Robust Estimation of Camera Location and Orientation From Noisy Data With Outliers

Rakesh Kumar and Allen Hanson

COINS TR 89–120

December 1989

Robust Estimation of Camera Location and Orientation from Noisy Data with Outliers

Rakesh Kumar and Allen R. Hanson

Computer and Information Science Dept.
Graduate Research Center
University of Massachusetts at Amherst
MA. 01003.
Phone: (413) 545-1519
EMAIL: kumar@cs.umass.EDU

December 11, 1989

[1]This research was supported by the following Defense Advanced Research Projects Agency grants F30602-87-C-0140, DACA76-85-C-0008, DACA76-86-C-0015 and National Science Foundation grant DCR8500332.

Abstract

This paper describes a solution and mathematical analysis of the problem of estimating camera location and orientation from a set of recognized landmarks appearing in the image, sometimes refered to as pose determination. The landmarks we use are real or virtual 3D lines represented in a world coordinate system. Given correspondences between these 3D lines and 2D lines found in the image, the goal is to find the Rotation and Translation which map the world coordinate system to the camera coordinate system. Some of these correspondences may be wrong (outliers). Our algorithm can handle up to 49.9 % outliers or gross errors in the data. The camera model assumes perspective projection.

We develop three algorithms, "R_then_T", "R_and_T" and "Med_R_and_T" to estimate the camera location and orientation. Algorithms "R_then_T" and "R_and_T" minimize the sum of squares of their error measures over all lines. Algorithm "R_then_T" solves for rotation first and uses the result to solve for translation. It is a variation of an algorithm developed by Liu, Huang and Faugeras [23]. The second algorithm "R_and_T" solves for both rotation and translation, simultaneously. The results from the second algorithm are much better that those of the first. However, both "R_then_T" and "R_and_T" are sensitive to outliers. Algorithm "Med_R_and_T" is robust with respect to outliers. It minimizes over all lines the median of the square of the same error measure used by the "R_and_T" algorithm. We also discuss the performance of our algorithms with respect to other error measures and robust algorithms.

A closed form expression is developed for the uncertainty in the output parameters as a function of the variance of the noise in the input parameters. Based on this analysis, statements are made about the kind of errors to expect in different situations. The algorithms have been tested over hundreds of simulated noisy data experiments and also over both indoor and outdoor real image sequences.

1 Introduction

This paper describes a solution and mathematical analysis of the problem of estimating camera location and orientation from a set of recognized landmarks appearing in the image. We have developed algorithms for handling both 3D line and 3D point landmarks. Since the point and line alogorithms are very similar, to save space we will shall only briefly discuss the point algorithms. The line landmarks employed are real or virtual 3D lines represented in a world coordinate system. Given correspondences between these 3D lines and 2D lines found in the image, the goal is to find the Rotation and Translation which map the world coordinate system to the camera coordinate system. We assume that correspondences established between model and data are line correspondences and not endpoint correspondences. Some of these correspondences may be wrong. Our algorithm can handle up to 49.9 % outliers or gross errors in the data. The camera model assumes perspective projection. In addition, intrinsic camera parameters, such as focal length, field of view, center of the image, size of image etc. are assumed to be known [5,19,30].

We are interested in applying our algorithm to aid in the navigation of a robot moving in a known outdoor environment. Among the real image data results which we present will be situations where the landmarks are on the order of hundreds of feet distant from the camera.

A mathematical analysis of an uncertainty measure is developed, which relates the variance in the output parameters to the noise present in the input parameters. For this analysis, we assume that there is no noise in the 3D model data and the only input noise is in the image data.

1.1 Previous Work

The problem of "determination of camera location and orientation" has been referred to by various other names including, "exterior orientation", "pose determination", "pose refinement", "perspective inversion" and "model alignment". We prefer the first name and will henceforth refer to it by its abbreviated form, "camera location determination". There have been many papers on camera location determination, but most assume point data is available and only a few have presented techniques for line data. Most solutions are also iterative in nature and require an initial estimate. They generally minimize the sum of squares of an error estimate and are sensitive to gross errors or outliers in the data. There are only a few techniques published which claim to handle data with outliers [7,11,22].

