The 3D Line Motion Matrix and Alignment of Line Reconstructions

Adrien Bartoli and **Peter Sturm**
INRIA Rhône-Alpes
655, avenue de l'Europe
38334 Saint Ismier cedex
France.
first.last@inria.fr

Note: This work was supported by the project IST-1999-10756, VISIRE.

Abstract

We study the problem of aligning two 3D line reconstructions in projective, affine or Euclidean space.

We introduce the 6×6 3D line motion matrix that acts on Plücker coordinates. We characterize its algebraic properties and its relation to the usual 4×4 point motion matrix, and propose various methods for estimating 3D motion from line correspondences, based on cost functions defined in images or 3D space. We assess the quality of the different estimation methods using simulated data and real images.

Keywords: Lines, Reconstruction, Motion.

1 Introduction

The goal of this paper is to align two reconstructions of a set of 3D lines (figure 1). The recovered motions can be used in many areas of computer vision [6, 7, 15].

Lines are widely used for tracking [11, 27], for visual servoing [1] or for pose estimation [18] and their reconstruction has been well studied (see e.g. [4] for image detection, [22] for matching and [12, 23, 24, 25] for structure and motion).

There are three intrinsic difficulties to motion estimation from 3D line correspondences, even in Euclidean space. Firstly, there is no global linear and minimal parameterization for lines representing their 4 degrees of freedom by 4 global parameters. Secondly, there is no universally agreed error metric for comparing lines. Thirdly, depending on the representation, it may be non trivial to transfer a line between two different bases.

In this paper, which is an extended version of [2], we address the problem of motion computation using line reconstructions from images. We assume that two sets of images are given and independently registered. We also assume that correspondences of lines between these two sets are known. Reconstructing lines from each of these image sets provides the two 3D line sets to be aligned.

In projective, affine, metric and Euclidean space, motion is usually represented by 4×4 matrices (homography, affinity, similarity or rigid displacement), with different numbers of parameters, see [13] for more details. This representation is well-suited to points and planes represented using homogeneous coordinates. We call it the *usual motion matrix*.

One way to represent 3D lines is to use Plücker coordinates. They are consistent in that they do not depend on particular points or planes used to define a line. On the other hand, transferring a line between bases is not direct (one must either recover two points lying on it, transfer them and form their Plücker coordinates or transform the 4×4 skew-symmetric Plücker matrix representating the line). The problem with the Plücker matrix representation is that applying the motion is quadratic in the entries of the usual motion matrix which therefore can not be estimated linearly from line matches.

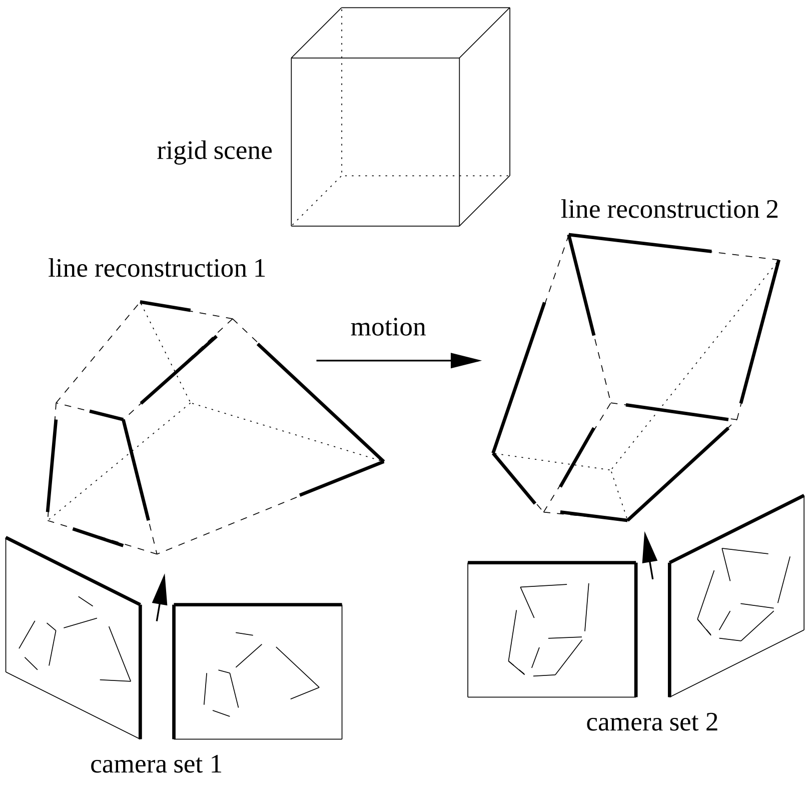


Figure 1: Our problem is to align two corresponding line reconstructions or, equivalently, to estimate the motion between the camera sets.

To overcome this, we derive a motion representation that is well-adapted to Plücker coordinates in that it transfers them linearly between bases. The transformation is represented by a 6×6 matrix that we call the *3D line motion matrix*. We characterize its algebraic properties in terms of the usual motion matrix. The expressions obtained were previously known in the Euclidean case [21]. We also deal with projective and affine spaces. We give a means of extracting the usual motion matrix from the 3D line motion matrix. A given general 6×6 matrix can then be corrected so that it exactly represents a motion[1].

Using this representation, we derive several estimators for 3D motion from line reconstructions. The motion allows lines to be transferred from the first reconstruction into the second one. Cost functions can therefore be expressed either directly in the second reconstruction basis using 3D entities or in image-related quantities, in terms of the observed and reprojected lines in the second set of images.

Our first method is based on the direct comparison of 3D Plücker coordinates. Two other methods are based on algebraic distances between reprojected lines and either observed lines or points lying on them (such as their end-points). A 6×6 matrix is recovered linearly, then corrected so that it exactly represents a motion. A fourth method uses a more physically meaningful cost function based on orthogonal distances between reprojected lines and points lying on measured lines. This requires non-linear optimization techniques that need an initialization provided by e.g. one of the proposed linear methods. We also propose a means to quasi-linearly optimize this cost function, which does not require a separate initialization method.

§2 gives some preliminaries and our notations. We introduce the 3D line motion matrix in §3 and show in §4 how to extract the usual motion matrix from it. §5 shows how these techniques can be used to estimate the motion between two reconstructions of 3D lines. We validate our methods on both simulated data and real images in §§6 and 7 respectively, and give our conclusions in §8.

[1]Compare this with the case of fundamental matrix estimation using the 8 point algorithm: the obtained 3×3 matrix is corrected so that its smallest singular value becomes zero [13].

2 Preliminaries and Notations

We make no formal distinction between coordinate vectors and physical entities. Equality up to a non-null scale factor is denoted by $\sim$, transposition and transposed inverse by $^\mathsf{T}$ and $^{-\mathsf{T}}$, and the skew-symmetric 3×3 matrix associated with the cross product by $[.]_\times$, i.e. $[\mathbf{v}]_\times \mathbf{q} = \mathbf{v} \times \mathbf{q}$. Vectors are typeset using bold fonts ($\mathbf{L}$, $\mathbf{l}$), matrices using sans-serif fonts (H, A, D) and scalars in italics. Everything is represented in homogeneous coordinates. Bars represent inhomogeneous leading parts of vectors or matrices, e.g. $\mathbf{M}^\mathsf{T} \sim \left(\bar{\mathbf{M}}^\mathsf{T} \; m\right)$. We use $\|\mathbf{v}\|$ to designate the $\mathcal{L}_2$-norm of vector $\mathbf{v}$.

Plücker line coordinates. Given two 3D points $\mathbf{M}^\mathsf{T} \sim \left(\bar{\mathbf{M}}^\mathsf{T} \; m\right)$ and $\mathbf{N}^\mathsf{T} \sim \left(\bar{\mathbf{N}}^\mathsf{T} \; n\right)$, one can form the Plücker coordinates of the line joining them as a 6-vector $\mathbf{L}^\mathsf{T} \sim \left(\mathbf{a}^\mathsf{T} \; \mathbf{b}^\mathsf{T}\right)$ defined up to scale [13]:

$$\left\{ \begin{array}{lcl} \mathbf{a} & = & \bar{\mathbf{M}} \times \bar{\mathbf{N}} \\ \mathbf{b} & = & m\bar{\mathbf{N}} - n\bar{\mathbf{M}}. \end{array} \right. \tag{1}$$

Note that other choices of constructing 6-vectors of Plücker coordinates are possible. Every choice goes with a bilinear constraint that 6-vectors have to satisfy in order to represent valid line coordinates. For our definition, the constraint is $\mathbf{a}^\mathsf{T}\mathbf{b} = 0$. An alternative representation is the *Plücker matrix* L, which is related to $\mathbf{L}$ via:

$$\mathsf{L} \sim \begin{pmatrix} [\mathbf{a}]_\times & \mathbf{b} \\ -\mathbf{b}^\mathsf{T} & 0 \end{pmatrix},$$

and to point coordinates by:

$$\mathsf{L} \sim \mathbf{M}\mathbf{N}^\mathsf{T} - \mathbf{N}\mathbf{M}^\mathsf{T}.$$

This is a skew-symmetric rank-2 4×4 matrix.

Usual motion representation. Transformations in projective, affine, metric and Euclidean spaces are usually represented by 4×4 matrices. In the general projective case, the matrices are unconstrained, while in the affine, metric and Euclidean cases they have the following forms, where R is a 3×3 rotation matrix and the other blocks do not have any special form:

projective: homography H	affine: affinity A	metric: similarity S	Euclidean: displacement D
$\begin{pmatrix} \bar{\mathsf{H}} & \mathbf{h}_1 \\ \mathbf{h}_2^\mathsf{T} & h \end{pmatrix}$	$\begin{pmatrix} \bar{\mathsf{A}} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}$	$\begin{pmatrix} s\mathsf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}$	$\begin{pmatrix} \mathsf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}$

This block partitioning of H, A, S and D will be used to define the corresponding 3D line motion matrices in §3.

Camera settings. We consider two sets of independently registered cameras. In the projective space, without loss of generality, we can express each set in a canonical reconstruction basis [20], and in particular, we can set reference cameras, e.g. the first ones to:

$$\mathsf{P} \sim \mathsf{P}' \sim (\mathrm{I}\ \mathbf{0}),$$

where P and P' are the reference cameras of the first and the second set respectively, that we call the *first* and the *second reference cameras*. Let H be the 4×4 usual homography matrix mapping points from the first basis to the second one and $\mathsf{P}'' \sim (\bar{\mathsf{P}}''\ \mathbf{p}'')$ the reference camera of the second set expressed in the first basis. These notations are illustrated on figure 2. We denote by $\boldsymbol{\pi}_\infty$ and $\boldsymbol{\pi}'_\infty$ the planes with coordinates $(0\ 0\ 0\ 1)^\mathsf{T}$ of the two reconstructions. In affine, metric or Euclidean reconstructions, these are the plane at infinity, while in projective reconstructions, they will in general correspond to two finite planes. Let $\boldsymbol{\pi}''_\infty$ be the plane in the first reconstruction corresponding to $\boldsymbol{\pi}'_\infty$. We also make use of the lines $\mathbf{L}_\infty$ and $\mathbf{L}'_\infty$ lying on $\boldsymbol{\pi}_\infty$ and $\boldsymbol{\pi}'_\infty$ respectively and related by H. In particular, $\mathbf{L}_\infty = \boldsymbol{\pi}_\infty \cap \boldsymbol{\pi}''_\infty = \{\mathbf{Q} \sim (\bar{\mathbf{Q}}^\mathsf{T}\ q)^\mathsf{T} \mid \bar{\mathbf{Q}}^\mathsf{T}\mathbf{h}_2 = q = 0\}$.

Let us give a geometrical interpretation of the components of H. This will be useful to subsequently investigate the corresponding properties of the 3D line motion matrices in §3. The fundamental matrix [19] between the reference views is given by:

$$\mathsf{F} \sim [\mathbf{p}'']_\times \bar{\mathsf{P}}'' \sim [\mathbf{h}_1]_\times \bar{\mathsf{H}}.$$

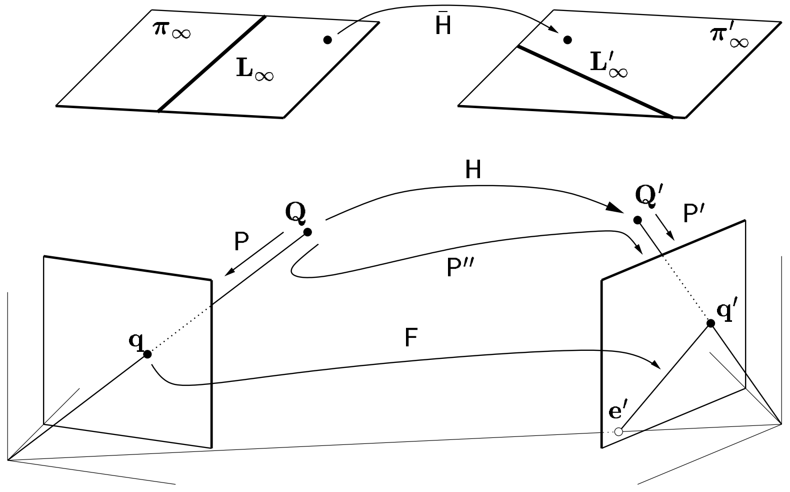


Figure 2: Camera settings. P is the perspective projection matrix from the first basis to the first reference camera and P' from the second basis to the second reference camera. P'' is the projection matrix of the second reference camera expressed in the first basis, i.e. it projects points expressed in the first basis in the second reference camera. P and P' are assumed known while P'', which depends on the motion parameters, is unknown. These settings can be easily transposed to the affine, metric and Euclidean cases.

