May 2003

Linear Pose Estimation from Points or Lines

Adnan Ansar
California Institute of Technology

Kostas Daniilidis
University of Pennsylvania, kostas@cis.upenn.edu

Copyright 2003 IEEE. Reprinted from *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 25, Issue 5, May 2003, pages 578-589.
Publisher URL: http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=26906&puNumber=34

Linear Pose Estimation from Points or Lines

Abstract

Estimation of camera pose from an image of *n* points or lines with known correspondence is a thoroughly studied problem in computer vision. Most solutions are iterative and depend on nonlinear optimization of some geometric constraint, either on the world coordinates or on the projections to the image plane. For real-time applications, we are interested in linear or closed-form solutions free of initialization. We present a general framework which allows for a novel set of linear solutions to the pose estimation problem for both *n* points and *n* lines. We then analyze the sensitivity of our solutions to image noise and show that the sensitivity analysis can be used as a conservative predictor of error for our algorithms. We present a number of simulations which compare our results to two other recent linear algorithms, as well as to iterative approaches. We conclude with tests on real imagery in an augmented reality setup.

Keywords

Pose estimation, exterior orientation, absolute orientation, camera localization

Linear Pose Estimation from Points or Lines

Adnan Ansar and Kostas Daniilidis, *Member*, *IEEE*

Abstract—Estimation of camera pose from an image of n points or lines with known correspondence is a thoroughly studied problem in computer vision. Most solutions are iterative and depend on nonlinear optimization of some geometric constraint, either on the world coordinates or on the projections to the image plane. For real-time applications, we are interested in linear or closed-form solutions free of initialization. We present a general framework which allows for a novel set of linear solutions to the pose estimation problem for both n points and n lines. We then analyze the sensitivity of our solutions to image noise and show that the sensitivity analysis can be used as a conservative predictor of error for our algorithms. We present a number of simulations which compare our results to two other recent linear algorithms, as well as to iterative approaches. We conclude with tests on real imagery in an augmented reality setup.

Index Terms—Pose estimation, exterior orientation, absolute orientation, camera localization.

1 Introduction

Pose estimation appears repeatedly in computer vision in many contexts, from visual servoing over 3D input devices to head pose computation. Our primary interest is in real-time applications for which only a small number of world objects (lines or points) is available to determine pose. Augmented reality [2], in which synthetic objects are inserted into a real scene, is a prime candidate since a potentially restricted workspace demands robust and fast pose estimation from few targets. The motion of the camera is usually unpredictable in such scenarios, so we also require algorithms which are noniterative and require no initialization.

In this paper, we propose a novel set of algorithms for pose estimation from n points or n lines. The solutions are developed from a general procedure for linearizing quadratic systems of a specific type. If a unique solution for the pose problem exists, then our algorithms are guaranteed to return it. They fail in those cases where there are multiple discrete solutions. Hence, we can guarantee a solution for $n \geq 4$, provided the world objects do not lie in a critical configuration [21], [26]. The only similar noniterative methods for an arbitrary number of points are those of Quan and Lan [24] and Fiore [7]. We are aware of no competing method for lines, but show that our results are qualitatively acceptable in comparison to an iterative algorithm of Kumar and Hanson [16].

1.1 Related Work

Our goal has been to develop fast pose estimation algorithms which produce stable results for a small number of point or line correspondences. In the point case, a similar approach to ours is taken by Quan and Lan [24]. They derive a set of eighth degree polynomial constraints in even powers on the depth of each reference point by taking sets of three inherently quadratic constraints on three variables and eliminating two using Sylvester resultants. They apply this method to each point in turn. Our algorithm, like theirs, is based on depth recovery, but our approach avoids the degree increase, couples all n points in a single system of equations, and solves for all n simultaneously. Recently, Fiore [7] has produced an algorithm for points which introduces two scale parameters in the world to camera transformation and solves for both to obtain the camera coordinates of points. Unlike our algorithm and that of Quan and Lan, Fiore's approach requires at least six points unless they are coplanar. We show in Section 4.1, that our algorithm outperforms both of these linear algorithms in terms of accuracy. We also mention the approach of Triggs [27] which uses multiresultants to solve a polynomial system derived from the image of the absolute quadric. This method is best suited to four or five points and does not perform as well as direct decomposition of the projection matrix for larger collections of points.

There are many closed form solutions to the three point problem, such as [4], [10], which return solutions with well understood multiplicities [15], [22]. Fischler and Bolles [8] extended their solution to four points by taking subsets and using consistency checks to eliminate the multiplicity for most point configurations. Horaud et al. [11] developed a closed form solution on four points, which avoids this reduction to a three point solution. These closed form methods can be applied to more points by taking subsets and finding common solutions to several polynomial systems, but the results are susceptible to noise and the solutions ignore much of the redundancy in the data.

There exist many iterative solutions based on minimizing the error in some nonlinear geometric constraints, either on the image or target. We mention just a few. Nonlinear optimization problems of this sort are normally solved with some variation of gradient descent or Gauss-Newton methods. Typical of these approaches is the work of Lowe [19] and of Haralick [5]. There are also approaches which

- *A. Ansar is with the Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109. E-mail: Adnan.I.Ansar@jpl.nasa.gov.*
- *K. Daniilidis is with the GRASP Laboratory, University of Pennsylvania, 3401 Walnut Street, Suite 300C, Philadelphia, PA 19104-6228. E-mail: kostas@grasp.cis.upenn.edu.*

Manuscript received 19 Dec. 2001; revised 4 Sept. 2002; accepted 10 Sept. 2002.
Recommended for acceptance by R. Sharma.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 115596.

more carefully incorporate the geometry of the problem into the update step. For example, Kumar and Hanson [16] have developed an algorithm based on constraints on image lines using an update step adapted from Horn's [13] solution of the relative orientation problem. We compare this algorithm to our line algorithm in Section 4.1. There are several such variations using image line data. Liu et al. [18] use a combination of line and point data. Lu et al. [20] combine a constraint on the world points, effectively incorporating depth, with an optimal update step in the iteration. We use this as a reference in Section 4, to compare the three linear point algorithms mentioned. Dementhon and Davis [3] initialize their iterative scheme by relaxing the camera model to scaled orthographic. These iterative approaches typically suffer from slow convergence for bad initialization, convergence to local minima, and the requirement of a large number of points for stability. Our algorithms require no initialization, can be used for a small number of points or lines, and guarantee a unique solution when one exists.

Another approach is to recover the world to image plane projection matrix and extract pose information. This technique is examined by [1], [9] among many others. This projective approach is inherently less stable for pose estimation because of the simultaneous solution for the calibration parameters. It also requires a large data set for accuracy. We compare this approach to ours in Section 4.1.

2 Pose Estimation Algorithm

Throughout this paper, we assume a calibrated camera and a perspective projection model. If a point has coordinates $(x, y, z)^T$ in the coordinate frame of the camera, its projection onto the image plane is $(x/z, y/z, 1)^T$.

2.1 Mathematical Framework

We begin with a general mathematical treatment from which we will derive both our point and line algorithms. Consider a system of m quadratic equations in n variables x_i of the form

$$b_i = \sum_{j=1}^{n} \sum_{k=j}^{n} a_{ijk} x_i x_j \quad (i = 1 \ldots m), \tag{1}$$

where the right-hand side of (1) is homogeneous in $\{x_i\}$. We present a linearization technique to solve this system in the special case where the solution is a single point in $\mathbb{R}^n$. Let $x_{ij} = x_i x_j$ and $\rho = 1$. We rewrite (1) as

$$\sum_{j=1}^{n} \sum_{k=j}^{n} a_{ijk} x_{ij} - b_i \rho = 0 \quad (i = 1 \ldots m). \tag{2}$$

Since $x_{ij} = x_{ji}$, this is a homogeneous linear system in the $\frac{n(n+1)}{2} + 1$ variables $\{\rho, x_{ij} | 1 \leq i \leq j \leq n\}$. Such a system can be solved by singular value decomposition. We first write the system as

$$\mathbf{M}\bar{x} = 0, \tag{3}$$

where $\bar{x} = (x_{11}\ x_{12} \ldots x_{nn}\ \rho)^T$ and $\mathbf{M}$ is the matrix of coefficients of the system (2). Then, $\bar{x} \in \mathrm{Ker}(\mathbf{M})$. If $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$ is the SVD, then $\mathrm{Ker}(\mathbf{M}) = \mathrm{span}(\{\mathbf{v}_i\})$ where $\{\mathbf{v}_i\}$ are the columns of $\mathbf{V}$ corresponding to the zero singular values in $\mathbf{\Sigma}$.[1] If $\mathrm{Ker}(\mathbf{M})$ is one-dimensional, then $\bar{x}$ is recovered up to scale. However, the condition $\rho = 1$ determines scale and returns the correct solution to (2), from which we recover the solution to (1) up to a uniform sign error. In practice, the physical interpretation of the problem will determine sign.

