Linear Pose Estimation from Points or Lines

Adnan Ansar

Presently at:
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109

Formerly at:
GRASP Laboratory
University of Pennsylvania
34th Walnut Street
Philadelphia, PA 19104

and

Kostas Daniilidis

GRASP Laboratory
University of Pennsylvania
34th Walnut Street
Philadelphia, PA 19104

This work has been funded by NSF IIS-0083209, ARO/MURI DAAH04-96-1-0007, NSF CDS-97-03220, DARP-ITO-DABT63-99-1-0017, Penn Research Foundation, and Advanced Network and Services.

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 present a number of simulations which compare our results to two other recent linear algorithm as well as to iterative approaches. We conclude with tests on real imagery in an augmented reality setup. We also present an analysis of the sensitivity of our algorithms to image noise.

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 non-iterative 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 [20, 24]. The only similar non-iterative methods for an arbitrary number of points are those of Quan and Lan [22] and Fiore [6]. 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 [15].

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 [22]. 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 [6] 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 6 points unless they are coplanar. We show in Sect. 4.1 that our algorithm outperforms both of the other linear algorithms.

There are many closed form solutions to the 3 point problem, such as [4, 9], which return solutions with well understood multiplicities [14, 21]. Fischler and Bolles [7] extended their solution to 4 points by taking subsets and using consistency checks to eliminate the multiplicity for most point configurations. Horaud *et al.* [10] developed a closed form solution on 4 points which avoids this reduction to a 3 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 on gradient descent or Gauss-Newton methods. Typical of these approaches is the work of Lowe [18] and of Haralick [5]. There are also approaches which more carefully incorporate the geometry of the problem into the update step. For example, Kumar and Hanson [15] have developed an algorithm based on constraints on image lines using an update step adapted from Horn's [12] solution of the relative orientation problem. We compare this algorithm to our line algorithm in Sect. 4.1. There are several such variations using image line data. Liu *et al.* [17] use a combination of line and point data. Lu, Hager and Mjolsness [19] 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 Sect. 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 guarantees 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, 8] among many others. This is also the basis for the calibration technique of Lenz and Tsai [16]. 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 Sect. 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 projections 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 solutions 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 $\{\rho,\ x_{ij} \mid 1 \leq i \leq j \leq n\}$. 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 $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^{\mathbf{T}}$ 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 Σ. 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

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

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 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 $\lambda_1\mathbf{v}_1^L + \lambda_2\mathbf{v}_2^L + ... + \lambda_N\mathbf{v}_N^L = \rho = 1$ 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 $sgn(x_i) = sgn(x_1)sgn(x_{1i})$.

Before presenting our pose estimation algorithms, we briefly present a more formal treatment of our approach. Let HQ($\mathbb{R}^n$) and 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 $f(t_it_j) = t_{ij},\ f(1) = \rho$, 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 $V_0 = \mathrm{Ker(M)}$ 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 $x_{ij}x_{kl} = x_{i'j'}x_{k'l'}$ for $\{i', j', k', l'\}$ a permutation of $\{i, j, k, l\}$. Let e_1 be one such equation, and let 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 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 $\{e_j \mid j < i\}$, 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 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 Sect. 4.

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

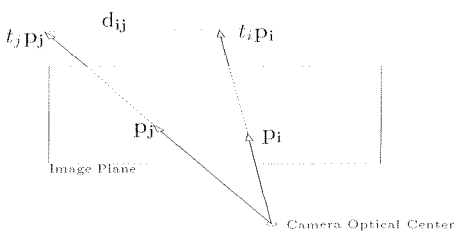

Fig. 1. The basic geometric constraint used in n point algorithm relates the distance between points in the world d_{ij} and the scale factors t_i and t_j associated with the projections $\mathbf{p}_i$ and $\mathbf{p}_j$.

$$\begin{aligned} b_{ij} &= (t_i\mathbf{p}_i - t_j\mathbf{p}_j)^T(t_i\mathbf{p}_i - t_j\mathbf{p}_j) \\ &= t_i^2\mathbf{p}_i^T\mathbf{p}_i + t_j^2\mathbf{p}_j^T\mathbf{p}_j - 2t_it_j\mathbf{p}_i^T\mathbf{p}_j \end{aligned} \tag{6}$$

Equation (6) 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 $\frac{n(n-1)}{2} \times (\frac{n(n+1)}{2} + 1)$ and a simple argument shows that it 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. It follows that 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 [11] or SVD of the cross-covariance matrix [13]. Given the rotation, translation between the two centroids is immediately recovered.