Indeed, we have $(\bar{\mathsf{H}}\ \mathbf{h}_1) \sim (\bar{\mathsf{P}}''\ \mathbf{p}'')$ since:

$$\begin{array}{rcl} \mathsf{P}'' & \sim & \mathsf{P}'\mathsf{H} \\ (\bar{\mathsf{P}}''\ \mathbf{p}'') & \sim & (\mathrm{I}\ \mathbf{0}) \begin{pmatrix} \bar{\mathsf{H}} & \mathbf{h}_1 \\ \mathbf{h}_2^\mathsf{T} & h \end{pmatrix} \\ (\bar{\mathsf{P}}''\ \mathbf{p}'') & \sim & (\bar{\mathsf{H}}\ \mathbf{h}_1). \end{array}$$

These results may be easily specialized to the affine, metric and Euclidean spaces. The other parts of H can be interpreted as follows:

- $\bar{\mathsf{H}}$ is the 2D plane homography for points between the reference views induced by the plane $\boldsymbol{\pi}_\infty$. Indeed, let us consider $\mathbf{Q} \in \boldsymbol{\pi}_\infty$. One can easily check that $\mathbf{Q}^\mathsf{T} \sim (\mathbf{q}^\mathsf{T}\ 0)$ where $\mathbf{q}$ is the corresponding point in the first reference image and $\mathsf{P}'\mathsf{H}\mathbf{Q} \sim \bar{\mathsf{H}}\mathbf{q}$. Hence, if we deal with affine, metric or Euclidean reconstructions, then $\bar{\mathsf{H}}$ is the infinite homography [13] between the two reference views;

- $\mathbf{h}_1$ contains the coordinates of the second epipole since:
$\mathsf{F}^\mathsf{T}\mathbf{h}_1 \sim \bar{\mathsf{P}''}^\mathsf{T}[\mathbf{p}'']_\times{}^\mathsf{T}\mathbf{h}_1 \sim \bar{\mathsf{H}}^\mathsf{T}[\mathbf{h}_1]_\times{}^\mathsf{T}\mathbf{h}_1 \sim \mathbf{0}$;

- $\boldsymbol{\pi}''_\infty \sim (\mathbf{h}_2^\mathsf{T}\ h)^\mathsf{T}$ contains the coordinates of the plane at infinity of the second basis expressed in the first basis. Indeed, $\boldsymbol{\pi}''_\infty \sim \mathsf{H}^\mathsf{T}\boldsymbol{\pi}'_\infty \sim (\mathbf{h}_2^\mathsf{T}\ h)^\mathsf{T}$.

Perspective projection matrix for lines. With our choice of Plücker coordinates (1), the image projection of a line [8, 13] becomes the 3×6 linear transformation $\widetilde{\mathsf{P}}$:

$$\widetilde{\mathsf{P}} \sim \left(\det(\bar{\mathsf{P}})\bar{\mathsf{P}}^{-\mathsf{T}}\ [\mathbf{p}]_\times\bar{\mathsf{P}}\right), \tag{2}$$

where $\mathsf{P} \sim (\bar{\mathsf{P}}\ \mathbf{p})$ is the 3×4 perspective camera matrix. This result can be easily demonstrated by finding the image line joining the projections of two points on the 3D line. An explicit proof is given in the appendix. Specific forms for affine cameras and calibrated perspective cameras are straightforward to derive.

3 The 3D Line Motion Matrix

In this section, we define and examine the properties of the *3D line motion matrix* for the projective space first and then specialize it to the affine, metric and Euclidean spaces respectively.

3.1 The 3D Line Homography Matrix

The Plücker coordinates of a line, expressed in two different bases, are linearly linked. The 6×6 *matrix* $\widetilde{\mathsf{H}}$ *that we call the* 3D line homography matrix *describes the transformation in the projective case and can be parameterized as:*

$$\widetilde{\mathsf{H}} \sim \begin{pmatrix} \det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}} & [\mathbf{h}_1]_\times \bar{\mathsf{H}} \\ -\bar{\mathsf{H}}[\mathbf{h}_2]_\times & h\bar{\mathsf{H}} - \mathbf{h}_1\mathbf{h}_2^\mathsf{T} \end{pmatrix}, \tag{3}$$

where H *is the usual* 4×4 *homography matrix for points. If* $\mathbf{L}^\mathsf{T} \sim \begin{pmatrix}\mathbf{a}^\mathsf{T} & \mathbf{b}^\mathsf{T}\end{pmatrix}$ *are Plücker line coordinates (i.e.* $\mathbf{a}^\mathsf{T}\mathbf{b} = 0$*), then* $\widetilde{\mathsf{H}}\mathbf{L}$ *are the Plücker coordinates of the transformed line (i.e.* $\widetilde{\mathsf{H}}\mathbf{L}$ *satisfies the bilinear Plücker constraint).*

The proof of this result is provided in the appendix. Note that $\widetilde{\mathsf{H}}$ is a 6×6 matrix defined up to scale and subject to 20 non-linear constraints since the projective motion has 15 degrees of freedom. Other algebraic properties directly follow from equation (3). Firstly, the determinant of $\widetilde{\mathsf{H}}$ can be expressed in terms of that of H as:

$$\det(\widetilde{\mathsf{H}}) = (\det(\mathsf{H}))^3,$$

which means that if H is full-rank, then $\widetilde{\mathsf{H}}$ is also full-rank. Secondly, let $\widetilde{\mathsf{G}}$ denote the 3D line motion matrix corresponding to the usual motion matrix G. Then:

$$\begin{aligned} \mathsf{G} \sim \mathsf{H}^\mathsf{T} \quad &\Longleftrightarrow \quad \widetilde{\mathsf{G}} \sim \widetilde{\mathsf{H}}^\mathsf{T} \\ \mathsf{G} \sim \mathsf{H}^{-1} \quad &\Longleftrightarrow \quad \widetilde{\mathsf{G}} \sim \widetilde{\mathsf{H}}^{-1}. \end{aligned}$$

These properties can be easily verified using any linear algebra symbolic manipulation software such as MAPLE.

Consistency constraints on $\widetilde{\mathsf{H}}$. Let $\widetilde{\mathsf{H}}$ be subdivided in 3×3 blocks as:

$$\widetilde{\mathsf{H}} \sim \begin{pmatrix} \widetilde{\mathsf{H}}_{11} & \widetilde{\mathsf{H}}_{12} \\ \widetilde{\mathsf{H}}_{21} & \widetilde{\mathsf{H}}_{22} \end{pmatrix}.$$

By $\mathbf{r}_{ij,k}$, we denote the k-th row of matrix $\widetilde{\mathsf{H}}_{ij}$ and by $\mathbf{c}_{ij,l}$ its l-th column. We express the 20 non-linear consistency constraints that must hold on $\widetilde{\mathsf{H}}$ as follows:

$$\begin{aligned}
(\mathbf{r}_{11,k}^{\mathsf{T}})(\mathbf{r}_{12,k}) &= 0,\ k = 1\ldots 3 \\
(\mathbf{r}_{21,k}^{\mathsf{T}})(\mathbf{r}_{22,k}) &= 0,\ k = 1\ldots 3 \\
(\mathbf{c}_{11,l}^{\mathsf{T}})(\mathbf{c}_{21,l}) &= 0,\ l = 1\ldots 3 \\
(\mathbf{c}_{12,l}^{\mathsf{T}})(\mathbf{c}_{22,l}) &= 0,\ l = 1\ldots 3 \\
(\mathbf{r}_{11,k}^{\mathsf{T}})(\mathbf{r}_{22,k'}) + (\mathbf{r}_{12,k}^{\mathsf{T}})(\mathbf{r}_{21,k'}) &= 0,\ k = 1\ldots 3,\ k' = 1\ldots 3,\ k \neq k' \\
(\mathbf{r}_{11,1}^{\mathsf{T}})(\mathbf{r}_{22,1}) + (\mathbf{r}_{12,1}^{\mathsf{T}})(\mathbf{r}_{21,1}) &= (\mathbf{r}_{11,2}^{\mathsf{T}})(\mathbf{r}_{22,2}) + (\mathbf{r}_{12,2}^{\mathsf{T}})(\mathbf{r}_{21,2}) \\
&= (\mathbf{r}_{11,3}^{\mathsf{T}})(\mathbf{r}_{22,3}) + (\mathbf{r}_{12,3}^{\mathsf{T}})(\mathbf{r}_{21,3}).
\end{aligned}$$

A detailled derivation is given in the appendix. Note that these contraints are bilinear in the entries of $\widetilde{\mathsf{H}}$. These constraints are important in that they characterize the algebraic structure of a 3D line homography matrix.

Geometric interpretation of $\widetilde{\mathsf{H}}$.

- *the upper-left* 3×3 *block* $\widetilde{\mathsf{H}}_{11} \sim \bar{\mathsf{H}}^{-\mathsf{T}}$ *is the 2D plane homography for lines, between the reference views, induced by* $\boldsymbol{\pi}_\infty$. This follows from the observation made in §2 that $\widetilde{\mathsf{H}}_{11}^{-\mathsf{T}} \sim \bar{\mathsf{H}}$ is the corresponding plane homography acting on points.

- *the upper-right* 3×3 *block* $\widetilde{\mathsf{H}}_{12}$ *is the fundamental matrix between the reference views,* i.e. $\widetilde{\mathsf{H}}_{12} = \mathsf{F}$. Indeed, $\widetilde{\mathsf{H}}_{12} = [\mathbf{h}_1]_\times \widetilde{\mathsf{H}} \sim [\mathbf{p}'']_\times \bar{\mathsf{P}}'' \sim \mathsf{F}$ (cf §2).

- *the upper* 3×6 *block* $(\widetilde{\mathsf{H}}_{11}\ \widetilde{\mathsf{H}}_{12})$ *is the perspective projection matrix for Plücker line coordinates from the first basis to the second reference view.* Indeed,

  $$(\widetilde{\mathsf{H}}_{11}\ \widetilde{\mathsf{H}}_{12}) = (\det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}}\ [\mathbf{h}_1]_\times \bar{\mathsf{H}}) \sim (\det(\bar{\mathsf{P}}'')\bar{\mathsf{P}}''\ [\mathbf{p}]_\times \bar{\mathsf{P}}) \sim \widetilde{\mathsf{P}},$$

  where $\widetilde{\mathsf{P}}$ corresponds to the perspective projection matrix (2) for Plücker line coordinates (1) and P'' is the perspective projection matrix from the first basis to the second reference view (see figure 2).

- *the lower-left* 3×3 *block* $\widetilde{\mathsf{H}}_{21} \sim \bar{\mathsf{H}}[\mathbf{h}_2]_\times$ *is a degenerate line-to-point homography between the reference views*, which can be interpreted as follows. Let $\mathbf{l}$ be an image line in the first reference view. The intersection of $\mathbf{l}$ and the line of equation $\mathbf{h}_2$ is a point $\mathbf{q} \sim [\mathbf{h}_2]_\times \mathbf{l}$. The backprojection of $\mathbf{q}$ onto $\boldsymbol{\pi}_\infty$ lies on $\mathbf{L}_\infty$ and is given by $\mathbf{Q}_\infty^\mathsf{T} \sim (\mathbf{q}^\mathsf{T}\ 0)$. Its corresponding point in the second reconstruction lies therefore on $\mathbf{L}'_\infty$ and is given by ${\mathbf{Q}'}_\infty^\mathsf{T} \sim ({\mathbf{q}'}^\mathsf{T}\ 0)$ with $\mathbf{q}' \sim \bar{\mathsf{H}}[\mathbf{h}_2]_\times \mathbf{l}$. Projecting $\mathbf{Q}'_\infty$ into the second reference view gives finally $\mathbf{q}' \sim \bar{\mathsf{H}}[\mathbf{h}_2]_\times \mathbf{l}$.

  So, $\widetilde{\mathsf{H}}_{21}$ is somehow reciprocal to a fundamental matrix that maps points to lines. Whereas a fundamental matrix gives matching constraints for image points, $\widetilde{\mathsf{H}}_{21}$ does not give any matching constraints for general lines (there are none between two views).

- *the lower-right* 3×3 *block* $\widetilde{\mathsf{H}}_{22}$ *is the 2D plane homography for points, induced by* $\boldsymbol{\pi}_\infty$ *(expressed in the second basis), between the reference views*. Indeed, we have shown that $\bar{\mathsf{H}}$ is a plane homography and $\mathbf{h}_1$ the second epipole corresponding to the pair of reference views. Using the formulation of [20], $\widetilde{\mathsf{H}}_{22} = h\bar{\mathsf{H}} - \mathbf{h}_1\mathbf{h}_2^\mathsf{T}$ is the 2D plane homography induced by the plane $(\mathbf{h}_2^\mathsf{T}\ h)^\mathsf{T}$, which are the coordinates of the plane at infinity of the second basis expressed in the first one.