Fischler and Bolles [7] assume point data and solve for the "legs" of the points (the lengths of rays from the optical center of the camera to the points in 3D space). The closed form solution they present is quite complex and involves solving a quartic equation iteratively. Their technique is one of the very few which attempts to deal with the "outlier" problem, i.e. situations where gross errors are present and smoothing by least squares will not work. Recently, Linnainmaa et. al. have come up with a generalized hough transform approach to find the coordinates of the 3D points in camera coordinates [22]. Horaud [14] has devised a closed form solution for the case of 3 line segments meeting at a point. In his case too, an quartic equation must be solved.

Lowe [24] presents iterative techniques for both point and line data. However, he does not assume perspective projection and therefore his solution is not applicable to our problem of camera location determination in outdoor scenes. His technique is similar to that of Wolf [31] appearing in the photogrammetry literature. Ganapathy [9] presents a linear closed form solution for point data. Besides solving for the rotation and translation parameters, he also solves for the center of the image and scaling along "x" and "y" directions in the image. We have an implementation of his technique and find it extremely susceptible to noise, probably due to his linear least squares minimization where he assumes all his parameters are independent when they are not. Recently, Faugeras et. al. [5] have come up with a technique to solve a similar system of equations with appropriate constraints. Here it is important to draw the distinction between techniques for "camera calibration" [5,19,30], also called "interior orientation" versus the techniques for "camera location determination". Camera calibration techniques solve for intrinsic camera parameters along with the rotation and translation. The techniques for camera calibration require very precise image measurements and are less tolerant to noise. Camera location determination techniques are less susceptible to image noise but one needs to know the intrinsic camera parameters accurately.

Liu, Huang and Faugeras present a solution to the "camera location determination" problem which works for both point and line data [23]. Our work is based on the constraints formulated by them. Their constraint uses the fact that the 3D lines in the camera coordinate system must lie on the projection plane formed by the corresponding image line and the optical center. Using this fact, constraints for rotation can be separated from those of translation. They first solve for the rotation and then use the rotation result to solve for the translation. They suggest two methods to solve for the rotation constraint. In the first method, they represent the rotation as an orthonormal matrix and devise an eigenvalue solution. However, they do not enforce the six orthonormality constraints for an orthonormal matrix. It is not clear how they would find the nearest orthonormal matrix to the matrix their algorithm returns, and whether they then would have a solution to the earlier problem. The second method represents rotation by Euler angles, and is a non-linear iterative solution obtained by linearizing the problem about the current estimate of the output parameters. The translation constraint is solved by a linear least-squares method.

Recently, Worrall et. al. [32] came up with a least square technique which minimizes the same error measure as our "R_and_T" algorithm. However, they represent rotations as

Euler angles and use a different non-linear technique from that presented here. They also do not handle outliers or provide a mathematical analysis of the errors. Note that our work was done independently of theirs.

Robust Statistics techniques are currently gaining popularity in Computer Vision [2,8,11,16]. Traditionally, least square techniques have been used for regression analysis or model fitting. Least squares is optimum and reliable when the underlying noise in the data is gaussian. However, when outliers are present in the data, the gaussian assumption is violated and the least squares result is skewed in order to make the data approximate a gaussian. Because of the skewing of the result, trying to detect outliers by thresholding on the residual errors of each line will not work. Throwing away one line at a time and doing least squares on the remaining subset also does not work when more than one outlier is present. To handle outliers, statisticians have suggested many different "robust" techniques [8,10,15,28]. Most of this work has been for linear problems. A measure to analyze these "robust" algorithms is the breakdown point : the smallest fraction of outliers present in the input data, which may cause the output estimate to be arbitrarily wrong. Algorithms based on minimizing L1, L2 or Ln error measures have breakdown points of 1/n where "n" is the number of data items. Standard robust statistical procedures can be classified as M-estimates (Maximum likelihood type estimates), L-estimates (linear combination of order statistics) and R-estimates (estimates based on rank transformations). Most of these techniques have been shown to have breakdown points of 1/(p+1) or lower [21,28] where "p" is the number of unknowns (p = 6 for the "camera location determination" problem). Finally, there are outlier detection techniques based on the diagonal entries of the "Hat" matrix, Mahalanobis distance etc. [8]; these generally work for only certain kind of outliers and often cannot handle more than one outlier.

