Structure from Planar Motions with Small Baselines

René Vidal[1] and John Oliensis[2]

[1] Department of EECS, 301 Cory Hall, Berkeley CA 94710
rvidal@eecs.berkeley.edu
[2] NEC Research Institute, 4 Independence Way, Princeton NJ 08540
oliensis@research.nj.nec.com

Abstract. We study the multi-frame structure from motion problem when the camera translates on a plane with small baselines and arbitrary rotations. This case shows up in many practical applications, for example, in ground robot navigation. We consider the framework for small baselines presented in [8], in which a factorization method is used to compute the structure and motion parameters accurately, efficiently and with guaranteed convergence. When the camera translates on a plane, the algorithm in [8] cannot be applied because the estimation matrix drops rank, causing the equations to be no longer linear. In this paper, we show how to linearly solve those equations, while preserving the accuracy, speed and convergence properties of the non-planar algorithm. We evaluate the proposed algorithms on synthetic and real image sequences, and compare our results with those of the optimal algorithm. The proposed algorithms are very fast and accurate, have less than 0.3% outliers and work well for small–to–medium baselines and non-planar as well as planar motions.

1 Introduction

The structure from motion problem has been studied extensively over the past decades and many algorithms have been proposed for general camera motion (see [12] for batch methods, [5, 10] for recursive methods, [11, 13, 14] for factorization methods and [2] for projective methods).

However, most of these algorithms are not designed to give accurate answers when the baselines are small, which is the most complex case because the signal-to-noise ratio is small. Since the small baseline case shows up in most vision applications in control and robotics, we believe it is fundamental to design algorithms that work well in this case.

In [8], an algorithm explicitly designed for small baselines was presented, showing the possibility of computing structure and motion parameters accurately, efficiently and with guaranteed convergence. The translation is taken as zero initially, and then the algorithm repeatedly updates the rotation, translation, and depth estimates until they converge (convergence depends explicitly on the small baseline assumption). The translation and depth estimates are obtained linearly from an approximate rank–3 factorization of a matrix which depends on the image displacements and the current rotation estimates.

The above algorithm works only for non-planar motion, *i.e.,* when the camera positions do not all lie in a plane or line. When the camera motion is planar or linear (as happens for example in ground robot navigation), the matrix used in [8] to compute the translations and depths has an approximate rank of 2 or 1, rather than 3, and thus the algorithm of [8] is not applicable.

In [7, 9] (see also [3]), an algorithm is proposed which is designed to give accurate results for small baselines and linear motion. Though it also works for planar or fully non-planar motions, it gives less accurate results for these cases.

In this paper, we study the *planar motion* case in more detail. We show that, even though the equations relating depth and translation are no longer linear (as they were in the non-planar or linear motion cases), it is possible to cancel the non-linearities to obtain linear solutions. The proposed algorithms are very fast and accurate, have less than 0.3% outliers and work well for small–to–medium baselines and non-planar as well as planar motions.

2 Non-planar motion algorithm

We consider an image sequence containing N_P points in N_F frames. We use $\mathbf{x}_p^i = (x_p^i, y_p^i, 1)^T$, $p = 1 \cdots N_P$, $i = 0 \cdots N_F - 1$, to denote the image coordinates of the p^{th} point in the i^{th} image and choose the zeroth image as the reference frame. By convention $\mathbf{x}_p = (x_p, y_p, 1)^T = (x_p^0, y_p^0, 1)^T$. The motion of the i^{th} camera frame with respect to the zeroth is described by a rotation matrix $R^i \in SO(3)$ and a translation vector $T^i = (T_x^i, T_y^i, T_z^i)^T \in \mathbb{R}^3$. Let Z_p be the depth of the p^{th} point in 3D with respect to the zeroth camera frame. The image points in the i^{th} frame are related to those of the zeroth by:

$$\lambda \mathbf{x}_p^i = R^i(\mathbf{x}_p - T^i/Z_p), \quad \lambda = \left[R^i(\mathbf{x}_p - T^i/Z_p)\right]_z . \tag{1}$$

Let τ be the ratio between the largest translation and the smallest depth, *i.e.,* $\tau = T_{\max}/Z_{\min}$. We say that the *baselines are small* if $\tau << 1$. Under this assumption, one can initialize all the translations to be zero and then solve linearly for the rotations from (1). It is shown in [6] that the errors $\|\Omega^i\|$ between these rotation estimates R_{est}^i and the true rotations R_{true}^i are approximately proportional to τ, where $\Omega^i \in \mathbb{R}^3$ is such that $R_{true}^i R_{est}^{iT} = \exp([\Omega^i]_\times) \in SO(3)$. (Here, $[u]_\times \in so(3)$ represents the skew-symmetric matrix generating the cross product, *i.e.,* for all $u, v \in \mathbb{R}^3$ we have $u \times v = [u]_\times v$).

The initial rotation estimates R_{est}^i are used to warp the image points $\mathbf{x}_p^i$ from the i^{th} to the reference frame. From the warped image points, we define a vector of *displacements* $\mathbf{d}_p^i$ with respect to the reference frame as:

$$\mathbf{d}_p^i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(\frac{R_{est}^{iT} \mathbf{x}_p^i}{(R_{est}^{iT} \mathbf{x}_p^i)_z} - \mathbf{x}_p \right),$$

from which we form a *displacement* matrix $D \in \mathbb{R}^{2N_P \times (N_F-1)}$. It is shown in [8] that the displacement matrix satisfies:

$$D \approx \Phi(Z^{-1})\, T + \Psi\, \Omega, \tag{2}$$

where

$$\begin{aligned}
\Phi(Z^{-1}) &= \begin{bmatrix} -\{Z^{-1}\} & 0 & \{xZ^{-1}\} \\ 0 & -\{Z^{-1}\} & \{yZ^{-1}\} \end{bmatrix} && \in \mathbb{R}^{2N_P \times 3} \\
T &= [T^1 \ \cdots \ T^{N_F-1}] && \in \mathbb{R}^{3 \times (N_F-1)} \\
\Psi &= \begin{bmatrix} -\{xy\} & \{1+x^2\} & -\{y\} \\ -\{1+y^2\} & \{xy\} & \{x\} \end{bmatrix} && \in \mathbb{R}^{2N_P \times 3} \\
\Omega &= [\Omega^1 \ \cdots \ \Omega^{N_F-1}] && \in \mathbb{R}^{3 \times (N_F-1)}.
\end{aligned}$$

In the above equations, $\{xZ^{-1}\}$ (for example) denotes a vector whose p^{th} component is $x_p Z_p^{-1}$. Notice that Ψ can be interpreted as the matrix of rotational flows and Ω as the matrix of (unknown) residual rotational velocities.