- *the lower* 3×6 *block* $(\widetilde{\mathsf{H}}_{21}\ \widetilde{\mathsf{H}}_{22})$ *transfers a line* $\mathbf{L}$ *from the first basis to the second one and projects its intersection point with* $\boldsymbol{\pi}'_\infty$ *into the second reference view.* This can be seen as follows. ${\mathbf{Q}'}_\infty^\mathsf{T} \sim ({\mathbf{q}'}^\mathsf{T}\ 0)$ with $\mathbf{q}' \sim (\widetilde{\mathsf{H}}_{21}\ \widetilde{\mathsf{H}}_{22})\mathbf{L}$ is the point at infinity of $\widetilde{\mathsf{H}}\mathbf{L}$ which projects to $\mathbf{q}'$ in the second reference view.

3.2 The 3D Line Affinity Matrix

For affine reconstructions, the 3D line motion matrix has the following form and we call it the 3D line affinity matrix*:*

$$\widetilde{\mathsf{A}} \sim \begin{pmatrix} \det(\bar{\mathsf{A}})\bar{\mathsf{A}}^{-\mathsf{T}} & [\mathbf{t}]_\times \bar{\mathsf{A}} \\ 0 & \bar{\mathsf{A}} \end{pmatrix}. \tag{4}$$

This result is obtained by specializing the 3D line homography matrix (3). The geometric interpretation of this matrix is very similar to the projective case. In particular, $\bar{\mathsf{A}}$ is the homography at infinity between the two reference views. Note that a 6×6 matrix defined up to scale representing an affinity is subject to 23 constraints, many of them being linear or bilinear.

3.3 The 3D Line Similarity Matrix

In metric space, the 3D line motion matrix has the following form and we call it the 3D line similarity matrix*:*

$$\widetilde{\mathsf{S}} \sim \begin{pmatrix} s\mathsf{R} & [\mathbf{t}]_\times \mathsf{R} \\ 0 & \mathsf{R} \end{pmatrix}. \tag{5}$$

This result is obtained by specializing the 3D line homography matrix (3). The geometric interpretation of this matrix is straightforward. The 3×3 upper-right block $[\mathbf{t}]_\times \mathsf{R}$ is the essential matrix between the reference views while the other two non-zero 3×3 blocks give the rotation matrix between the reference cameras, as well as the relative scale of the two reconstructions. Note that a 6×6 matrix defined up to scale representing a similarity is subject to 28 constraints.

3.4 The 3D Line Displacement Matrix

In Euclidean space, the 3D line motion matrix has the following form and we call it the 3D line displacement matrix*:*

$$\widetilde{\mathsf{D}} \sim \begin{pmatrix} \mathsf{R} & [\mathbf{t}]_\times \mathsf{R} \\ 0 & \mathsf{R} \end{pmatrix}. \tag{6}$$

This result is obtained by specializing the 3D line homography matrix (3). It coincides with that obtained in [21]. The geometric interpretation of this matrix is very similar to the metric case. Note that an homogeneous 6×6 matrix representing a rigid displacement is subject to 29 constraints.

4 Extracting the Usual Motion Matrix from the 3D Line Motion Matrix

Given a 6×6 3D line motion matrix, one can extract the corresponding motion parameters, i.e. the usual 4×4 motion matrix. In this section, we show how to extract a usual motion matrix from a 3D line motion matrix in projective, affine, metric and Euclidean spaces. We also give solutions for the cases where the 6×6 matrix considered is corrupted by noise and therefore does not exactly correspond to a motion, i.e. it differs from the forms (3), (4), (5) or (6).

4.1 Projective Space

We give an algorithm for the noise-free case in table 1. Its simple proof is given in the appendix. In the presence of noise, $\widetilde{\mathsf{H}}$ does not exactly satisfy the constraints (3) and steps 2-4 have to be achieved in a least squares sense (see below). From there, one can further improve the result, e.g. by non-linear minimization of the Frobenius norm between the given (inexact) line homography matrix and the one corresponding to the recovered usual motion parameters.

Let $\widetilde{\mathsf{H}}$ be subdivided in 3×3 blocks as: $\widetilde{\mathsf{H}} \sim \begin{pmatrix} \widetilde{\mathsf{H}}_{11} & \widetilde{\mathsf{H}}_{12} \\ \widetilde{\mathsf{H}}_{21} & \widetilde{\mathsf{H}}_{22} \end{pmatrix}$.

1. $\bar{\mathsf{H}}$: compute $\bar{\mathsf{H}} = \pm\sqrt{|\det(\widetilde{\mathsf{H}}_{11})|}\ \widetilde{\mathsf{H}}_{11}^{-\mathsf{T}}$
2. $\mathbf{h}_1$: compute $[\mathbf{h}_1]_\times = \pm\widetilde{\mathsf{H}}_{12}\bar{\mathsf{H}}^{-1}$
3. $\mathbf{h}_2$: compute $[\mathbf{h}_2]_\times = \mp\bar{\mathsf{H}}^{-1}\widetilde{\mathsf{H}}_{21}$
4. h: compute h via $h\mathrm{I}_{3\times 3} = \pm(\widetilde{\mathsf{H}}_{22} + \mathbf{h}_1\mathbf{h}_2^{\mathsf{T}})\bar{\mathsf{H}}^{-1}$.

Table 1: Extracting the homography matrix from the 3D line homography matrix. Note that extracted values are defined up to a global change of sign.

We give one way to perform a least squares estimation. Other algorithms might be possible. Steps 2 and 3 require to fit a skew-symmetric 3×3-matrix $[\mathbf{w}]_\times$ to a general

3×3 matrix W. The following solution minimizes the Frobenius norm of $[\mathbf{w}]_\times - \mathsf{W}$: $\mathbf{w} = \frac{1}{2}(W_{32} - W_{23} \quad W_{13} - W_{31} \quad W_{21} - W_{12})$. Step 4 requires to fit a scaled 3×3 identity matrix $\lambda \mathrm{I}$ to a general 3×3-matrix Λ. The following solution minimizes the Frobenius norm of $\lambda \mathrm{I} - \Lambda$: $\lambda = \frac{1}{3} \sum_i \Lambda_{ii}$.

4.2 Affine Space

We give a straightforward algorithm in table 2. This algorithm is valid for the noise-free case. For the noisy case, one can modify the proposed steps as follows.

Let $\widetilde{\mathsf{A}}$ be subdivided in 3×3 blocks as: $\widetilde{\mathsf{A}} \sim \begin{pmatrix} \widetilde{\mathsf{A}}_{11} & \widetilde{\mathsf{A}}_{12} \\ 0 & \widetilde{\mathsf{A}}_{22} \end{pmatrix}$.

1. $\bar{\mathsf{A}}$: compute $\bar{\mathsf{A}} = \widetilde{\mathsf{A}}_{22}$
2. $\mathbf{t}$: compute $[\mathbf{t}]_\times = \widetilde{\mathsf{A}}_{12} \bar{\mathsf{A}}^{-1}$.

Table 2: Extracting the affinity matrix from the 3D line affinity matrix.

For step 1, $\bar{\mathsf{A}}$ can be recovered from $\widetilde{\mathsf{A}}_{11}$ as well as $\widetilde{\mathsf{A}}_{22}$. Their average, possibly weighted, can be used to recover $\bar{\mathsf{A}}$. Step 2 can be conducted as indicated for the projective case in the previous section.

4.3 Metric Space

We give an algorithm for the noise-free case in table 3. For the noisy case, one might

Let $\widetilde{\mathsf{S}}$ be subdivided in 3×3 blocks as: $\widetilde{\mathsf{S}} \sim \begin{pmatrix} \widetilde{\mathsf{S}}_{11} & \widetilde{\mathsf{S}}_{12} \\ 0 & \widetilde{\mathsf{S}}_{22} \end{pmatrix}$.

1. normalize $\widetilde{\mathsf{S}}$ such that $\det(\widetilde{\mathsf{S}}_{22}) = 1$;
2. s: compute $s = \sqrt[6]{\det(\widetilde{\mathsf{S}}_{11})}$;
3. R: compute $\mathsf{R} = \widetilde{\mathsf{S}}_{22}$;
4. $\mathbf{t}$: compute $[\mathbf{t}]_\times = \widetilde{\mathsf{S}}_{12} \mathsf{R}^\mathsf{T}$.

Table 3: Extracting the similarity matrix from the 3D line similarity matrix.

average the two versions of R available from $\widetilde{\mathsf{D}}$ as $\mathsf{R} = (\widetilde{\mathsf{D}}_{11}/s^2 + \widetilde{\mathsf{D}}_{22})/2$ and correct the result so that the obtained matrix exactly represents a rotation by using e.g. [16] or $\mathsf{R} \leftarrow \mathsf{U}\mathsf{V}^\mathsf{T}$ where $\mathsf{R} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$ is the SVD (Singular Value Decomposition) of R. Step 4 can be achieved as indicated for the affine and projective cases.

4.4 Euclidean Space

We give an algorithm for the noise-free case in table 4. For the noisy case, one might

Let $\widetilde{\mathsf{D}}$ be subdivided in 3×3 blocks as: $\widetilde{\mathsf{D}} \sim \begin{pmatrix} \widetilde{\mathsf{D}}_{11} & \widetilde{\mathsf{D}}_{12} \\ \mathsf{0} & \widetilde{\mathsf{D}}_{22} \end{pmatrix}$.
1. normalize $\widetilde{\mathsf{D}}$ such that $\det(\widetilde{\mathsf{D}}_{11}) = 1$;
2. R: compute $\mathsf{R} = \widetilde{\mathsf{D}}_{11}$;
3. $\mathbf{t}$: compute $[\mathbf{t}]_\times = \widetilde{\mathsf{D}}_{12}\mathsf{R}^\mathsf{T}$.

Table 4: Extracting the displacement matrix from the 3D line displacement matrix.

average the two versions of R available from $\widetilde{\mathsf{D}}$ as $\mathsf{R} = (\widetilde{\mathsf{D}}_{11} + \widetilde{\mathsf{D}}_{22})/2$ and correct the result so that the obtained matrix exactly represents a rotation as in the metric case. Step 3 can be achieved as indicated for the affine and projective cases.

5 Aligning Two Line Reconstructions

We now describe how the 3D line motion matrix can be used to align two sets of N corresponding 3D lines expressed in Plücker coordinates. We examine the projective case but the method can also be used for affine, metric or Euclidean frames. We assume that the two sets of cameras are independently weakly calibrated, i.e. their projection matrices are known up to a 3D homography, so that a projective basis is attached to each set [20]. Lines can be projectively reconstructed in these two bases. Our goal is to align these 3D lines i.e. to find the projective motion between the two bases, using the line reconstructions.

5.1 General Estimation Scheme

Estimation is performed by finding $\arg\min_{\widetilde{\mathsf{H}}} \mathcal{C}$ where $\mathcal{C}$ is the cost function considered. The scale ambiguity is removed by using the additional constraint $||\widetilde{\mathsf{H}}||^2 = 1$. Non-linear optimization is performed directly on the usual motion parameters (the 16 entries of H and using the constraint $||\mathsf{H}||^2 = 1$) whereas the other estimators determine an approximate 3D line homography matrix $\widetilde{\mathsf{H}}$ first, then recover the usual homography matrix using the algorithm in table 1. The employed cost functions are non-symmetric, taking into account the errors only in the second set of images.

5.2 Estimation Based on a 3D Cost Function

Our first alignment method is "*Lin_3D*". One of the difficulties of using 3D lines for motion estimation is that there is no universally agreed error metric between two 3D lines. For that reason, we propose only one estimator based on 3D entities. This estimator is based on a direct comparison of Plücker coordinates. A measure of the distance between two 3D lines $\mathbf{L}$ and $\widehat{\mathbf{L}}$ is given by:

$$d_{3D}^2(\mathbf{L}, \widehat{\mathbf{L}}) = \sum_{k \in \{1\ldots 6\}} \left(\sum_{l \in \{1\ldots 6\}, l \neq k} \left(L_k \hat{L}_l - L_l \hat{L}_k \right)^2 \right).$$

It can be constructed as follows. Consider the 6×6 matrix $\mathsf{Z} = \mathbf{L}\widehat{\mathbf{L}}^\mathsf{T} - \widehat{\mathbf{L}}\mathbf{L}^\mathsf{T}$. If $\mathbf{L}$ and $\widehat{\mathbf{L}}$ are identical, all entries of Z vanish. Therefore, $d_{3D}(\mathbf{L}, \widehat{\mathbf{L}}) = \frac{||\mathsf{Z}||_\mathcal{F}}{\sqrt{2}}$, where $||.||_\mathcal{F}$ is the Frobenius matrix norm, can be used as a distance measure between $\mathbf{L}$ and $\widehat{\mathbf{L}}$.

