $O\left(N_p\right)$.

4 Algorithm II

The algorithm presented above assumes that the translational motion is sufficiently general, i.e., that the camera locations do not all lie close to a single line or plane. In this section, we present a version of our algorithm that deals with the common case of a camera moving along a line. One can extend it to deal with more general motions such as a camera moving on a plane or in 3D, as we describe below.

Definitions. Let the unit vector $\mathbf{T}'$ denote the direction of the $\mathbf{T}'^i$, and let $\tau^i \equiv \left|\mathbf{T}'^i\right|$.

Let

$$\left[\left[\frac{\mathbf{p}-\underline{\mathbf{T}'}}{|\mathbf{p}-\underline{\mathbf{T}'}|} \times \mathbf{d}\right]\right]$$

denote the $(N_I - 1) \times N_p$ matrix whose (i,m)–th element equals $(\mathbf{p}_m - \underline{\mathbf{T}'}) \times \mathbf{d}_m^i / |\mathbf{p}_m - \underline{\mathbf{T}'}|$.

Let $\mathbf{Q}_5(\mathbf{T}')$ be the $N_p \times N_p$ projection matrix that annihilates the subspace generated by the *eight* length–N_p vectors

$$\left\{\frac{\mathbf{p}-\underline{\mathbf{T}'}}{|\mathbf{p}-\underline{\mathbf{T}'}|} \times \mathbf{h}^{(b)}\right\}.$$

One can show that this subspace is *five*–dimensional.

4.1 Algorithm Description

Step L0: Rescaling. We transform all images so that they center on the origin and have the same scale. We choose the scale so that $\left\langle x^2 + y^2 \right\rangle^{1/2} = 1$ for the reference image. (At the end of the algorithm, we transform the recovered unknowns back to the coordinate system of the original reference image.)

Step L1: Homography compensation. Assuming that the translations are *zero*, we recover the homographies $M^i \sim K^i R^i (K)^{-1}$ separately between the reference image and each subsequent image i. Define the compensated image i by $\mathbf{p}^i_{cm} \equiv \left(M^i\right)^{-1} * \mathbf{p}^i_m$. Let the image displacements $\mathbf{d}^i_m$ and displacement matrix D now refer to the compensated image points $\mathbf{p}^i_{cm}$.

Step L2a: Using Householder matrices [25][4], we compute a $(N_p - 6) \times N_p$ matrix H_L such that: 1) H_L annihilates the subspace generated by the *six* length–N_p vectors $\{1\}, \{x\}, \{y\}, \{x^2\}, \{xy\}, \{y^2\}$; 2) the rows of H_L are unit norm and orthogonal.

Step L2b: Linear translation recovery. We solve the linear system of $(N_I - 1)(N_p - 6)$ equations

$$0 \approx C^{-1/2} \left[\left[(T'_z \mathbf{p} - [\mathbf{T}']_2) \times \mathbf{d} \right] \right] H_L^T \tag{14}$$

for $\mathbf{T}'$.

Step L3: Iterative improvement of recovered translation. Starting from the previous estimate $\mathbf{T}'_E$ for $\mathbf{T}'$, we minimize the error

$$E_L(\mathbf{T}') \equiv \text{trace}\left(C^{-1} \left[\left[\frac{\mathbf{p} - \underline{\mathbf{T}'}}{|\mathbf{p} - \underline{\mathbf{T}'}|} \times \mathbf{d} \right] \right] \mathbf{Q}_5(\mathbf{T}') \left[\left[\frac{\mathbf{p} - \underline{\mathbf{T}'}}{|\mathbf{p} - \underline{\mathbf{T}'}|} \times \mathbf{d} \right] \right]^T \right). \tag{15}$$

with respect to $\mathbf{T}'$. Take $\mathbf{T}' = \arg\min \mathbf{E}_L$.

Step L4: Improved homography recovery. The same as in Step P4 of our general–motion approach.

Step L5: Iteration (optional). The same as in Step P5 of our general–motion approach.

4.2 Discussion

Step L0. As in [6] and [19][13], this step reduces the bias of our linear algorithm in Step L2.

