HAL
open science

Camera Pose Revisited – New Linear Algorithms

Marc-André Ameller, Bill Triggs, Long Quan

To cite this version:

Marc-André Ameller, Bill Triggs, Long Quan. Camera Pose Revisited – New Linear Algorithms. 2000. inria-00548306

HAL Id: inria-00548306

https://inria.hal.science/inria-00548306v1

Submitted on 20 Dec 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Camera Pose Revisited — New Linear Algorithms

Marc-André Ameller, Bill Triggs, and Long Quan

GRAVIR-INRIA-CNRS
655 avenue de l'Europe
38330 Montbonnot, France
Email: {*Marc-Andre.Ameller,Bill.Triggs,Long.Quan*}*@inrialpes.fr*

Abstract. Camera pose estimation is the problem of determining the position and orientation of an internally calibrated camera from known 3D reference points and their images. We briefly survey several existing methods for pose estimation, then introduce four new linear algorithms. The first three give a unique linear solution from four points by SVD null space estimation. They are based on resultant matrices: the 24×24 method is the raw resultant matrix, and the 12×12 and 9×9 methods are compressed versions of this obtained by Gaussian elimination with pivoting on constant entries. The final method returns the four intrinsic solutions to the pose from 3 points problem. It is based on eigendecomposition of a 5×5 matrix. One advantage of all these methods is that they are simple to implement. In particular, the matrix entries are simple functions of the input data. Numerical experiments are given comparing the performance of the new algorithms with several existing algebraic and linear methods.
Keywords: Calibration, Camera Pose Estimation / Space Resection, Polynomial Solving, Resultant Matrices.

1 Introduction

Camera pose estimation is the problem of determining the position and orientation of an internally calibrated camera from known 3D reference points and their images. It is also called *space resection* in the photogrammetry community. It is one of the oldest and commonest tasks in computer vision and photogrammetry, and has often been studied in the past. With 3 points, the problem generically has four possible solutions. Many closed form methods are known for finding these. The earliest was perhaps by Lagrange in 1795. See [17, 6–8, 15]. Fischler & Bolles [6] gave a method that has become popular in computer vision when they introduced their RANSAC paradigm for detecting outliers in the data. Haralick *et al* [8] review many old and new variants of the basic 3-point method

Submitted to ECCV'00. This work was supported in part by Esprit LTR project 21914 Cumuli.

and carefully examine their numerical stabilities under different orders of substitution and elimination. For handling redundant data, iterative methods have also been developed in [12, 18, 3]. Methods for camera pose from line segments instead of point features have also been developed [9, 4, 2, 13, 11].

3 point methods intrinsically give multiple solutions. If a unique solution is required, additional information must be given. A fourth point generically suffices, but even with an infinite number of points there are certain degenerate cases for which no unique solution is possible. These *critical configurations* are known precisely. See [16, 17] for details, but briefly, all 3D points including the camera centre must lie on a special twisted cubic space curve (the *horopter*) that wraps around a circular cylinder (the *dangerous cylinder*). Notable degenerate cases of this geometry include: (*i*) all object points at infinity (camera translation not estimable); (*ii*) the projection center is coplanar with any three of the four object points; (*iii*) a 3D line and a circle in an orthogonal plane touching the line. This last case is particularly troublesome for pose from any three points, or from a square or rectangle of four coplanar points, when the camera is in the region directly above the points. We will show its effect on the pose methods tested below.

Motivated by a lack of methods that directly provide the unique pose solution in the redundant data case, a family of linear algorithms was presented in [14]. Unfortunately these methods are similar to algebraic ones in the sense that the matrix entries are complicated coefficients extracted from fourth degree polynomials. This makes implementation somewhat cumbersome. In this paper, we propose: (*i*) three new linear algorithms for pose from 4 points, based on finding the null vectors of 9×9, 12×12 and 24×24 linear systems with simple matrix entries; (*ii*) a new linear algorithm for pose from 3 points, based on the eigenspace of a 5×5 matrix obtained using the Bezout-Cayley-Dixon resultant method.

The paper is organized as follows. In §2, the basic geometry of camera pose is reviewed and discussed. In §3 we present the linear algebra based resultant matrix approach to solving redundant polynomial equations, and also the three new 4 point algorithms. The 3 point eigenspace method is presented in §4. §5 gives some initial experimental results evaluating the new pose methods against some old ones on simulated data. Finally, §6 summarizes the contributions and gives some conclusions.

2 Geometry of camera pose from points

Given a calibrated camera centered at $\mathbf{c}$ and correspondences between some 3D reference points $\mathbf{p}_i$ and their images $\mathbf{u}_i$, each pair of correspondences i and j gives a constraint on the unknown camera-point distances $x_i = ||\mathbf{p}_i - \mathbf{c}||$ (cf.