The distance d_{3D} induces the following cost function:

$$\mathcal{C}_{3D} = \sum_i d_{3D}^2(\mathbf{L}_{2_i}, \widehat{\mathbf{L}}_{2_i}),$$

where $\mathbf{L}_{b\,i}$ is the i-th reconstructed line in the b-th basis ($b \in \{1, 2\}$) and $\widehat{\mathbf{L}}_{2_i} \sim \widetilde{\mathsf{H}}\mathbf{L}_{1_i}$ is the estimated line after transfer from the first to the second basis.

Finding $\widetilde{\mathsf{H}}$ that minimizes $\mathcal{C}_{3D}$ is a linear least squares problem. Concretely, it may be solved by finding the null-vector of a $30N \times 36$ matrix, using e.g. SVD.

5.3 Estimation Based on 2D (Image-Related) Cost Functions

The following cost functions are based on minimizing discrepancies between reprojected lines and lines extracted in images. Concretely, we consider the projections in the second set of images, of 3D lines in the first set, after transfer using the $\widetilde{\mathsf{H}}$ to be estimated. The discrepancy of these reprojected lines and observed ones is measured by comparing 2D lines directly or by computing the distance between reprojected lines and points on observed ones. The following cost functions are expressed in terms of observed image lines, their end-points or an arbitrary number of points along the line, and reprojected lines in the second set of images. These points need not be corresponding points.

If end-points are not available then they can be hallucinated, e.g. by intersecting the image lines with the image boundaries. The linear and quasi-linear methods need at least 7 lines to solve for the motion while the non-linear one needs 5 (which is the minimum number) but requires an initial guess.

We derive a joint projection matrix mapping a 3D line to a set of image lines in the second set of images. Let P'_j be the projection matrices of the M images corresponding to the second set of cameras (expressed in the second basis) and $\widetilde{\mathsf{P}}'_j$ the corresponding 3×6 line projection matrices (cf equation (2)). Similarly, let P_j and $\widetilde{\mathsf{P}}_j$ be the point and line projection matrices for the first set of cameras (expressed in the first basis).

Linear estimation 1. Our second alignment method "*Lin_2D_1*" directly uses the line equations in the images. End-points need not be defined. We define an algebraic measure of the distance between two image lines $\mathbf{l}$ and $\widehat{\mathbf{l}}$ by $d^2(\mathbf{l}, \widehat{\mathbf{l}}) = ||\mathbf{l} \times \widehat{\mathbf{l}}||^2$. This distance does not have any direct physical meaning, but it is zero if the two lines are identical and simple in that it is bilinear. If $\mathbf{l}$ and $\widehat{\mathbf{l}}$ had unit norm and if they were interpreted as vectors in Euclidean 3D space, then $d^2(\mathbf{l}, \widehat{\mathbf{l}})$ would be the absolute value of the sine of the angle between them.

This distance induces the cost function:

$$\mathcal{C}_1 = \sum_i \sum_j d^2(\mathbf{l}_{ij}, \widehat{\mathbf{l}}_{ij}),$$

where $\mathbf{l}_{ij}$ is the i-th observed line in the j-th image of the second set and $\widehat{\mathbf{l}}_{ij}$ the corresponding reprojection: $\widehat{\mathbf{l}}_{ij} \sim \widetilde{\mathsf{P}}'_j \widetilde{\mathsf{H}} \mathbf{L}_{1_i}$. We normalize observed image lines such that $||\mathbf{l}_{ij}||^2 = 1$. Finding $\widetilde{\mathsf{H}}$ that minimizes $\mathcal{C}_1$ is a linear least squares problem that may be solved by computing the null-vector of a $3MN \times 36$ matrix using e.g. SVD.

Linear estimation 2. Our third method "*Lin_2D_2*" uses points lying on the lines in the second image set and the algebraic distance $d_a^2(\mathbf{x}, \mathbf{l}) = \left(\mathbf{x}^\mathsf{T}\mathbf{l}\right)^2$ between an image point $\mathbf{x}$ and a line $\mathbf{l}$. This distance would have a physical meaning, i.e. the orthogonal distance between $\mathbf{x}$ and $\mathbf{l}$, if they were normalized such that $x_3 = 1$ and $l_1^2 + l_2^2 = 1$.

If we consider the end-points $\mathbf{x}_{ij}$ and $\mathbf{y}_{ij}$ of each line i of the j-th image of the second image set, this gives the cost function:

$$\mathcal{C}_2 = \sum_i \sum_j \left(d_a^2(\mathbf{x}_{ij}, \widehat{\mathbf{l}}_{ij}) + d_a^2(\mathbf{y}_{ij}, \widehat{\mathbf{l}}_{ij}) \right).$$

One can observe that an arbitrary number of points could be incorporated in $\mathcal{C}_2$ in a straightforward manner. Finding $\widetilde{\mathsf{H}}$ that minimizes $\mathcal{C}_2$ is a linear least squares problem that may be solved by computing the null-vector of a $2MN \times 36$ matrix using e.g. SVD.

Non-linear estimation 1. Our fourth method "*NLin_2D_1*" uses a geometrical cost function based on the orthogonal distance between reprojected 3D lines and their measured end-points [17], defined as $d_\perp^2(\mathbf{x}, \mathbf{l}) = \frac{\left(\mathbf{x}^\mathsf{T}\mathbf{l}\right)^2}{l_1^2 + l_2^2}$ (provided $x_3 = 1$):

$$\mathcal{C}_3 = \sum_i \sum_j \left(d_\perp^2(\mathbf{x}_{ij}, \widehat{\mathbf{l}}_{ij}) + d_\perp^2(\mathbf{y}_{ij}, \widehat{\mathbf{l}}_{ij}) \right).$$

This is non-linear in the reprojected lines $\widehat{\mathbf{l}}_{ij}$ and consequently in the entries of $\widetilde{\mathsf{H}}$, which implies the use of non-linear optimization techniques. We use the Levenberg-Marquardt algorithm [10] with numerical differentiation. The unknowns are minimally parameterized (we optimize directly the entries of H, not $\widetilde{\mathsf{H}}$), so no subsequent correction is needed to recover the motion parameters.

Quasi-linear estimation. The drawbacks of non-linear optimization are that the implementation may be complicated. For these reasons, we also developed a quasi-linear estimator "*Qlin_2D*" that minimizes the same cost function $\mathcal{C}_3$. Consider the cost functions $\mathcal{C}_2$ and $\mathcal{C}_3$. Both depend on the same data, measured image points on the line (such as end-points) and reprojected lines, the former using an algebraic and the latter the orthogonal distance. We can relate these distances by:

$$d_\perp^2(\mathbf{x}, \mathbf{l}) = w \; d_a^2(\mathbf{x}, \mathbf{l}) \quad \text{where} \quad w = \frac{1}{l_1^2 + l_2^2}, \tag{7}$$

and rewrite $\mathcal{C}_3$ as:

$$\mathcal{C}_3 = \sum_i \sum_j w_{ij} \left(d_a^2(\mathbf{x}_{ij}, \widehat{\mathbf{l}}_{ij}) + d_a^2(\mathbf{y}_{ij}, \widehat{\mathbf{l}}_{ij}) \right). \tag{8}$$

The non-linearity is hidden in the weight factors w_{ij}. If they were known, the cost function would be a sum of squares of terms that are linear in the entries of $\widetilde{\mathsf{H}}$. This leads to the following iterative algorithm. Weights, assumed unknown, are initialized to unity and iteratively updated. The loop is ended when the weights or equivalently the residual errors converge. The algorithm is summarized in table 5. It is a quasi-linear optimization that converges from the minimization of an algebraic error to the geometrical one. Its main advantages are that it is simple to implement (as a loop over a linear method) and that it gives reliable results as will be seen in the next sections.

1. *initialization*: set w_{ij}=1;
2. *estimation*: estimate $\widetilde{\mathsf{H}}$ using standard weighted least squares; the 6×6 matrix obtained is corrected so that it represents a motion, see algorithm 1;
3. *weighting*: use $\widetilde{\mathsf{H}}$ to update the weights w_{ij} according to equation (7);
4. *iteration*: iterate steps 2 and 3 until convergence (see text).

Table 5: Quasi-linear motion estimation from 3D line correspondences.

Non-linear estimation 2. The cost function $\mathcal{C}_3$ employed in methods *NLin_2D_1* and *QLin_2D* is expressed only in terms of entities of the second set of images. Hence, it is not symmetric with respect to the two sets of images. Our sixth method "*NLin_2D_2*" is based on a cost function $\mathcal{C}_4$, similar to $\mathcal{C}_3$, but that is symmetric with respect to the two sets of images.

Let $\mathbf{x}'_{ij}$ and $\mathbf{y}'_{ij}$ designate the end-points of line i in the j-th image of the first set. In order to write the expression of $\mathcal{C}_4$, we need the lines $\widehat{\mathbf{l}}'_{ij}$ projected in the first image set, after transfer from the second basis. They are given by $\widehat{\mathbf{l}}'_{ij} \sim \widetilde{\mathsf{P}}_j \widetilde{\mathsf{H}}^{-1} \mathbf{L}_{2_i}$. The cost function $\mathcal{C}_4$ can then be expressed as:

$$\mathcal{C}_4 = \mathcal{C}_3 + \sum_i \sum_j \left(d_\perp^2(\mathbf{x}'_{ij}, \widehat{\mathbf{l}}'_{ij}) + d_\perp^2(\mathbf{y}'_{ij}, \widehat{\mathbf{l}}'_{ij}) \right). \tag{9}$$

This is a non-linear function. As for method *NLin_2D_1*, we optimize the entries of H using the Levenberg-Marquardt algorithm with numerical differentiation. At each optimization step, we form matrix $\widetilde{\mathsf{H}}$. We do not compute directly the inverse of the 6×6 matrix $\widetilde{\mathsf{H}}$ to get $\widetilde{\mathsf{H}}^{-1}$, but we rather compute the 4×4 matrix H^{-1} and form the corresponding 3D line homography matrix, which is $\widetilde{\mathsf{H}}^{-1} = \widetilde{(\mathsf{H}^{-1})}$.

Other cost functions. Other distance measures between image lines have been proposed in the literature. In [24], the authors proposed the total squared distance, i.e. the sum of squared orthogonal distances along two line segments. In [25], the overlap between line segments is considered. These distances could be used for motion estimation within the above described framework, requiring non-linear optimization.

It would also be possible to use a symmetric cost function, i.e. over the two sets of images, as in [5] for the case of points.

Singularities. It has been shown that there exist critical sets of 3D lines [3]. In the case of motion estimation from 3D line correspondences, [21] shows that these sets may contain an infinite number of lines. For example, if all observed lines are coplanar, motion estimation is ambiguous. We did not encounter such a singular situation during our experiments.

6 Results Using Simulated Data

We first compare our estimators using simulated data. The test bench consists of four cameras that form two stereo pairs observing a set of $N = 50$ 3D lines randomly chosen in a sphere lying in the fields of view of all cameras. Lines are projected onto the image planes, end-points are hallucinated at the image boundaries and corrupted by additive Gaussian noise, and the equations of the image lines are estimated from these noisy end-points.

A canonical projective basis [20] is attached to each camera pair and used to reconstruct the lines in projective space. We then compute the 3D homography between the two projective bases using the estimators given in §5. We assess the quality of an estimated motion by measuring the RMS (Root Mean Square) of the Euclidean reprojection error (orthogonal distances between reprojected lines and end-points in both image pairs). This corresponds to the symmetric cost function $\mathcal{C}_4$ (equation (9)) minimized by the non-linear algorithm *NLin_2D_2*. We also monitor the computational cost of each method. We show median results over 100 trials.

Accuracy. Figure 3 shows the error as the level of added noise varies. The non-linear method is initialized using the quasi-linear one (an initialization from *Lin2_2D_2* gives similar results). We observe that the methods *Lin_3D* (based on an algebraic distance between 3D Plücker coordinates), *Lin_2D_1* and *Lin_2D_2* perform worse than the others. This is due to the fact that the cost functions used in these methods are not physically meaningful and biased compared to $\mathcal{C}_4$. Method *QLin_2D* gives results close to those obtained using *NLin_2D_1*. It is therefore a good compromise between the linear and non-linear methods, achieving good results while keeping simplicity of implementation. However, we observed that in a few cases (about 4%), the quasi-linear method does not enhance the result obtained by *Lin_2D_2* while *NLin_2D* does. *QLin_2D* estimates more parameters than necessary and this may cause numerical instabilities. The method that best minimizes the reprojection error is *NLin_2D_2*. This results could have been expected since this method consists in minimizing the reprojection error.