Step L2. This linear step of our projective algorithm is *exactly* the same as in our Euclidean algorithm of [19]; the only difference is that the projective algorithm computes an estimate of $K\mathbf{T}$ rather than $\mathbf{T}$. However, the projective computation exploits a bigger fraction of the available information and gives a better approximation to the result of minimizing the full error. Since H_L annihilates only one more degree of freedom than $\mathbf{Q}_5$, the projective computation uses all but one of the available constraints, while the Euclidean computation forgoes *three* of the available constraints. One can show that the additional length–N_p vector

annihilated by H_L is

$$\left\{x^2T_x'^2 + xyT_y'T_x' + y^2T_y'^2 + xT_z'T_x' + yT_y'T_z' + T_z'^2\right\}.$$

For linear motion, (3) becomes

$$\mathbf{d}_m^i = \mathbf{d}_{Tm}^i + \mathbf{d}_{Rm} = \frac{\tau^i Z_m^{-1}(T_z'\mathbf{p}_m - [\mathbf{T}']_2)}{1 - Z_m^{-1}\tau^i T_z'} + \mathbf{d}_{Rm}. \tag{16}$$

Assume we have compensated for the homographies in Step L1. At the ground–truth value for $\mathbf{T}'$, $(T_z'\mathbf{p}_m - [\mathbf{T}']_2)\times$ $\mathbf{d}_{Tm}^i = 0$ up to noise, and

$$\left[\left[(T_z'\mathbf{p} - [\mathbf{T}']_2) \times \mathbf{d}\right]\right] H_L^T \approx O\left(\eta, |\delta M|^2\right)$$

independent of the denominator in (16). Thus, if the residual homographies δM^i are small, our algorithm works for translations of *any* size [19]. If Step L1 does not suffice to make the δM^i small, the iteration in Step L5 may. Thus, we expect the algorithm to work even for very large translations, as the Euclidean version of our approach has been shown to do [15]. These comments also apply to Step L3.

Step L3. For $\mathbf{e} = \underline{\mathbf{T}'}$ in or near the FOV, instead of Steps L2 and L3, one could use the recent method of Srinivasan [29][28] to find the *true* global minimum of (15) directly. This is important, since [15][2] showed that the least–squares error typically has several local minima for $\mathbf{e}$ near the image region. We have not yet implemented Srinivasan's approach.

Remark. One could also recover the Z_m^{-1} as in [19][23] and improve the homography computation as in Step P4 of the general–motion algorithm. Since these are straightforward transcriptions of previous methods, we do not discuss them here.

4.3 Linear–Translation Experiments

We generated sequences as in Experiments 1–4 except that the camera translated in constant steps of 0.2 along a line (with random rotations as before). We created 400 sequences, choosing random directions for the image–plane projections of $\hat{\mathbf{T}}$ and systematically varying $T_z/\sqrt{T_x^2 + T_y^2}$ from .01 to 4 in steps of .01.

Figure 2 shows the mean angular errors in recovering $\mathbf{T}'$ for three versions of our algorithms: a pure Euclidean approach [19][23], the projective approach **Algorithm II**, and **Algorithm II** with Step L1 replaced by initial compensation of the rotations. We derived the curve labeled **MLE** by first using Levenberg–Marquardt to compute the maximum–likelihood least–squares projective reconstruction and then computing the calibration and motions from this by least–squares minimization.

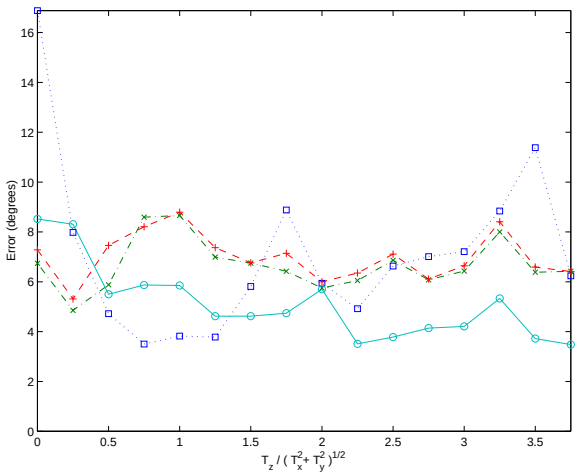

Figure 2: Angular errors in recovering $\mathbf{T}'$. x axis shows the lower limit of a bin of size 0.25 in $T_z/\sqrt{T_x^2+T_y^2}$; each data point is an average over 25 trials from the indicated bin. Circles: MLE; squares: Euclidean. '$*$' and '$+$' are for our linear projective algorithm, with initial compensation for the rotations and homographies, respectively.

The Euclidean estimate is worse than the projective ones when $\mathbf{T}$ is near–parallel to the image plane or to $\hat{z}$, but it is comparable[4] for intermediate $\mathbf{T}$. The overall median of these results is slightly better than those of the projective algorithms.