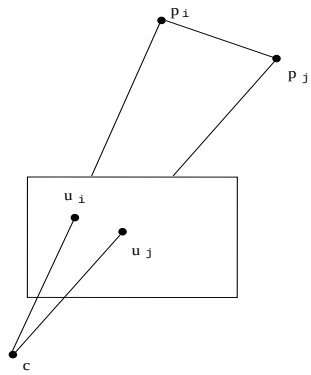

Fig. 1. The basic geometry of camera pose determination for each pair of correspondences $\mathbf{p}_i \leftrightarrow \mathbf{u}_i$ and $\mathbf{p}_j \leftrightarrow \mathbf{u}_j$ between the 3D reference points and their images.

Figure 1):

$$P_{ij}(x_i, x_j) \;\equiv\; {x_i}^2 + {x_j}^2 + c_{ij}\, x_i\, x_j - {d_{ij}}^2 \;=\; 0 \tag{1}$$

$$c_{ij} \;\equiv\; -2\,\cos\theta_{ij} \tag{2}$$

where $d_{ij} = ||\mathbf{p}_i - \mathbf{p}_j||$ is the known inter-point distance between the i-th and j-th reference points and θ_{ij} is the 3D viewing angle subtended at the camera center by the i-th and j-th points. The cosine of this viewing angle is directly computed from the image points and the calibration matrix $\mathbf{K}$ of the internal parameters of the camera as

$$\cos\theta_{ij} \;=\; \frac{\mathbf{u}_i^T\,\mathbf{C}\,\mathbf{u}_i}{\sqrt{(\mathbf{u}_i^T\,\mathbf{C}\,\mathbf{u}_i)(\mathbf{u}_j\mathbf{C}\,\mathbf{u}_j)}},$$

where $\mathbf{C} = (\mathbf{K}\,\mathbf{K}^T)^{-1}$.

For $n = 3$, we obtain the following polynomial system

$$\begin{cases} P_{12}(x_1, x_2) = 0, \\ P_{13}(x_1, x_3) = 0, \\ P_{23}(x_2, x_3) = 0 \end{cases}$$

for the three unknown distances x_1, x_2, x_3. This system has a Bezout bound of $8 = 2 \times 2 \times 2$ solutions. However since it has no odd-order terms, $x_i \mapsto -x_i$ preserves its form and the solutions occur in four $x_i \leftrightarrow -x_i$ pairs. Using the classical Sylvester resultant twice, x_2 and x_3 can be eliminated to obtain an 8th degree polynomial in x_1 with only even terms, i.e. a 4th degree polynomial in $x = x_1^2$:

$$g(x) = a_4\, x^4 + a_3\, x^3 + a_2\, x^2 + a_1\, x + a_0 = 0.$$

This has at most four solutions for x and can be solved in closed form. As x_i is positive, $x_1 = \sqrt{x}$. Then x_2 and x_3 are uniquely determined from x_1.

In this method and all of those below, the recovered camera-point distances x_i are used to estimate the coordinates of the 3D reference points in a camera-centered 3D frame: $\tilde{\mathbf{p}}_i = x_i \, \mathbf{K}^{-1}\mathbf{u}_i$. To find the camera pose, the rigid 3D motion that best aligns these points with their known world-frame coordinates is then estimated. The centres of gravity of the two point clouds are aligned by translation, then the aligning rotation(s) are found by the quaternion or SVD methods [10, 5].

3 Linear algorithms for pose from 4 points

3.1 Linear methods for polynomial system solving

Linear algebra is a useful tool for manipulating polynomials. A polynomial $P_i(x) = \sum_\alpha c_{\alpha,i}\, x^\alpha$ in variables $x = (x_1, \ldots, x_n)$ is a finite sum of *coefficients* $c_{\alpha,i}$ times *monomials* $x^\alpha = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \ldots \cdot x_n^{\alpha_n}$. Here $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n$ is called a multi-index or exponent vector. The basic idea is to fix a sufficiently large set of working monomials, and to regard polynomials as row vectors of coefficients times this column vector of monomials. Similarly, a list of polynomials is regarded as a coefficient matrix containing the row vectors of the polynomials, times the column vector of monomials. The internal structure of the vector of monomials is only enforced indirectly — a kind of over-parametrization of the problem, but one that has the advantage of being linear.