Equation (2) depends on Ω which is of order τ and thus approximately proportional to the size of the translations. Therefore, if we solved for T from (2) after neglecting the unknown Ω, we would obtain translation estimates with an error of order τ, *i.e.,* of the order of the translations themselves. In order to obtain translation estimates with a small error of order τ^2, we define a matrix $H \in \mathbb{R}^{(2N_P-3) \times 2N_P}$ annihilating the rotational flows, *i.e.,* such that $H\Psi = 0$. Multiplying (2) by H gives:

$$HD \approx H\Phi(\{Z^{-1}\})\, T. \tag{3}$$

We conclude that HD (which can be computed from the given image points and the initial rotations) has a rank that is approximately equal to the rank of T, which is either 1, 2 or 3, depending on whether the motion is linear, planar or non-planar, respectively. In the non-planar motion case, one can use the singular value decomposition to factorize HD into its structure $S \in \mathbb{R}^{(2N_P-3) \times 3}$ and motion $M \in \mathbb{R}^{(N_F-1) \times 3}$ components as:

$$HD = SM^T = SUU^{-1}M^T \approx H\Phi(\{Z^{-1}\})\, T, \tag{4}$$

where $U \in \mathbb{R}^{3 \times 3}$ is an arbitrary nonsingular matrix. Given this factorization, one can solve linearly for $\{Z^{-1}\}$ and U from the equation $H\Phi(\{Z^{-1}\}) = SU$ and obtain the translation vectors from $T = U^{-1}M^T$.

Given these new estimates for the translations and depths, the algorithm in [8] improves the initial estimates for the rotation linearly from (1). Then, the algorithm repeatedly updates the rotation, translation, and depth estimates until they converge. Under the small baseline assumption, one can show that the incremental change in the unknowns between the k^{th} and $(k+1)^{th}$ iteration is approximately proportional to τ^k, and hence the algorithm has good convergence properties. (See [8] for details).

We summarize the algorithm for non-planar motion as follows[1]:

[1] The algorithm in [8] differs from this description by including an iteration that corrects the small–baseline approximation in (2).

Algorithm 1 (Non-planar Motion Algorithm) *Given a set of N_P corresponding image points $\{\mathbf{x}_p^i\}$, $p = 1, ..., N_P$, $i = 0, ..., N_F - 1$, with respect to N_F camera frames, compute the motion (R, T) and the depth $\{Z\}$ as follows:*

```
1. Initialize $T = 0$.
2. Solve for $R$ linearly from (1), given $T$ and $\{Z\}$. $\{Z\}$ is unnecessary if $T = 0$.
3. Given $R$ compute $D$. Then compute $S$ and $M$ from the SVD of $HD$.
   (a) Solve for $\{Z\}$ and $U$ linearly from $H\Phi(\{Z^{-1}\}) = SU$, given $S$.
   (b) Solve for $T = U^{-1}M^T$, given $U$ and $M$.
4. Goto 2. until $(R, T, \{Z\})$ converge.
```

3 Planar motion algorithms

In the non-planar motion case, translation and depth parameters are estimated linearly from the equation:

$$H\Phi(\{Z^{-1}\})V = SU \tag{5}$$

where the columns of $V \in \mathbb{R}^{3\times 3}$ are a basis for the translation vectors and one can choose $V = I_3$ without loss of generality.

Assume now that all translation vectors lie in a plane, so that rank$(T) = 2$. The data matrix HD now factors into rank–2 matrices $S \in \mathbb{R}^{(2N_P-3)\times 2}$ and $M \in \mathbb{R}^{(N_F-1)\times 2}$. In (5), we now have $U \in \mathbb{R}^{2\times 2}$ and $V \in \mathbb{R}^{3\times 2}$. The matrix V is defined by its orthogonality to the normal $\pi \in \mathbb{S}^2$ to the true plane of motion, and thus has two degrees of freedom. Since there is no global parameterization for $\mathbb{S}^2$, hence for V, in order to solve (5) we will need to choose a set of local parameterizations for V. Further, notice that since (5) is bilinear in $\{Z^{-1}\}$ and V, we cannot proceed as in step 3(a) of the non-planar motion algorithm.

In the following subsections, we show how to cancel the nonlinearity in (5) due to V and hence obtain a linear solution for $\{Z\}$ and T.

3.1 Multiple b algorithm

Let π be the normal to the plane of motion. We consider vectors $b^i \in \mathbb{R}^3, i = 1, ..., m$ that are not perpendicular to π, *i.e.,* vectors that do not lie in the true plane of motion, and parameterize V depending on the directions specified by these b^i's (see Fig. 1). We start with the simplest case of a single b.

$\boldsymbol{b = [0, 0, 1]^T}$**.** Assume that the normal π to the true plane of motion does not lie in the X–Y plane. Then (5) can be written as:

$$H \begin{bmatrix} -\{Z^{-1}\} & \{0\} & \{xZ^{-1}\} \\ \{0\} & -\{Z^{-1}\} & \{yZ^{-1}\} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ v_1 & v_2 \end{bmatrix} = SU, \tag{6}$$

where $v_1 \in \mathbb{R}$ and $v_2 \in \mathbb{R}$ are unknowns chosen to make the columns of V perpendicular to π. Define the following matrices in $\mathbb{R}^{(2N_P-3)\times N_P}$: $H_x = H(:, 1\!:\!N_P)$,

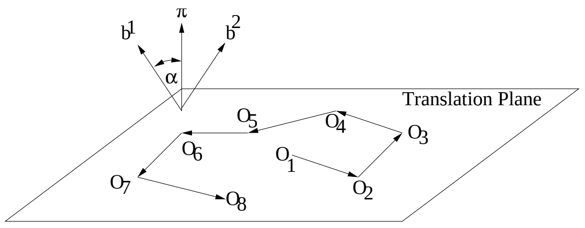

Fig. 1. Multiple b algorithm for planar motions.