We also ran **Algorithm II** (with initial projective compensation) on all 55 choices of image pairs from the Castle sequence (Figure 1). Our algorithm recovered $\mathbf{T}'$ with an average error of 1.1°, compared to 0.71° for the MLE. The median errors for the two approaches were respectively .80° and .40°, and the maximum errors were 6.6° and 4.4°. Though the assumed focal length was incorrect by a factor of 12.2 (Section 3.4), our approach did nearly as well as the MLE.

Because of the large error in the focal length, the variation of initially compensating for a rotation instead of a homography yielded relatively large errors of about 10° in the rotation. Our Euclidean algorithm usually recovered $\mathbf{T}'$ accurately. Apart from 13 outliers (possibly due to local minima), the Euclidean algorithm recovered $\mathbf{T}'$ with an average error of 5.1°.

Comparison to Sturm/Triggs Iterative Factorization. We experimentally compared **Algorithm II** to the iterative extension of the Sturm/Triggs approach (**STIF**). We created synthetic sequences of 16 images and 32 points using the structure from the PUMA sequence. Let $\hat{\mathbf{T}}_{\text{true}}$ denote the true translation direction and $\hat{\mathbf{z}}$ the viewing direction in the reference image. We systematically varied the angle $\theta_{\text{true}} \equiv \angle\left(\hat{\mathbf{T}}_{\text{true}}, \hat{\mathbf{z}}\right)$

[4]We verified that the Euclidean approach gives bad results for $\mathbf{T} \perp \hat{\mathbf{z}}$ simply because of the calibration error. With no calibration error, it does perform well for $\mathbf{T}$ parallel to the image plane.

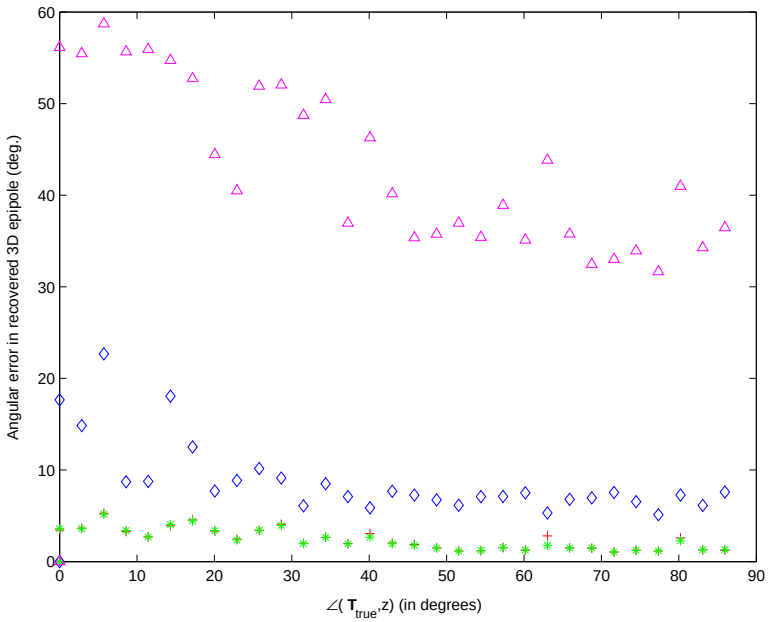

Figure 3: The angular errors in recovering the 3D epipole $\mathbf{T}'$ for **STIF** and Algorithm II, plotted versus $\theta_{\text{true}} = \angle(\mathbf{T}_{\text{true}}, \hat{\mathbf{z}})$. All data points represent averages over 30 trials. 'Triangles' show the result of one Sturm/Triggs iteration; 'diamonds' show the **STIF** result after convergence or after a cutoff of 300 iterations if the algorithm did not converge by this number. '+' shows the linear estimate of Algorithm II, Step L2, and '*' shows the result after the nonlinear minimization in Step L3 of Algorithm II.

from 0° to 86° in increments of 2.9°. For each selected θ_{true}, we created 30 sequences, where we chose the projection of $\hat{\mathbf{T}}_{\text{true}}$ on the x–y plane randomly for each sequence. For each image, we randomly chose the rotation R^i with $\sigma_{\text{ROT}} = 5°$, i.e., with a standard deviation of 5° for the rotation around each axis, and we randomly and uniformly chose the translation magnitudes up to a maximum of 1. Recall that the PUMA depths range from 13–32. We introduced varying calibrations as in Experiments 6–19, and added Gaussian noise of 0.05 pixels, assuming that the maximum magnitude of the image point coordinates corresponded to 256 pixels.