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 $\{s_i = (\bar{\alpha}_i, \mathbf{c}_i)\}$, where $\bar{\alpha}_i$ 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 $\bar{\alpha}_i$ normalized. Consider the point $\mathbf{d}_i$ on s_i which is closest to the origin of the image plane. Then $\mathbf{d}_i = \mathbf{c}_i - (\mathbf{c}_i^T\bar{\alpha}_i)\bar{\alpha}_i$. Let $\bar{\gamma}_i = \frac{\mathbf{d}_i}{\|\mathbf{d}_i\|}$. It follows that $\bar{\gamma}_i^T\bar{\alpha}_i = 0$ so that $\{\bar{\alpha}_i, \bar{\gamma}_i\}$ 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

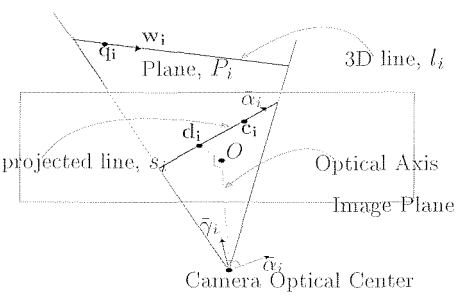

Fig. 2. Geometric constraint used in n line algorithm. The plane P_i determined by the line l_i and the optical center is spanned by $\{\bar{\alpha}_i, \bar{\gamma}_i\}$. Thus, $\mathbf{w}_i = \mathbf{R}_i\mathbf{v}_i$ can be written as a linear combination of these two vectors.

it as $\mathbf{w}_i = (\mathbf{w}_i^T\bar{\alpha}_i)\bar{\alpha}_i + (\mathbf{w}_i^T\bar{\gamma}_i)\bar{\gamma}_i$. Substituting $\mathbf{w}_i = \mathbf{R}\mathbf{v}_i$ we obtain $\mathbf{R}\mathbf{v}_i = (\mathbf{R}\mathbf{v}_i^T\bar{\alpha}_i)\bar{\alpha}_i + (\mathbf{R}\mathbf{v}_i^T\bar{\gamma}_i)\bar{\gamma}_i$. From this we develop a set of quadratic equations in the entries of $\mathbf{R}$ to to obtain a system of the form (1) and directly recover the

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

$$K_{i,j} = \\ [(\mathbf{R}\mathbf{v}_i^T \bar{\alpha}_i)\bar{\alpha}_i + (\mathbf{R}\mathbf{v}_i^T \bar{\gamma}_i)\bar{\gamma}_i]^T [(\mathbf{R}\mathbf{v}_j^T \bar{\alpha}_j)\bar{\alpha}_j + (\mathbf{R}\mathbf{v}_j^T \bar{\gamma}_j)\bar{\gamma}_j] \tag{7}$$

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

$$\mathbf{R}\mathbf{v}_i \times \mathbf{R}\mathbf{v}_j = \\ [(\mathbf{R}\mathbf{v}_i^T \bar{\alpha}_i)\bar{\alpha}_i + (\mathbf{R}\mathbf{v}_i^T \bar{\gamma}_i)\bar{\gamma}_i] \times [(\mathbf{R}\mathbf{v}_j^T \bar{\alpha}_j)\bar{\alpha}_j + (\mathbf{R}\mathbf{v}_j^T \bar{\gamma}_j)\bar{\gamma}_j] \tag{8}$$

Observe that (7) and (8) do not enforce the requirement that $\mathbf{R} \in SO(3)$. We accomplish this using the 12 quadratic constraints derived from

$$\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{9}$$

Note that in general, there are only 6 independent constraints in (9), but by employing our linearization procedure, we introduce more relations on the 45 linearized terms $\{r_{ij} = r_i r_j\}$, where $\{r_i\}$ are the 9 entries in $\mathbf{R}$. Using (7), (8) and (9), 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,

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

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.