$H_y = H(:, N_P+1:2N_P)$ and $H_z = H_x\text{diag}(\{x\}) + H_y\text{diag}(\{y\})$. Then (6) can be rewritten as:

$$\begin{aligned}(-H_x + v_1 H_z)\left\{Z^{-1}\right\} &= SU_1,\\ (-H_y + v_2 H_z)\left\{Z^{-1}\right\} &= SU_2,\end{aligned} \tag{7}$$

where $U = [U_1, U_2] \in \mathbb{R}^{2\times 2}$. Define a $(N_P-3)\times(2N_P-3)$ matrix N_z to annihilate the columns of H_z. N_z can be computed quickly using Householder matrices. We obtain:

$$\begin{bmatrix} N_zH_x & N_zS & 0\\ N_zH_y & 0 & N_zS\end{bmatrix}\begin{bmatrix}\{Z^{-1}\}\\ U_1\\ U_2\end{bmatrix} \equiv G\begin{bmatrix}\{Z^{-1}\}\\ U_1\\ U_2\end{bmatrix} = 0. \tag{8}$$

Multiplying by N_z introduces an additional solution $[\{Z\}; U_1; U_2] = [\{1\}; 0; 0]$ to the system of equations in (6). In order to show that this is a solution, we just need to prove that $N_zH_x\{1\} = N_zH_y\{1\} = 0$. Since $H\Psi = 0$, we have $H_x\{1+x^2\} + H_y\{xy\} = 0$. Therefore $-N_zH_x\{1\} = N_z(H_x\{x^2\} + H_y\{xy\}) = N_zH_z\{x\} = 0$. Similarly, one can show that $-N_zH_y\{1\} = N_zH_z\{y\} = 0$.

In the presence of noise, the solution $[\{1\}; 0; 0] \in \mathbb{R}^{N_P+4}$ will still correspond to the zero singular value of G, while the solution we are looking for will be a linear combination of the two smallest singular vectors of G. That is, the first and second singular vectors of G equal $[\{1\}; 0; 0]$ and $[\zeta; U_1; U_2]$, respectively, where $\zeta \in \mathbb{R}^{N_P}$ represents the first N_P entries of the second singular vector. Therefore, U_1 and U_2 can be obtained directly from the second singular vector of G, while the inverse depths equal ζ up to an unknown additive mixture of the constant term $\{1\}$, *i.e.*, $\{Z^{-1}\} = \lambda\{1\} + \zeta$ for some $\lambda \in \mathbb{R}$.

Replacing the above expression for $\{Z^{-1}\}$ in (7) we obtain:

$$\begin{aligned}-H_x\{1\}\,\lambda + H_z\zeta\,v_1 + H_z\{1\}\,\lambda v_1 &= H_x\zeta + SU_1\\ -H_y\{1\}\,\lambda + H_z\zeta\,v_2 + H_z\{1\}\,\lambda v_2 &= H_y\zeta + SU_2.\end{aligned}$$

The bilinear terms λv_1 and λv_2 can be canceled by multiplying these equations by $\tilde{N}_z \in \mathbb{R}^{(2N_P-4)\times(2N_P-3)}$ such that $\tilde{N}_zH_z\{1\} = 0$. We obtain:

$$\begin{bmatrix} -\tilde{N}_zH_x\{1\} & \tilde{N}_zH_z\zeta & 0\\ -\tilde{N}_zH_y\{1\} & 0 & \tilde{N}_zH_z\zeta\end{bmatrix}\begin{bmatrix}\lambda\\ v_1\\ v_2\end{bmatrix} = \begin{bmatrix}\tilde{N}_z(H_x\zeta + SU_1)\\ \tilde{N}_z(H_y\zeta + SU_2)\end{bmatrix},$$

from which one can solve linearly for λ, v_1 and v_2. Note that $\{Z^{-1}\}$ is now known since ζ and λ are. One can improve these initial estimates for $\{Z^{-1}\}$ and U by solving (7) given v_1 and v_2. The plane normal is given by $\pi = (-v_1; -v_2; 1)$ and the translations are $T = VU^{-1}M^T$, where M is the motion factor of HD.

Arbitrary single *b*. Assume that $b \in \mathbb{R}^3$ is a given vector with a nonzero component along the true plane normal, *i.e.,* $b^T\pi \neq 0$. Let a_1, a_2 and b form an orthonormal basis of $\mathbb{R}^3$. Then a basis for the translation plane is given by $V = [a_1 + v_1 b \quad a_2 + v_2 b]$ for some unknown $v_1 \in \mathbb{R}$ and $v_2 \in \mathbb{R}$. Equation (5) can be written as:

$$\begin{aligned} (H_{a_1} + v_1 H_b)\left\{Z^{-1}\right\} &= SU_1, \\ (H_{a_2} + v_2 H_b)\left\{Z^{-1}\right\} &= SU_2, \end{aligned} \tag{9}$$

where $H_{a_1} = -a_{1x}H_x - a_{1y}H_y + a_{1z}H_z$, $H_{a_2} = -a_{2x}H_x - a_{2y}H_y + a_{2z}H_z$ and $H_b = -b_xH_x - b_yH_y + b_zH_z$.

As before, we define $N_b \in \mathbb{R}^{(N_P-3)\times(2N_P-3)}$ to annihilate the columns of H_b and obtain the system of equations:

$$\begin{bmatrix} -N_bH_{a_1} & N_bS & 0 \\ -N_bH_{a_2} & 0 & N_bS \end{bmatrix} \begin{bmatrix} \{Z^{-1}\} \\ U_1 \\ U_2 \end{bmatrix} \equiv G \begin{bmatrix} \{Z^{-1}\} \\ U_1 \\ U_2 \end{bmatrix} = 0.$$

The two smallest singular vectors of G are now $[\zeta_1; 0; 0]$ and $[\zeta_2; U_1; U_2]$, with $\zeta_1 = b_x\{x\} + b_y\{y\} + b_z\{1\}$. The inverse depths are $\{Z^{-1}\} = \lambda\zeta_1 + \zeta_2$, where λ is obtained by solving:

$$\begin{bmatrix} \tilde{N}_bH_{a_1}\zeta_1 & \tilde{N}_bH_b\zeta_2 & 0 \\ \tilde{N}_bH_{a_2}\zeta_1 & 0 & \tilde{N}_bH_b\zeta_2 \end{bmatrix} \begin{bmatrix} \lambda \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \tilde{N}_b(SU_1 - H_{a1}\zeta_2) \\ \tilde{N}_b(SU_2 - H_{a_2}\zeta_2) \end{bmatrix},$$

with $\tilde{N}_b \in \mathbb{R}^{(2N_P-4)\times(2N_P-3)}$ defined to annihilate $H_b\zeta_1$.