Figure 3 compares the results of **Algorithm II** and **STIF** for the 3D epipoles. Each data point represents the mean over the 30 trials for the indicated value of $\theta_{\text{true}} = \angle\left(\hat{\mathbf{T}}_{\text{true}}, \hat{\mathbf{z}}\right)$. For our approach, we plot results for the error $\angle(\mathbf{T}'_{\text{true}}, \mathbf{T}'_{\text{calc}})$, where $\mathbf{T}'_{\text{true}}$ is the true value of the 3D epipole and $\mathbf{T}'_{\text{calc}}$ is the value recovered. Since the iterative factorization approach recovers a different 3D epipole $\mathbf{T}'^i_{\text{STIF}}$ for each image, we plot the average error $\sum_i \angle\left(\mathbf{T}'_{\text{true}}, \mathbf{T}'^i_{\text{STIF}}\right) / (N_I - 1)$. For our approach, we show the errors for the initial linear estimate Step L2 and for one iteration of the nonlinear estimate Step L3. For comparison, we show the estimates after one iteration and after the algorithm either converges or reaches 300 iterations.

One iteration of **STIF** does much worse than our linear estimate, and its converged result is also much

worse. Our linear estimate is almost as good as our nonlinear estimate. Our nonlinear algorithm averaged about 2 seconds of computation and the linear algorithm took fractions of a second, while the Sturm/Triggs iterative factorization approach averaged about 9 seconds.

We also applied the iteration of Step L5 and allowed **Algorithm II** to converge. The average over all sequences of the number of cycles till convergence was < 4, and the average time till convergence was 8 seconds, less than the 9 seconds for **STIF**. **STIF** averaged 177 iterations. However, we used a much stricter convergence criterion for this approach, to give it a chance to converge to an accurate result. As proven in [16], the Sturm/Triggs iterative factorization approach minimizes a particular error function. We defined the algorithm to have converged when that error changed by less than 10^{-8} between iterations. We defined **Algorithm II** to have converged when the residual homography recovered in Step L4 satisfied $\left|\left(F^i\left(:\right);I^i;J^i\right)\right| < 6 \times 10^{-2}$.

For an additional comparison, we ran the Step L5 iteration of **Algorithm II** on 100 sequences similar to those above, defining convergence by $\left|\left(F^i\left(:\right);I^i;J^i\right)\right| < 6 \times 10^{-9}$. On average, Step L5 converged in 7.5 cycles and always in less than 15.

Figure 5 shows similar results for the homography recovery, for the homography error defined in Section 3.4.

We also tested **Algorithm II** and **STIF** on the rocket–field real–image sequence [3], see Figure 4. This sequence has large translations ranging up to 7.5 in magnitude (recall that the depths vary from 17 to 67). The camera moves approximately along a line, with $\mathbf{T}^i$ deviating from its average direction by up to 1.5°. Steps L2 and L3 of our approach recovered $\mathbf{T}'$ with errors of 5.8° and 6.4°, respectively, and Steps L1 and L4 recovered the homographies with average errors of 2.8° and 2.0°. The first iteration of **STIF** had average errors of 14.7° and 15.5° for $\mathbf{T}'$ and the homographies, respectively. After 100 iterations, **STIF** gave errors of 3.3° and 7.1°.

We have found that the Sturm/Triggs iterative factorization approach gives poor results for linear motion when the motion is forward (or backward). We created 30 sequences with 16 images of 32 points for randomly chosen structures, where the depths varied between 40 and 60, the translation direction was $(0.1; 0.1; 1)$, the $\left|\mathbf{T}^i\right|$ varied randomly up to 1, the rotations had $\sigma_{\mathrm{ROT}} = 5°$, and there was *zero noise.* We allowed **STIF** up to 3000 iterations to converge, and defined it to have converged when its error changed by less than 10^{-10} between iterations. We proved in [16] that **STIF** does eventually converge to a local minimum of the error. On 9 of the 30 trials, the algorithm had large errors in recovering the epipoles, with an average error of 44° for these trials. The *maximum* error of our linear estimate in Step L2 was 0.15°. For a similar set of 30 sequences with rotations of up to 15°, the Sturm/Triggs iterative factorization approach gave large errors

Figure 4: The first image of the rocket–field sequence.

on 25 out of 30 trials, while our approach again gave nearly perfect results. We also found that **STIF** often failed to converge correctly when we used the structure from the PUMA sequence to generate sequences.

These failures are produced by a bad initial choice of the projective depths, which causes **STIF** to converge to an incorrect local minima of its error. For small forward or backward motions and small FOV, the translational image displacements are small, and there is no way to get good initial estimates for the projective depths from any small number of images, as Sturm/Triggs originally proposed to do [30]. Also, the projective error surface is flat for epipoles away from the forward–motion direction [18], and it tends to have local minima near the image points [15][2][18]. If the initial choice of projective depths corresponds to an epipole far from the image region, these factors could cause a local minimum to intercept the **STIF** algorithm and prevent it from converging to the true global minimum.