3 Sensitivity Analysis

We analyze the sensitivity of our algorithms and show that the error in our solutions is bounded by the error in the image data. For basic definitions and information on matrix perturbation theory, see [23]. We consider first 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. Let $|\cdot|_F$ indicate the Frobenius norm and $|\cdot|$ the 2-norm. Suppose that $\mathbf{M}$ in (3) is a perturbation of the real system $\tilde{\mathbf{M}}$ due to noise, with $\tilde{\mathbf{M}} = \mathbf{M} + \mathbf{M}_e$. Since M is derived from polynomials in image measurements, a bound on image noise implies that we can bound $|\mathbf{M}_e|$. Suppose that the real physical solution to the pose problem is given by some $\tilde{x} = \bar{x} + \bar{x}_e$, where $\bar{x}$ solves the perturbed system (i.e. $\mathbf{M}\bar{x} = 0$) with $\tilde{\mathbf{M}}\tilde{x} = 0$. It is our goal to bound $|\bar{x}_e|$.

Expanding out the expression for the unperturbed system results in $\mathbf{M}\bar{x} + \mathbf{M}\bar{x}_e = -\mathbf{M}_e(\bar{x} + \bar{x}_e)$. Using $\mathbf{M}\bar{x} = 0$ and multiplying by $\mathbf{M}^\dagger$, the pseudoinverse of $\mathbf{M}$, we obtain $\mathbf{M}^\dagger\mathbf{M}\bar{x}_e = -\mathbf{M}^\dagger\mathbf{M}_e(\bar{x} + \bar{x}_e)$.

For the case of five or more lines, recall that $\mathbf{M}$ has full column rank. It follows that $\mathbf{M}^\dagger \mathbf{M} \bar{x}_e = \bar{x}_e$. Now, applying simple properties of the norm to both sides, we obtain $|\bar{x}_e| \leq |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F (|\bar{x}| + |\bar{x}_e|)$. We can either ignore the quadratic term $|\mathbf{M}_e|_F |\bar{x}_e|$, or use the highly conservative estimate that $|\bar{x}_e| \leq |\bar{x}|$ for any reasonable situation. Thus, for $1 < \lambda \leq 2$

$$|\bar{x}_e| \leq \lambda |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F |\bar{x}|$$

The point case is complicated by the fact that $\mathbf{M}$ is rank deficient. We write $\bar{x}_e = \bar{x}_n + \bar{x}_p$, where $\bar{x}_p \in \mathrm{K} = \mathrm{Ker}(M)$ and $\bar{x}_n$ is orthogonal to K. Then applying the procedure above, we find that for points,

$$|\bar{x}_n| \leq \lambda |\mathbf{M}^\dagger|_F |\mathbf{M}_e|_F |\bar{x}| \tag{10}$$

but we have no constraint on $\bar{x}_p$. In order to bound this component of the error, we must use the fact that both $\bar{x}$ and $\tilde{x}$ must lie on the variety above if they are valid solutions, whether perturbed or not. We write the ij component of $\bar{x}_p$ as $\hat{x}_{ij}$ and of $\bar{x}_n$ as $\check{x}_{ij}$, so that the ij component of $\bar{x}_e$ is $\hat{x}_{ij} + \check{x}_{ij}$. Using the notation for inner products from section 2.2, we state that

$$-2p_{ij}\hat{x}_{ij} + p_{ii}\hat{x}_{ii} + p_{jj}\hat{x}_{jj} = 0 \tag{11}$$

This is a consequence of the simple form $\mathbf{M}$ takes in the point case. [1] Consider now the relation $x_{ii}x_{jj} = x_{ij}^2$ which must be satisfied by $\tilde{x}$. Substituting approriate terms results in

$$(x_{ii} + \hat{x}_{ii} + \check{x}_{ii})(x_{jj} + \hat{x}_{jj} + \check{x}_{jj}) = (x_{ij} + \hat{x}_{ij} + \check{x}_{ij})^2 \tag{12}$$