Finally, given v_1 and v_2 one can re-solve for $\{Z^{-1}\}$ and U from (9) to improve the estimates. The plane normal is given by $\pi = (a_1 + v_1 b) \times (a_2 + v_2 b)$ and the translations are obtained as $T = VU^{-1}M^T$.

Multiple b algorithm. We conclude from the previous section that using a single b to parameterize V has the disadvantage of introducing an additional solution to the system of equations in (9). Although this is not a problem from an algebraic point of view since we have shown how to resolve the ambiguity, in the presence of noise it can lead to solutions which are not robust.

For a single b, the additional solution has the form $[b_x\{x\}+b_y\{y\}+b_z\{1\}; 0; 0]$. Therefore, it can be eliminated by choosing more than one b. Let a_1^i, a_2^i, b^i, $i = 1, \ldots, m$ be a set of orthonormal bases for $\mathbb{R}^3$ and assume none of the b^i's lies in the translation plane. One can solve for $\{Z^{-1}\}$, U_1^i and U_2^i uniquely from the set of equations:

$$\begin{bmatrix} -N_{b^1}H_{a_1^1} & N_{b^1}S & 0 & \cdots & 0 & 0 \\ -N_{b^1}H_{a_2^1} & 0 & N_{b^1}S & & 0 & 0 \\ \vdots & \vdots & & \ddots & & \vdots \\ -N_{b^m}H_{a_1^m} & 0 & 0 & & N_{b^m}S & 0 \\ -N_{b^m}H_{a_2^m} & 0 & 0 & \cdots & 0 & N_{b^m}S \end{bmatrix} \begin{bmatrix} \{Z^{-1}\} \\ U_1^1 \\ U_2^1 \\ \vdots \\ U_1^m \\ U_2^m \end{bmatrix} = 0. \qquad (10)$$

Given $\{Z^{-1}\}$ and U^i, one can solve for v_j^i, $i = 1, \dots m$ and $j = 1, 2$ from:

$$H_{b^i}\{Z^{-1}\}v_j^i = SU_j^i - H_{a_j^i}\{Z^{-1}\}. \qquad (11)$$

Given v_j^i one can improve the estimates for $\{Z^{-1}\}$ and U_j^i by re-solving (11). Finally, let $\pi^i = (a_1^i + v_1^i b^i) \times (a_2^i + v_2^i b^i)$ and $\Pi = [\pi^1 \cdots \pi^m] \in \mathbb{R}^{3\times m}$. The plane normal is obtained as the leading left singular vector of Π and the translations are obtained as:

$$T = \frac{1}{m}\left(\sum_{i=1}^{m} V^i {U^i}^{-1}\right) M^T. \qquad (12)$$

Choosing the b's. Since the b^i's cannot be perpendicular to the true plane normal, some estimate of π is needed in order to choose the b^i's. If $\pi = b$, we have $v_1 = v_2 = 0$. Therefore, we can obtain an initial estimate of π as follows:

1. Obtain three estimates of (π, v_1, v_2) from the single b algorithm with b equal to the X, Y and Z axes.
2. Choose the π that minimizes $v_1^2 + v_2^2$.

Given an initial estimate for π, one has to choose b^i's so that they are not perpendicular to π and not close to each other. We choose all the b's to be randomly distributed on a cone with angle α from π. Experimentally, the best performance is obtained with $\alpha \approx 37^o$ and three different b's. Using more b's has the disadvantage of over-parameterizing U, while one can show theoretically that using two b's does not exploit all the image data at some image points.

Algorithm 2 (Multiple b algorithm for planar motions) *Given a set of corresponding image points $\{\mathbf{x}_p^i\}$, $p = 1, ..., N_P$, $i = 0, ..., N_F - 1$, compute the motion (R, T) and the depth $\{Z\}$ as follows:*

1. *Initialization*
   (a) *Solve for R linearly from (1), given $T = 0$.*
   (b) *Given R compute D. Then compute S and M from the SVD of HD.*
   (c) *Compute $(\{Z\}, T, \pi)$ from "best" single b algorithm along X, Y or Z.*
   (d) *Use π to generate multiple b's in a cone with angle α along π.*
2. *Solve for R linearly from (1), given T and $\{Z\}$.*
3. *Given R compute D. Then compute S and M from the SVD of HD.*
   (a) *Solve for $\{Z\}$ and $U^1 \cdots U^m$ from (10), given the current b's.*
   (b) *Solve for π from SVD of $\Pi = [\pi^1 \cdots \pi^m]$ and for T from (12).*
   (c) *Use π to generate multiple b's in a cone with angle α along π.*
4. *Goto 2. until $(R, T, \{Z\})$ converge.*

3.2 The intersection algorithm

Let the columns of $V = [V_1\ ,\ V_2] \in \mathbb{R}^{3\times 2}$ be a basis for the translation plane and let $N_s \in \mathbb{R}^{(2N_P-5)\times(2N_P-3)}$ be a matrix annihilating S. From (5), we obtain:

$$\begin{aligned}(-H_xV_{jx} - H_yV_{jy} + H_zV_{jz})\{Z^{-1}\} &= SU_j\\ N_s(-H_xV_{jx} - H_yV_{jy} + H_zV_{jz})\{Z^{-1}\} &= 0\\ \begin{bmatrix} N_sH_x\{Z^{-1}\} & N_sH_y\{Z^{-1}\} & -N_sH_z\{Z^{-1}\}\end{bmatrix} V_j &= 0,\end{aligned}$$

for $j = 1$ and 2. We conclude that the matrix

$$\mathcal{I} = \begin{bmatrix} N_sH_x\{Z^{-1}\} & N_sH_y\{Z^{-1}\} & -N_sH_z\{Z^{-1}\}\end{bmatrix} \in \mathbb{R}^{(2N_P-5)\times 3}$$

has rank 1. Since $\pi^TV = 0$ we further have

$$\begin{bmatrix} N_sH_x\{Z^{-1}\} & N_sH_y\{Z^{-1}\} & -N_sH_z\{Z^{-1}\}\end{bmatrix} = B\pi^T \qquad (13)$$

for some $B \in \mathbb{R}^{2N_P-5}$ that belongs to the *intersection* of the subspaces generated by the columns of N_sH_x, N_sH_y and N_sH_z.