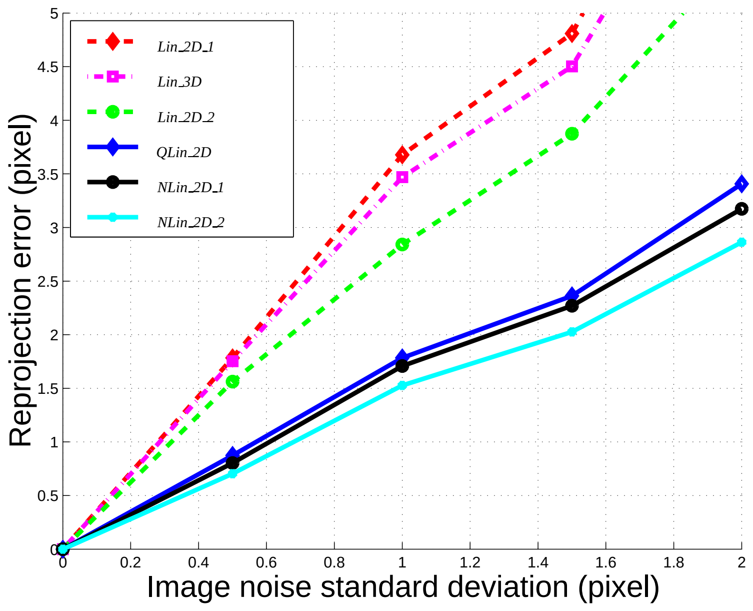


Figure 3: Comparison of the reprojection error versus added image noise for different motion estimators. The order of methods in the legend corresponds to the curves from top to bottom.

Computational cost. Figure 4 shows the computational cost as the level of added noise varies. These results have been obtained using our C implementation, on a 850 Mhz. Pentium III PC running under Windows. As expected, the linear meth-

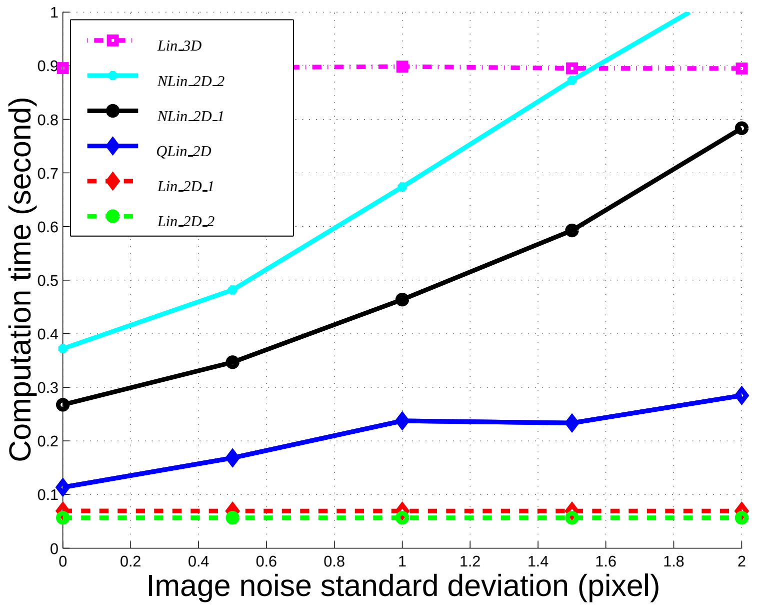


Figure 4: Comparison of the computation time versus added image noise for different motion estimators. The order of methods in the legend corresponds to the curves from top to bottom.

ods (*Lin_2D_1*, *Lin_2D_2* and *Lin_3D*) give constant results, i.e. that do not depend upon the level of added noise. Note that for these methods, the computational cost is dominated by the singular value decomposition needed to solve the linear system. Hence, the computational cost is directly linked to the size of the linear system associated to the method. This explains that method *Lin_3D*, with a $30N \times 36$ linear system to solve, has a much higher computational cost than methods *Lin_2D_1* and *Lin_2D_2* which have respectively $6N \times 36$ and $4N \times 36$ systems to solve.

On the other hand, non-linear methods (*QLin_2D*, *NLin_2D_1* and *NLin_2D_2*) give results depending on the added noise level. Indeed, the noise level influences the number of iterations needed by the method to convergence and hence, the computational cost. As for linear methods, the computational cost for method *QLin_2D* is dominated by the singular value decomposition of successive linear systems, i.e. step 2 of the algorithm shown in table 5. Hence, the computational cost is roughly given by the number of iterations multiplied by the computational cost of method *Lin_2D_2*, on which *QLin_2D* is based. This corresponds to what can be observed on figure 4. For methods *NLin_2D_1* and *NLin_2D_2*, we observe that the computational cost of each Levenberg-Marquardt iteration is dominated by the computation of the differentiation of the cost function, and slightly influenced by the resolution of the normal equations. The fact that *NLin_2D_2* has roughly twice the computational cost of *NLin_2D_1* is explained by the fact that the number of terms in the symmetric cost function $\mathcal{C}_4$ is twice that of the cost function $\mathcal{C}_3$.

7 Results on Real Images

We test our algorithms using real images. Two scenarios are considered, described in the following two sections.

The first one considers two projective line reconstructions obtained from a weakly calibrated stereo rig. The overlap between the two reconstructions is large and the recovered motion is expected to be accurate. A stereo self-calibration technique is then applied to upgrade the reconstructions to metric space.

The second scenario is based on metric reconstructions obtained from multiple views of an indoor scene. The overlap between the two reconstructions is small. Hence, the alignment is expected to be unstable.

7.1 Largely Overlapping Reconstructions

We use images taken with a weakly calibrated stereo rig, shown on figure 5. Weakly calibrated means that the fundamental matrix between the left and right images is known. In practice, we estimate it from point correspondences using the maximum likelihood estimation technique given in [26]. The epipolar geometry is the same for both image pairs. Hence, stereo self-calibration techniques can be applied to recover camera calibration from the computed 3D motion. From the fundamental matrix, we

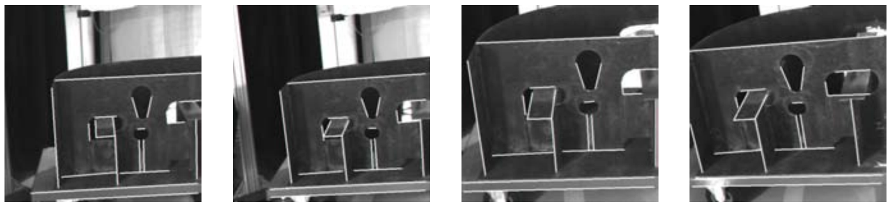
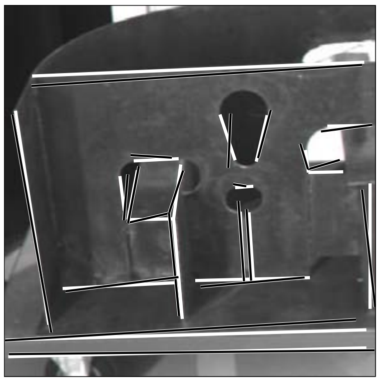


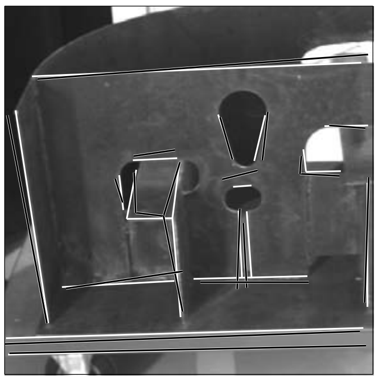
Figure 5: The two image pairs of a ship part used in the experiments, overlaid with extracted lines. Note that the extracted end-points do not necessarily correspond.

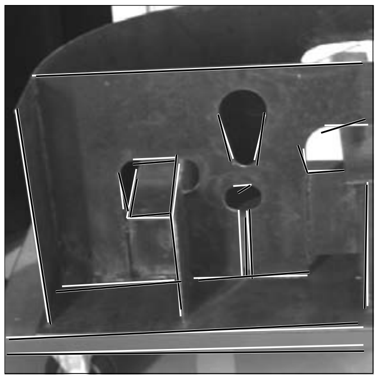
define a canonical reconstruction basis for each pair [20]. This also gives the line projection matrices $\widetilde{\mathsf{P}}_j$ and $\widetilde{\mathsf{P}}'_j$. We track 21 lines across images by hand and projectively reconstruct them for each image pair. One can observe that all lines are visible in each of the 4 images of figure 5.

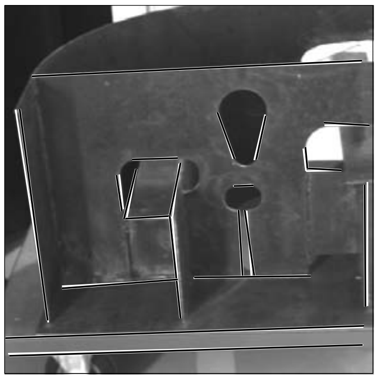
Motion estimation. We use the methods of §5 to estimate the projective motion between the two reconstructions, but since we have no 3D ground truth we will only show the result of transferring the set of reconstructed lines from the first to the second 3D frame, using the 3D line homography matrix, and reprojecting them. Figure 6 shows these reprojections, which confirms that the non-linear and quasi-linear methods achieve better results than the linear ones. Note that the results appear visually slightly worse for method *NLin_2D_2* compared to method *NLin_2D_1* since the former minimizes the reprojection error in both image sets, while the latter uses only the second image set.

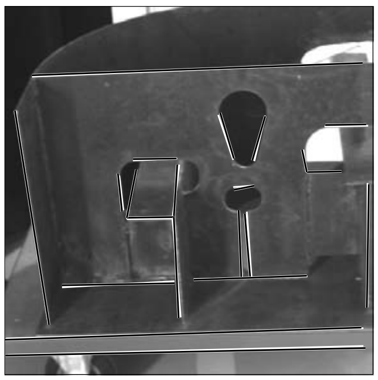
We measure the reprojection error (in both image sets) and computational cost for each method:


Lin_3D


Lin_2D_1


Lin_2D_2


QLin_2D


NLin_2D_1

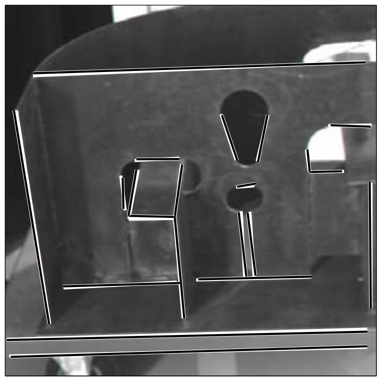

NLin_2D_2

Figure 6: The lines transferred from the first reconstruction and reprojected onto the second image pair are shown overlaid on the left image of the second pair in black for different methods while observed image lines are shown in white.

method	reprojection error (pixel)	computation time (second)
Lin_3D	3.45	0.16
Lin_2D_1	3.36	0.03
Lin_2D_2	2.83	0.02
NLin_2D_1	2.05	0.15
QLin_2D	1.93	0.09
NLin_2D_2	1.53	0.26

These results confirm those observed on figure 6: non-linear and quasi-linear methods achieve better results than linear ones. The methods with the lowest computational costs are *Lin_2D_1* and *Lin_2D_2*, while the method with the highest computational cost is *NLin_2D_2*. Methods *Lin_3D* and *NLin_2D_1* have roughly the same computational cost.

Even if there are some differences between the methods, it can be said that all of them give a correct result, i.e. the reprojection error is reasonable.

Self-calibration. We use the usual 4×4 homography matrix estimated with the method *NLin_2D_1* to self-calibrate the stereo rig using the method described in [14] with a three-parameter camera (zero skew and unit aspect ratio).

The usual 4×4 upgrade matrix, which converts a projective point reconstruction into a metric one, has the following form [14], where K is the matrix of intrinsic parameters of the left camera, $\mathbf{a}^\mathsf{T} \sim (\bar{\mathbf{a}}^\mathsf{T} \; a)$ the coordinates of the plane at infinity in the reconstruction basis and f the focal length:

$$\mathsf{H}_u \sim \begin{pmatrix} \mathsf{K}^{-1} & \mathbf{0} \\ \bar{\mathbf{a}}^\mathsf{T} & a \end{pmatrix},$$

which gives the 6×6 *3D line upgrade matrix* as:

$$\widetilde{\mathsf{H}}_u \sim \begin{pmatrix} \frac{1}{f^2}\mathsf{K}^\mathsf{T} & 0 \\ -\mathsf{K}^{-1}[\bar{\mathbf{a}}]_\times & a\mathsf{K}^{-1} \end{pmatrix}.$$

We compare the intrinsic parameters recovered for the left camera to those estimated using off-line calibration [9]:

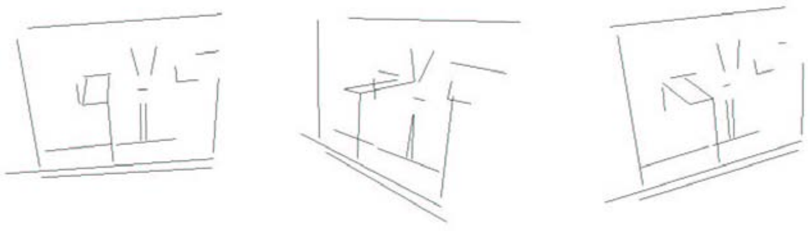


Figure 7: Reconstructed lines after self-calibration.

parameter	self-calib.	off-line calib.	% error
f	1514.22	1461.02	3.51
u_0	252.93	267.98	5.95
v_0	250.68	241.03	3.84

where (u_0, v_0) are the image coordinates of the principal point.

Figure 7 shows different points of view of the reconstruction we obtained from the first pair. We hallucinate 3D end-points by intersecting the viewing rays of image end-points with the reconstructed lines from the first pair (we use those from the left image of the first pair). Qualitatively, the result seems to be correct.