Consider a general polynomial system :

$$\begin{cases} P_1(x_1, \ldots, x_n) = 0 \\ \qquad\qquad\qquad \vdots \\ P_q(x_1, \ldots, x_n) = 0, \end{cases}$$

We can form its matrix — called a *resultant matrix* — which has the interesting property that the monomial vector corresponding to any root of the system lies in its null space. But usually there will be many more monomials than input polynomials so this will not help us much — the matrix will have quite low column rank and hence a large null space. We can do better by including additional polynomials selected from the algebraic *ideal* generated by the inputs, *i.e.* the set of all sums of polynomial multiples of the input polynomials. Although this forces the number of columns (monomials) to increase, the number of additional ideal polynomials that can be found within that set of monomials increases faster, so the null space tends to reduce in size as we add further monomials. If the original system has only a single solution, with luck we will eventually reduce the null space of the matrix to just one dimension giving the monomials of that root, from which the root itself is easily extracted. The required ideal elements can be guessed by hand (as here), or generated more systematically using several classical and modern resultant-building methods [1]. Although they give generically sufficient monomial sets, these constructions are often far from minimal, and also may not suffice for certain special values of the input coefficients.

3.2 Linear algorithms for pose estimation from 4 points

Now we apply the resultant principal to the problem of camera pose. For 4 points, the input system is 6 equations (1) in 4 variables: $\{P_{ij}(x_i, x_j) = 0 \mid 1 \leq i < j \leq 4\}$. Any solution of this system trivially also satisfies:

$$\{\, x_i \, P_{jk}(x_j, x_k) = 0 \,|\, i, j < k = 1, 2, 3, 4\}$$

This new system contains 24 polynomials in 24 monomials: 4 x_i^3, 12 $x_i^2 \, x_j \, x_k$ and 4 $x_i \, x_j \, x_k$ (where i, j, k are distinct), and finally 4 x_i. It turns out that (if the inputs c_{ij} and d_{ij} are correct) the corresponding 24×24 resultant matrix generically has a 1D null space, which contains the two possible algebraic solutions x and $-x$, and also a spurious solution $x = 0$ that arose when we multiplied the P_{jk} by x_i. But it is easy to pick the desired solution, as the depths x_i must be positive. In detail, the 24×24 matrix and the list of monomials labeling its columns are:

$$\left(\begin{array}{cccccccccccc|cccc|cccc}
1&0&0&0&c_{12}&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&c_{12}&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&c_{12}&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&c_{12}&0&0&0&0&0\\
\hline
1&0&0&0&0&c_{13}&0&0&0&0&0&1&0&0&0&0&0&0&0&0&d_{13}^2&0&0&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&c_{13}&0&0&0&0&d_{13}^2&0&0\\
0&0&1&0&0&1&0&0&0&0&0&c_{13}&0&0&0&0&0&0&0&0&0&0&d_{13}^2&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&c_{13}&0&0&0&0&d_{13}^2\\
\hline
1&0&0&0&0&0&c_{14}&0&0&0&0&0&0&0&0&1&0&0&0&0&d_{14}^2&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&c_{14}&0&0&0&d_{14}^2&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&c_{14}&0&0&d_{14}^2&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&c_{14}&0&0&0&0&0&0&d_{14}^2\\
\hline
0&0&0&0&0&0&0&1&0&0&0&0&c_{23}&0&0&0&0&0&0&0&d_{23}^2&0&0&0\\
0&1&0&0&0&0&0&0&c_{23}&0&1&0&0&0&0&0&0&0&0&0&0&d_{23}^2&0&0\\
0&0&1&0&0&0&0&1&0&c_{23}&0&0&0&0&0&0&0&0&0&0&0&0&d_{23}^2&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&c_{23}&0&0&0&d_{23}^2\\
\hline
0&0&0&0&0&0&0&1&0&0&0&0&0&c_{24}&0&0&0&0&0&0&d_{24}^2&0&0&0\\
0&1&0&0&0&0&0&0&0&c_{24}&0&0&0&0&0&0&0&0&0&0&0&d_{24}^2&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&c_{24}&0&0&d_{24}^2&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&c_{24}&0&0&0&0&0&d_{24}^2\\
\hline
0&0&0&0&0&0&0&1&0&0&0&0&0&c_{34}&0&0&0&0&0&0&d_{34}^2&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&c_{34}&0&d_{34}^2&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&c_{34}&0&0&d_{34}^2&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&d_{34}^2
\end{array}\right)
\left(\begin{array}{c}
x_1^3\\ x_2^3\\ x_3^3\\ x_4^3\\ \hline
x_1^2\,x_2\\ x_1^2\,x_3\\ x_1^2\,x_4\\ x_2^2\,x_1\\ x_2^2\,x_3\\ x_2^2\,x_4\\ x_3^2\,x_2\\ x_3^2\,x_1\\ x_3^2\,x_4\\ x_4^2\,x_2\\ x_4^2\,x_3\\ x_4^2\,x_1\\ \hline
x_1\,x_2\,x_3\\ x_1\,x_2\,x_4\\ x_1\,x_3\,x_4\\ x_2\,x_3\,x_4\\ \hline
-x_1\\ -x_2\\ -x_3\\ -x_4
\end{array}\right) = 0$$