After eliminating B from (13) we obtain:

$$\begin{bmatrix} N_sH_x & -N_sH_y & 0\\ N_sH_x & 0 & N_sH_z\end{bmatrix}\begin{bmatrix}\{Z^{-1}\}/\pi_x\\ \{Z^{-1}\}/\pi_y\\ \{Z^{-1}\}/\pi_z\end{bmatrix} = 0. \qquad (14)$$

Rather than solving (14) directly, we first eliminate N_s from the equations. One can show that this reduces the bias in solving (14), since, in the presence of noise, the matrix S is white. We define $Y_w = \{Z^{-1}\}/\pi_w$, for $w = x, y, z$. Then, since $N_s^TN_s$ is a projection matrix, equation (14) is equivalent to:

$$\begin{bmatrix} H_x & -H_y & 0 & S & 0\\ H_x & 0 & H_z & 0 & S\end{bmatrix}\begin{bmatrix}Y_x\\ Y_y\\ Y_z\\ U_1\\ U_2\end{bmatrix} = 0. \qquad (15)$$

If we disregard the fact that Y_x, Y_y, Y_z are dependent, we can solve equation (15) linearly. When $\pi_x \neq 0$, $\pi_y \neq 0$ and $\pi_z \neq 0$, unique solutions for Y_x, Y_y, Y_z are obtained, from which $\{Z^{-1}\}$ can be uniquely recovered as the leading left singular vector of $[Y_x, Y_y, Y_z] \in \mathbb{R}^{N_P\times 3}$ using SVD. When at least one of the π_w equals zero, there are two possible solutions for the linear system (15). For example, if $\pi_x = \pi_y = 0$, the two solutions are $[Y_x; Y_y; Y_z] = [\{Z^{-1}\}; \{Z^{-1}\}; 0]$ or $[\{Z^{-1}\}; -\{Z^{-1}\}; 0]$. One can still recover $\{Z^{-1}\}$ from these solutions.

We improve the above estimate for $\{Z^{-1}\}$ by first recovering B from the SVD of $\mathcal{I}$, using the previously recovered $\{Z^{-1}\}$, and then solving linearly for $\{Z^{-1}\}$ and π from (13), given B. Given $\{Z^{-1}\}$ and π one can improve the initial estimate of U by solving (5), with $V \in \mathbb{R}^{3\times 2}$ obtained from the equation $\pi^TV = 0$. Finally, the translations are given by $T = VU^{-1}M^T$ as usual.

Algorithm 3 (Intersection algorithm for planar motions) *Given a set of corresponding image points* $\{\mathbf{x}_p^i\}$, $p = 1, ..., N_P$, $i = 0, ..., N_F - 1$, *compute the motion* (R, T) *and the depth* $\{Z\}$ *as follows:*

```
1. Initialize $T = 0$
2. Solve for $R$ linearly from (1), given $T$ and $\{Z\}$. $\{Z\}$ is unnecessary if $T = 0$.
3. Given $R$ compute $D$. Then compute $S$ and $M$ from the SVD of $HD$.
   (a) Solve for $Y_x$, $Y_y$, $Y_z$ and $U$ from (15).
   (b) Solve for $\{Z\}$ as the leading left singular vector of $[Y_x \, Y_y \, Y_z]$ using SVD.
   (c) Solve for $B$ from $\mathcal{I}$ and then solve for $\pi$ and $\{Z\}$ from (13).
   (d) Solve for $U$ and $V$ from (5), and let $T = VU^{-1}M^T$.
4. Goto 2. until $(R, T, \{Z\})$ converge.
```

3.3 Hybrid algorithm

Both the multiple b and the intersection algorithms solve for depth, translation and plane normal in two main stages:

(a) First depth is obtained from either (10) or (15).
(b) Then the plane normal π is obtained from the SVD of Π or from (13).

In the first stage, the multiple b algorithm solves for a unique $\{Z\}$ and multiple copies of U, while the intersection algorithm solves for multiple copies of $\{Z\}$ and a unique U. In the presence of noise, this causes the multiple b algorithm to give a very accurate estimate for $\{Z\}$ and a less accurate estimate for U, and vice-versa for the intersection algorithm.

In the second stage, the multiple b algorithm solves for the plane normal and translations based on the estimates of U and $\{Z\}$, while the intersection algorithm solves for the plane normal from the estimates of $\{Z\}$ only.

This suggests to combine the best part of both algorithms in a hybrid manner as follows: first solve for $\{Z\}$ using the multiple b algorithm and then solve for π and T using the intersection algorithm.

3.4 General properties of the planar algorithms

Convergence. Since the only modification of the non-planar algorithm is to replace step 3(a) by a method that remains linear, all the planar algorithms that we propose inherit the convergence guarantees of the non-planar algorithm.

Detection of planar motion. We detect when the motion is roughly planar, and thus when to apply our new algorithms, from the singular values s_1, s_2 and s_3 of HD: if $s_3/s_2 \geq \epsilon$, where ϵ is a pre–fixed threshold, we use the non-planar motion algorithm; if $s_1/s_2 < \epsilon$ we use the algorithm for linear motion in [7].

Non-planar motion. Note that our planar algorithm works even on fully non–planar motions: in the non-planar case, restricting to two of the singular vectors of HD does not corrupt the reconstruction but simply omits the additional information from the third singular vector [9].