4.4 Extension to General Motion

We describe how to extend **Algorithm II** to any motion [26].

Step L'0. Rescaling.

Step L'1: Homography compensation.

Step L'2: Compute H, H_L, and the SVD of D_{CH}. Let $N_S \leq 3$ be the number of large singular values of D_{CH}, and let $\ddot{A}^{(s)}$, $\ddot{B}^{(s)}$ be the right and left singular vectors corresponding to these singular values. Let $\sigma^{(s)}$ denote the leading singular values of D_{CH}, and define the $(2N_p - 8) \times N_S$ matrix $B_\sigma \equiv \left[\sigma^{(1)}\ddot{B}^{(1)}, \ldots, \sigma^{(N_S)}\ddot{B}^{(N_s)}\right]$.

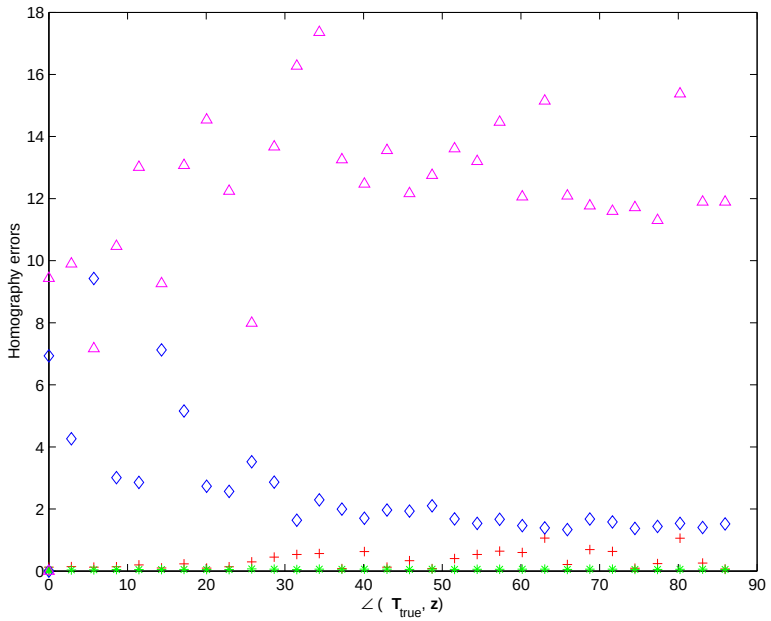

Figure 5: The errors in recovering the homographies for the Sturm/Triggs iterative factorization approach and Algorithm II, plotted versus $\theta_{\text{true}} = \angle\left(\mathbf{T}_{\text{true}}, \hat{\mathbf{z}}\right)$. All data points represent averages over 30 trials and over all images in each sequence. 'Triangles' show the result of one Sturm/Triggs iteration; 'diamonds' show the iterative factorization result after convergence or after a cut off of 300 iterations if the algorithm did not converge by this number. '+' shows the initial estimate of Algorithm II, Step L1, and '*' shows the result after Step L5 of Algorithm II has converged.

Step L'0 is the same as Step L0, and Steps L'1–L'2 are the same as Steps P1, P2 in the general–motion algorithm.

Step L'3a. For each s with $1 \le s \le N_S$, define

$$\breve{\mathbf{d}}_m^{(s)} = \left(\left[H^T \ddot{A}^{(s)} \right]_m ; \left[H^T \ddot{A}^{(s)} \right]_{N_p+m} \right)$$

for $m = 1 \ldots N_p$. That is, we define the $\breve{\mathbf{d}}_m^{(s)}$ so that $H^T \ddot{A}^{(s)} = \left[\left\{ \breve{d}_x^{(s)} \right\} ; \left\{ \breve{d}_y^{(s)} \right\} \right]$.

Step L'3b: Linear estimates of effective translation directions. Let the unit vector $\mathbf{T}'^{(s)}$ be the translation direction corresponding to $\ddot{A}^{(s)}$. For each s, we solve the system of $N_p - 6$ equations

$$H_L \left\{ \left(T_z'^{(s)} \mathbf{p} - \left[\mathbf{T}'^{(s)} \right]_2 \right) \times \breve{\mathbf{d}}^{(s)} \right\} = 0$$

in the least–squares sense for $\mathbf{T}'^{(s)}$.