If we now intersect this quadric in $\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{ij}$ with the plane described by (11), we obtain a conic in $\hat{x}_{ii}, \hat{x}_{jj}$. Using the same procedure we can find two other conics in $\hat{x}_{ii}, \hat{x}_{kk}$ and $\hat{x}_{jj}, \hat{x}_{kk}$ and intersect them all to obtain a discrete set of solutions for $\hat{x}_{ii}, \hat{x}_{jj}, \hat{x}_{kk}$, hence $\hat{x}_{ij}, \hat{x}_{ik}, \hat{x}_{jk}$ using (11). The depend only on $\bar{x}$, $\{p_{ij}\}$ and the bounded $\bar{x}_n$. An alternative approach is to ignore second order terms in error the quadric (12) to obtain

$$x_{ii}\hat{x}_{jj} + xjj\hat{x}_{ii} - 2x_{ij}\hat{x}_{ij} \approx 2x_{ij}\check{x}_{ij} - (x_{ii}\check{x}_{jj} + x_{jj}\check{x}_{ii}) \tag{13}$$

Now, the intersection of the plane defined by (13) with (11) results in a line in $\hat{x}_{ii}, \hat{x}_{jj}$. We find two other lines (as with the conics above), and solve. In this case, we can write down an explicit solutions. Using ii, jj, kk, we have

$$\hat{x}_{ii} = \frac{f_{ij}b_{ij}b_{jk} + f_{ik}b_{ij}b_{jk} - f_{jk}b_{ij}b_{ik}}{a_{ij}a_{jk}b_{ik} + a_{ik}b_{ij}b_{jk}} \tag{14}$$

with

$$\begin{aligned} a_{ij} &= x_{jj} - x_{ij}\frac{p_{ii}}{p_{ij}} \\ b_{ij} &= x_{ii} - x_{ij}\frac{p_{jj}}{p_{ij}} \\ f_{ij} &= 2x_{ij}\check{x}_{ij} - (x_{ii}\check{x}_{jj} + x_{jj}\check{x}_{ii}) \end{aligned}$$

[1] Proof omitted because of space constraints; reference omitted for anonymity purposes.

Since these relations must hold, for all integers i, j, k, we select the smallest $\hat{x}_{ii}$ in absolute value.

Thus, we see by inspections of (14) that up to first approximation, $|\bar{x}_p|$ is of the same magnitude as $|\bar{x}_n|$, which is already bounded by (10).

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 6 or more points by SVD methods. We use a triangle ($\triangle$) to indicate this algorithm on all graphs.

F The n point linear algorithm of Fiore [6]. We signify this by a square($\square$).

QL The n point linear algorithm of Quan and Lan [22]. We signify this by a diamond ($\Diamond$).

LHM The iterative algorithm of Lu, Hager and Mjolsness [19] initialized at ground truth. We signify this by a circle ($\circ$) 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 [15]. We initialize **KH** at the ground truth translation and rotation (**KHRT** signified by $\triangle$) and at ground truth translation and identity rotation (**KHT** signified by $\square$).

4.1 Simulation

All simulations are performed in MATLAB. We assume calibrated virtual cameras with effective focal length (diagonal terms in calibration matrix) 1500 in the point case and 600 in the line case. We report errors in terms or relative rotation error and relative translation 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, $|\bar{q} - \bar{q}_r|$. 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 [25]. 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 6 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.

