CVPR 01 - IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, HAWAII, USA, PP. 287-292, VOL. 1, DECEMBER 2001.

The 3D Line Motion Matrix and Alignment of Line Reconstructions

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

Abstract

We study the problem of aligning two 3D line reconstructions expressed in Plücker line coordinates.

We introduce the 6×6 3D line motion matrix that acts on Plücker coordinates in projective, affine or Euclidean space. 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 image-related and 3D cost functions. We assess the quality of the different estimation methods using simulated data and real images.

1. Introduction

The goal of this paper is to align two reconstructions of *3D* lines (figure 1). The recovered motions can be used in many areas of computer vision, including tracking and motion segmentation, visual servoing and self-calibration.

Lines are widely used for tracking [5, 17], for visual servoing [1] or for pose estimation [8] and their reconstruction has been well studied (see e.g. [2] for image detection, [12] for matching and [13, 14, 15] 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 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, we address the problem of motion computation using projective line reconstructions. This is the most general case, so our results can easily be specialized to affine and Euclidean spaces. See [18] for a review of previous work on the Euclidean case.

In each of these spaces, motion is usually represented by 4×4 matrices (homography, affinity or rigid displacement), with different numbers of parameters. See [6] for more details. This representation is well-suited to points and planes. We call it the usual motion matrix.

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

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

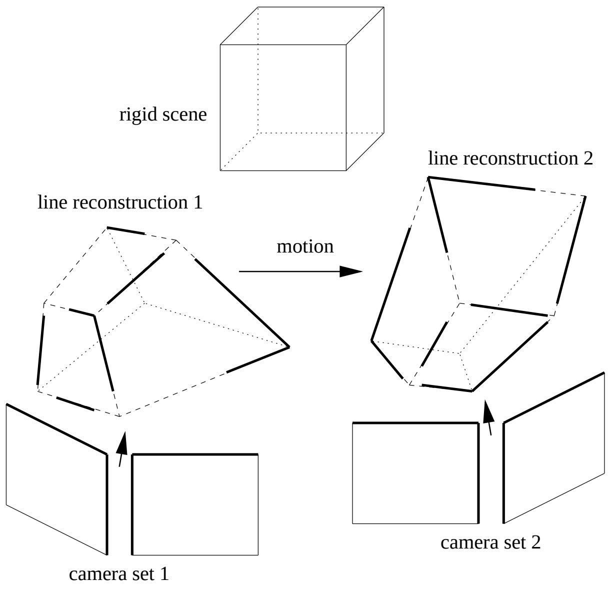

Figure 1. Our problem is to estimate the motion of the cameras between two corresponding line reconstructions.

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 the algebraic properties of this in terms of the usual motion matrix. The expressions obtained were previously known in the Euclidean case [10, 18]. We give a means of extracting the usual motion matrix from the *3D* line motion matrix and show how to correct a general 6×6 matrix so that it represents a motion (compare this with the case of the fundamental matrix estimation using the 8 point algorithm: the obtained 3×3 matrix is corrected so that its smallest singular value becomes zero [16]).

Using this representation, we derive several estimators for *3D* motion from line reconstructions. The motion allows lines to be transfered and reprojected from the first reconstruction onto the images of the second one. Optimization criteria can therefore be expressed in image-related quantities, in terms of the actual and reprojected lines in the second set of images.

Our two first methods are based on algebraic distances between reprojected lines and either actual lines or their end-points. A third method is based on direct comparison of Plücker coordinates. A 6×6 matrix is recovered linearly, then corrected so that it exactly represents a motion. A fourth method uses a more physically meaningful criterion based on orthogonal distances between reprojected lines and actual end-points. This requires non-linear optimization techniques that need an initialization provided by a linear method. To avoid the use of non-linear optimization while keeping the criterion, we devise a method that quasi-linearly optimizes it and that does not require a separate initialization.