Haralick and Joo [11] present both least square and robust algorithms for the "camera location determination" problem using point data. Their robust algorithm uses an

M-estimate technique based on Huber's work [15]. They claim in their paper to be able to handle between 20 to 30 % outliers in the input data. This is slightly higher than the upper bound of 1/(p+1) as noted above (where p = 6). Their robust technique is based on linearizing the error function about a current estimate and then reweighting the contribution of each point to the linear least square solution based on the residual error value of that point. They use Huber and Tukey's "ψ" functions on the residual error for reweighting. One consequence of this may be that the initial estimate of the pose must be fairly close to the final solution or else the wrong lines may get higher weights.

1.2 Our approach

One of the results of this paper is that the decomposition of the solution into the two stages of solving first for rotation and then translation does not use the set of constraints effectively. This same observation was made by other researchers working on the structure from motion problem [4]. The rotation and translation constraints, when used separately, are very weak constraints. When solving for them separately, even small errors in the rotation stage get amplified to large errors in the translation stage. This is particularly true with the large distances of the landmarks from the camera in our application. If we solve for them simultaneously, we get much better noise immunity.

We use the same constraints as Liu, Huang and Faugeras, but a different non-linear technique. The technique we use was adapted from one used by Horn [13] to solve the problem of relative orientation (similar to structure from motion). We believe that the application of Horn's technique gives us much better convergence properties than their solution using Euler angles. With Horn's technique an implementation has been developed where a solution for rotation is obtained first, and then is used to solve for translation. We call this algorithm "R_then_T". Again using Horn's technique, another algorithm ("R_and_T") was developed to solve for the rotation and translation simultaneously. We also discuss using other error functions based on similar constraints for minimization. Algorithm "R_and_T" gives the best performance in all cases. Both "R_and_T" and "R_then_T" minimize the sum of squares of their error measures and are non-robust. "R_and_T" and "R_then_T" need initial estimates of the rotation and translation. If the user is unable to provide these initial estimates we give techniques where the rotation space is sampled to provide initial estimates and the solution with the minimum error from these sample runs is picked as the final output.

The algorithm "Med_R_and_T" minimizes the Median of the square of the error over all lines or the LMS (least median of squares) estimate. It is based on a robust algorithm by Rousseeuw [28]. LMS algorithms have been proven to have a 49.9 % breakdown point. The outliers can be arbitrarily large. Also, unlike the M-estimate algorithms, the distance (for successful convergence) of the initial estimate from the final result is not affected.

Liu, Huang and Faugeras extend their technique to point data by drawing virtual lines between pairs of points [23]. They use the same rotational constraint; however, the translation constraint is different from that of lines. The techniques and mathematical analysis developed here apply equally well to 3D/2D point data. Although the algorithms presented here are for lines, we have also developed equivalent algorithms for point data. In section 3, we give the objective functions which are to be minimized for the point case. These algorithms can be extended to deal with combinations of point and line data. We will make the following comments about using point data. Firstly for points too, we find that solving rotation and translation simultaneously instead of separately (as they propose) gives much better results. Another observation we make is that a point algorithm using "n" points seems to be more robust than a line algorithm using "n" lines. The results for both points and lines depends on the particular data set one has. The point algorithm returns better results chiefly because the results of the first stage, i.e. the rotation stage, are much better. Intuitively, this is because using "n" points we can draw $O(n^2)$ lines.

1.3 Minimum number of lines

Both rotation and translation in the 3D world can be represented by three parameters each. Each line or point data gives us 2 constraints. Thus, a minimum of three lines or points are needed. However, in many cases, with three lines or points, there is no unique solution. If the three lines are parallel in 3D space or lie on the same projection plane, then an infinite number of solutions can be found. If the three lines meet at a common point in 3D space, then we can get two solutions for rotation (the Necker cube phenomena) and an infinite number of solutions for translation.