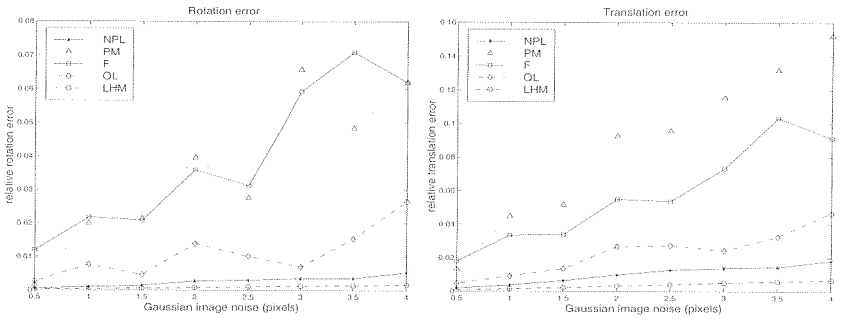

Fig. 3. (Point Simulation 1) Rotation and Translation errors for 6 points vs. noise level. We plot results for the five algorithms, **NPL**, **PM**, **F**, **QL**, **LHM**. Note that **NPL** outperforms all but the iterative **LHM** with ground truth initialization.

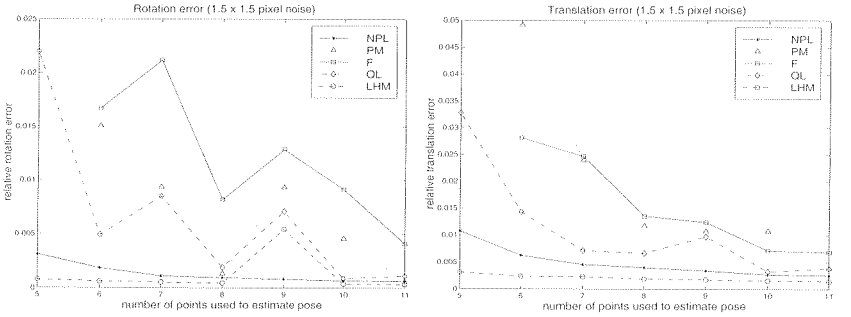

Fig. 4. (Point Simulation 2) Rotation and Translation errors vs. number of points used for pose estimation with 1.5×1.5 pixel Gaussian noise. We plot results for the five algorithms, **NPL**, **PM**, **F**, **QL**, **LHM**.We see that **NPL** outperforms all but the iterative **LHM** with ground truth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

Point Simulation 2 (Dependence on number of points): We demonstrate that all 5 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 5 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 5 points, since these algorithms require at least 6 points.

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 [25]. For each noise level, we generate 400 poses and 6 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. 5). 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 immediately see the advantage of having no initialization requirement for **NLL**.

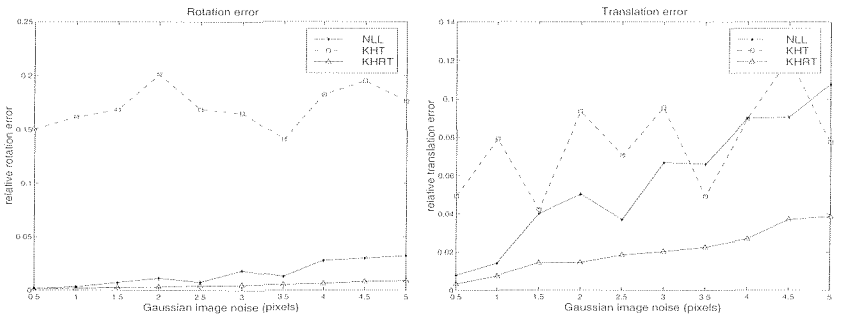

Fig. 5. (Line Simulation 1) Rotation and Translation errors vs. noise level for **NLL** and **KH**. We initialize **KH** at ground truth **R** and **T** (**KHRT**) to evaluate absolute performance and at ground truth **T** and $\mathbf{R} = \mathbf{I}$ (**KHT**) to demonstrate the advantage of requiring no initialization in **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 4 to 11 and with fixed noise of 1.5×1.5 pixels. We see in Fig. 6 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**.