§2 gives some preliminaries and our notation. We introduce the *3D* line motion matrix in §3 and show how this can be used to estimate the motion between two reconstructions of *3D* lines in §4. We validate our methods on both simulated data and real images in §§5 and 6 respectively, and give our conclusions and perspectives in §7.

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 ($\mathbf{L}$, $\mathbf{l}$) and matrices using sans-serif fonts (H, A, D). Everything is represented in homogeneous coordinates. Bars represent inhomogeneous parts, e.g. $\mathbf{M}^\mathsf{T} \sim \begin{pmatrix}\bar{\mathbf{M}}^\mathsf{T} & m\end{pmatrix}$.

Plücker line coordinates. Given two *3D* points $\mathbf{M}^\mathsf{T} \sim \begin{pmatrix}\bar{\mathbf{M}}^\mathsf{T} & m\end{pmatrix}$ and $\mathbf{N}^\mathsf{T} \sim \begin{pmatrix}\bar{\mathbf{N}}^\mathsf{T} & n\end{pmatrix}$, one can form the Plücker matrix representing the line joining them 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 [6]. The Plücker coordinates $\mathbf{L}^\mathsf{T} \sim \begin{pmatrix}\mathbf{a}^\mathsf{T} & \mathbf{b}^\mathsf{T}\end{pmatrix}$ of the line are its 6 different (up to sign) off-diagonal entries, written as a vector. There are many ways of arranging them. We choose the following:

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

i.e. $\mathsf{L} \sim \begin{pmatrix} [\mathbf{a}]_\times & -\mathbf{b} \\ \mathbf{b}^\mathsf{T} & 0 \end{pmatrix}$. The constraint $\det \mathsf{L} = 0$ corresponds to $\mathbf{a}^\mathsf{T}\mathbf{b} = 0$.

Standard motion representation. Motions in projective, affine and Euclidean spaces are usually represented by 4×4 matrices. In the general projective case, the matrices are unconstrained, while in the affine and Euclidean cases they have the following forms, where R is a 3×3 rotation matrix:

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

3. The *3D* Line Motion Matrix

In this section, we define the *3D* line motion matrix in the projective case, and then specialize it to the affine and Euclidean cases.

Proposition 1 *The Plücker coordinates of a line, expressed in two different bases, are linearly linked. The 6×6 matrix $\widetilde{\mathsf{H}}$ describing the transformation in the projective case is called the 3D line homography matrix and can be parameterized as :*

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

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 the Plücker coordinates of a line (i.e. $\mathbf{a}^\mathsf{T}\mathbf{b} = 0$), then $\widetilde{\mathsf{H}}\mathbf{L}$ are the Plücker coordinates of the transformed line.

Proof: consider a line with coordinates $\mathbf{L}_1^\mathsf{T}$ defined by two points $\mathbf{M}_1^\mathsf{T}$ and $\mathbf{N}_1^\mathsf{T}$ in the first projective basis and coordinates $\mathbf{L}_2^\mathsf{T}$ defined by points $\mathbf{M}_2^\mathsf{T}$ and $\mathbf{N}_2^\mathsf{T}$ 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 $\widetilde{\mathsf{H}}$:

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

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

Corollary 1 *In affine and Euclidean coordinates, the 3D line motion matrix takes the forms:*

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

This result coincides with that obtained in [10, 18] in the Euclidean case.

Extracting the motion 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. An algorithm is given in table 1 for the projective case. In the presence of noise $\widetilde{\mathsf{H}}$ does not exactly satisfy the constraints and steps 2-4 have to be achieved in a least square sense. From there, one can further improve the result by non-linear minimization of the Frobenius norm between the given line homography and the one corresponding to the recovered motion. This algorithm can be specialized by

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

Table 1. Extracting the point homography from the *3D* line homography matrix.

considering the special structure of the *3D* line motion matrix in the affine and Euclidean cases.

4. 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 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 [9]. 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.

General estimation scheme. For the reasons mentioned in the introduction, we have chosen to use image-based cost functions. Alternatively, we could use an algebraic distance between Plücker coordinates to linearly estimate the motion using *3D* lines (see [3] in the case of points). This estimator is called "*Lin3D*".