7.2 Slightly Overlapping Reconstructions

We use two sets of images, shown on figures 8 and 9, taken with a calibrated camera. For each image set, we use point correspondences to get camera positions and used them to reconstruct 3D lines, as shown on figure 10 and 11.

The observed scene is composed of two stacks of boxes and a laptop. In the first image set, the leftmost stack of boxes is not visible, while in the second image set, the laptop is not visible. Hence, even if 40 lines are reconstructed from each image set, the overlap is constituted by 15 lines only, lying on the middle stack of boxes and closely located in space.

We apply our alignment algorithms to these data. The results are visible on figure 12. Visually, the results lie in two categories. The linear methods give very biased

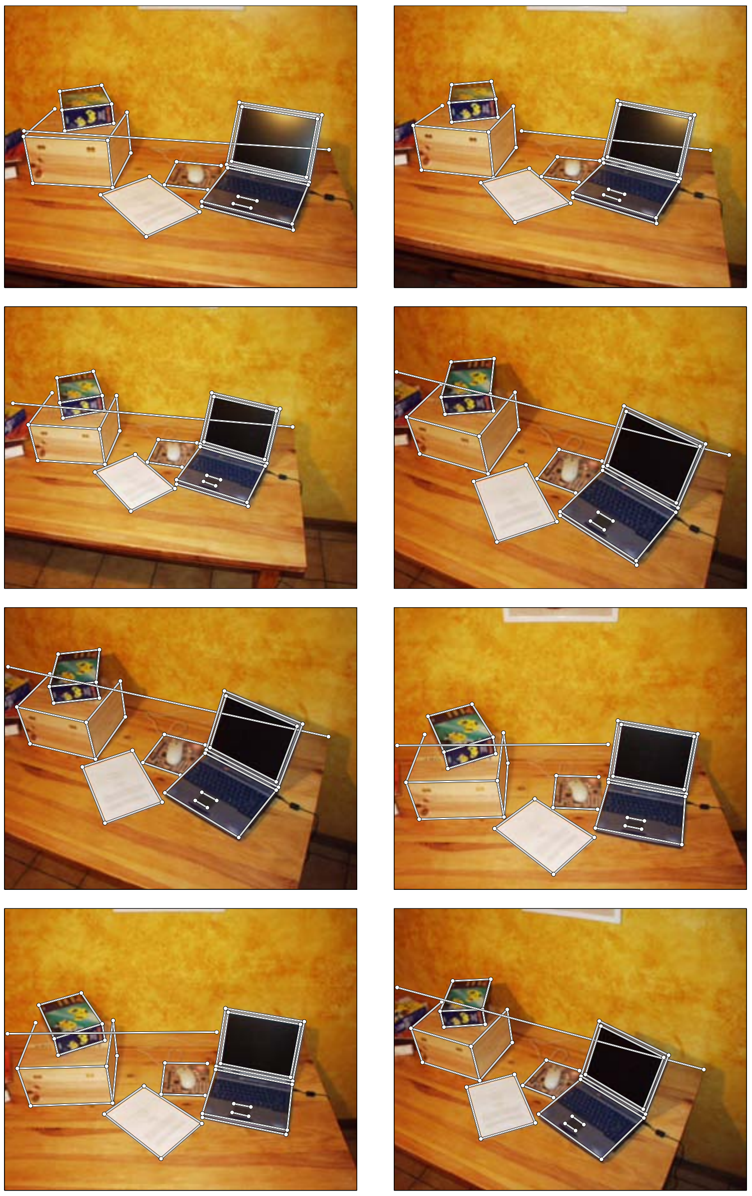


Figure 8: The first set of images, overlaid with extracted lines.

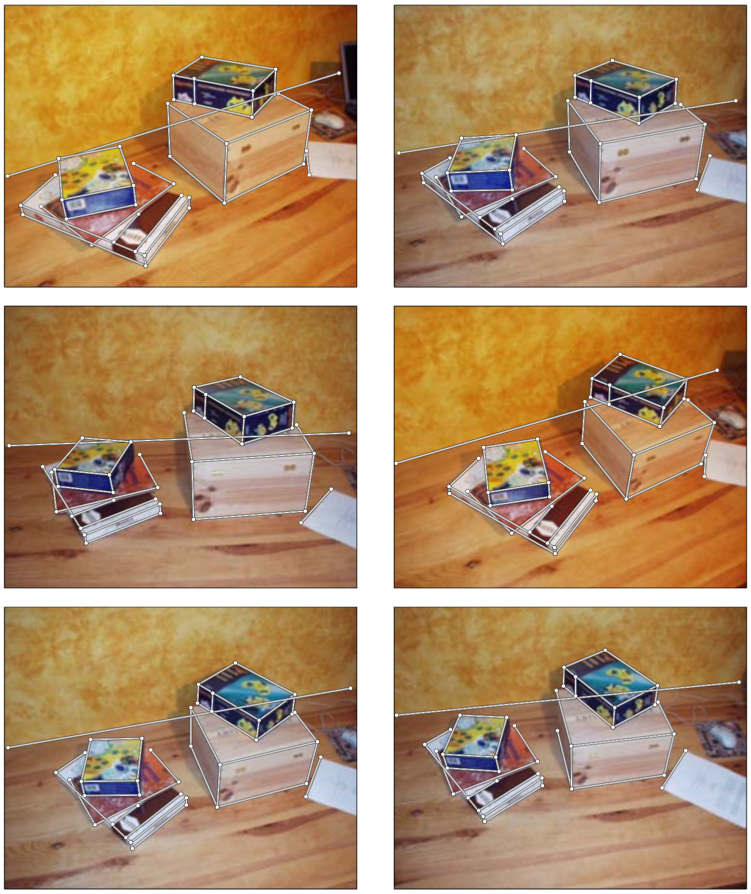


Figure 9: The second set of images, overlaid with extracted lines.

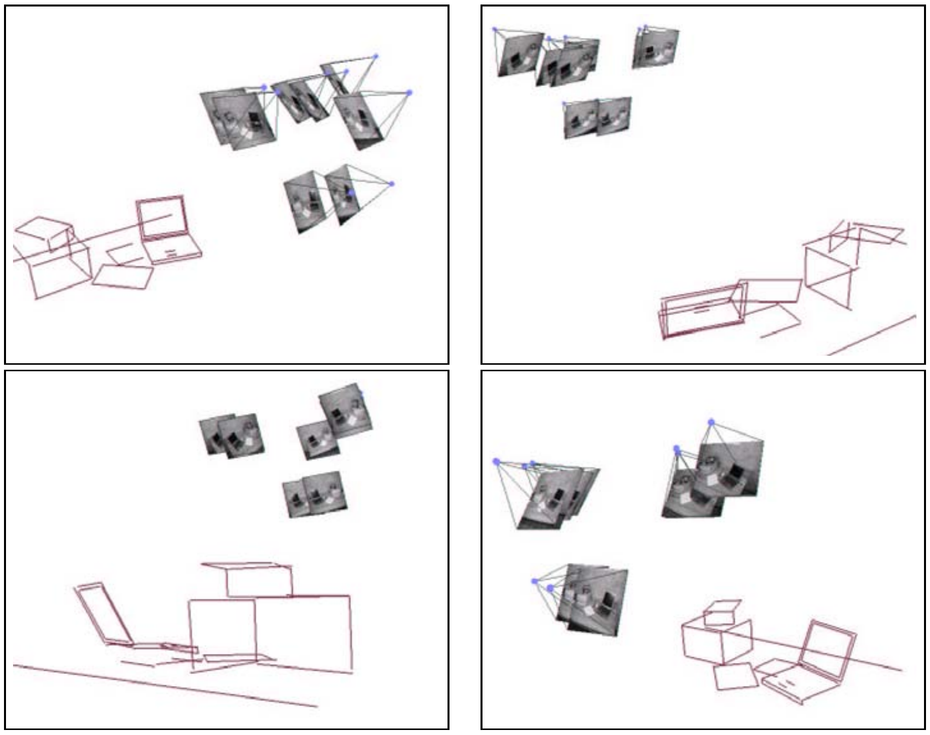


Figure 10: The 40 lines reconstructed from the first set of images.

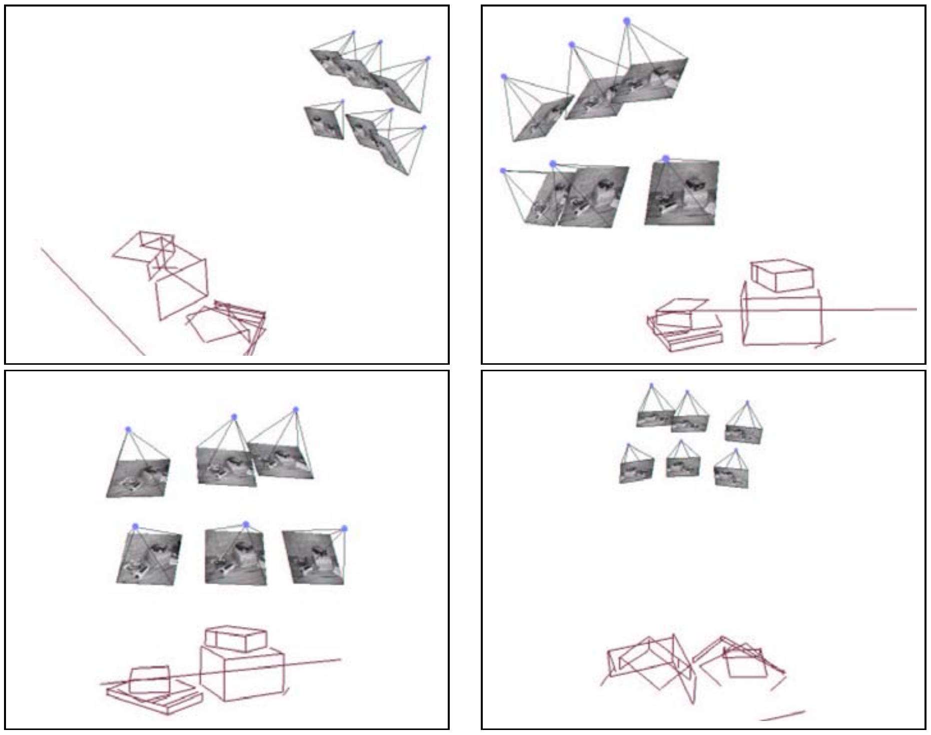


Figure 11: The 40 lines reconstructed from the second set of images.

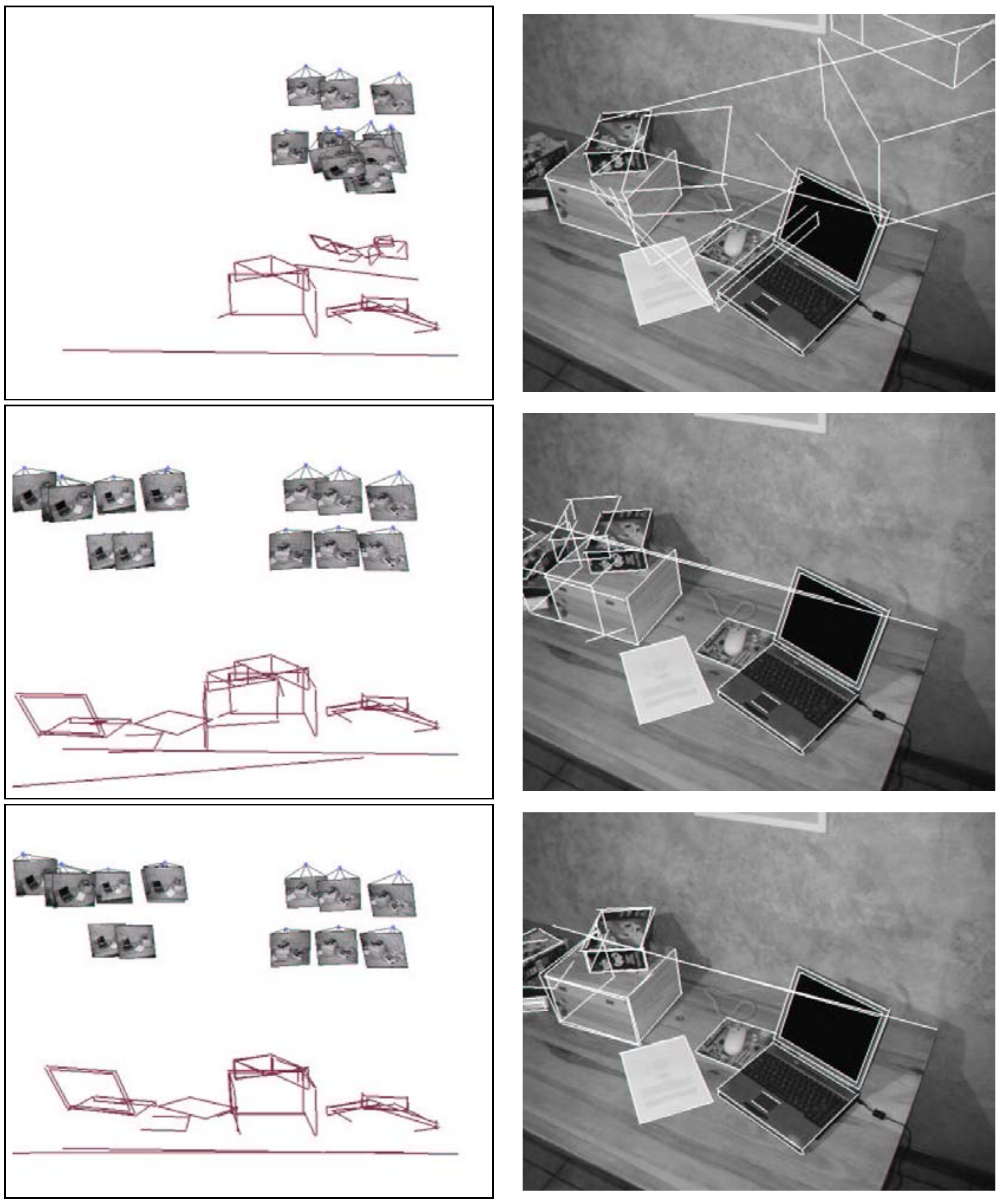


Figure 12: The two reconstructed sets of lines, from to to bottom: without alignment, alignment with linear methods and alignment with non-linear/quasi-linear methods. The left column shows views of the reconstructions, while the right column shows the reprojection in an original image.

results, leading to bad alignment, while the non-linear methods (including the quasi-linear one) give reliable results.