If the dimension of $\mathrm{Ker}(\mathbf{M})$ is $N > 1$, we attempt to isolate the solution to (1) by reimposing the quadratic nature of the original problem. Since $\bar{x} \in \mathrm{Ker}(\mathbf{M})$, there exist real numbers $\{\lambda_i\}$ such that

$$\bar{x} = \sum_{i=1}^{N} \lambda_i \mathbf{v}_i. \tag{4}$$

For any integers $\{i, j, k, l\}$ and any permutation $\{i', j', k', l'\}$, observe that $x_{ij}x_{kl} = x_{i'j'}x_{k'l'}$. Substituting individual rows from the right-hand side of (4) into relations of this sort results, after some algebra, in constraints on the λ_i of the form

$$\begin{aligned} &\sum_{a=1}^{N} \lambda_{aa} (\mathbf{v}_a^{ij}\mathbf{v}_a^{kl} - \mathbf{v}_a^{i'j'}\mathbf{v}_a^{k'l'}) + \\ &\sum_{a=1}^{N} \sum_{b=a+1}^{N} 2\lambda_{ab} (\mathbf{v}_a^{ij}\mathbf{v}_b^{kl} - \mathbf{v}_a^{i'j'}\mathbf{v}_b^{k'l'}) = 0, \end{aligned} \tag{5}$$

where we use the notation $\lambda_{ab} = \lambda_a \lambda_b$ for integers a and b, and $\mathbf{v}_a^{ij}$ refers to the row of $\mathbf{v}_a$ corresponding to the variable x_{ij} in $\bar{x}$. We again have the obvious relation $\lambda_{ab} = \lambda_{ba}$. It follows that equations of the form (5) are linear and homogeneous in the $\frac{N(N+1)}{2}$ variables $\{\lambda_{ab}\}$. These can be written in the form $\mathbf{K}\bar{\lambda} = 0$, where $\mathbf{K}$ is the matrix of coefficients from (5) and $\bar{\lambda}$ is the vector formed by the terms $\{\lambda_{ab}\}$. We again solve this system by SVD, where $\mathbf{K} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^\mathbf{T}$. Observe that $\mathrm{Ker}(\mathbf{K})$ must be one-dimensional since two independent solutions would allow us to derive two solutions to (1), contradicting our original assumption. Having recovered $\bar{\lambda}$ up to scale, we recover the correct scale by imposing the condition implied by the last row of (4), specifically that $\lambda_1 \mathbf{v}_1^L + \lambda_2 \mathbf{v}_2^L + \ldots + \lambda_N \mathbf{v}_N^L = \rho = 1$ where $\mathbf{v}_i^L$ is the last row of $\mathbf{v}_i$. Having solved for $\bar{\lambda}$, hence, $\bar{x}$, we obtain x_i as $\pm\sqrt{x_{ii}}$, where the choice of sign for x_1 determines the sign of x_i by $\mathrm{sgn}(x_i) = \mathrm{sgn}(x_1)\mathrm{sgn}(x_{1i})$.

Before presenting our pose estimation algorithms, we briefly present a more formal treatment of our approach. Let $\mathrm{HQ}(\mathbb{R}^n)$ and $\mathrm{HL}(\mathbb{R}^n)$ be the set of quadratic and linear equations on $\mathbb{R}^n$, respectively, which are homogeneous in the variables. Our approach was to linearize the quadratic system in (1) to the linear one in (2) by applying the map $f : \mathrm{HQ}(\mathbb{R}^n) \to \mathrm{HL}(\mathbb{R}^{\tilde{n}})$ defined by $f(t_i t_j) = t_{ij}$, $f(1) = \rho$, where $\tilde{n} = \frac{n(n+1)}{2} + 1$. This increases the dimension of the solution space to $N \geq 1$ by artificially disambiguating related quadratic terms. Let $V_0 = \mathrm{Ker}(\mathbf{M})$ as above. We think of V_0 as an N-dimensional affine variety in $\mathbb{R}^{\tilde{n}}$. V_0 assumes an especially simple form since it is a vector subspace of $\mathbb{R}^{\tilde{n}}$. To recover the original solution to (1), we impose additional constraints of the form $x_{ij}x_{kl} = x_{i'j'}x_{k'l'}$

1. Ker() refers to the kernel or nullspace of a linear transformation, i.e., the set of vectors mapped to zero. Span() refers to the span of a set of vectors, the set of all linear combinations of these vectors.

for $\{i', j', k', l'\}$ a permutation of $\{i, j, k, l\}$. Let e_1 be one such equation, and let $\mathrm{Var}(e_1)$ be the algebraic variety in $\mathbb{R}^{\tilde{n}}$ defined by it. Then, $V_1 = V_0 \cap \mathrm{Var}(e_1)$ is a subvariety of V_0 defined by the e_i and the system (2). Since $\mathrm{Var}(e_1)$ is not in any linear subspace of $R^{\tilde{n}}$, it follows that V_1 is a proper subvariety of V_0. Given a sequence of such constraints $\{e_i\}$ with e_i independent of $\{e_j | j < i\}$, we obtain a nested sequence of varieties $V_0 \supset V_1 \supset V_2 \ldots$ of decreasing dimension. Since we have more quadratic constraints than the dimension of V_0, we eventually arrive at the desired solution. Observe that this procedure is entirely generic and does not depend on the coefficients of the original system (1). It follows that an abstract description of the subspace $S = \mathrm{Var}(\{e_i\}) \subset \mathbb{R}^{\tilde{n}}$, which we do not yet have, would allow us to eliminate the second, often more computationally intensive, SVD needed to find $\mathrm{Ker}(\mathbf{K})$ in our procedure. Note that we are aware of the problems overdimensioning can cause when seeking solutions in a given parameter space in the presence of noise, for example, in determining the Essential matrix. However, these effects are determined by the geometry of the underlying space. In our case, the genericity of S and the linear nature of V_0 contributes to the robustness which we see in Section 4.

We now examine the variety S in more detail. For the moment, we will ignore the constant ρ since it was introduced only as a computational trick. As seen below, for the problems under consideration, we always know the signs of $\{x_i\}$, hence, $\{x_{ij}\}$. We now observe that the only essential relations of the form $x_{ij}x_{kl} = x_{i'j'}x_{k'l'}$ are those which can be written as

$$x_{ii}x_{jj} = x_{ij}^2. \tag{6}$$

Since $\{i', j', k', l'\}$ is a permutation of $\{i, j, k, l\}$, we have trivially that

$$x_{ii}x_{jj}x_{kk}x_{ll} = x_{i'i'}x_{j'j'}x_{k'k'}x_{l'l'}.$$

Now, substituting (6), we obtain

$$x_{ij}^2 x_{kl}^2 = x_{i'j'}^2 x_{k'l'}^2.$$

Since we know the signs of all x_{ij}, taking the square roots of both sides of the above equation and applying the correct signs results in the desired $x_{ij}x_{kl} = x_{i'j'}x_{k'l'}$. Observe also that, for four integers i, j, k, l, the polynomials $x_{ii}x_{jj} - x_{ij}^2$ and $x_{kk}x_{ll} - x_{kl}^2$ have zero sets which are coincident (or subsets of one another in either direction) only if $\{i, j\} = \{k, l\}$. It follows immediately that $S = \cap\{\mathrm{Var}(x_{ii}x_{jj} - x_{ij}^2)\}$, and that this is a basis for the variety. There are $\frac{n(n-1)}{2}$ such polynomials in an $\frac{n(n+1)}{2}$ dimensional Euclidean space. We expect then that S is n-dimensional. This is clear if we notice that only the variables $\{x_{ii}\}$ are independent. If there is a solution to the quadratic system (1), it follows immediately that the linearized version of this solutions satisfies (2) and that it lies in the variety S. Suppose the physical solution is unique. We have already established that there is no sign ambiguity in $\{x_i\}$ or $\{x_{ij}\}$. Suppose that our algorithm produces two solutions $\bar{x}$ and $\bar{y}$. Both of these must then be in $\mathrm{Ker}(\mathbf{M})$ and in S. However, if they are in S, it follows that for $x_{ij}^2 = x_{ii}x_{jj}$ and $y_{ij}^2 = y_{ii}y_{jj}$.

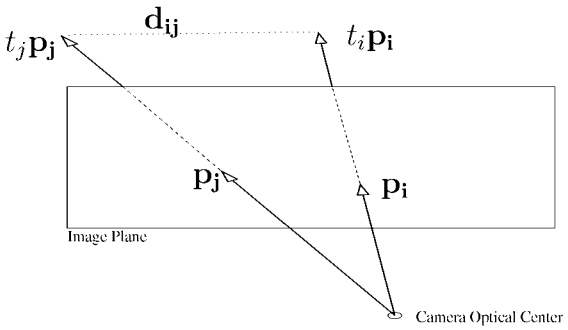


Fig. 1. The basic geometric constraint used in n point algorithm relates the distance between points in the world d_{ij} and the scale factors t_i and t_j associated with the projections $\mathbf{p}_i$ and $\mathbf{p}_j$.

However, the knowledge of sign implies that we can write this as

$$\begin{aligned} x_{ij} &= \mathrm{sgn}(x_i)\mathrm{sgn}(x_j)\sqrt{x_{ii}x_{jj}} = x_i x_j y_{ij} \\ &= \mathrm{sgn}(y_i)\mathrm{sgn}(y_j)\sqrt{y_{ii}y_{jj}} = y_i y_j. \end{aligned}$$

This explicit decomposition implies that $\bar{x}$ and $\bar{y}$ correspond to solutions consistent with the original quadratic system (1). This contradicts the uniqueness assumption on the solution to the original system.

2.2 Point Algorithm

We assume that the coordinates of n points are known in some global frame, and that for every reference point in the world frame, we have a correspondence to a point on the image plane. Our approach is to recover the depths of points by using the geometric rigidity of the target in the form of the $\frac{n(n-1)}{2}$ distances between n points.