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 motion parameters (the entries of H) whereas the other estimators determine $\widetilde{\mathsf{H}}$ first, then recover the motion using algorithm 1.

Our cost functions are expressed in terms of observed image lines or their end-points, and reprojected lines in the second set of images. They are therefore non-symmetric, taken into account only the error in the second set of images. We derive a perspective projection matrix for *3D* lines expressed in Plücker coordinates and a joint projection matrix mapping a *3D* line to a set of image lines in the second set of images.

If end-points are not available they can be hallucinated, e.g. by intersecting the image lines with the image boundaries. The linear and quasi-linear methods need at least 9 lines to solve for the motion while the non-linear one needs 4 but requires an initial guess.

Perspective projection matrix for lines. With our choice of Plücker coordinates (1), the image projection of a line [6] becomes the linear transformation $\widetilde{\mathsf{P}} \sim \left(\det(\bar{\mathsf{P}})\bar{\mathsf{P}}^{-\mathsf{T}} \ [\mathbf{p}]_\times \bar{\mathsf{P}}\right)_{3\times 6}$, where $\mathsf{P} \sim (\bar{\mathsf{P}} \ \mathbf{p})$ is the perspective camera matrix. This result can be easily demonstrated by finding the image line joining the projections of two points on the line.

Let P_j be the projection matrices of the m images corresponding to the second reconstruction. We define the joint projection matrix for lines as:

$$\mathcal{P}^{\mathsf{T}} = \left(\widetilde{\mathsf{P}}_1^{\mathsf{T}} \ \ldots \ \widetilde{\mathsf{P}}_m^{\mathsf{T}}\right).$$

Linear estimation 1. Our first alignment method "*Lin1*" directly uses the line equations in the images. End-points need not be available. We define an algebraic measure of 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 significance, but it is zero if the two lines are identical and simple in that it is bilinear. This distance induces the error criterion:

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

where $\mathbf{l}_{ij}$ is the i-th observed line in the j-th image and $\widehat{\mathbf{l}}_{ij}$ the corresponding reprojection. Each term of the sum over i can be written as $\mathsf{B}_i \mathcal{P} \widetilde{\mathsf{H}} \mathbf{L}_i$ where B_i is a $3m \times 3m$ rank-$2m$ matrix defined as:

$$\mathsf{B}_i = \begin{pmatrix} [\mathbf{l}_{i1}]_\times & & \\ & \ddots & \\ & & [\mathbf{l}_{im}]_\times \end{pmatrix}.$$

These equations can be rearranged to form a linear system in the unknown entries of $\widetilde{\mathsf{H}}$, where each line correspondence provides $3m$ equations. The system can be solved using SVD (Singular Value Decomposition) [11] to obtain a solution that satisfies $||\widetilde{\mathsf{H}}||^2 = 1$ as the null-vector of a $3mn \times 36$ matrix.

Linear estimation 2. Our second method "*Lin2*" uses observed end-points 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 line $\mathbf{l}$. This gives the criterion:

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

where $\mathbf{x}_{ij}$ and $\mathbf{y}_{ij}$ designate the end-points of the i-th line in the j-th image. Each term of the sum over i can be written as $\mathsf{C}_i \mathcal{P} \widetilde{\mathsf{H}} \mathbf{L}_i$ where

$$\mathsf{C}_i = \begin{pmatrix} \mathbf{x}_{i1}{}^\mathsf{T} & & \\ \mathbf{y}_{i1}{}^\mathsf{T} & & \\ & \ddots & \\ & & \mathbf{x}_{im}{}^\mathsf{T} \\ & & \mathbf{y}_{im}{}^\mathsf{T} \end{pmatrix}$$

is a full-rank $2m \times 3m$ matrix. These equations can be rearranged to form a linear system in the unknown entries of $\widetilde{\mathsf{H}}$. Each line correspondence accounts for $2m$ equations. The system can be solved by SVD [11] of a $2mn \times 36$ matrix.