We measure the reprojection error and computational cost for each method:

method	reprojection error (pixel)	computation time (second)
Lin_3D	14.49	0.12
Lin_2D_1	15.10	0.02
Lin_2D_2	13.28	0.01
NLin_2D_1	2.95	0.17
QLin_2D	2.91	0.08
NLin_2D_2	1.76	0.31

These measurements confirm the previous observation: the reprojection error is of an order of 10 pixels for linear methods, which is large, while it is of an order of a few pixels for non-linear/quasi-linear methods. These results are explained by the fact that only a few line correspondences are available and that they are closely located in space.

8 Conclusions

We addressed the problem of estimating the motion between two line reconstructions in the general projective case. We used Plücker coordinates to represent 3D lines and showed that they could be transferred linearly between two reconstruction bases using a 6×6 3D line homography matrix. We specialized this result to the affine, metric and Euclidean cases. We investigated the algebraic properties of this matrix and showed how to extract the usual 4×4 motion matrices (i.e. homography, affinity or rigid displacement) from them.

We then proposed several 3D and image-based estimators for the motion between two line reconstructions. Experimental results on both simulated and real data show that the linear estimators perform worse than the non-linear ones, especially when the cost function is expressed in 3D space. The non-linear and quasi-linear estimators, based on orthogonal image errors give similar good results. Concerning the computational cost, we show that linear methods based on 2D cost functions are not expensive,

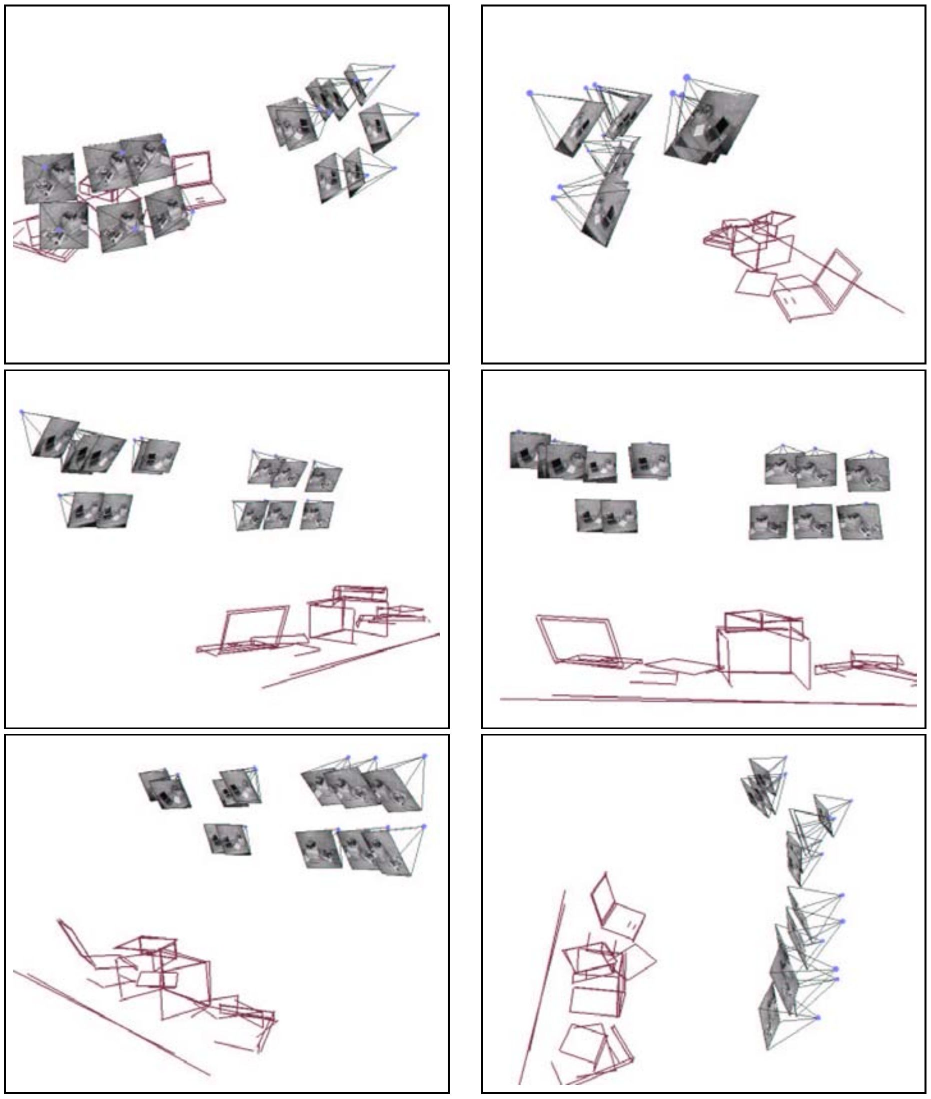


Figure 13: Some views of the reconstructions aligned with the non-linear method *NLin_2D_2*, consisting of 65 lines and 14 cameras.

compared to non-linear methods, while the linear method based on a 3D cost function may be as expensive as a non-linear method.

More specifically, when the overlap between the two reconstructions is large, as it can be expected when a continuous image sequence is processed, the alignement obtained with linear methods is reliable. Hence these methods could be used for real-time stereo tracking of lines, in a manner similar to [6].

A Proofs and Derivations

In this appendix, we derive and prove some results mentioned in this paper.

Perspective projection matrix for lines. Consider a line with Plücker coordinates $\mathbf{L}^\mathsf{T} \sim (\mathbf{a}^\mathsf{T}\ \mathbf{b}^\mathsf{T})$ defined by two points $\mathbf{M}^\mathsf{T} \sim (\bar{\mathbf{M}}^\mathsf{T}\ m)$ and $\mathbf{N}^\mathsf{T} \sim (\bar{\mathbf{N}}^\mathsf{T}\ n)$ that are projected onto $\mathbf{m}$ and $\mathbf{n}$ respectively by the perspective projection matrix P. The corresponding image line is $\mathbf{l}$. Expanding its expression leads to the perspective projection matrix for lines, $\widetilde{\mathsf{P}}$ as:

$$\begin{aligned}
\mathbf{l} &\sim \mathbf{m} \times \mathbf{n} \\
&\sim (\mathsf{P}\mathbf{M}) \times (\mathsf{P}\mathbf{N}) \\
&\sim (\bar{\mathsf{P}}\bar{\mathbf{M}} + m\mathbf{p}) \times (\bar{\mathsf{P}}\bar{\mathbf{N}} + n\mathbf{p}) \\
&\sim (\bar{\mathsf{P}}\bar{\mathbf{M}}) \times (\bar{\mathsf{P}}\bar{\mathbf{N}}) + m\mathbf{p} \times (\bar{\mathsf{P}}\bar{\mathbf{N}}) - n\mathbf{p} \times (\bar{\mathsf{P}}\bar{\mathbf{M}}) \\
&\sim \det(\bar{\mathsf{P}})\bar{\mathsf{P}}^{-\mathsf{T}}(\bar{\mathbf{M}} \times \bar{\mathbf{N}}) + [\mathbf{p}]_\times \bar{\mathsf{P}}(m\bar{\mathbf{N}} - n\bar{\mathbf{M}}) \\
&\sim \widetilde{\mathsf{P}}\mathbf{L} \text{ where } \widetilde{\mathsf{P}} \sim (\det(\bar{\mathsf{P}})\bar{\mathsf{P}}^{-\mathsf{T}}\ [\mathbf{p}]_\times \bar{\mathsf{P}}). \blacksquare
\end{aligned}$$

Deriving the 3D line homography matrix. Consider a line with coordinates $\mathbf{L}_1^\mathsf{T} \sim (\mathbf{a}_1^\mathsf{T}\ \mathbf{b}_1^\mathsf{T})$ defined by two points $\mathbf{M}_1^\mathsf{T} \sim (\bar{\mathbf{M}}_1^\mathsf{T}\ m_1)$ and $\mathbf{N}_1^\mathsf{T} \sim (\bar{\mathbf{N}}_1^\mathsf{T}\ n_1)$ in the first projective basis and Plücker coordinates $\mathbf{L}_2^\mathsf{T} \sim (\mathbf{a}_2^\mathsf{T}\ \mathbf{b}_2^\mathsf{T})$ defined by points $\mathbf{M}_2^\mathsf{T} \sim (\bar{\mathbf{M}}_2^\mathsf{T}\ m_2)$ and $\mathbf{N}_2^\mathsf{T} \sim (\bar{\mathbf{N}}_2^\mathsf{T}\ n_2)$ in the second projective basis. Expanding the expressions for $\mathbf{a}_2$ and $\mathbf{b}_2$ according to the definition of Plücker coordinates (1) gives respectively the 3×6 upper and lower parts of $\widetilde{\mathsf{H}}$:

$$\begin{aligned}
\mathbf{a}_2 &= \bar{\mathbf{M}}_2 \times \bar{\mathbf{N}}_2 \\
&= \left(\bar{\mathsf{H}}\bar{\mathbf{M}}_1 + m_1\mathbf{h}_1\right) \times \left(\bar{\mathsf{H}}\bar{\mathbf{N}}_1 + n_1\mathbf{h}_1\right) \\
&= \det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}}(\bar{\mathbf{M}}_1 \times \bar{\mathbf{N}}_1) + [\mathbf{h}_1]_\times \bar{\mathsf{H}}\left(m_1\bar{\mathbf{N}}_1 - n_1\bar{\mathbf{M}}_1\right) \\
&= \det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{a}_1 + [\mathbf{h}_1]_\times\bar{\mathsf{H}}\mathbf{b}_1,
\end{aligned}$$

$$\begin{aligned}
\mathbf{b}_2 &= m_2\bar{\mathbf{N}}_2 - n_2\bar{\mathbf{M}}_2 \\
&= \mathbf{h}_2^{\mathsf{T}}\left(\bar{\mathbf{M}}_1\bar{\mathsf{H}}\bar{\mathbf{N}}_1 - \bar{\mathbf{N}}_1\bar{\mathsf{H}}\bar{\mathbf{M}}_1\right) \\
&\quad + \mathbf{h}_2^{\mathsf{T}}\left(\bar{\mathbf{M}}_1\mathbf{h}_1 n_1 - \bar{\mathbf{N}}_1\mathbf{h}_1 m_1\right) \\
&\quad + h\bar{\mathsf{H}}(m_1\bar{\mathbf{N}}_1 - n_1\bar{\mathbf{M}}_1) \\
&= -\bar{\mathsf{H}}[\mathbf{h}_2]_\times\mathbf{a}_1 - \mathbf{h}_1\mathbf{h}_2^{\mathsf{T}}\mathbf{b}_1 + h\bar{\mathsf{H}}\mathbf{b}_1. \ \blacksquare
\end{aligned}$$

The specialization of this result to the affine, metric and Euclidean spaces shown in §§3.2 and 3.4 respectively is straightforward.

Deriving the 20 consistency constraints on the 3D line homography matrix. We prove 20 non-linear consistency constraints that must hold on the entries of a 3D line homography matrix. Note that there exist other possible constraints. We use the notation defined in §3.1.

Consider the product $\widetilde{\mathsf{H}}_{11}\widetilde{\mathsf{H}}_{12}^{\mathsf{T}}$. Its expansion leads to:

$$\begin{aligned}
\widetilde{\mathsf{H}}_{11}\widetilde{\mathsf{H}}_{12}^{\mathsf{T}} &\sim \bar{\mathsf{H}}^{-\mathsf{T}}\bar{\mathsf{H}}^{\mathsf{T}}[\mathbf{h}_1]_\times \\
&\sim [\mathbf{h}_1]_\times,
\end{aligned}$$

which is a skew-symmetric matrix. It means that its diagonal entries vanish, which corresponds to the following 3 constraints on the 3D line homography matrix:

$$(\mathbf{r}_{11,k}^{\mathsf{T}})(\mathbf{r}_{12,k}) = 0, \ k = 1 \ldots 3.$$

Note that 3 other constraints could be derived from this equation based on the off-diagonal entries.