Let $\mathbf{w}_i$ and $\mathbf{w}_j$ be two points with projections $\mathbf{p}_i$ and $\mathbf{p}_j$. We indicate by d_{ij} the distance between $\mathbf{w}_i$ and $\mathbf{w}_j$. Let t_i and t_j be positive real numbers so that $|t_i\mathbf{p}_i|$ is the distance of the point $\mathbf{w}_i$ from the optical center of the camera, similarly for t_j. It follows that $d_{ij} = |t_i\mathbf{p}_i - t_j\mathbf{p}_j|$. This is our basic geometric constraint (see Fig. 1). Let $b_{ij} = d_{ij}^2$. Then, we have

$$\begin{aligned} b_{ij} &= (t_i\mathbf{p}_i - t_j\mathbf{p}_j)^T(t_i\mathbf{p}_i - t_j\mathbf{p}_j) \\ &= t_i^2\mathbf{p}_i^T\mathbf{p}_i + t_j^2\mathbf{p}_j^T\mathbf{p}_j - 2t_it_j\mathbf{p}_i^T\mathbf{p}_j. \end{aligned} \tag{7}$$

Equation (7) is exactly of the form (1), and we apply the solution described to recover the depth scalings t_i. In this case, $\mathbf{M}$ in (3) has size $\frac{n(n-1)}{2} \times \left(\frac{n(n+1)}{2} + 1\right)$, and can be written as $\mathbf{M} = (\mathbf{M}'|\mathbf{M}'')$, where $\mathbf{M}'$ is $\frac{n(n-1)}{2} \times \frac{n(n-1)}{2}$ diagonal. $\mathbf{M}$ has the explicit form

$$\mathbf{M} = \left(\begin{array}{ccccc|ccccc} -2p_{12} & 0 & 0 & \ldots & 0 & p_{11} & p_{22} & 0 & \ldots & -c_{12} \\ 0 & -2p_{13} & 0 & \ldots & 0 & p_{11} & 0 & p_{33} & \ldots & -c_{13} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & -2p_{n-1,n} & 0 & \ldots & \ldots & p_{n-1,n-1} & p_{nn} & -c_{n-1,n} \end{array}\right). \tag{8}$$

It follows that $\mathrm{Ker}(\mathbf{M})$ is $\frac{n(n+1)}{2} + 1 - \frac{n(n-1)}{2} = n + 1$ dimensional. Hence, we must compute $\mathbf{K}$ and find its

kernel. $\mathbf{K}$ will have $\frac{(n+1)(n+2)}{2}$ rows and there are $O(n^3)$ equations of the form (5). We use only the $\frac{n^2(n-1)}{2}$ constraints derived from expressions of the form $t_{ii}t_{jk} = t_{ij}t_{ik}$.

The choice of sign for $\{t_i\}$ is clear, since these are all positive depth scalings. Given these scale factors, we have the coordinates of world points in the frame of the camera. Now, the recovery of camera rotation and translation simply amounts to solving the absolute orientation problem. We translate the two clouds of points, in the camera and world frames, to their respective centroids and recover the optimal rotation using unit quaternions [12] or SVD of the cross-covariance matrix [14]. Given the rotation, translation between the two centroids is immediately recovered.

We summarize the point algorithm:

1. Establish quadratic equations in point depths with coefficients depending on image measurements and distances between 3D points using (7).
2. Rewrite quadratic depth terms t_it_j as t_{ij}.
3. Solve resulting linear system in (8).
4. Express real solution as linear combination of basis vectors of $\mathrm{Ker}(\mathbf{M})$ with unknown $\{\lambda_a\}$ as in (4).
5. Use relations of the form $x_{ij}x_{kl} = x_{i'j'}x_{k'l'}$ for $\{i', j', k', l'\}$ a permutation of $\{i, j, k, l\}$ to establish quadratic relations between $\{\lambda_a\}$.
6. Rewrite as linear relations in terms $\{\lambda_{ab}\}$ and solve using (5).
7. Recover depths $\{t_i\}$ using $\{\lambda_a\}$.
8. Solve absolute orientation problem to recover pose.

2.3 Line Algorithm

Unlike the point case, direct recovery of line parameters does not appear feasible since the number of linearized variables (derived, for example, from Plücker coordinates) grows too fast in comparison to the number of available constraints. Instead, we show how to directly recover the rotation and translation.

Let $\{l_i = (\mathbf{v}_i, \mathbf{p}_i)\}$ be a collection of 3D lines such that in the world coordinate frame $\{\mathbf{v}_i\}$ are normalized vectors giving the directions of the lines and $\{\mathbf{p}_i\}$ are points on the lines. It follows that, in parametric form, points on l_i are given by $t_i\mathbf{v}_i + \mathbf{p}_i$ for the real parameter t_i. If $(\mathbf{R}, \mathbf{T}) \in SE(3) = SO(3) \times \mathbb{R}^3$ is the transformation relating the world and camera frames, then the corresponding representations of the lines in the camera frame are $\{l_i = (\mathbf{w}_i, \mathbf{q}_i)\}$ where $\mathbf{w}_i = \mathbf{R}\mathbf{v}_i$ and $\mathbf{q}_i = \mathbf{R}\mathbf{p}_i + \mathbf{T}$. Let P_i be the plane defined by the optical center of the camera and the line l_i.

Let the corresponding lines in the image plane of the camera be $\{s_i = (\bar{\boldsymbol{\alpha}}_i, \mathbf{c}_i)\}$, where $\bar{\boldsymbol{\alpha}}_i$ and $\mathbf{c}_i$ are of the forms $(\alpha_{i,x}, \alpha_{i,y}, 0)^T$ and $(c_{i,x}, c_{i,y}, 1)^T$, respectively, with $\bar{\boldsymbol{\alpha}}_i$ normalized. Consider the point $\mathbf{d}_i$ on s_i which is closest to the origin of the image plane. Then, $\mathbf{d}_i = \mathbf{c}_i - (\mathbf{c}_i^T\bar{\boldsymbol{\alpha}}_i)\bar{\boldsymbol{\alpha}}_i$. Let $\bar{\boldsymbol{\gamma}}_i = \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|}$. It follows that $\bar{\boldsymbol{\gamma}}_i^T\bar{\boldsymbol{\alpha}}_i = 0$ so that $\{\bar{\boldsymbol{\alpha}}_i, \bar{\boldsymbol{\gamma}}_i\}$ is an orthonormal frame spanning the plane P_i (see Fig. 2). Since $\mathbf{w}_i$ lies entirely in the plane P_i, we can write it as $\mathbf{w}_i = (\mathbf{w}_i^T\bar{\boldsymbol{\alpha}}_i)\bar{\boldsymbol{\alpha}}_i + (\mathbf{w}_i^T\bar{\boldsymbol{\gamma}}_i)\bar{\boldsymbol{\gamma}}_i$. Substituting $\mathbf{w}_i = \mathbf{R}\mathbf{v}_i$, we obtain $\mathbf{R}\mathbf{v}_i = (\mathbf{R}\mathbf{v}_i^T\bar{\boldsymbol{\alpha}}_i)\bar{\boldsymbol{\alpha}}_i + (\mathbf{R}\mathbf{v}_i^T\bar{\boldsymbol{\gamma}}_i)\bar{\boldsymbol{\gamma}}_i$. From this, we develop a set of quadratic equations in the entries of $\mathbf{R}$ to obtain a system of the form (1) and directly recover the rotation matrix. Let

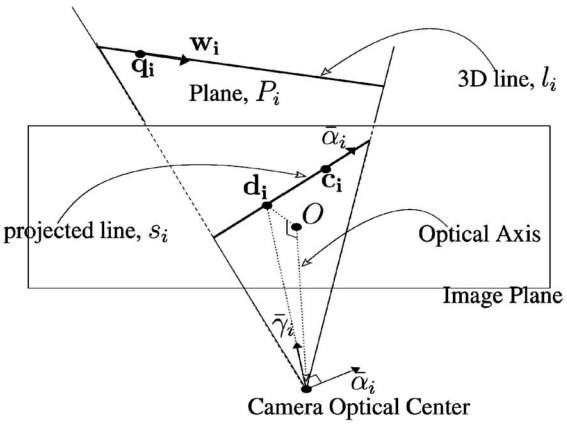


Fig. 2. Geometric constraint used in n line algorithm. The plane P_i determined by the line l_i and the optical center is spanned by $\{\bar{\alpha}_i, \bar{\gamma}_i\}$. Thus, $\mathbf{w}_i = \mathbf{R}_i\mathbf{v}_i$ can be written as a linear combination of these two vectors.

$K_{i,j} = \mathbf{v}_i^T\mathbf{v}_j$. We have the equation

$$K_{i,j} = [(\mathbf{R}\mathbf{v}_i^T\bar{\boldsymbol{\alpha}}_i)\bar{\boldsymbol{\alpha}}_i + (\mathbf{R}\mathbf{v}_i^T\bar{\boldsymbol{\gamma}}_i)\bar{\boldsymbol{\gamma}}_i]^T[(\mathbf{R}\mathbf{v}_j^T\bar{\boldsymbol{\alpha}}_j)\bar{\boldsymbol{\alpha}}_j + (\mathbf{R}\mathbf{v}_j^T\bar{\boldsymbol{\gamma}}_j)\bar{\boldsymbol{\gamma}}_j]. \tag{9}$$

For $i \neq j$, we obtain three additional equations from

$$\begin{aligned}\mathbf{R}\mathbf{v}_i \times \mathbf{R}\mathbf{v}_j = {} & [(\mathbf{R}\mathbf{v}_i^T\bar{\boldsymbol{\alpha}}_i)\bar{\boldsymbol{\alpha}}_i + (\mathbf{R}\mathbf{v}_i^T\bar{\boldsymbol{\gamma}}_i)\bar{\boldsymbol{\gamma}}_i] \\ & \times [(\mathbf{R}\mathbf{v}_j^T\bar{\boldsymbol{\alpha}}_j)\bar{\boldsymbol{\alpha}}_j + (\mathbf{R}\mathbf{v}_j^T\bar{\boldsymbol{\gamma}}_j)\bar{\boldsymbol{\gamma}}_j].\end{aligned} \tag{10}$$

Observe that (9) and (10) do not enforce the requirement that $\mathbf{R} \in SO(3)$. We accomplish this using the 12 quadratic constraints derived from

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}. \tag{11}$$

Note that, in general, there are only six independent constraints in (11), but by employing our linearization procedure, we introduce more relations on the 45 linearized terms $\{r_{ij} = r_ir_j\}$, where $\{r_i\}$ are the nine entries in $\mathbf{R}$. Using (9), (10), and (11), we obtain $n(2n-1) + 12$ equations of the form (1) in the 46 variables $\{\rho,\ r_{ij}\}$. For $n \geq 5$, we obtain a solution for $\mathbf{R}$ directly from the SVD of the corresponding $\mathbf{M}$ from (3). For $n = 4$, the additional step involving the SVD of $\mathbf{K}$ is required. Observe that the sign convention is also determined. Since $\mathbf{R} \in SO(3)$, we need only choose the global sign so that $\det(\mathbf{R}) = 1$.