4.2 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 [16]. All image processing was done offline using MATLAB. Note that the more computationally intensive point algorithm **NPL** can be run in real-time (> 30 Hz) on a 600 MHz PIII using the implementation of SVD from numerical recipes in C for up to 9 points 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 8 marked points in Fig. 7 by magnifying the relevant region and marking by

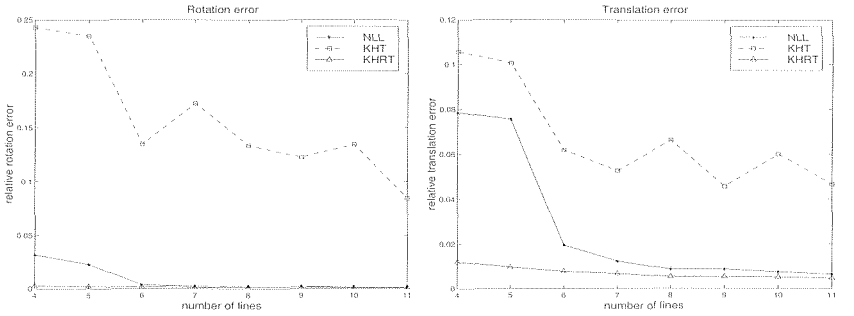

Fig. 6. (Line Simulation 2) Rotation and Translation errors vs. number of points for **NLL** and **KH**. Noise is fixed at 1.5×1.5 pixels. We initialize **KH** at ground truth $\mathbf{R}$ and $\mathbf{T}$ (**KHRT**) to evaluate absolute performance and at ground truth $\mathbf{T}$ and $\mathbf{R} = \mathbf{I}$ (**KHT**) to demonstrate the advantage of requiring no initialization in **NLL**.

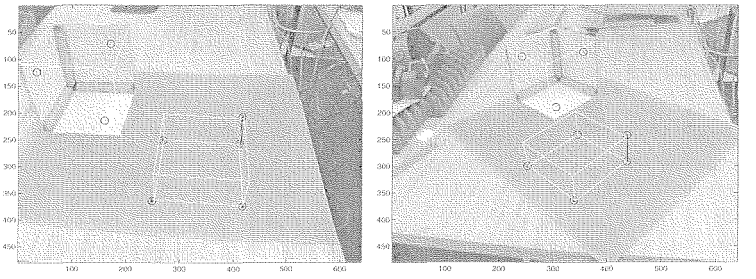

Fig. 7. (Point Experiment 1) Reprojection of a virtual box and three edges of a cube onto real-world reference objects. We estimate camera pose using the 8 circled points and NPL.

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 7 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.

Point Experiment 2: We repeat **Point Experiment 1** on a different scale. In Fig. 8, the box is approximately 18 inches on each side, and the camera is approximately 8 feet from the nearest corner of the box. We estimate pose from the 8 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. 9 we mark 9 points on the calibration grid in the image. The points have a uniform spacing of 8 inches. The camera is placed approximately 11 feet from the marked points. We recover the coordinates of the 9 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 in. with a standard deviation of 0.07 in. 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. 10(a), we indicate the 7 line segments used to estimate camera pose. In Fig. 10(b), 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 inserted into the images only after pose estimation and transformation to camera coordinates.

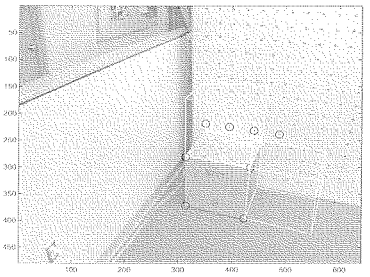

Fig. 8. (Point Experiment 2) Reprojection of 2 virtual boxes of dimensions identical to a real box. We estimate camera pose using the 8 circled points and **NPL**.