Fischler and Bolles [7] provide a geometric construction, where there could be up to four solutions for 3 points or lines lying in a 3D plane. The same construction can be used to demonstrate more than one solution for cases of four and five points lying in a plane. Given a solution, they demonstrate that another solution can be constructed if the two normals drawn to a side of the triangle (formed by the 3 points or lines) from the optical center and the opposite vertex meet at a common point on the side. In this manner, for each side, we could possibly construct another solution therefore getting a maximum of four possible solutions.

The rest of the paper is divided as follows: in section 2, we discuss the geometry of perspective projection and the rotation and translation constraints. In section 3 the least squares problem is set up. We present the least squares non linear technique and methods for situations when there is no good initial estimate. Section 4 provides the least median squares algorithm. Uncertainty analysis is done in section 5 and section 6 has results and discussions for all three algorithms. Appendix A discusses various representations for rotation and motivates our particular choice of quaternions for large rotations and the 3D rotation vector for small angles.

2 Rotation and Translation Constraints

The rigid body transformation from the world coordinate system to the camera coordinate system can be represented as a rotation (R) followed by a translation (T). The i'th point p_i in world coordinates gets mapped to the point p_{ci} in camera coordinates. Lines in 3D can be represented by two points p_i^R and p_i^L or a point p_i and a direction d_i. The mapping is represented by the following equation:

$$p_{ci} = R(p_i) + T \tag{1}$$

In the above equation, except for the rotation R, all the other terms are column vectors with 3 components each. We refer to the components by the subscripts x, y and z. R is the rotation operator and can be expressed in many ways, e.g. orthonormal matrices, quarternions, axis and angle, etc. We discuss the various representations for rotation in the appendix. Fig. 1 shows the camera and world coordinate systems. X_w, Y_w and Z_w represent the axes of the world coordinate system. O is the optical center of the lens and the origin of the camera coordinate system $OX_cY_cZ_c$. OZ_c is the optical axis. In equation (1) the translation vector "T" represents the location of the origin of the world coordinate system in camera coordinates. Equation (1) can be rewritten to map points in the camera coordinate system to the world coordinate system:

$$p_i = R^T(p_{ci}) + T_w \tag{2}$$

In the above equation (2) R^T is the inverse of the rotation operator (transpose if rotation is expressed as an orthonormal matrix) in equation (1). "T_w" represents the location of the origin of the camera coordinate system in world coordinates. "T_w" is related to "T" by the following equation:

$$T_w = -R^T(T) \tag{3}$$

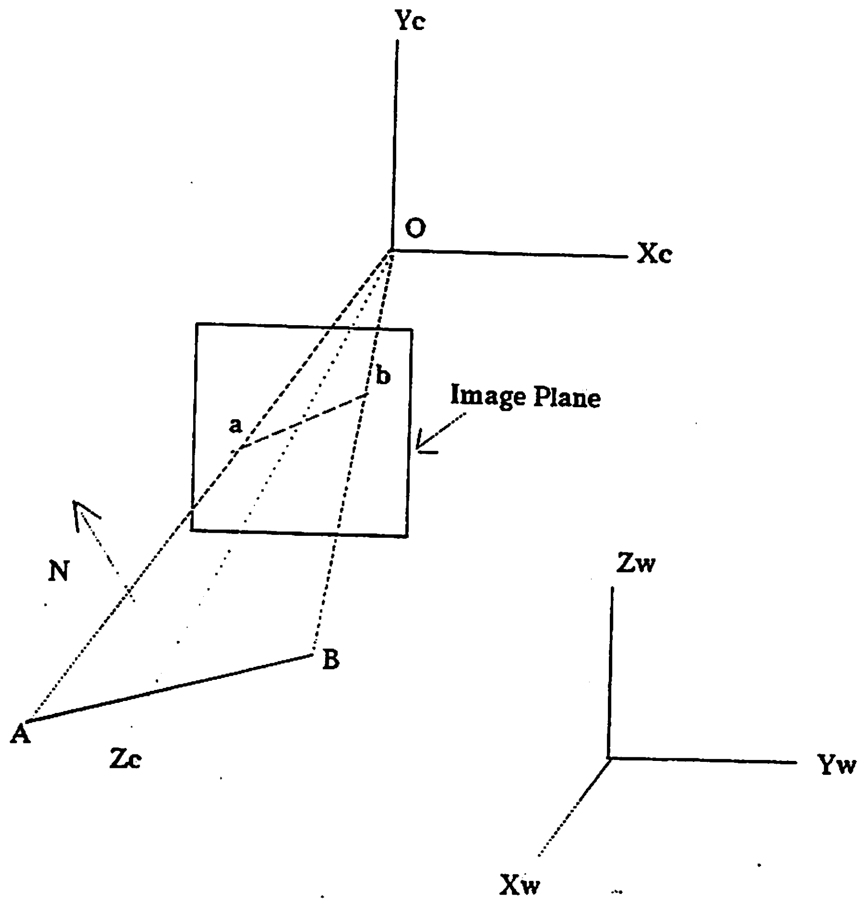

Figure 1. Camera and world coordinate system (perspective projection).

The 3D line (Fig. 1) "AB" projects to the image line "ab". A 3D point p_{ci} projects to an image pixel I_i by the following equations:

$$I_{ix} = s_x \frac{p_{cix}}{p_{ciz}} \qquad I_{iy} = s_y \frac{p_{ciy}}{p_{ciz}} \tag{4}$$

where s_x and s_y are the scale factors along the "X" and "Y" directions respectively. They are related to the field of view angles ϕ_x, ϕ_y and the image size N_r (number of rows or columns, assuming a square image) :

$$s_x = \frac{N_r}{2}\cot(\frac{\phi_x}{2}) \quad s_y = \frac{N_r}{2}\cot(\frac{\phi_y}{2})$$

A line in the image plane can be represented in (ρ, θ) parameters by the following equation:

$$I_{ix}\cos\theta_i + I_{iy}\sin\theta_i = \rho_i \tag{5}$$

Subsituting I_{ix} and I_{iy} from equation (4) into equation (5) we get the equation of the projection plane formed by the image line and the optical center :

$$(s_x\cos\theta_i)p_{cix} + (s_y\sin\theta_i)p_{ciy} - \rho_i p_{ciz} = 0 \tag{6}$$

In Fig. 1, the projection plane formed by the image line "ab" is given by the plane "Oab" and the 3D line "AB" must lie in this plane. "N" is the normal to the projection plane, given by the vector N_i:

$$N_i = (s_x\cos\theta_i, s_y\sin\theta_i, -\rho_i)^T \tag{7}$$

N_i can be normalized to be a unit vector and henceforth we shall assume that N_i is the unit normal vector to the projection plane.

The rotation constraint for lines, formulated by Liu, Huang and Faugeras [23], is that the 3D line must lie in the projection plane formed by its corresponding image line is:

$$N_i \cdot R(d_i) = 0 \tag{8}$$

We noted above that a rigid body transformation can be represented as a rotation followed by a translation. The translation does not change the direction of the line. Therefore, the direction of the 3D line after rotation must be perpendicular to the normal of the projection plane of the image line.

The translation constraint formulated by Liu et. al. uses the fact that any point on the 3D line in camera coordinates must lie on the projection plane. The vector formed from the origin (optical center) to this point must be perpendicular to the normal of the projection plane. Note that we can choose any point p_i on the 3D line. This can be expressed as follows:

$$N_i \cdot (R(p_i) + T) = 0 \quad or \quad N_i \cdot T = -N_i \cdot R(p_i) \tag{9}$$

At this point, we would like to make clear the first two algorithms we have developed and will be comparing in this paper. In the first algorithm "R_then_T" we solve for rotation using the constraint in equation (8). Then the rotation result returned from this step is used in conjunction with equation (9) to solve for translation.

In the second algorithm "R_and_T", only equation (9) is used to solve for both rotation and translation simultaneously. For each line, two points are used which must satisfy equation (9). As the tables in our results section will show, "R_and_T" performs much better then "R_then_T".