The matrix entries are very simple functions of the input data. Once the null space monomial vector $\mathbf{v}$ is found by SVD of the input matrix, the solution for the depths x can be calculated as follows. We could simply take some fixed ratios of elements, such as

$$(x_1, x_2, x_3, x_4) \;=\; \left(\sqrt{\frac{\mathbf{v}_1}{\mathbf{v}_{21}}},\; \sqrt{\frac{\mathbf{v}_2}{\mathbf{v}_{22}}},\; \sqrt{\frac{\mathbf{v}_3}{\mathbf{v}_{23}}},\; \sqrt{\frac{\mathbf{v}_4}{\mathbf{v}_{24}}}\right).$$

A somewhat more accurate method is to divide the components into 6 blocks, each containing (*something*) $\cdot\ (x_1, x_2, x_3, x_4)$ (the $x_i\, x_j\, x_k$ block gives instead $x_1\, x_2\, x_3\, x_4 \cdot (\frac{1}{x_1}, \frac{1}{x_2}, \frac{1}{x_3}, \frac{1}{x_4})$). Choose the block with the largest entries and read off the ratios $x_1 : x_2 : x_3 : x_4$ from this. Then estimate the overall scale by taking $x_i = \sqrt{\frac{x_i^2\, x_j}{x_j}}$ where $x_i^2\, x_j$ and x_j are components of $\mathbf{v}$ chosen to be as large as possible, and j may equal i. The aim is to select vector components of largest possible size for all calculations, as these are less affected by noise.

Generically, if the coefficients $c_{i,j}$ and $d_{i,j}$ fail to correspond to a coherent 3D geometry, the matrix will have full rank. So the determinant of the 24×24 matrix is a (nontrivial) multiple of the resultant polynomial of the system.

3.3 The 12×12 and 9×9 linear methods

The 24×24 matrix can be reduced to a smaller matrix by partial symbolic Gaussian elimination. Any of the resulting reduced submatrices (*i.e.* lower right hand corners, with zeros to the left of them in the elimination matrix) can be used as resultant matrices, simply by ignoring the monomials that do not appear. The fact that the input matrix is very sparse and has a large number of constant entries that can be used as known-stable pivots, allows symbolic elimination to go quite far before the coefficients get too complicated for convenient implementation and/or numerical conditioning is lost. This is mainly a compromise between the complexity of the entries and the size of the resulting matrix, but the elimination should probably not go so far that it becomes difficult to extract the solution from the remaining monomials. Also, reduction changes the effective error metric, so it may affect the precision of the results (either upwards of downwards).

In this case, we experimented with 12×12 and 9×9 reductions. At each step of Gaussian reduction we chose a constant pivot so that polynomial division was avoided, and permuted the rows accordingly. The 12×12 version eliminates all terms with x_1, x_2 or x_3 squared or cubed. Eliminating also the x_4^2 and x_4^3 terms would give an 8×8 matrix in $(x_1\, x_2\, x_3, \ldots, x_2\, x_3\, x_4,\ x_1, \ldots, x_4)$. But we judged the coefficients to be inconveniently complex, and eliminating the x_4^3 term makes scale recovery for the solution more difficult, so we preferred to leave x_4^3 in and have a 9×9 resultant matrix. The 12×12 and 9×9 matrices are a little too big to display here, but are available from the authors.

4 Eigenvector algorithm for 3 points

4.1 The Bezout-Cayley-Dixon Method

Another linear algebra method for solving redundant polynomial systems is the Bezout-Cayley-Dixon method [1]. Consider a general polynomial system:

$$\begin{cases} P_1(x_1, \dots, x_n) &= 0 \\ & \vdots \\ P_{n+1}(x_1, \dots, x_n) &= 0. \end{cases}$$

Introduce new variables $y_1, \dots, y_n$ and construct the matrix:

$$\mathbf{M} = \begin{pmatrix} P_1(x_1, x_2, \dots, x_n) & P_1(y_1, x_2, \dots, x_n) & \dots & P_1(y_1, y_2, \dots, y_n) \\ \vdots & \vdots & & \vdots \\ P_{n+1}(x_1, x_2, \dots, x_n) & P_{n+1}(y_1, x_2, \dots, x_n) & \dots & P_{n+1}(y_1, y_2, \dots, y_n) \end{pmatrix}.$$