Non-linear estimation. Our third method "*NLin*" uses a physical cost function based on the orthogonal distance between reprojected *3D* lines and their measured end-points [7], defined as $d_\perp^2(\mathbf{x}, \mathbf{l}) = \frac{(\mathbf{x}^\mathsf{T}\mathbf{l})^2}{l_1^2 + l_2^2}$:

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

This is non-linear in the image lines and consequently in the entries of $\widetilde{\mathsf{H}}$, which implies the use of non-linear optimization techniques. 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 is complicated and the computational cost is high. For these reasons, we also developed a quasi-linear estimator "*Qlin*" that minimizes the same cost function. Consider the cost functions $\mathcal{C}_2$ and $\mathcal{C}_3$. Both depend on the same data, measured end-points and reprojected lines, the former using an algebraic and the latter the orthogonal distance. We can relate these distances by:

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

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

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

The non-linearity is hidden in the weight factors $w_{\mathbf{l}_{ij}}$. If they were known, the criterion would be linear in the entries of $\widetilde{\mathsf{H}}$. This leads to the following iterative algorithm. Weights, assumed unknown, are initialized to 1 and iteratively updated. The loop is ended when the weights or equivalently the error converge. The algorithm is summarized in table 2. It is a quasi-linear optimization that converges from the algebraic minimum error to the geometrical one. It is simple to implement (as a loop over a linear method) and less sensitive to local minima than the non-linear method [4].

1. *initialization*: set $w_{\mathbf{l}_{ij}}$=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 $w_{\mathbf{l}_{ij}}$ according to equation (2);
4. *iteration*: iterate steps 2. and 3. until convergence (see text).

Table 2. Quasi-linear motion estimation from *3D* line correspondences.

5. 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 *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 [9] 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 §4. We assess the quality of an estimated motion by measuring the RMS (Root Mean Square) of the Euclidean reprojection errors (orthogonal distances between reprojected lines and end-points in the second image pair). This corresponds to the criterion minimized by the non-linear and the quasi-linear algorithms. We compute the median error over 100 trials.

Figure 2 shows the error as the level of added noise varies. The non-linear method is initialized using the quasi-linear one. We observe that the methods *Lin3D* (based on an

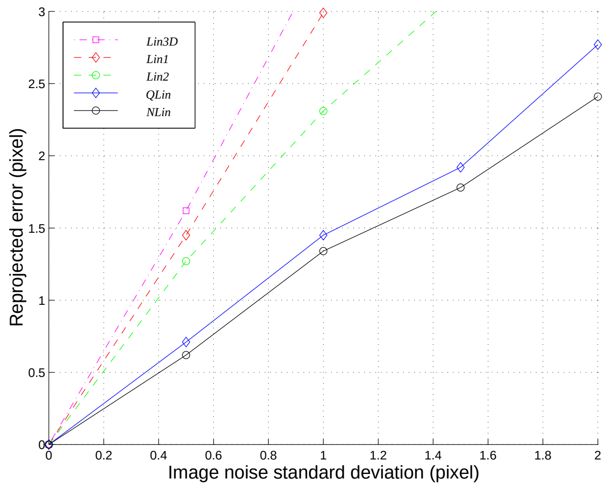

Figure 2. Comparison of reprojected error versus added image noise for different motion estimators.

algebraic distance between *3D* Plücker coordinates), *Lin1* and *Lin2* perform worse than the others. This is due to the fact that the criteria used in these methods are not physically meaningful and biased compared to $\mathcal{C}_3$. Method *QLin* gives results close to those obtained using *NLin*. 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 *Lin2* while *NLin* does. *QLin* estimates more parameters than necessary and this may cause numerical instabilities.