Similarly, consider the product $\widetilde{\mathsf{H}}_{21}\widetilde{\mathsf{H}}_{22}^{\mathsf{T}}$. Its expansion leads to:

$$\begin{aligned}
\widetilde{\mathsf{H}}_{21}\widetilde{\mathsf{H}}_{22}^{\mathsf{T}} &\sim \bar{\mathsf{H}}[\mathbf{h}_2]_\times \left(h\bar{\mathsf{H}}^{\mathsf{T}} - \mathbf{h}_2\mathbf{h}_1^{\mathsf{T}}\right) \\
&\sim \bar{\mathsf{H}}[\mathbf{h}_2]_\times h\bar{\mathsf{H}}^{\mathsf{T}} - \bar{\mathsf{H}}[\mathbf{h}_2]_\times \mathbf{h}_2\mathbf{h}_1^{\mathsf{T}} \\
&\sim \bar{\mathsf{H}}[\mathbf{h}_2]_\times \bar{\mathsf{H}}^{\mathsf{T}} \\
&\sim [\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{h}_2]_\times,
\end{aligned}$$

where we used the rule:

$$[\mathsf{J}\mathbf{g}]_\times = \det(\mathsf{J})\mathsf{J}^{-\mathsf{T}}[\mathbf{g}]_\times\mathsf{J}^{-1} \sim \mathsf{J}^{-\mathsf{T}}[\mathbf{g}]_\times\mathsf{J}^{-1}. \tag{10}$$

As previously, we end up with a skew-symmetric matrix whose diagonal entries vanish, giving another 3 constraints:

$$(\mathbf{r}_{21,k}^{\mathsf{T}})(\mathbf{r}_{22,k}) = 0,\ k = 1\ldots 3.$$

Similarly, observe that:

$$\begin{aligned}
\widetilde{\mathsf{H}}_{11}^{\mathsf{T}}\widetilde{\mathsf{H}}_{21} &\sim \bar{\mathsf{H}}^{-1}\bar{\mathsf{H}}[\mathbf{h}_2]_\times \\
&\sim [\mathbf{h}_2]_\times,
\end{aligned}$$

and that:

$$\begin{aligned}
\widetilde{\mathsf{H}}_{12}^{\mathsf{T}}\widetilde{\mathsf{H}}_{22} &\sim \bar{\mathsf{H}}^{\mathsf{T}}[\mathbf{h}_1]_\times(h\bar{\mathsf{H}} - \mathbf{h}_1\mathbf{h}_2^{\mathsf{T}}) \\
&\sim \bar{\mathsf{H}}^{\mathsf{T}}[\mathbf{h}_1]_\times\bar{\mathsf{H}} \\
&\sim [\bar{\mathsf{H}}^{-1}\mathbf{h}_1]_\times,
\end{aligned}$$

which gives, respectively, the following 6 constraints:

$$(\mathbf{c}_{11,l}^{\mathsf{T}})(\mathbf{c}_{21,l}) = 0,\ l = 1\ldots 3$$

$$(\mathbf{c}_{12,l}^{\mathsf{T}})(\mathbf{c}_{22,l}) = 0,\ l = 1\ldots 3.$$

The first 12 constraints are derived. We derive the remaining 8 contraints as follows. Consider the following equation:

$$\widetilde{\mathsf{H}}_{11}\widetilde{\mathsf{H}}_{22}^{\mathsf{T}} + \widetilde{\mathsf{H}}_{12}\widetilde{\mathsf{H}}_{21}^{\mathsf{T}} \sim \det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}}(h\bar{\mathsf{H}}^{\mathsf{T}} - \mathbf{h}_2\mathbf{h}_1^{\mathsf{T}}) + [\mathbf{h}_1]_\times\bar{\mathsf{H}}[\mathbf{h}_2]_\times\bar{\mathsf{H}}^{\mathsf{T}}.$$

After expansion and by using equation (10) for the last term, we obtain:

$$\begin{aligned}\widetilde{\mathsf{H}}_{11}\widetilde{\mathsf{H}}_{22}^{\mathsf{T}}+\widetilde{\mathsf{H}}_{12}\widetilde{\mathsf{H}}_{21}^{\mathsf{T}} &\sim h\mathrm{I}-\det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{h}_2\mathbf{h}_1^{\mathsf{T}}+[\mathbf{h}_1]_\times\det(\bar{\mathsf{H}})[\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{h}_2]_\times\\ &\sim h\mathrm{I}-\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{h}_2\mathbf{h}_1^{\mathsf{T}}-[\mathbf{h}_1]_\times[\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{h}_2]_\times^{\mathsf{T}}.\end{aligned}$$

Define $\mathbf{a}=\bar{\mathsf{H}}^{-\mathsf{T}}\mathbf{h}_2$ and $\mathbf{b}=\mathbf{h}_1$ and use the following rule:

$$\mathbf{a}\mathbf{b}^{\mathsf{T}}+[\mathbf{b}]_\times[\mathbf{a}]_\times^{\mathsf{T}}=(\mathbf{a}^{\mathsf{T}}\mathbf{b})\mathrm{I},$$

which gives:

$$\begin{aligned}\widetilde{\mathsf{H}}_{11}\widetilde{\mathsf{H}}_{22}^{\mathsf{T}}+\widetilde{\mathsf{H}}_{12}\widetilde{\mathsf{H}}_{21}^{\mathsf{T}} &\sim h\mathrm{I}-(\mathbf{h}_2^{\mathsf{T}}\bar{\mathsf{H}}^{-1}\mathbf{h}_1)\mathrm{I}\\ &\sim \mathrm{I}.\end{aligned}$$

From this equation, we deduce that all diagonal entries of matrix $\widetilde{\mathsf{H}}_{11}\widetilde{\mathsf{H}}_{22}^{\mathsf{T}}+\widetilde{\mathsf{H}}_{12}\widetilde{\mathsf{H}}_{21}^{\mathsf{T}}$ are equal and all off-diagonal entries are zero, which gives the remaining 8 constraints:

$$\begin{aligned}(\mathbf{r}_{11,k}^{\mathsf{T}})(\mathbf{r}_{22,k'})+(\mathbf{r}_{12,k}^{\mathsf{T}})(\mathbf{r}_{21,k'}) &= 0,\ k\in 1\ldots 3,\ k'\in 1\ldots 3,\ k\neq k'\\ (\mathbf{r}_{11,1}^{\mathsf{T}})(\mathbf{r}_{22,1})+(\mathbf{r}_{12,1}^{\mathsf{T}})(\mathbf{r}_{21,1}) &= (\mathbf{r}_{11,2}^{\mathsf{T}})(\mathbf{r}_{22,2})+(\mathbf{r}_{12,2}^{\mathsf{T}})(\mathbf{r}_{21,2})\\ &= (\mathbf{r}_{11,3}^{\mathsf{T}})(\mathbf{r}_{22,3})+(\mathbf{r}_{12,3}^{\mathsf{T}})(\mathbf{r}_{21,3}).\ \blacksquare\end{aligned}$$

Extracting the usual homography matrix from the 3D line homography matrix. To prove the correctness of algorithm 1 we may reform the 3D line homography matrix $\widetilde{\mathsf{H}}'$ corresponding to the extracted usual motion parameters (given by equation (3)) and verify that each of its blocks corresponds to the original block of $\widetilde{\mathsf{H}}$, as follows:

$$\begin{aligned}
\widetilde{\mathsf{H}}'_{11} &= \det(\bar{\mathsf{H}})\bar{\mathsf{H}}^{-\mathsf{T}} \\
&= \det\left(\pm\sqrt{|\det(\widetilde{\mathsf{H}}_{11})|}\widetilde{\mathsf{H}}_{11}^{-\mathsf{T}}\right)\frac{\pm 1}{\sqrt{|\det(\widetilde{\mathsf{H}}_{11})|}}\widetilde{\mathsf{H}}_{11} \\
&= \pm\sqrt{|\det(\widetilde{\mathsf{H}}_{11})|}^{3}\frac{1}{\det(\widetilde{\mathsf{H}}_{11})}\frac{\pm 1}{\sqrt{|\det(\widetilde{\mathsf{H}}_{11})|}}\widetilde{\mathsf{H}}_{11} \\
&= \pm\widetilde{\mathsf{H}}_{11}
\end{aligned}$$

$$\begin{aligned}
\widetilde{\mathsf{H}}'_{12} &= [\mathbf{h}_1]_\times\bar{\mathsf{H}} \\
&= \pm\widetilde{\mathsf{H}}_{12}(\pm\bar{\mathsf{H}}^{-1})\bar{\mathsf{H}} \\
&= \pm\widetilde{\mathsf{H}}_{12}
\end{aligned}$$

$$\begin{aligned}
\widetilde{\mathsf{H}}'_{21} &= -\bar{\mathsf{H}}[\mathbf{h}_2]_\times \\
&= -\mp\bar{\mathsf{H}}(\pm\bar{\mathsf{H}}^{-1})\widetilde{\mathsf{H}}_{21} \\
&= \pm\widetilde{\mathsf{H}}_{21}
\end{aligned}$$

$$\begin{aligned}
\widetilde{\mathsf{H}}'_{22} &= h\bar{\mathsf{H}} - \mathbf{h}_1\mathbf{h}_2^{\mathsf{T}} \\
&= \pm(\widetilde{\mathsf{H}}_{22} + (\pm\mathbf{h}_1)(\pm\mathbf{h}_2^{\mathsf{T}}))(\pm\bar{\mathsf{H}}^{-1})\bar{\mathsf{H}} - \mathbf{h}_1\mathbf{h}_2^{\mathsf{T}} \\
&= \pm\widetilde{\mathsf{H}}_{22}. \ \blacksquare
\end{aligned}$$

References

[1] N. Andreff, B. Espiau, and R. Horaud. Visual servoing from lines. In *International Conference on Robotics and Automation*, San Francisco, April 2000.

[2] A. Bartoli and P. Sturm. The 3D line motion matrix and alignment of line reconstructions. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Kauai, Hawaii, USA*, volume I, pages 287–292. IEEE Computer Society Press, December 2001.

[3] T. Buchanan. Critical sets for 3D reconstruction using lines. In G. Sandini, editor, *Proceedings of the 2nd European Conference on Computer Vision, Santa Margherita Ligure, Italy*, pages 730–738. Springer-Verlag, May 1992.

[4] J. Canny. A computational approach to edge detection. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[5] G. Csurka, D. Demirdjian, and R. Horaud. Finding the collineation between two projective reconstructions. *Computer Vision and Image Understanding*, 75(3):260–268, September 1999.

[6] D. Demirdjian and R. Horaud. Motion-egomotion discrimination and motion segmentation from image-pair streams. *Computer Vision and Image Understanding*, 78(1):53–68, April 2000.

[7] F. Devernay and O. Faugeras. From projective to euclidean reconstruction. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, San Francisco, California, USA*, pages 264–269, June 1996.

[8] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 951–956, June 1995.

[9] O.D. Faugeras and G. Toscani. Camera calibration for 3D computer vision. In *Proceedings of International Workshop on Machine Vision and Machine Intelligence, Tokyo, Japan*, 1987.

[10] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, 1981.

[11] G. Hager and K. Toyama. X vision : A portable substrate for real-time vision applications. *CVIU*, 69(1):23–37, 1998.

[12] R.I. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision*, 22(2):125–140, 1997.

[13] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June 2000.

[14] R. Horaud, G. Csurka, and D. Demirdjian. Stereo calibration from rigid motions. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1446–1452, December 2000.

[15] R. Horaud, F. Dornaika, and B. Espiau. Visually guided object grasping. IEEE *Transactions on Robotics and Automation*, 1997.

[16] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, July 1988.

[17] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, USA*, pages 482–488, June 1998.

[18] Y. Liu, T.S. Huang, and O.D. Faugeras. Determination of camera location from 2D to 3D line and point correspondences. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990.

[19] Q.T. Luong and O. Faugeras. The fundamental matrix: Theory, algorithms and stability analysis. *International Journal of Computer Vision*, 17(1):43–76, 1996.

[20] Q.T. Luong and T. Vieville. Canonic representations for the geometries of multiple projective views. *Computer Vision and Image Understanding*, 64(2):193–229, 1996.

[21] N. Navab and O. D. Faugeras. The critical sets of lines for camera displacement estimation: a mixed euclidean-projective and constructive approach. *International Journal of Computer Vision*, 23(1):17–44, 1997.

[22] C. Schmid and A. Zisserman. Automatic line matching across views. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA*, pages 666–671, 1997.

[23] M. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990.

[24] C.J. Taylor and D.J. Kriegman. Structure and motion from line segments in multiple images. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, November 1995.

[25] Z. Zhang. Estimating motion and structure from correspondences of line segments between two perspective images. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 17(12):pp. 1129–1139, June 1994.

[26] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, March 1998.

[27] Z. Zhang and O. D. Faugeras. Tracking and grouping 3D line segments. In *Proceedings of the 3rd International Conference on Computer Vision, Osaka, Japan*, pages p. 577–580, December 1990.