4 Experimental Results

In this section, we evaluate the proposed algorithms on synthetic and real images. We compare our results with those of the following two algorithms:

1. *Linear motion algorithm*: the algorithm of [7] first estimates the rotations linearly assuming that the translations are zero. It computes the SVD of a matrix that depends on the image displacements and the rotation estimates. For each of the first three singular values of this matrix that is above a pre-fixed noise threshold, it recovers a translation direction from the corresponding singular vector, using a linear algorithm similar to that of [3] for optical flow. It refines the estimates of the translation directions by minimizing appropriate error functions. It linearly computes the depths from the recovered translation directions and the leading singular vectors, and it finally computes the translation magnitudes. This algorithm has been shown in [9] to give better results than the Sturm/Triggs algorithm [11] for arbitrary small motions. In our experiments, we always apply this algorithm under the assumption of planar motion, *i.e.,* with the fixed threshold set so that just the two largest singular values are used.
2. *Optimal re-projection error*: motion parameters (R, T) and 3D structure X are obtained by minimizing the function:

$$F(R,T,X) = \sum_{p=1}^{N_P} \sum_{i=0}^{N_F-1} \left\| \mathbf{x}_p^i - \frac{R^i(X_p - T^i)}{(R^i(X_p - T^i))_z} \right\|^2$$

starting from the ground truth. The depth of each point is $Z_p = (X_p)_z$.

In our comparison, we use the following error measures for the different parameters, averaged over the number of trials (and frames if appropriate):

$$\begin{aligned}
\text{Rotation error} &= \text{acos}\left((\text{trace}(R_{true}R_{est}^T) - 1)/2\right) \\
\text{Translation error} &= \text{acos}(T_{true}^T T_{est})/(\|T_{true}\|\|T_{est}\|) \\
\text{Depth error} &= \text{acos}(Z_{true}^T Z_{est})/(\|Z_{true}\|\|Z_{est}\|) \\
\text{Normal error} &= \text{acos}(\pi_{true}^T \pi_{est})/(\|\pi_{true}\|\|\pi_{est}\|).
\end{aligned}$$

4.1 Experiments on Synthetic Images

In our simulations, we varied the motion, plane normal, structure and image noise randomly at each trial. We generated the structure by randomly picking each coordinate of the 3D point according to a uniform distribution taken from a truncated pyramid specified by the depth variation and the field of view, as shown in Fig. 2. Simulation parameters are shown in Table 1[2].

[2] In the table u.f.l. stands for units of focal length.

Table 1. Simulation parameters.

Parameter	Unit	Value
Number of trials		1000
Number of points		20
Number of frames		8
Field of view	degrees	90
Depth variation	u.f.l.	100 - 400
Image size	pixels	500 $\times$ 500
$\tau = T_{\max}/Z_{\min}$		0.1-0.6

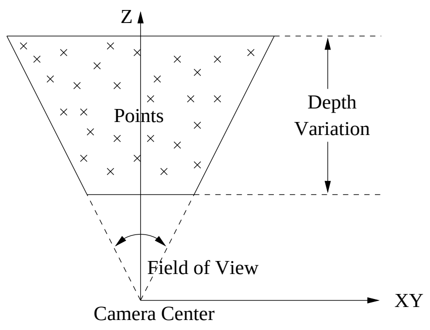

Fig. 2. Truncated pyramid used to generate the structure.

Error vs. noise. Figures 3 and 4 compare the performance of the different algorithms for different levels of noise[3]. For small τ, we observe that the new algorithms significantly outperform the *linear motion algorithm*. The best algorithm for rotation is the *multiple b algorithm*, and the best one for translation, depth and normal is the *hybrid algorithm*. The new algorithms give very accurate estimates for rotation, translation and normal to the plane of motion (almost indistinguishable from the optimal). Depth estimates are suboptimal due to the bas–relief ambiguity[4]. As τ increases, so does the error in the noise free case, which decreases the accuracy of our algorithms with respect to the optimal. This is expected, since the approximation $T_{z,\max}/Z_{\min} \approx 0$ is no longer valid. Notice that the slope of the error decreases with τ.

Table 2 includes the number of outliers for each algorithm. We define a trial to be an outlier if the error for that trial in any of the motion or structure parameters exceeds the mean error by 8 times the standard deviation. The *intersection algorithm* has between 1% and 3% outliers, while the *multiple b algorithm* has less than 0.4%. As explained in Section 3.3, this is because the initial estimation of depth from (15) is not as robust as that of the *multiple b algorithm* from (10). The *hybrid algorithm* has less than 0.3% outliers.

Overall, the *hybrid algorithm* is the one with the best performance: it is more accurate and has fewer outliers.

[3] Mean errors do not include outliers.

[4] [8] describes methods for repairing a similar problem in depth recovery observed for the non–planar motion algorithm, which could also be applied here.

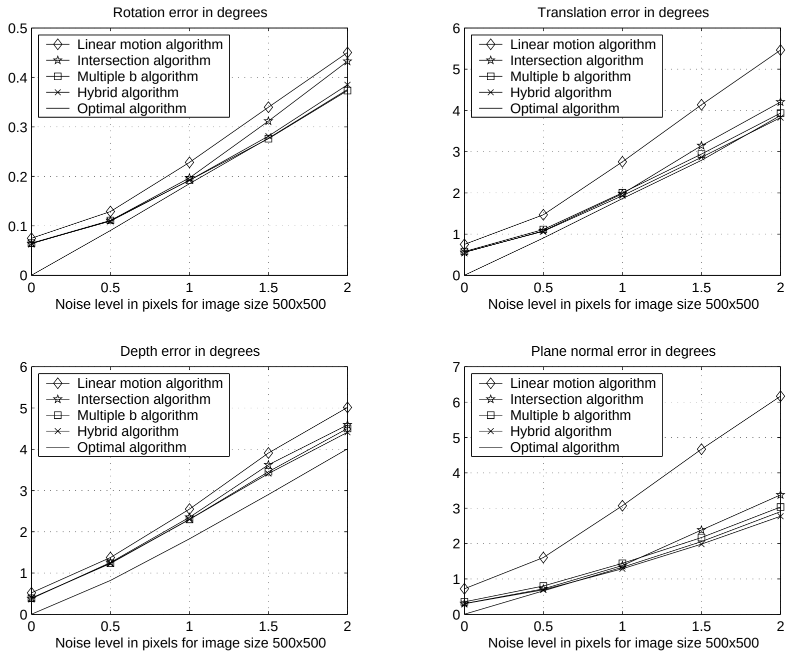

Fig. 3. Error vs. noise for $\tau = T_{\max}/Z_{\min} \in (0.1, 0.2)$.