Step L'3c: Refinement of effective translation directions. For each s, we refine the estimate for $\mathbf{T}'^{(s)}$ by minimizing

$$\left\{ \frac{\mathbf{p} - \mathbf{T}'^{(s)}}{\left| \mathbf{p} - \mathbf{T}'^{(s)} \right|} \times \breve{\mathbf{d}}^{(s)} \right\}^T \mathbf{Q}_5 \left\{ \frac{\mathbf{p} - \mathbf{T}'^{(s)}}{\left| \mathbf{p} - \mathbf{T}'^{(s)} \right|} \times \breve{\mathbf{d}}^{(s)} \right\}$$

with respect to $\mathbf{T}'^{(s)}$.

Step L'4a: Isolate translational flow. Recall that $\mathbf{h}_m$ is a 2×8 matrix such that the b–th column of $\mathbf{h}_m$ equals $\mathbf{h}_m^{(b)}$. For each s, we compute a length–8 vector $\mathbf{w}^{(s)}$ by solving the linear system of N_p equations

$$\frac{\mathbf{p}_m - \underline{\mathbf{T}'^{(s)}}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^{(s)}}\right|} \times \breve{\mathbf{d}}_m^{(s)} = \frac{\mathbf{p}_m - \underline{\mathbf{T}'^{(s)}}}{\left|\mathbf{p}_m - \underline{\mathbf{T}'^{(s)}}\right|} \times \left(\mathbf{h}_m \mathbf{w}^{(s)}\right). \tag{17}$$

Compute the $\ddot{\Phi}\left(\mathbf{T}'^{(s)}, Z^{-1}\right)$ from $H^T \ddot{A}^{(s)} = \ddot{\Phi}\left(\mathbf{T}'^{(s)}, Z^{-1}\right) + \Psi \mathbf{w}^{(s)}$.

Step L'4b: Depth recovery. Solve the linear system of $2N_sN_p$ equations

$$\lambda^{(s)} \breve{\mathbf{d}}_m^{(s)} = \left(Z_m^{-1} + a^{(s)} + b^{(s)} x_m + c^{(s)} y_m\right) \left(T_z^{(s)} \mathbf{p}_m - \left[\mathbf{T}'^{(s)}\right]_2\right) \sigma^{(s)} \tag{18}$$

for the Z_m^{-1} and the $4N_S$ constants $\lambda^{(s)}$, $a^{(s)}$, $b^{(s)}$, $c^{(s)}$.

Step L'4c: Full translation recovery. We recover the translations via

$$\left[\ \left\{T_x'^i\right\},\quad \left\{T_y'^i\right\},\quad \left\{T_z'^i\right\}\ \right] = C^{1/2} B_\sigma \begin{bmatrix} \left(\sigma^{(1)}/\lambda^{(1)}\right)\left[\mathbf{T}'^{(1)}\right]^T; \\ \vdots \\ \left(\sigma^{(N_S)}/\lambda^{(N_s)}\right)\left[\mathbf{T}'^{(N_s)}\right]^T \end{bmatrix}$$

Step L'5: Improved homography recovery. The same as in Steps L4 and P4 of the algorithms described above.

Step L'6: Iteration (optional). The same as in Steps P5 and L5 of the algorithms described above.

4.4.1 Discussion

This algorithm is the projective version of our Euclidean algorithm in [26].

Step L'3: Translation–direction recovery. Each leading singular vector corresponds to an effective translation direction $\mathbf{T}'^{(s)}$. We recover the $\mathbf{T}'^{(s)}$ exactly as before in **Algorithm II**.

Step L'4a,b: Isolate translational flow; depth recovery. For each leading singular vector, the $\mathbf{h}_m \mathbf{w}^{(s)}$ is the effective infinitesimal homography. As before, one cannot determine this uniquely due to projective covariance, and we specify it by setting to zero the components of $\mathbf{w}^{(s)}$ that would add a plane to the Z_m^{-1}.

The constants $a^{(s)}$, $b^{(s)}$, $c^{(s)}$ are necessary since Step L'4a recovers the homographies up to an ambiguity. If $s' \neq s$, the Z_m^{-1} corresponding to the recovered $\ddot{\Phi}\left(\mathbf{T}'^{(s)}\right)$ may differ by an additive plane from the Z_m^{-1} for $\ddot{\Phi}\left(\mathbf{T}'^{(s')}\right)$.

We need the scales $\lambda^{(s)}$ to fix the scale of the Z_m^{-1} between different singular vectors.

We introduce the singular value $\sigma^{(s)}$ in (18) to emphasize the singular vectors with larger singular values, since these have less noise sensitivity.