Having recovered the rotation, we describe how to recover the translation. Given the point $\mathbf{q}_i$ on the line l_i in camera coordinates, we project to a point $\mathbf{k}_i = (q_{i,x}/q_{i,z}, q_{i,y}/q_{i,z}, 1)$ on the image plane. Since this point is on the line s_i, we have, using the notation of this section,

$$q_{i,z}(\mathbf{k}_i^T\bar{\boldsymbol{\gamma}}_i)\bar{\boldsymbol{\gamma}}_i = q_{i,z}\mathbf{d}_i.$$

Substituting $\mathbf{q}_i = \mathbf{R}\mathbf{p}_i + \mathbf{T}$ for each line, we obtain two linear equations in the entries of $\mathbf{T}$. A solution can be obtained by directly applying SVD.

We summarize the line algorithm:

1. Use invariance of inner products (9), expression of cross product in two independent forms (10), and

membership of $\mathbf{R}$ in $SO(3)$ (11) to establish quadratic equations in entries of $\mathbf{R}$, the rotation matrix between world an image coordinate frames.
2. Rewrite quadratic terms $r_i r_j$ from entries of $\mathbf{R}$ as r_{ij}.
3. Solve resulting linear system.
4. For four lines, proceed as with points for multidimensional kernel.
5. For five or more lines, first linear step is sufficient.
6. Using known rotation, write overdetermined linear system in entries of translation using projection equations and solve using SVD.

3 Sensitivity Analysis

We now analyze the sensitivity of the linear system and its intersection with the variety described above to image noise. Consider the linear system of the form (3).

$$\mathbf{M}\overline{x} = 0. \tag{12}$$

Recall that the entries of $\mathbf{M}$ are polynomials in the image measurements. In the presence of noise, the true coefficient matrix for this system is $\tilde{\mathbf{M}} = \mathbf{M} + \mathbf{M}_e$, for some error matrix $\mathbf{M}_e$, and the true physical solution to the pose problem is $\tilde{x} = \overline{x} + \overline{x}_e$, for some error vector $\overline{x}_e$. In our notation, $\{\tilde{\mathbf{M}}, \tilde{x}\}$ represents the true physical system, and $\{\mathbf{M}, \overline{x}\}$ represents the system perturbed by image noise. We proceed by using standard techniques from matrix perturbation theory [25].

If we assume knowledge of the noise in image measurements, we can estimate $\mathbf{M}_e$. In particular, we can bound $|\mathbf{M}_e|$ for some appropriate matrix norm. For the line case, these are complicated polynomial expressions, but bounds can be computed experimentally. In the point case, the polynomials are simple products and sums of products of image measurement errors. Our concern is not with the computation of bounds on $\mathbf{M}_e$ since these depend on a clearly defined way on image measurements, but with the nature of $\tilde{x}_e$, the error in the recovered solution. We make no further mention of the computation of $|\mathbf{M}_e|$ in this section.

We first consider the simpler case of five or more lines and then apply a more elaborate analysis to the point algorithm. We omit the four line case in this treatment. In the following, we will use $|\cdot|_F$ to indicate the Frobenius norm of a matrix, and $|\cdot|$ to indicate the 2-norm of vectors or matrices, as appropriate.

3.1 Sensitivity for Lines

Using the notation developed above, we write

$$\begin{aligned} &\tilde{\mathbf{M}}\tilde{x} = 0 \\ &(\mathbf{M} + \mathbf{M}_e)(\overline{x} + \overline{x}_e) = 0 \\ &\mathbf{M}\overline{x} + \mathbf{M}\overline{x}_e = -\mathbf{M}_e(\overline{x} + \overline{x}_e). \end{aligned} \tag{13}$$

First, observe that $\mathbf{M}\tilde{x} = 0$.[2] We multiply both sides of (13) by the pseudoinverve of $\mathbf{M}$ to obtain

$$\mathbf{M}^\dagger\mathbf{M}\overline{x}_e = -\mathbf{M}^\dagger\mathbf{M}_e(\overline{x} + \overline{x}_e). \tag{14}$$

In the case of five or more lines, recall that $\mathbf{M}$ has full column rank. It follows that $\mathbf{M}^\dagger\mathbf{M}\tilde{x}_e = \tilde{x}_e$. Using this fact, the triangle inequality and the properties of the Frobenius norm, we obtain

$$|\overline{x}_e| \leq |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F (|\overline{x}| + |\overline{x}_e|).$$

At this point, we can choose to ignore the quadratic error term, or we can state, based on observation of a given physical situation and camera setup, that $|\tilde{x}_e| \leq \kappa|\tilde{x}|$, where $\kappa \leq 1$ for any reasonable situation and noise level. It follows that

$$|\overline{x}_e| \leq \lambda |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F |\overline{x}| \tag{15}$$

with $1 \leq \lambda \leq 2$, where the experimental evaluation below indicates that the upper limit is a highly conservative estimate. In addition, for the line case $\overline{x}$ consists of products of terms from rotation matrices, all of which are bounded above by one. Thus, we can restate (15) as

$$|\overline{x}_e| \leq \sqrt{46} \cdot \lambda |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F.$$

3.2 Sensitivity for Points

The point case becomes more complicated by the fact that $\mathbf{M}$ is rank deficient and has a multidimensional kernel. We begin by writing the error term as $\overline{x}_e = \overline{x}_p + \overline{x}_n$, where $\overline{x}_p \in \mathbf{K} = \mathrm{Ker}(\mathbf{M})$ and $\overline{x}_n$ is orthogonal to $\mathbf{K}$. Then,

$$\mathbf{M}^\dagger\mathbf{M}\overline{x}_e = \mathbf{M}^\dagger\mathbf{M}(\overline{x}_p + \overline{x}_n) = \mathbf{M}^\dagger\mathbf{M}(\overline{x}_n) = \overline{x}_n,$$

and (15) becomes

$$|\overline{x}_n| \leq \lambda |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F |\overline{x}|. \tag{16}$$

In other words, we only have an estimate of the error in a direction normal to the kernel of M. Nothing more can be obtained from the linear system. We must now use the fact that both the perturbed solution $\overline{x}$ and the correct solution $\overline{x} = \overline{x} + \overline{x}_e$ lie on the variety defined by the relations in (6). Each of the equations of the form $x_{ii}x_{jj} = x_{ij}^2$ defines a differentiable manifold in $\mathbf{R}^N$, but their intersection does not. This lack of regularity means we cannot use differential properties in a straightforward manner.

Instead, we begin by examining the kernel more closely. Recall that $\mathbf{K}$ is $n+1$-dimensional for n points. If $\overline{x}$ is written as

$$\overline{x} = (x_{12} \ \ldots x_{1n} \ x_{23} \ \ldots x_{2n} \ \ldots x_{(n-1)n} \ x_{11} \ \ldots x_{nn} \ \rho)^T,$$

then using the notation of Section 2.2 with xs replacing ts to maintain consistency with this section, we write down an explicit basis for $\mathbf{K}$. Let $\mathbf{w} = (1\ 1\ 1\ 1\ \ldots 1)^T$ and

$$\mathbf{v}_i = \left(\frac{\delta_{12}}{p_{12}} \ldots \frac{\delta_{1n}}{p_{1n}} \ \frac{\delta_{23}}{p_{23}} \ \ldots \frac{\delta_{2n}}{p_{2n}} \ \ldots \frac{\delta_{(n-1)(n)}}{p_{(n-1)n}} \ \frac{\delta_{11}}{p_{11}} \ \ldots \frac{\delta_{nn}}{p_{nn}} \ 0 \right)^T,$$

where

$$\delta_{kl} = \begin{cases} \frac{1}{2} & \text{if } k \neq l \text{ and } (k = i \text{ or } l = i) \\ 1 & \text{if } l = k = i \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\{\mathbf{v}_i | i = 1..n\} \cup w$ is the required basis. The fact that $\mathbf{M}\mathbf{v}_i = \mathbf{M}\mathbf{w} = 0$ for the form of $\mathbf{M}$ in (8) is a simple calculation. Linear independence is easily established by inspecting the last $n+1$ entries in each vector. Note that this is not an orthogonal basis, but we do not require this for our purposes.

2. Strictly speaking, $\mathbf{M}\tilde{x}$ is not zero. However, in practice, $|\mathbf{M}\tilde{x}| << 1$, since $\tilde{x}$ is the closest solution (e.g., via SVD) to the system defined by $\mathbf{M}$. This does not depend on the magnitude of the error in $\mathbf{M}$.

Since $\overline{x}_p$ lies in the kernel of $\mathbf{M}$, we have $\overline{x}_p = \sum_{i=1}^{n} c_i \mathbf{v}_i + d\mathbf{w}$ for real numbers $\{c_i, d\}$. Observe, however, that the only basis vector which contributes a component to ρ is $\mathbf{w}$. Since $\rho = 1$ is known exactly, there is no error in this direction. It follows that $d = 0$. An alternative interpretation of this fact is that the ρ variable can be eliminated from the outset if we do not require that the linear equation in the first step of our algorithm be homogeneous. The result is an expression of the sort

$$(\mathbf{M} + \mathbf{M}_e)(\overline{x} + \overline{x}_e) = \bar{w} \Rightarrow \mathbf{M}\overline{x}_e = -\mathbf{M}_e(\overline{x} + \overline{x}_e),$$

since $\mathbf{M}\overline{x} = \bar{w}$ and where the equations are over one fewer variable. From this point, we would proceed as above.