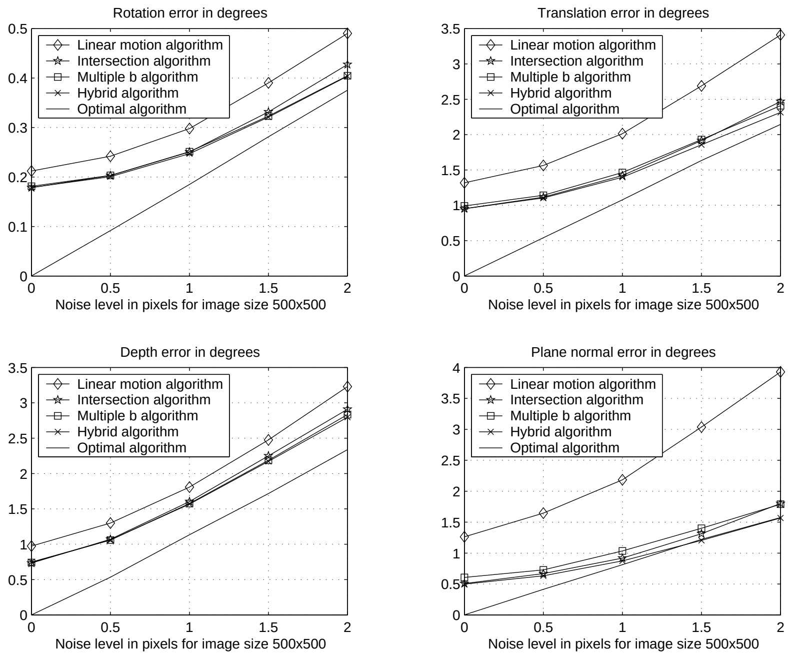

Fig. 4. Error vs. noise for $\tau = T_{\max}/Z_{\min} \in (0.2, 0.3)$.

Table 2. Number of outliers for each algorithm as a function of noise.

Algorithm	τ	Number of outliers				
		Noise level in pixels				
		0.0	0.5	1.0	1.5	2.0
Linear Motion	0.1–0.2	0	1	2	2	9
Intersection	0.1–0.2	0	11	18	20	28
Multiple b	0.1–0.2	0	0	0	1	4
Hybrid	0.1–0.2	0	0	0	1	3
Linear Motion	0.2–0.3	0	1	0	1	4
Intersection	0.2–0.3	7	6	5	19	18
Multiple b	0.2–0.3	0	0	0	0	1
Hybrid	0.2–0.3	0	0	0	1	1
Linear Motion	0.3–0.4	0	1	0	0	3
Intersection	0.3–0.4	7	4	9	17	19
Multiple b	0.3–0.4	0	0	1	0	1
Hybrid	0.3–0.4	0	1	0	1	1

Error vs. τ. Figure 5 compares the performance of the proposed algorithms for different values of τ, for a noise level of 1 pixel. We observe that the proposed algorithms have very good performance in the range $0.1 \leq \tau \leq 0.6$, with the best performance for $\tau \approx 0.3$. When $\tau < 0.1$, the signal–to–noise ratio is too small, causing an increase of both the error and the number of outliers[5]. When $\tau > 0.6$ the small baseline assumption is violated, and hence the mean error increases. Notice that we still get good results for $\tau = 0.6$, which corresponds to a relatively large translational motion.

Efficiency. Table 3 shows the average execution time on a Pentium III 800 MHz for a MATLAB implementation of each algorithm. The average is taken over 1000 trials, 8 frames and 20 points. We can observe that the fastest of the new algorithms is the *hybrid algorithm*, which is approximately 130 times faster than the optimal algorithm minimizing the re-projection error. Since the optimal algorithm is initialized from the ground truth, the ratio could be higher in practice.

Table 3. Execution time of each algorithm for $0.1 \leq \tau \leq 0.2$.

	Linear Motion	Intersection	Multiple b	Hybrid	Optimal
Time (sec)	0.11	0.21	0.44	0.20	25.75

[5] A generalization of the discussion in [8, 9] implies that, when the signal–to-noise ratio is small and the motion is partly forward, one can get $s_1 \gg s_2 \sim s_3$, *i.e.,* the translational motion can be effectively linear rather than planar. In this situation, one may get better results by applying our algorithms under the assumption of linear rather than planar motion, as [9] verified experimentally for an analogous case.

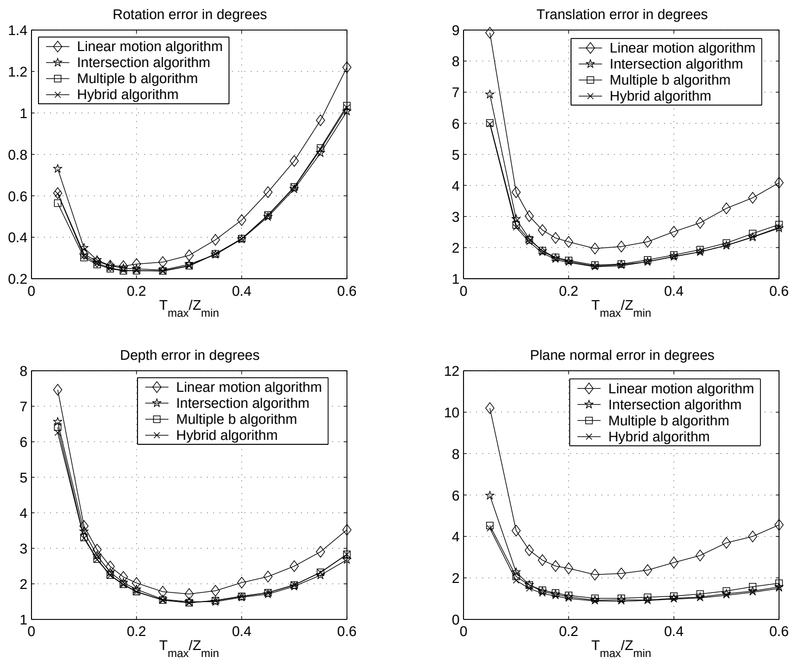

Fig. 5. Error vs. $\tau = T_{\max}/Z_{\min}$ for a noise level of 1 pixel.