Step L'4b takes $O\left(N_p^3\right)$ computation. For larger problems, one can use instead an $O\left(N_p\right)$ approach which sacrifices some information in solving for the $a^{(s)}$, $b^{(s)}$, $c^{(s)}$, $\lambda^{(s)}$.

4.4.2 Experiments

We created 100 sequences with *general motion* as in Experiments 6–19, using the PUMA structure, varying calibration, $\sigma_T = 0.5$, $\sigma_{\mathrm{ROT}} = 5°$, noise= 0.1 pixels (refer to Table 2). We ran the extended version of **Algorithm II** *without* the iterative refinement of translations in Step L'3c and without the iteration of Step L'6. Without Steps L'3c and L'6, the extended algorithm is purely linear, with *no* iteration apart from the SVD and linear equation solving. We compared our results to those of the Sturm/Triggs iterative factorization algorithm (**STIF**), which we allowed up to 300 iterations to converge. On average, **STIF** required 135 iterations to converge.

This noniterative version of our algorithm averaged 0.21 seconds of computation, compared to 7.0 seconds for **STIF**. It gave an average error for the projected Z_m^{-1} of 1.39° compared to 1.63° for **STIF**. Its average error in computing the epipolar directions was 8.8°, while **STIF** gave 5.3°. Its average homography error was 0.16° and that of **STIF** was 0.52°.

A single iteration of **STIF** gave an average error of 29.7° for the epipoles and 3.2° for the homographies.

We created a second set of 100 sequences with *planar motion* with the same parameters as the first. We achieved planar motions by creating the $\mathbf{T}^i$ as before and then setting the third singular value of the matrix of translations to zero. The average errors for the projected Z_m^{-1} were 1.62° (ours) and 1.97° (**STIF**). The average errors for the epipolar directions were 4.4° (ours) and 7.2° (**STIF**). The average errors for the homographies were 0.11° (ours) and 0.66° (**STIF**).

On average, **STIF** took 140 iterations to converge. The first iteration of **STIF** gave an average error of 33° for the epipolar directions and 3.7° for the homographies. We also ran **STIF** for 11 iterations, to check its speed of convergence. This took on average 0.61 seconds and gave average errors of 5.1°, 22.9°, and 2.4° for the projected inverse depths, epipolar directions, and homographies, respectively.

4.4.3 Approximately Linear or Planar Motions

What happens when we apply **Algorithm II** or its extension with a too restrictive motion model—for example, assuming planar motion when the camera centers do not lie exactly on a plane? To the extent that we can neglect the denominator in (3)—and doing so causes small second order effects when the translations

are small—*this causes no additional error in the structure recovery.* "Applying our algorithm with a too restrictive motion model" means using N_S of the leading singular vectors when the dimensionality of the translational motion is greater than N_S, i.e., we neglect the smaller components of the translational motion. This does not prevent us from recovering a good approximation to the translations projected into an N_S–dimensional subspace—or from recovering the full translational motion subsequently. Since each singular vector gives a separate estimate of the Z_m^{-1}, the only effect on the structure recovery of restricting to a subset of the singular vectors is that we lose information that could have been used to improve the estimates.

We ran our extended **Algorithm II** with $N_S = 2$ on 10 *general–motion* sequences created as in the previous section. As before, we ran the algorithm without the iterative refinement of translations in Step L'3c and without the iteration of Step L'6. We obtained an average error of 1.37° for the projected inverse depths, which is less than our result with $N_S = 3$ and also less than the **STIF** error. We projected the true 3D epipoles into the plane of the epipolar directions recovered by our algorithm. The average error in recovering the 3D epipoles in this plane was 6.4°. The average error of Step L'5 in recovering the homographies was 0.20°.

These results suggest that it can be advantageous to ignore the smaller of the leading singular values and vectors when these have a good deal of noise contamination. In the sequences above, the x, y, and z components of the translations have similar magnitudes, but the smallness of the FOV causes the signal (i.e., the image displacements) from the z–translations to be much smaller than for the other directions. One can detect this from the smallness of the third leading singular value of D_{CH}, which for these sequences is often not much bigger than the noise effects.