We write the ij component of $\overline{x}_p$ as $\hat{x}_{ij}$ and the ij component of $\overline{x}_n$ as $\breve{x}_{ij}$, and focus on the three vector obtained by projection of $\overline{x}_p$ onto the x_{ii}, x_{jj}, and x_{ij} directions. We see from inspection of the basis vectors that, for any i, j, only v_i, v_j contribute to $(\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{ij})$. This can then be rewritten as $(\frac{c_i}{p_{ii}}, \frac{c_j}{p_{jj}}, \frac{c_i + c_j}{2p_{ij}})$ by substituting the explicit form of $\{v_i\}$. We now obtain a simple relationship between $\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{ij}$, namely,

$$-2p_{ij}\hat{x}_{ij} + p_{ii}\hat{x}_{ii} + p_{jj}\hat{x}_{jj} = 0. \tag{17}$$

If $\overline{x} + \overline{x}_e$ is the solution to the unperturbed system $\mathbf{M} + \mathbf{M}_e$, it must satisfy $x_{ii}x_{jj} = x_{ij}^2$ for all i, j. If we substitute $\overline{x}_e = \overline{x}_p + \overline{x}_n$, we obtain

$$(x_{ii} + \hat{x}_{ii} + \breve{x}_{ii})(x_{jj} + \hat{x}_{jj} + \breve{x}_{jj}) = (x_{ij} + \hat{x}_{ij} + \breve{x}_{ij})^2. \tag{18}$$

Note that $\breve{x}_{ii}, \breve{x}_{jj}, \breve{x}_{ij}$ are bounded by (16). We now combine (17) and (18) and try to impose bounds on $\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{ij}$. We mention in passing that, although $\overline{x}_p$ is orthogonal to $\overline{x}_n$, we can make no such statement about the vectors $(\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{kk})$ and $(\breve{x}_{ii}, \breve{x}_{jj}, \breve{x}_{kk})$.

As a first approximation, we ignore all quadratic terms in error in (18), which then becomes

$$x_{ii}\hat{x}_{jj} + x_{jj}\hat{x}_{ii} - 2x_{ij}\hat{x}_{ij} \approx 2x_{ij}\breve{x}_{ij} - (x_{ii}\breve{x}_{jj} + x_{jj}\breve{x}_{ii}). \tag{19}$$

We rewrite the right-hand side of (19) as f_{ij}. Since f_{ij} depends only on terms in $\overline{x}$ and $\overline{x}_n$, we can bound it explicitly. Solving for $\hat{x}_{ij}$ and substituting into (17), we obtain a line in $\hat{x}_{ii}, \hat{x}_{jj}$ given by

$$\left(x_{jj} - x_{ij}\frac{p_{ii}}{p_{ij}}\right)\hat{x}_{ii} + \left(\tilde{x}_{ii} - x_{ij}\frac{p_{jj}}{p_{ii}}\right)\hat{x}_{jj} = f_{ij}. \tag{20}$$

If we take any k with $i \neq k \neq j$, we obtain two more equations of the form

$$(x_{kk} - x_{ik}\frac{p_{ii}}{p_{ik}})\hat{x}_{ii} + (x_{ii} - x_{ik}\frac{p_{kk}}{p_{ii}})\hat{x}_{kk} = f_{ik}$$
$$(x_{kk} - x_{jk}\frac{p_{jj}}{p_{jk}})\hat{x}_{jj} + (x_{jj} - x_{jk}\frac{p_{kk}}{p_{jj}})\hat{x}_{kk} = f_{jk}.$$

However, these are all linear in $\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{kk}$, and we solve them simultaneously to obtain

$$\hat{x}_{ii} = \frac{f_{ij}a_{jk}b_{ik} + f_{ik}b_{ij}b_{jk} - f_{jk}b_{ij}b_{ik}}{a_{ij}a_{jk}b_{ik} + a_{ik}b_{ij}b_{jk}}, \tag{21}$$

where

$$a_{ij} = x_{jj} - x_{ij}\frac{p_{ii}}{p_{ij}} \quad b_{ij} = x_{ii} - x_{ij}\frac{p_{jj}}{p_{ij}}.$$

The other terms are obtained by transposing the indices appropriately.

For each i, we compute $\hat{x}_{ii}$ using all combinations of j, k with $j \neq i \neq k$ and for a bound on the fs obtained from $\overline{x}_n$. We need only consider the smallest of these since all relations must be satisfied. Combining the terms $\hat{x}_{ii}$, bounded as above, and using (17), we obtain a bound on $|\overline{x}_p|$ of the same order as $|\overline{x}_n|$. It follows that we have bounded $|\overline{x}_e|$.

We mention that the linear approximation above is not necessary. If we choose to include the terms which are quadratic in the errors, we obtain a quadric from (18) in $\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{ij}$ instead of a plane as in (19). The intersection of this quadric with the plane defined by (17) is a conic in $\hat{x}_{ii}, \hat{x}_{jj}$ instead of a line as in (20). We then proceed as above, but with three conics instead of three lines. We do not attempt to write down a closed form expression for this approach.

3D Errors: Note that the above procedure can be adapted with little modification to handle the case of errors in the 3D coordinates of fiducials. The key point is that the error in ρ can no longer be assumed to be zero. Hence, the right-hand side of (17) will contain a term depending on estimated error in the distances between world points. Consequently, one of the two error planes intersected above is not a linear subspace of $\mathbf{R}^n$, but rather an affine space, whose distance from the origin is a function of the estimated error in world coordinates.

4 Results

We conduct a number of experiments, both simulated and real, to test our algorithms (hereafter referred to as **NPL** and **NLL** for n point linear and n line linear, respectively) under image noise. We compare to the following algorithms:

For points:

- **PM**: Direct recovery and decomposition of the full projection matrix from six or more points by SVD methods [6]. We use a triangle (△) to indicate this algorithm on all graphs.
- **F**: The n point linear algorithm of Fiore [7]. We signify this by a square (□).
- **QL**: The n point linear algorithm of Quan and Lan [24]. We signify this by a diamond (⋄).
- **LHM**: The iterative algorithm of Lu et al. [20] initialized at ground truth. We signify this by a circle (○) and include it primarily as a reference to compare the absolute performance of the linear algorithms. We expect it to achieve the best performance.

For lines:

- **KH**: The iterative algorithm of Kumar and Hanson referred to as R_and_T in [16]. We initialize **KH** at the ground truth translation and rotation (**KHRT** signified by △) and at ground truth translation and identity rotation (**KHT** signified by □).

4.1 Simulation

All simulations are performed in MATLAB. We assume calibrated virtual cameras with effective focal length (diagonal terms in calibration matrix) $1,500$ in the point case and

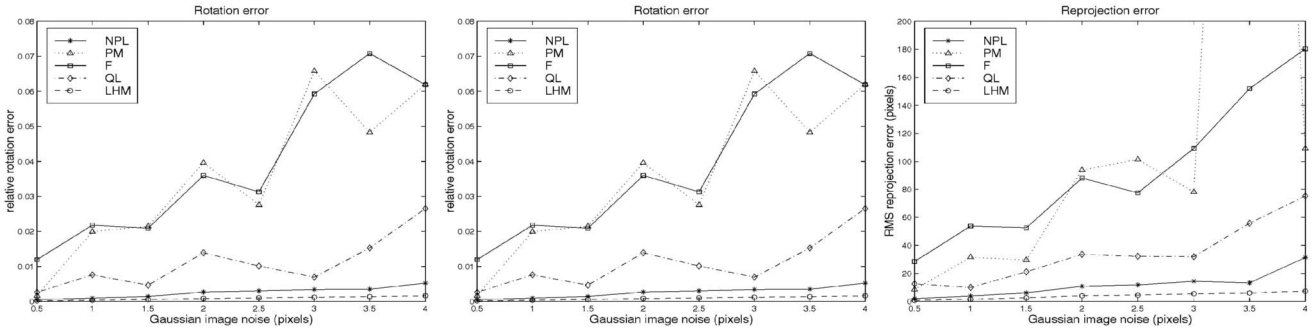


Fig. 3. **(Point Simulation 1)** Rotation, Translation, and Reprojection errors for six points versus noise level. We plot results for the five algorithms, **NPL**, **PM**, **F**, **QL**, and **LHM**. Note that **NPL** outperforms all but the iterative **LHM** with ground truth initialization.

600 in the line case. We report errors in terms of relative rotation error and relative translation error. For the point case, we also show RMS reprojection error. Each pose $(\mathbf{R}, \mathbf{T})$ is written as $(\bar{q}, \mathbf{T})$, where $\bar{q}$ is a unit quaternion. For recovered values $(\bar{q}_r, \mathbf{T}_r)$, the relative translation error is computed as $2\frac{|T-T_r|}{|T|+|T_r|}$ and the relative rotation error as the absolute error in the unit quaternion, $|\bar{q} - \bar{q}_r|$. For n points with real projections $\{\mathbf{p}_i\}$ and recovered projections $\{\mathbf{p}_{ir}\}$, the RMS reprojection error is

$$\sqrt{\left(\sum_{i=1}^{n} |p_i - p_{ir}|^2\right)/n}.$$

Note that reprojection errors are computed with a **different set of random points** than those used to estimate pose. In the case of **PM**, the reprojection is performed with the recovered projection matrix rather than by applying the world to camera transformation. Hence, the reprojection error can be smaller than for some linear methods, but never in comparison to **NPL**. Noise levels in image measurements are reported in terms of the standard deviation of a zero mean Gaussian. For the point case, when we add Gaussian noise with standard deviation σ to image coordinates, we do so independently in the x and y directions. We also only admit noise between -3σ and 3σ. In the line case, we again report pixel noise and propagate to noise in the line parameters following [28]. Unless indicated, all plots represent mean values over 400 trials.