4.2 Experiments on Real Images

We tested our algorithms on the *castle* [1] and *puma* [4] sequences. For the castle sequence we used the first 7 frames only so that the motion is approximately planar. The singular values of the matrix HD for the castle sequence are $s_1 = 125.0480$, $s_2 = 39.3361$ and $s_3 = 0.0301$ and the largest baseline is $\tau = 0.0443$. For the puma sequence we considered the first 16 frames, for which the motion is approximately linear, and added some frames so that the motion is approximately planar. The resulting singular values are $s_1 = 5.5359$, $s_2 = 3.7313$ and $s_3 = 0.0714$, and the largest baseline is $\tau = 0.1090$. The first frames of the sequences are shown in Figure 6.

Figure 7 compares the performance of the proposed algorithms for the castle sequence. Again, we observe that the new algorithms outperform the *linear motion algorithm*. Further, motion and structure parameters are estimated very accurately (within 0.5 degrees). The algorithm with best performance is the *hybrid algorithm*, except for translation for which the best algorithm is the *multiple b algorithm*. For the puma sequence, the *multiple b algorithm* is the best for rotation and depth, and the *intersection algorithm* is the best for translation and normal. This result is not surprising, since theoretically the best performance of the intersection algorithm is for $\pi \approx [0; 0; 1]$. Notice that all the algorithms worked well on this sequence though it is slightly non-planar.

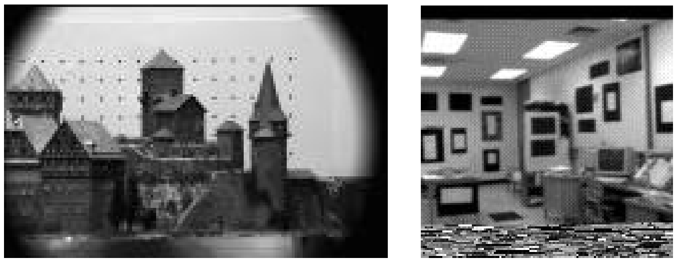

Fig. 6. First frames of the castle and puma sequences.

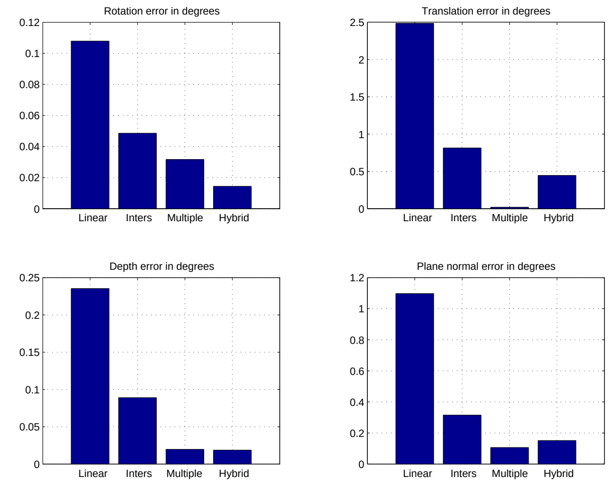

Fig. 7. Mean error of each algorithm for the castle sequence.

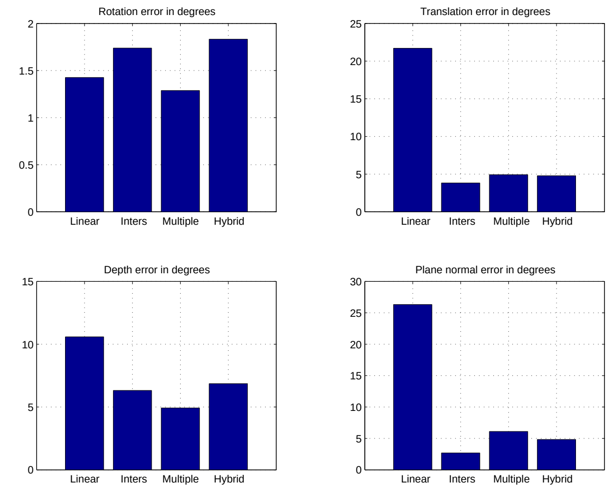

Fig. 8. Mean error of each algorithm for the puma sequence.

5 Conclusions

We presented a set of linear algorithms for motion and structure estimation when the camera translates on a plane with small baselines and arbitrary rotations. Our algorithms are based on an approximate rank-2 factorization of a matrix which depends on the image displacements and current rotation estimates. Translation and depth are obtained from the factorization after an appropriate parameterization of the plane of translation.

We tested our algorithms on both synthetic and real sequences. Experimental results show that the proposed algorithms are able to compute the structure and motion parameters accurately and efficiently for baselines in the range $0.1 \leq \tau \leq 0.6$. The proposed algorithms have good convergence properties and the best algorithm presents less than 0.3% outliers.

References

1. CMU CIL-0001 castle image sequence. http://www.cs.cmu.edu/˜cil/cil-ster.html.
2. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2000.
3. D. Heeger and A. Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.
4. R. Kumar and A. Hanson. Sensitivity of the pose refinement problem to accurate estimation of camera parameters. In *IEEE International Conference on Computer Vision*, pages 365–369, 1990.
5. P. McLauchlan and D. Murray. A unifying framework for structure and motion recovery from image sequences. In *International Conference on Computer Vision and Pattern Recognition*, pages 314–20, 1995.
6. J. Oliensis. Rigorous bounds for two–frame structure from motion. In *European Conference on Computer Vision*, pages 184–195, 1996.
7. J. Oliensis. Structure from linear or planar motions. In *International Conference on Computer Vision and Pattern Recognition*, pages 335–42, 1996.
8. J. Oliensis. A multi-frame structure-from-motion algorithm under perspective projection. *International Journal of Computer Vision*, 34(2-3):163–192, 1999.
9. J. Oliensis and Y. Genc. Fast and accurate algorithms for projective multi–image structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):546–559, 2001.
10. S. Soatto, R. Frezza, and P. Perona. Motion estimation via dynamic vision. *IEEE Transactions on Automatic Control*, 41(3):393–413, March 1996.
11. P. Sturm and B. Triggs. A factorization based algorithm for multi–image projective structure and motion. In *European Conference on Computer Vision*, pages 709–720, 1996.
12. R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using non-linear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.
13. C. Tomasi and T. Kanade. Shape and motion from image streams under orthography. *International Journal of Computer Vision*, 9(2):137–154, 1992.
14. B. Triggs. Factorization methods for projective structure and motion. In *International Conference on Computer Vision and Pattern Recognition*, pages 845–51, 1996.