Each column converts one more x to a y (the results in general depend on the variable ordering chosen for this). If $(x_1, \dots, x_n)$ is a solution of the system, the first column vanishes and hence $\det(\mathbf{M}) = 0$. The determinant is in fact divisible by $(x_1 - y_1) \dots (x_n - y_n)$ because $P_i(\dots, x_i, \dots) - P_i(\dots, y_i, \dots)$ vanishes at $y_i = x_i$ and hence contains a multiple of $(x_i - y_i)$, and adding a previous column to each column of a determinant does not change it. If we do this division and split the resulting polynomial $P(x, y)$ into monomials in x and monomials in y we have

$$P(x, y) \;=\; \sum c_{\alpha,\beta} x^\alpha y^\beta \;=\; \mathbf{w}^T \mathbf{C}\, \mathbf{v}$$

where $\mathbf{v} = (\dots x^\alpha \dots)^T$ and $\mathbf{w} = (\dots y^\beta \dots)^T$ are monomial vectors and $\mathbf{C}$ is a matrix of coefficients $c_{\alpha,\beta}$. Solving the polynomial system reduces to the resolution of the linear system

$$\mathbf{C} \cdot \mathbf{v} = 0.$$

The y's can take arbitrary values, so each monomial y^α is linearly independent of all the others.

4.2 Pose from 3 points

We now apply the Bezout-Cayley-Dixon method to pose estimation from 3 points. We consider the three polynomials P_{12}, P_{13}, P_{23} to be polynomials in x_2, x_3 with coefficients in x_1 and c_{ij}, d_{ij}. The above construction in (x_2, x_3) gives a 5×5 $\mathbf{C}$ whose entries turn out to be linear in x_1^2. We can write it as

$$\mathbf{C}(x_1^2) \;=\; \mathbf{C}_0 + x_1^2\, \mathbf{C}_2,$$

If there is a solution then $\mathbf{C}$ has a null vector $\mathbf{C}\,\mathbf{v} \;=\; 0$, as the above determinant vanishes. $\mathbf{C}_2$ is always singular, but $\mathbf{C}_0$ is generically nonsingular. Multiply through by its inverse and divide by $x_1^2 \neq 0$ to give

$$\left(\mathbf{C}_2^{-1}\,\mathbf{C_0} + x_1^{-2}\,\mathbf{I}\right)\mathbf{v} \;=\; 0$$

This is an eigenvector problem. The five eigenvalues are 0 (a false root) and the x_1^{-2} of the four solutions of the pose problem. The corresponding eigenvectors give x_2, x_3. The matrices are:

$$\mathbf{C}_2 \;=\; \begin{pmatrix} -d_{23}\,c_{12}\,c_{13} & -d_{13}\,c_{12}\,c_{23}{+}c_{13}\,(d_{12}{-}d_{23}) & c_{12}\,(-d_{23}{+}d_{13}) & -d_{23}{+}d_{13}{+}d_{12} & -d_{13}\,c_{23} \\ c_{13}\,(d_{12}{-}d_{23}) & -d_{13}\,c_{23} & -d_{23}{+}d_{13}{+}d_{12} & 0 & 0 \\ -d_{12}\,c_{13}\,c_{23}{+}c_{12}\,(d_{13}{-}d_{23}) & -d_{23}{+}d_{13}{+}d_{12} & -d_{12}\,c_{23} & 0 & 0 \\ -d_{12}\,c_{23} & 0 & 0 & 0 & 0 \\ -d_{23}{+}d_{13}{+}d_{12} & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{C}_0 \;=\; \begin{pmatrix} 0 & c_{12}\,c_{23}{-}c_{13} & -c_{12} & -2 & c_{23} \\ -c_{13} & c_{23}{-}c_{12}\,c_{13} & -2 & -c_{12} & 0 \\ -c_{12}{+}c_{13}\,c_{23} & -2{+}c_{13}\,c_{12}\,c_{23} & c_{23}{-}c_{12}\,c_{13} & -c_{13} & c_{13}\,c_{23} \\ c_{23} & c_{12}\,c_{23} & 0 & 0 & c_{23} \\ -2 & -c_{12} & -c_{13} & -c_{23} & 0 \end{pmatrix} \qquad \mathbf{v} \;=\; \begin{pmatrix} 1 \\ x_1 \\ x_1^2 \\ x_2 \\ x_1\,x_2 \end{pmatrix}$$

5 Experimental results

This section presents comparative experiments testing the new algorithms developed here against some old ones. The methods are given abbreviated names as follows

4pt 3×3	The 4 point algorithm presented in [14].
3pt	The classical elimination based 3 point method [14].
3pt 5×5	The 5×5 eigensystem 3 point method.
4pt 9×9	The 9×9 method for 4 points.
4pt 12×12	The 12×12 method for 4 points.
4pt 24×24	The 24×24 method for 4 points.