5 Conclusion

Our goal was to develop fast, 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. 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

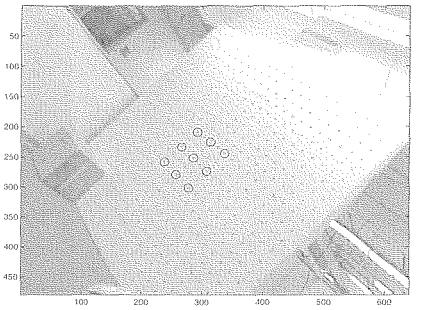

Fig. 9. (Point Experiment 3) We recover the coordinates of the 9 coplanar points marked above using **NPL** and calculate the mean distance between recovered points and a best fit plane as $< 1\%$ of the size of the square defined by the 9 points.

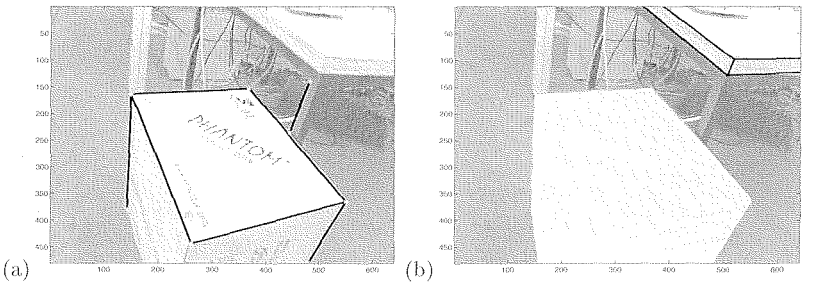

Fig. 10. (Line Experiment 1) (a) Line segments used to estimate camera pose. (b) Texture is added to the faces of the real box. A virtual cube is placed on the box. The edges of the table are highlighted.

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.

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*, pages 1–18, 1971.
2. R.T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 7:355–385, 1997.
3. D. Dementhon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–144, 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*, 11:1265–1278, 1989.

5. R.M. Haralick et al. Pose estimation from corresponding point data. *IEEE Trans. Systems, Man, and Cybernetics*, 19(6):1426–1446, 1989.
6. P.D. Fiore. Efficient linear solution of exterior orientation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23:140–148, 2001.
7. M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.
8. S. Ganapathy. Decomposition of transformation matrices for robot vision. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 130–139, 1984.
9. R. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 592–598, Maui, Hawaii, June 3-6, 1991.
10. 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.
11. B.K.P. Horn. Closed-form solution of absolute orientation using quaternions. *Journal Opt. Soc. Am. A*, A4:629–642, 1987.
12. B.K.P. Horn. Relative orientation. *International Journal of Computer Vision*, 4:59–78, 1990.
13. B.K.P. Horn, H.M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal Opt. Soc. Am. A*, A5:1127–1135, 1988.
14. T.S. Huang, A.N. Netravali, and H.H. Chen. Motion and pose determination using algebraic methods. In V. Cappellini, editor, *Time-Varying Image Processing and Moving Object Recognition, 2*, pages 243–249. Elsevier Science Publication, 1990.
15. R. Kumar and A. R. Hanson. Robust methods for estimaging pose and a sensitivity analysis. *Computer Vision and Image Understanding*, 60:313–342, 1994.
16. 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*, 10:713–720, 1988.
17. 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*, 12:28–37, 1990.
18. D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13:441–450, 1991.
19. C.-P. Lu, G. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:610–622, 2000.
20. S. Maybank. *Theory of Reconstruction from Image Motion*. Springer-Verlag, Berlin et al., 1993.
21. N. Navab and O.D. Faugeras. Monocular pose determination from lines: Critical sets and maximum number of solutions. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 254–260, New York, NY, June 15-17, 1993.
22. L. Quan and Z. Lan. Linear n-point camera pose determination. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21:774–780, 1999.
23. G.W. Stewart and Ji g. Sun. *Matrix Perturbation Theory*. Academic Press, Inc., Boston, MA, 1990.
24. E.H. Thompson. Space resection: Failure cases. *Photogrammetric Record*, X:201–204, 1966.
25. S. Yi, R.M. Haralick, and L.G. Shapiro. Error propagation in machine vision. *MVA*, 7:93–114, 1994.