Once one has recovered the structure accurately using a subset of the leading singular vectors, it is straightforward to calculate the neglected translational components from the initially neglected leading singular vectors. For each neglected singular vector, one solves for the corresponding translation direction $\mathbf{T}'^{(s)}$ from

$$\ddot{A}^{(s)} = H \left(T_z'^{(s)} \begin{pmatrix} \{Z^{-1}x\} \\ \{Z^{-1}y\} \end{pmatrix} - T_x'^{(s)} \begin{pmatrix} \{Z^{-1}\} \\ \{0\} \end{pmatrix} - T_y'^{(s)} \begin{pmatrix} \{0\} \\ \{Z^{-1}\} \end{pmatrix} \right),$$

using the previously computed Z_m^{-1}. The translational contribution from the s–th neglected singular vector is

$$\sigma^{(s)} \left(C^{1/2} \ddot{B}^{(s)} \right) \left(\mathbf{T}'^{(s)} \right)^T.$$

5 Conclusions

We presented fast projective structure–from–motion algorithms which have comparable accuracy to the MLE and appear superior to the Sturm/Triggs iterative factorization approach [30][30][33][1][7]. Our algorithms work for any motion as long as the camera displacements are not too big, with $|\mathbf{T}|/Z < 1/3$ [17]. We showed experimentally that Sturm/Triggs iterative factorization often fails when the camera moves along a line. We speculated that the recent characterizations of the projective least–squares error surface in [18][2][15] may offer part of the explanation for this. We studied the advantages of a pure projective approach, versus a mixed Euclidean/projective strategy, for the common situation when the calibration is fixed and partly known. We showed that algorithms can recover the (projected) inverse depths and homographies even in projective SFM. We clarified the nature of dominant–plane compensation, showing that it can be considered a small–translation approximation rather than a planar–scene approximation.

References

[1] R. Berthilsson, A. Heyden, G. Sparr, "Recursive Structure and Motion from Image Sequences using Shape and Depth Spaces," *CVPR* 444–449, 1997.

[2] A. Chiuso, R. Brockett, and S. Soatto, "Optimal Structure from Motion: Local Ambiguities and Global Estimates," Washington University Technical Report, 1999.

[3] R. Dutta, R. Manmatha, L.R. Williams, and E.M. Riseman, "A data set for quantitative motion analysis," *CVPR*, 159-164, 1989.

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

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

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

[7] A. Heyden, "Projective structure and motion from from image sequences using subspace methods," *SCIA* II 963–968, 1997.

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

[9] M. Irani and P. Anandan, "A Unified Approach to Moving Object Detection in 2D and 3D Scenes," **PAMI** 20, 577–589, 1998.

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

[11] R. Kumar, P. Anandan, and K. Hanna, "Direct recovery of shape from multiple views: A parallax based approach," *ICPR* A, 685–688, 1994.

[12] R. Kumar and A.R. Hanson, "Sensitivity of the Pose Refinement Problem to Accurate Estimation of Camera Parameters," *ICCV*, 365-369, 1990.

[13] W. J. MacLean, "Removal of translation bias when using subspace methods," *ICCV* 753–758, 1999. Also, PhD thesis, 1996.

[14] J. Oliensis, "A Critique of Structure from Motion Algorithms," discussion paper in **CVIU** 80, 172–214, 2000.

[15] J. Oliensis and Y. Genc, "Three Algorithms for 2–Image and $\geq$ 2–Image Structure from Motion," NECI TR 2000. This is a greatly extended version of "New Algorithms for Two–Frame Structure from Motion," *ICCV* 737–744, 1999.

[16] J. Oliensis, "Fast and Accurate Self–Calibration" *ICCV* 745-752, 1999.

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

[18] J. Oliensis, "A New Structure from Motion Ambiguity," **PAMI** 22:7, 685–700, 2000.

[19] J. Oliensis, " Recovering Heading and Structure for Constant–Direction Motion," NEC TR 1997

[20] J. Oliensis, "Direct Multi–Frame Structure from Motion for Hand–Held Cameras," *ICPR* Vol. I: 889–895, 2000.

[21] J. Oliensis and M. Werman, "Structure from Motion using Points, Lines, and Intensities," *CVPR* Vol. 2, 599–606, 2000.

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

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

[24] J. Oliensis, "Multiframe Structure from Motion in Perspective," *Workshop on the Representations of Visual Scenes*, 77–84, 1995.

[25] J. Oliensis, "A Linear Solution for Multiframe Structure from Motion," *IUW*, 1225–1231. 1994.

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

[27] H. S. Sawhney, "Simplifying Motion and Structure Analysis Using Planar Parallax and Image Warping," *ICPR* 403–408, 1994.

[28] S. Srinivasan, "Extracting Structure from Optical Flow Using the Fast Error Search Technique," **IJCV** 37(3): 203–230, 2000.

[29] S. Srinivasan, "Fast Partial Search Solution to the 3D SFM Problem," *ICCV* 528–535, 1999.

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

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

[32] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle Adjustment—A Modern Synthesis," *Workshop on Vision Algorithms: Theory and Practice*, 298–372, 1999.

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