6. Results on Real Images

We also tested our algorithms using images taken with a stereo rig, so that the epipolar geometry is the same for both image pairs, see figure 3. We use the technique given in [16]

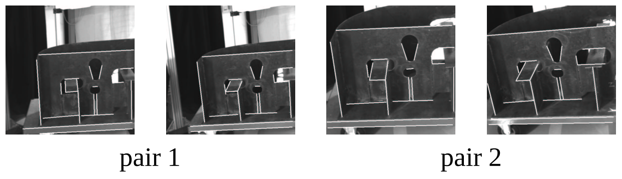

Figure 3. The two image pairs of a ship part used in the experiments, overlaid with actual lines. Note that the extracted end-points do not necessarily correspond.

to estimate the fundamental matrix and define a canonical reconstruction basis for each pair [9]. This also gives the joint line projection matrix $\mathcal{P}$. We track lines across images by hand and projectively reconstruct them for each image pair.

Motion estimation. We used the methods of §4 to estimate the projective motion between the two reconstruction bases, 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 4 shows these reprojections, which confirms that the non-linear and quasi-linear methods achieve much better results than the linear ones.

7. Conclusions and Perspectives

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 transfered linearly between two reconstruction bases using a 6×6 *3D* line motion matrix. We investigate the algebraic properties of this matrix and show how to extract the usual 4×4 motion matrix (i.e. homography, affinity or rigid displacement) from it.

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 (the residuals are at least twice as large) than the non-linear ones, especially

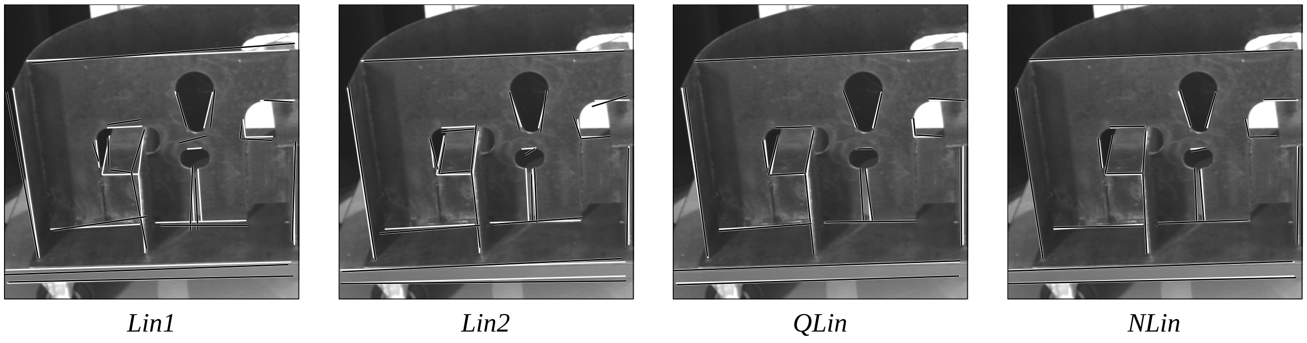

Figure 4. The lines transferred from the first reconstruction and reprojected onto the second image pair are shown overlaid on the left image of the second pair in black with white boundaries for different methods while actual image lines are shown in white. The method *Lin3D* (not shown here) gives results worse than *Lin1*.

when the cost function is expressed in *3D* space. The non-linear and quasi-linear estimators, based on orthogonal image errors give similar results.

In future work we plan to investigate the use of the *3D* line motion matrix for line tracking without using an a priori known model of the scene. Another promising avenue will be to reconstruct piecewise linear scenes by aligning line reconstructions obtained from multiple sequences of images.

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

[3] 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.

[4] 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.

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

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

[7] 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*, 1998.

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

[9] 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.

[10] N. Navab, O.D. Faugeras, and T. Viéville. The critical sets of lines for camera displacement estimation: A mixed euclidean-projective and constructive approach. In *Proceedings of the 4th International Conference on Computer Vision, Berlin, Germany*, pages 713–723. IEEE Computer Society Press, May 1993.

[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] Z. Zhang and O. D. Faugeras. Determining motion from 3D line segments: A comparative study. *Image and Vision Computing*, 9(1):10–19, February 1991.