Point Simulation 1 (Dependence on noise level). We vary noise from $\sigma = 0.5$ to 4. For each noise level, we generate 400 random poses. For each pose, we generate six points at random with distances between 0 and 200 from the camera. We restrict translations to $|T| < 100$. In Fig. 3, observe that **NPL** outperforms **PM**, **F**, and **QL** for all noise levels.

Point Simulation 2 (Dependence on number of points). We demonstrate that all five algorithms perform better as the number of points used for pose estimation is increased. Points and poses are generated exactly as in **Point Simulation 1**, but the number of points is varied from five to 11. We add 1.5×1.5 pixel Gaussian noise to all images. Note, in Fig. 4, that NPL outperforms the other linear algorithms, but that the performance difference is greatest for fewer points, which is our primary concern as mentioned in the introduction. Note that we do not plot results for **PM** or **F** for five points since these algorithms require at least six points.

Point Simulation 3 (Dependence on effective field of view). We generate poses as in **Point Simulation 1**. However, we now constrain the six points to lie on six of the vertices of a $10 \times 10 \times 10$ cube with arbitrary orientation, but centered on the optical axis of the camera. Once again, we take 31 point configurations for each of the 31 random poses. We add 1.5×1.5 pixel Gaussian noise to all images. Our goal is to evaluate the performance of our algorithm as the object occupies a smaller fraction of the image. Results are recorded in Fig. 5. **NPL** outperforms **QL**, **PM**, and **F** for pose estimation when the object is

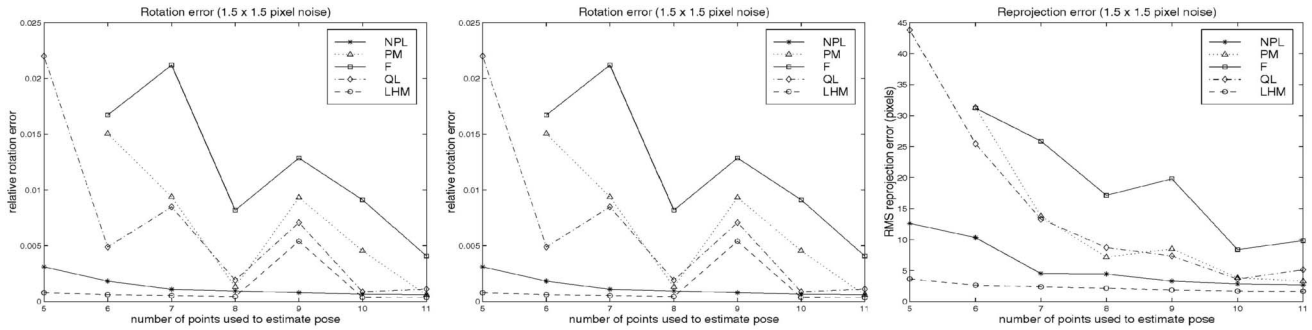


Fig. 4. **(Point Simulation 2)** Rotation, Translation, and Reprojection errors versus number of points used for pose estimation with 1.5×1.5 pixel Gaussian noise. We plot results for the five algorithms **NPL**, **PM**, **F**, **QL**, and **LHM**. We see that **NPL** outperforms all but the iterative **LHM** with ground truth initialization for all numbers of points considered. The difference is largest for a small number of points.

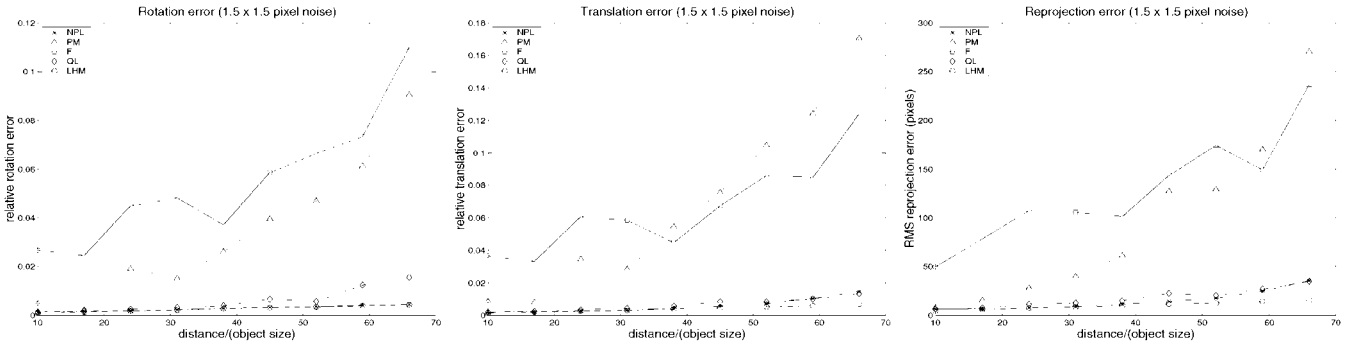


Fig. 5. **(Point Simulation 3)** Rotation, Translation, and Reprojection errors for six points versus extent of object given as distance/size with 1.5×1.5 pixel Gaussian noise. We plot results for the five algorithms, **NPL**, **PM**, **F**, **QL**, and **LHM**. We see that NPL outperforms all but the iterative KH with ground truth initialization in the region in which we are interested.

approximately seven times as far away as its extent. Note that this is our primary region of interest.

Line Simulation 1 (Dependence on noise level). We vary pixel noise from $\sigma = 0.5$ to 5 and propagate to noise in line parameters following [28]. For each noise level, we generate 400 poses and six line segments for each pose. World line segments are contained in a $20 \times 20 \times 20$ box in front of the camera and translations are restricted to $|T| < 10$. We plot relative rotation and translation errors for **NLL** and **KH** (see Fig. 6). As expected, the iterative algorithm performs better for good initialization (ground truth in the case of **KHRT**). However, we cannot predict convergence time. With poor initialization, even at ground truth translation and $\mathbf{R} = \mathbf{I}$ for **KHT**, our linear algorithm shows better mean performance. This is a result of convergence to local minima in some trials. We demonstrate this by plotting not only mean relative rotation and translation errors, but also the standard deviation of the relative rotation error. We immediately see the advantage of having no initialization requirement for **NLL**.

Line Simulation 2 (Dependence on number of lines). We generate poses and points as in **Line Simulation 1**, but for the numbers of lines varying from four to 11 and with fixed noise of 1.5×1.5 pixels. We see in Fig. 7, that the performance of both algorithms improves with increasing number of lines. Note also that **KH** is less likely to converge to local minima for larger numbers of lines. The absolute performance of **NLL** is again comparable to **KH**.

4.1.1 Timings in Simulation

We compare the runtimes of our procedure to several others using MATLAB implementations of all algorithms on a 1.1 GHz Pentium III processor. Note that realtime performance is not expected for any of the algorithms under MATLAB, and our only goal is to provide comparison. The iterative algorithms (**KH** and **LHM**) were set to terminate after 100 iterations if convergence had not yet been achieved. All results are averaged over 1,000 trials and for points or lines ranging from 6 to 11. The results are recorded in Table 1. The notation used is identical to that above with the following additions:

- **LHMPM**: refers to **LHM** initialized with the output of **PM**.
- **LHMT**: refers to **LHM** initialized at identity rotation and ground truth translation (analogous to **KHT** as defined above for lines).
- **KHNLL**: refers to **KH** initialized with **NLL**.

We see that in the implementations tested, **LHMPM** is consistently faster than our point algorithm (**NPL**), with the difference increasing with the number of points. However, as indicated in Section 4.2, for the number of points under consideration our algorithm runs in realtime under a C implementation on a 600 MHz Pentium III. In addition, unlike **LHM**, we guarantee recovery of the solution when it exists and require no initialization. For the line case (**NLL**), our algorithm is faster than **KH**. Since **NLL** is less computationally intensive

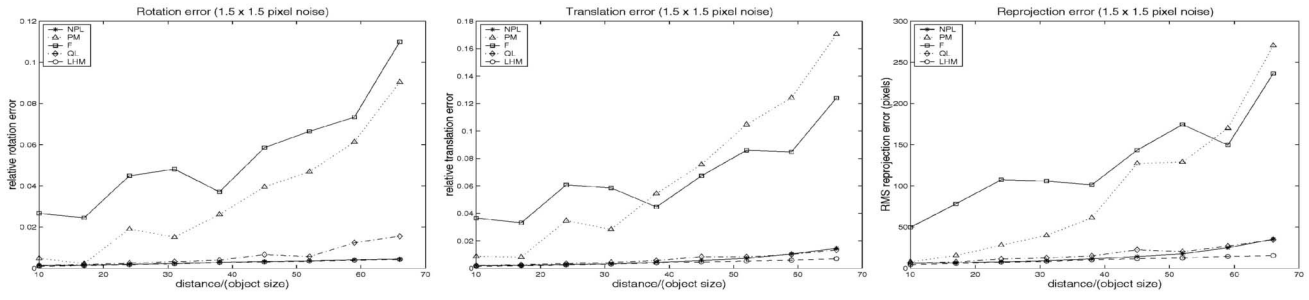


Fig. 6. **(Line Simulation 1)** Rotation and Translation errors and Standard Deviation of Rotation error versus noise level for **NLL** and **KH**. We initialize **KH** at ground truth $\mathbf{R}$ and $\mathbf{T}$ (**KHRT**) to evaluate absolute performance and at ground truth $\mathbf{T}$ and $\mathbf{R} = \mathbf{I}$ (**KHT**) to demonstrate the advantage of requiring no initialization in **NLL**.