We have also tested a variant of each of the algorithms that heuristically reorders the input points so that any of the reference points that are used especially by the algorithm are as widely spread as possible in the image.

In all experiments, the points are projected with a focal length of 1024 pixels into a 512×512 image. Gaussian noise is added, with default sigma 1 pixel. Each data point represents 200 trials on 3D points generated randomly in a Gaussian cloud of standard deviation 1 unit, seen from about 5 units away. However, in the singularity experiment, there were 500 trials per point and only the noise varied between trials, not the 3D points. In the coplanarity experiment, the cloud is flattened onto a plane. The results represent median errors, relative to the total size of the translation and to 1 radian rotation. The failure rate was measured rather arbitrarily, as the percentage of total trials where either the rotation or the translation error was over 0.5.

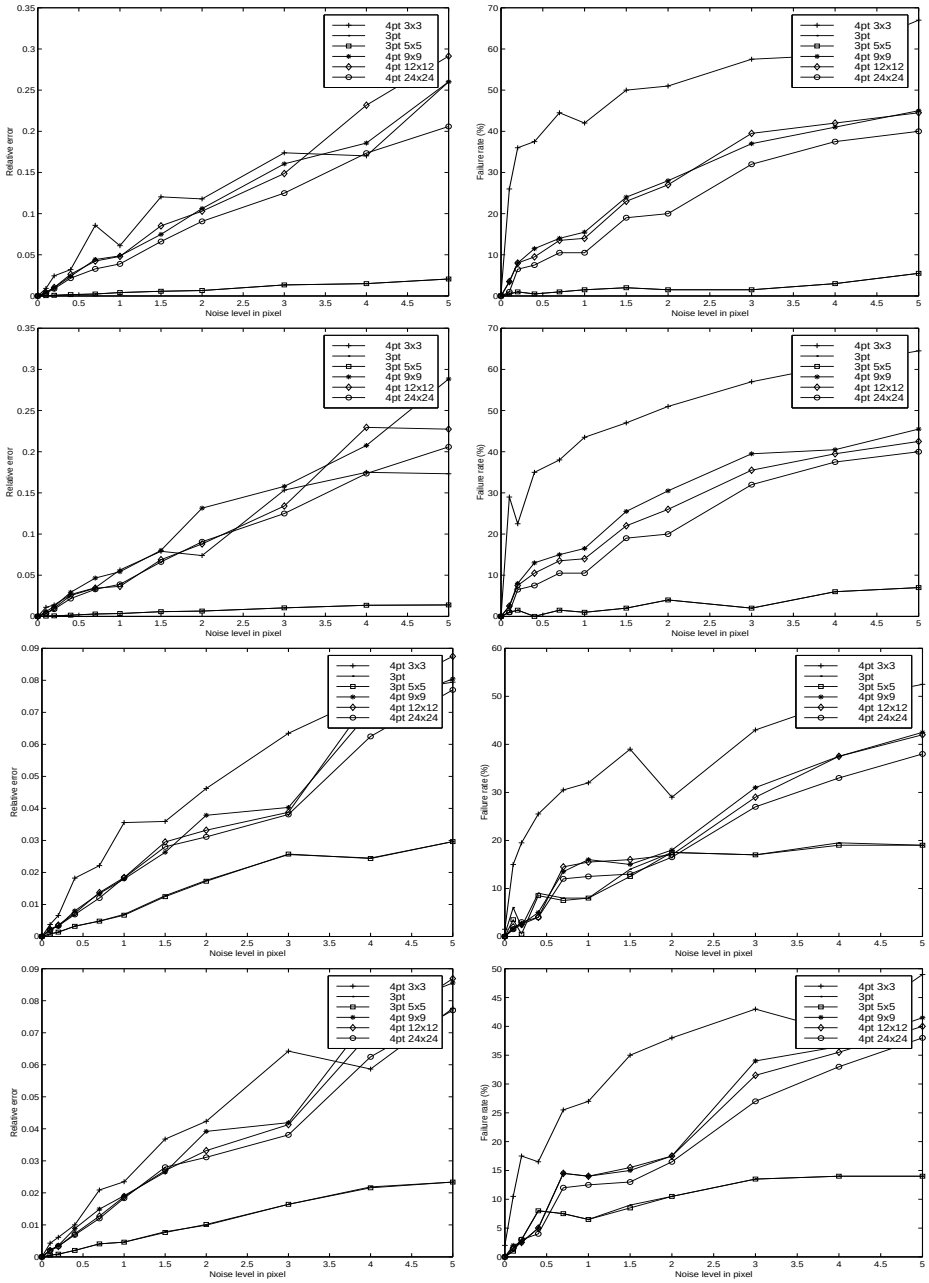

Fig. 2. The relative translation error and failure rate versus noise level in pixels, for 4 points. The first two rows are for non-coplanar, the second two for coplanar data. The first and third row use the input point ordering, the second and fourth heuristically reorder the points for better stability before running the pose methods. The trend of the rotation error is broadly similar to that of the translation error, and is not shown for lack of space.