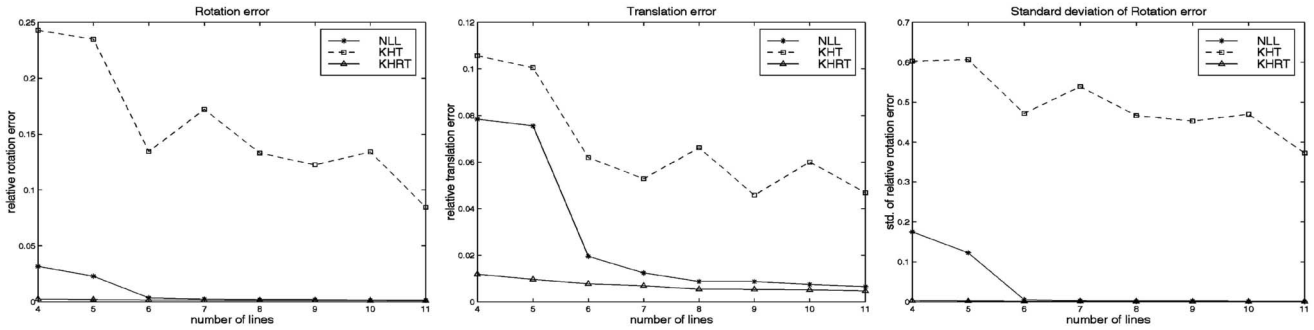


Fig. 7. **(Line Simulation 2)** Rotation and Translation errors and Standard Deviation of Rotation error versus number of points for **NLL** and **KH**. Noise is fixed at 1.5×1.5 pixels. We initialize **KH** at ground truth $\mathbf{R}$ and $\mathbf{T}$ (**KHRT**) to evaluate absolute performance and at ground truth $\mathbf{T}$ and $\mathbf{R} = \mathbf{I}$ (**KHT**) to demonstrate the advantage of requiring no initialization in **NLL**.

than **NPL**, we expect performance from a C implementation at video framerate with this algorithm as well.

4.2 Real Experiments

All images were taken with a Sony XC-999 camera and Matrox Meteor II frame grabber. The camera was calibrated using Lenz and Tsai's algorithm [17]. All image processing was done offline using MATLAB. Note that the more computationally intensive point algorithm **NPL** has been run in real-time (> 30 Hz) on a 600 MHz Pentium III using the implementation of SVD from numerical recipes in C [23] for the number of points discussed above and without any attempt to optimize the algorithm.

Point Experiment 1. We demonstrate that virtual objects are correctly registered into a real scene using **NPL** for pose estimation. We obtain the coordinates of the eight marked points in Fig. 8, by magnifying the relevant region and marking by hand with a MATLAB program. We take the vertex coordinates of a virtual box and the corners of the metal edge in the world frame, transform to the camera frame using the three recovered poses, and reproject. The metal edge, which we augment to a full cube, is seven inches on each side, and the camera distance varies from 30 to 40 inches from the nearest corner of the cube. Notice that the virtual boxes are properly placed and aligned with the world reference objects for all three poses.

TABLE 1
Timing Results (in Seconds) for Algorithms Under MATLAB on a 1.1 GHz Pentium III

Algorithm	6 points	7 points	8 points	9 points	10 points	11 points
NPL	0.085	0.145	0.239	0.409	0.721	1.240
PM	0.002	0.002	0.002	0.002	0.003	0.003
QL	0.097	0.166	0.264	0.397	.0564	0.775
F	0.002	0.002	0.002	0.002	0.003	0.003
LHMPM	0.072	0.058	0.053	0.052	0.051	0.050
LHMT	0.136	0.126	0.119	0.129	0.129	0.128
Algorithm	6 lines	7 lines	8 lines	9 lines	10 lines	11 lines
NLL	0.019	0.024	0.029	0.037	0.050	0.145
KHNLL	3.039	2.944	2.901	2.914	2.893	2.872
KHT	13.957	13.053	12.351	12.551	11.883	12.530

*Results are given for **NPL**, **PM**, **QL**, **F**, **LHMPM** (**LHM** initialized with **PM**), **LHMT** (**LHM** initialized with identity rotation and ground truth translation), **KHNLL** (**KH** initialized with **NLL**), and **KHT** (**KH** initialized with identity rotation and ground truth translation).*

Point Experiment 2. We repeat **Point Experiment 1** on a different scale. In Fig. 9, the box is approximately 18 inches on each side, and the camera is approximately eight feet from the nearest corner of the box. We estimate pose from the eight marked points using **NPL**. We then take the coordinates of two virtual boxes of identical size, stacked on top of and next to the real one, transform to camera coordinates, and reproject into the image. Note that the virtual boxes are very closely aligned with the real one and appear to be the correct size.

Point Experiment 3. We test **NPL** on coplanar points. In Fig. 10, we mark nine points on the calibration grid in the image. The points have a uniform spacing of eight inches. The camera is placed approximately 11 feet from the marked points. We recover the coordinates of the nine points using **NPL** and compute a best fit plane from the recovered points. The mean distance from the recovered points to the best fit plane is 0.15 inches with a standard deviation of 0.07 inches. We see that our algorithm does not degenerate for coplanar points.

Line Experiment 1. We demonstrate the correct registration of virtual objects into a real scene using **NLL**. In Fig. 11a, we indicate the seven line segments used to estimate camera pose. In Fig. 11b, we overlay a texture on the faces of the pictured box by transforming the world coordinates of the box vertices to camera coordinates and warping the texture onto the resulting quadrangles via homographies. We also place a virtual cube on the original box. The cube is aligned with the real box in world coordinates. Observe that, after transformation to the camera frame and reprojection, it remains aligned. Finally, we highlight the edges of the table by transforming its world coordinates to camera coordinates and reprojecting the appropriate line segments. We emphasize that all virtual objects are constructed in world coordinates and

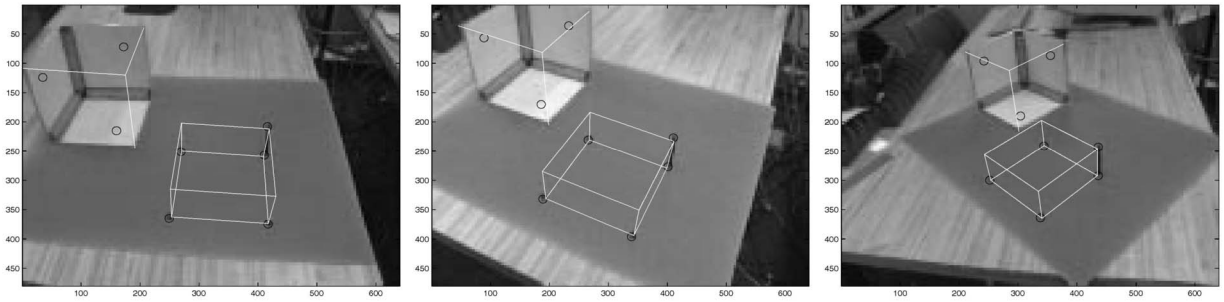


Fig. 8. **(Point Experiment 1).** Reprojection of a virtual box and three edges of a cube onto real-world reference objects. We estimate camera pose using the eight circled points and NPL.

inserted into the images only after pose estimation and transformation to camera coordinates.

4.3 Error Prediction

We show that the sensitivity analysis of Section 3 can be used to estimate errors in the recovered depths for the point algorithm, given some idea of the geometry of the problem. We focus on the point algorithm since the result for the line algorithm is a direct application of linear algebra techniques. Since our goal is to show the applicability of the overall procedure, we do not attempt to estimate $\mathbf{M}_e$. Rather, we will use the ground truth $\bar{\mathbf{M}}$ to find $\mathbf{M}_e$ exactly, and then calculate $\overline{x}_n$ using this. For each point i, we estimate $\overline{x}_p$, the error in the kernel direction using (21) over all j and k. We then take the smallest of these since all must be approximately satisfied.

We plot results for various levels of Gaussian noise ranging from 0.5 to five pixel standard deviation. For each noise level, we take 200 trials of six points with translation restricted to half the maximum scene depth. We plot in Fig. 12, the ratio of the norm of the real error (computed from ground truth) to the estimated error from the sensitivity analysis on a semilog scale. The horizontal lines represent a ratio of one. Points above the line are underestimation of error, and points below the line are overestimations. Note that underestimation in some cases is to be expected. First, the linear approximation in (19) will have an effect. Also, (21) has obvious singularities for certain configurations which we have not treated separately. Fig. 12a represents $\lambda = 1$ in (16). In this case, approximately 4.5 percent of the trials resulted in underestimation of errors. Fig. 12b represents $\lambda = 2$. Here, approximately 1 percent of the trials resulted in underestimation of the error.

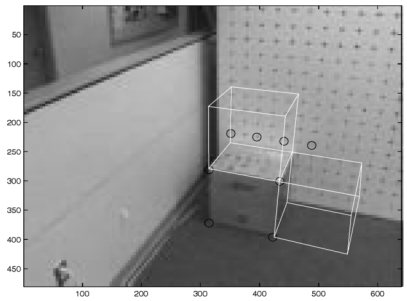


Fig. 9. **(Point Experiment 2).** Reprojection of two virtual boxes of dimensions identical to a real box. We estimate camera pose using the eight circled points and **NPL**.

5 Conclusion

Our goal was to develop fast and accurate pose estimation algorithms for a limited numbers of points or lines. We have presented a general mathematical procedure from which we derive a pair of linear algorithms which guarantee the correct solution in the noiseless case, provided it is unique. We develop a sensitivity analysis for our algorithms which also allows for coarse estimation of errors (with underestimation in very few cases) in pose recovery based on known image errors. Our point algorithm shows performance superior to competing linear algorithms and comparable to a recent iterative algorithm. For our line algorithm, there is no competing linear approach. We show results comparable to a robust iterative algorithm when it is correctly initialized and avoid the problems associated with local minima for such algorithms.