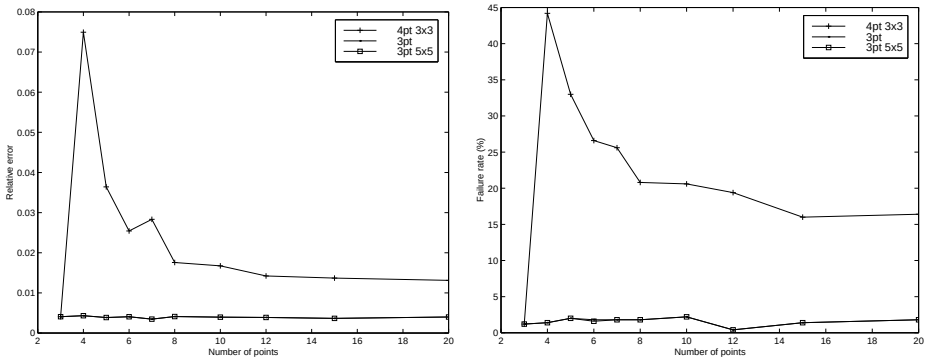

Fig. 3. Relative translation error and failure rate versus number of input points.

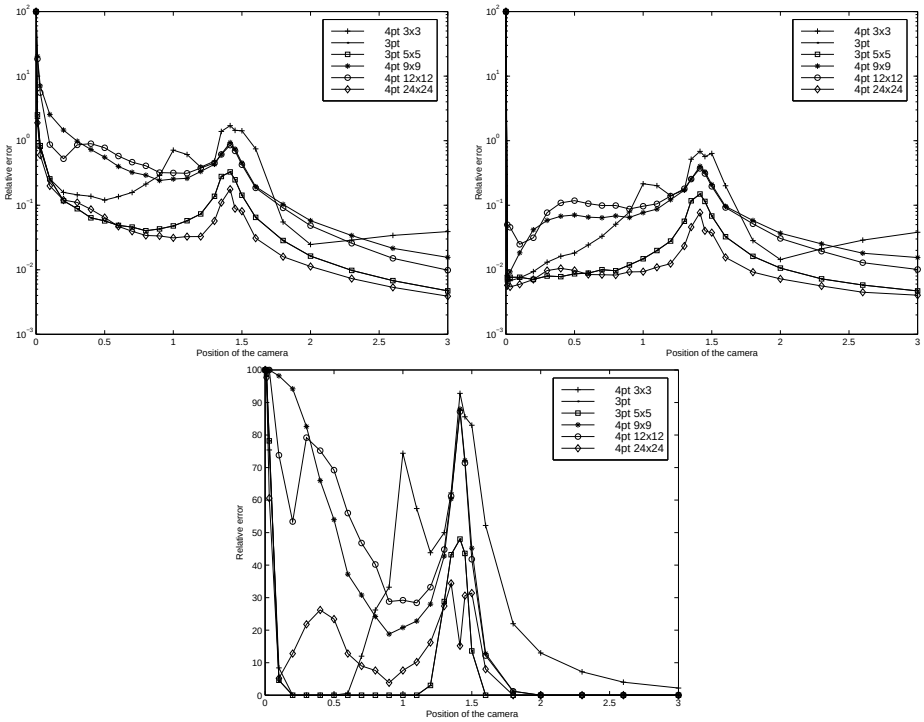

Fig. 4. Relative translation and rotation error and failure rate as the camera moves through two critical configurations for the pose problem, at position parameter 0 and $\sqrt{2}$.

Figure 2 shows the behaviour of the various methods with 4 input points, as the noise level is increased. Figure 3 shows the behaviour as additional points are added to the data set, for those methods that can handle additional points.

As mentioned in the introduction, the pose problem has some obtrusive singular cases that often cause problems in practice. Figure 4 shows error and failure rate results for one such configuration. The data is 4 coplanar points in a square $[-1,1]\times[-1,1]$ and the camera starts at 'position=0', at a singular point directly above their centre. The camera then moves sideways parallel to one edge of the square. At position= $\sqrt{2}$ units it crosses the side of the vertical circular cylinder through the 4 data points, where another singularity occurs. (The situation is even less well conditioned if the camera moves diagonally to a point above one vertex of the square — not shown).