Acknowledgments

The authors are grateful for support through the following grants: NSF-IIS-0083209, NSF-EIA-0218691, NSF-IIS-0121293, NSF-EIA-9703220, and a Penn Research Foundation grant. A portion of this work was completed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

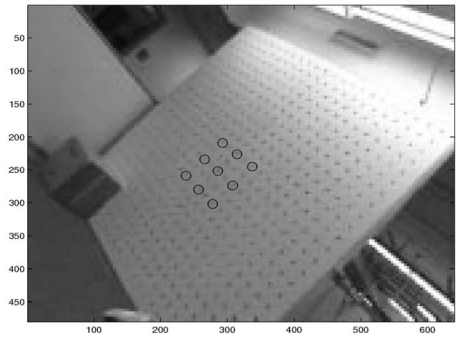
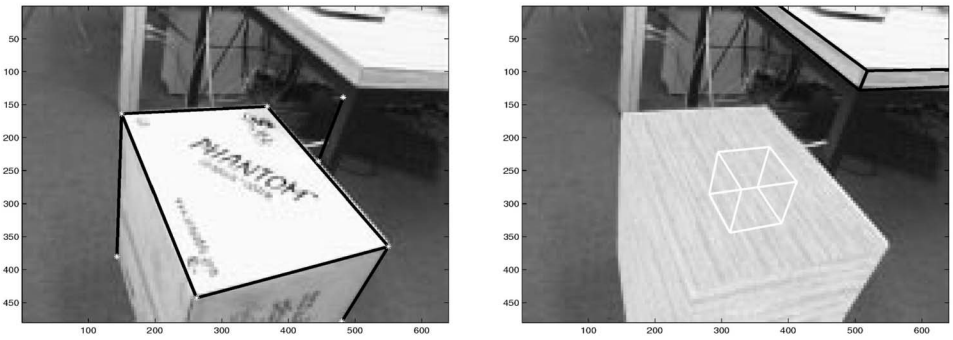
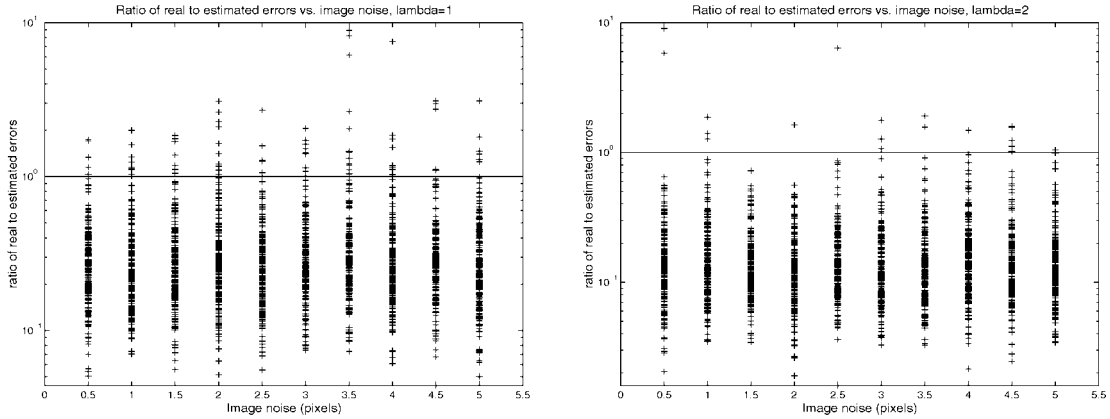


Fig. 10. **(Point Experiment 3).** We recover the coordinates of the nine coplanar points marked above using **NPL** and calculate the mean distance between recovered points and a best fit plane as $< 1\%$ of the size of the square defined by the nine points.



(a) (b)

Fig. 11. **(Line Experiment 1).** (a) Line segments used to estimate camera pose. (b) Texture is added to the faces of the real box. A virtual cube is placed on the box. The edges of the table are highlighted.



(a) (b)

Fig. 12. Ratio of the real error (computed using ground truth and recovered values) to error estimated from sensitivity analysis in depth recovery for **NPL**. (a) λ is set to 1. (b) λ is set to 2.

REFERENCES

[1] Y.I. Abdel-Aziz and H.M. Karara, "Direct Linear Transformation Into Object Space Coordinates in Close-Range Photogrammetry," *Proc. Symp. Close-Range Photogrammetry,* pp. 1-18, 1971.

[2] R.T. Azuma, "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments,* vol. 7, pp. 355-385, 1997.

[3] D. Dementhon and L.S. Davis, "Model-Based Object Pose in 25 Lines of Code," *Int'l J. Computer Vision,* vol. 15, pp. 123-144, 1995.

[4] M. Dhome, M. Richetin, J.T. Lapreste, and G. Rives, "Determination of the Attitude of 3-D Objects From a Single Perspective View," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, pp. 1265-1278, 1989.

[5] R.M. Haralick, "Pose Estimation From Corresponding Point Data," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 19, no. 6, pp. 1426-1446, 1989.

[6] O. Faugeras, *Three-Dimensional Computer Vision.* Cambridge, Mass.: M.I.T. Press, 1993.

[7] P.D. Fiore, "Efficient Linear Solution of Exterior Orientation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, pp. 140-148, 2001.

[8] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. ACM,* vol. 24, pp. 381-395, 1981.

[9] S. Ganapathy, "Decomposition of Transformation Matrices for Robot Vision," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 130-139, 1984.

[10] R. Haralick, C. Lee, K. Ottenberg, and M. Nölle, "Analysis and Solutions of the Three Point Perspective Pose Estimation Problem," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 592-598, June 1991.

[11] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle, "An Analytic Solution for the Perspective 4-Point Problem," *Computer Vision, Graphics, and Image Processing,* vol. 47, pp. 33-44, 1989.

[12] B.K.P. Horn, "Closed-Form Solution of Absolute Orientation Using Quaternions," *J. Optical Soc. Am. A,* vol. 4, pp. 629-642, 1987.

[13] B.K.P. Horn, "Relative Orientation," *Int'l J. Computer Vision,* vol. 4, pp. 59-78, 1990.

[14] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour, "Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices," *J. Optical Soc. Am. A,* vol. 5, pp. 1127-1135, 1988.

[15] T.S. Huang, A.N. Netravali, and H.H. Chen, "Motion and Pose Determination Using Algebraic Methods," *Time-Varying Image Processing and Moving Object Recognition,* Elsevier Science Publication, V. Cappellini, ed., no. 2, pp. 243-249, 1990.

[16] R. Kumar and A.R. Hanson, "Robust Methods for Estimaging Pose and a Sensitivity Analysis," *Computer Vision and Image Understanding,* vol. 60, pp. 313-342, 1994.

[17] R. Lenz and R.Y. Tsai, "Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, pp. 713-720, 1988.

[18] Y. Liu, T.S. Huang, and O.D. Faugeras, "Determination of Camera Location from 2-D to 3-D Line and Point Correspondences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, pp. 28-37, 1990.

[19] D.G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, pp. 441-450, 1991.

[20] C.-P. Lu, G. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation From Video Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, pp. 610-622, 2000.

[21] S. Maybank, *Theory of Reconstruction from Image Motion.* Springer-Verlag, 1993.
[22] N. Navab and O.D. Faugeras, "Monocular Pose Determination From Lines: Critical Sets and Maximum Number of Solutions," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 254-260, June 1993.
[23] W.H. Press, S. Teukolsky, W. Vetterling, and B.P. Flannery, *Numerical Recipes in C.* Cambridge, Cambridge Univ. Press, 1992.
[24] L. Quan and Z. Lan, "Linear N-Point Camera Pose Determination," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, pp. 774-780, 1999.
[25] G.W. Stewart and J.-G. Sun, *Matrix Perturbation Theory.* Boston, Mass.: Academic Press, Inc., 1990.
[26] E.H. Thompson, "Space Resection: Failure Cases," *Photogrammetric Record,* pp. 201-204, 1966.
[27] B. Triggs, "Camera Pose and Calibration from 4 or 5 Known 3D Points," *Proc. Int'l Conf. Computer Vision,* pp. 278-284, Sept. 1999.
[28] S. Yi, R.M. Haralick, and L.G. Shapiro, "Error Propagation in Machine Vision," *Machine Vision Assoc.,* vol. 7, pp. 93-114, 1994.

Adnan Ansar received the PhD degree in computer science in 2001, a Masters degree in computer science in 1998, a Masters degree in mathematics in 1993, and a Bachelors degree in physics and mathematics in 1993, all from the University of Pennsylvania. He is a member of Technical Staff in the Machine Vision Group at the Jet Propulsion Laboratory, California Institute of Technology. His current research interests are in camera calibration, multibaseline stereo, and vision-based spacecraft position and attitude determination.

Kostas Daniilidis received the PhD degree in computer science from the University Karlsruhe in 1992 and a Master's degree in electrical engineering from the National Technical University of Athens in 1986. He is an assistant professor of Computer and Information Science, University of Pennsylvania, affiliated with the interdisciplinary GRASP Laboratory. Prior to his current appointment, he was with the Cognitive Systems Group, University of Kiel, Germany. Currently, his research centers on omnidirectional vision and vision techniques for teleimmersion and augmented reality. Daniilidis was the chair of the IEEE Workshop on Omnidirectional Vision 2000. He is the cochair of the computer vision TC of the Robotics and Automation Society and has been reviewing for the main journals, conferences, and funding panels of computer vision. He received the 2001 Motor Company Award for the Best Penn Engineering Faculty Advisor. He is a member of the IEEE and IEEE Computer Society.