The main conclusion from these experiments is that — setting aside the fact that they return multiple possible solutions which may be inconvenient — the 3-point algorithms significantly outperform all of the current linear 4-point ones. Even though the linear methods use more data and have built in redundancy, their relative errors and failure rates are often 2 to 10 times higher than those of the 3 point methods. This is disappointing as it means that the linear model has managed to capture only a very crude approximation to the underlying error surface. We are currently trying to understand this and correct it. Conventional data normalization does not seem to help here, but more sophisticated methods may be possible.

Coplanar points are not a singular case for any of the methods tested here, and on the whole their performance is similar to the non-coplanar case, except that the performance advantage of the 3 point methods is decreased. Increasing the number of points for the linear algorithms does improve the results slightly, but not by enough to equal the 3 point methods. For the methods that treat points asymmetrically, choosing well spread image points as the basis points does seem to help on average, but only very irregularly in individual cases. There are probably better heuristics than simple image spread for finding stable basis configurations. In particular, with many points from a Gaussian cloud, there is a slight tendancy for image spreading to choose configurations near the 'camera above circle of points' degeneracy, which reduces the average stability.

Traditional elimination and the new eigenvector based 3-point method have very similar performance in all cases tested, with the eigenvector method having perhaps a slight edge.

The performance of the 24×24, 12×12 and 9×9 linear methods is also similar, with 24×24 having a slight advantage in overall accuracy, but being significantly slower than the 9×9 method. These are only statistical trends — in any one problem it is very difficult to guess which will work best.

The errors and especially the failure rates of all of the methods are significantly higher than one would like, especially for the 4 points methods. This is in part due to the randomly generated data happening to fall near a singular configuration,

and to the wide zone that these seem to occupy, but a large part of the failure must certainly be algorithmic.

6 Conclusions

We have presented three new linear algorithms for pose estimation from 4 points, and an eigenspace based one that finds the 4 solutions of pose from 3 points. The main advantage of the linear algorithms is that they generate a unique solution from the outset. However the present implementations still seem quite far from the optimal accuracy and reliability. None of the methods degenerate for (generic) coplanar points. All of the methods developed here are easy to implement in the sense that their matrices are relatively simple functions of the input coordinates.

We believe that the linear methods are still far from their full potential, and in future work we will look more closely at performance issues in an attempt to improve them.

Acknowledgments

This work was partly supported by European project CUMULI.

References

1. V.Y. Pan B. Mourrain. Multivariate polynomials, duality and structured matrices. Technical report, INRIA Sophia-Antipolis, October 1998.
2. H.H. Chen. Pose determination from line-to-plane correspondence: Existence condition and closed-form solutions. In *Proceedings of the 3rd International Conference on Computer Vision, Osaka, Japan*, pages 374–378, 1990.
3. D. Dementhon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, 1995.
4. M. Dhome, M. Richetin, J.T. Lapresté, and G. Rives. Determination of the attitude of 3D objects from a single perspective view. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.
5. O. Faugeras and M. Hebert. The representation, recognition, and locating of 3D objects. *The International Journal of Robotics Research*, 5:27–52, 1986.
6. M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381 – 395, June 1981.
7. W. Förstner. Reliability analysis of parameter estimation in linear models with applications to mensuration problems in computer vision. *Computer Vision, Graphics and Image Processing*, 40:273–310, 1987.
8. R.M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, USA*, pages 592–598, 1991.

9. R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics and Image Processing*, 47:33–44, 1989.
10. B.K.P. Horn. Closed form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 5(7):1127–1135, 1987.
11. 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.
12. D. Lowe. Fitting parameterized three-dimensional models to images. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.
13. D.G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, Massachusets, 1885.
14. L. Quan and Z.D. Lan. Linear n-point camera pose determination. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, August 1999.
15. C.C. Slama, editor. *Manual of Photogrammetry, Fourth Edition*. American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, USA, 1980.
16. E.H. Thompson. Space resection: Failure cases. *Photogrammetric Record*, X(27):201–204, 1966.
17. B.P. Wrobel. Minimum solutions for orientation. In *Proc. of the Workshop on Calibration and Orientation of Cameras in Computer Vision, Washington D.C., USA*. Springer-Verlag, August 1992.
18. J.S.C. Yuan. A general phogrammetric solution for the determining object position and orientation. IEEE *Transactions on Robotics and Automation*, 5(2):129–142, April 1989.