3 Least Square Solution methods

Ideally we would like to find the rotation "R" and translation "T" by which equation (9) is satisfied for each line. With noise, however, this will not be possible. In the "R_and_T" case, the objective function "E_1" we minimize is given by :

$$E_1 = \sum_{i=1}^{2n} (N_i \cdot (R(p_i) + T))^2 \tag{10}$$

For each image line two 3D points are used and therefore, each line contributes twice to the objective function. A physical interpretation of the objective function above is that it is the sum-of-the-squares of the perpendicular distances from the end-points of the 3D lines to their corresponding projection planes formed using the image lines. We minimize "E_1" to find an "R" and "T" such that the end-points of the 3D lines are mapped as close as possible (in terms of sum of squares of perpendicular distances) to their corresponding projection planes.

The above objective function E_1 gives higher weighting to endpoints of 3D lines which are further away from the camera. In order to give equal weighting to all endpoints, the following objective function is to be minimized :

$$E_2 = \sum_{i=1}^{2n} \left(\frac{N_i \cdot (R(p_i) + T)}{\mid R(p_i) + T \mid} \right)^2 \tag{11}$$

E_2, unlike E_1, is a rational function and therefore more difficult to optimize. To minimize E_2, at each iteration in our non-linear technique, we hold the denominator $\mid R(p_i) + T \mid$ for each point to be constant. In the next iteration, the value of $\mid R(p_i) + T \mid$ is updated with the new "R" and "T". Therefore, we are able to employ the same algorithm as used for E_1. This seems to work for all the cases we have run our algorithm on.

In contrast to E_1 and E_2, we could construct an objective function E_3, which minimizes the sum-of-the-squares of the perpendicular distances of the end-points of the image lines to the projection plane formed using the 3D line and the optical center.

$$E_3 = \sum_{i=1}^{n} \sum_{j=1}^{2} \left(\frac{p_{i1} - T_w}{\mid p_{i1} - T_w \mid} \times \frac{p_{i2} - T_w}{\mid p_{i2} - T_w \mid} \cdot (R^T I_{i,j}) \right)^2 \tag{12}$$

where T_w and R^T are the translation and rotation in the world coordinate system (2). We minimize E_3 by a method similar to the techniques used for E_1 and E_2. We found the performance of algorithm optimizing E_3 to be poorer, with respect to noise, than algorithms minimizing E_1 and E_2. We suspect this is because the numerator of E_3 is a fourth order function of "R" and "T" as compared to the second order numerators of E_1 and E_2.

In the "R_then_T" case, the rotation objective function E_R and translation objective function E_T are :

$$E_R = \sum_{i=1}^{n}(N_i \cdot R(d_i))^2 \qquad E_T = \sum_{i=1}^{n}(N_i \cdot (R(p_i) + T))^2 \tag{13}$$

We first find the "R" that minimizes E_R and then use that "R" to find a "T" which minimizes E_T.

For point data, we could draw virtual lines between all pairs of points and then use algorithms "R_then_T" or "R_and_T". We however minimize the following error expression:

$$E_{p1} = \sum_{i=1}^{n}(((s_x, 0, -I_{ix}) \cdot (R(p_i) + T))^2 + ((0, s_y, -I_{iy}) \cdot (R(p_i) + T))^2) \tag{14}$$

The two vectors $(s_x, 0, -I_{ix})$ and $(0, s_y, -I_{iy})$ form a basis for the plane normal to the projection ray from the optical center to the image point I_i. E_{p1} represents the sum of squares of the distance of the 3D points p_i to the point where their corresponding projection rays pierce the plane which passes through p_i and is normal to the projection ray.

Again, to give equal weightage to all points irrespective of their distance to the optical center, we can normalize the above objective function as before:

$$E_{p2} = \sum_{i=1}^{n}(((s_x, 0, -I_{ix}) \cdot \frac{(R(p_i) + T)}{|(R(p_i) + T)|})^2 + ((0, s_y, -I_{iy}) \cdot \frac{(R(p_i) + T)}{|(R(p_i) + T)|})^2) \tag{15}$$

E_{p2} is again minimized by keeping $|(R(p_i + T)|$ constant during an iteration and then updating it for the next using the new "R" and "T". The vectors $(s_x, 0, -I_{ix})$ and $(0, s_y, -I_{iy})$ can also be normalized to give equal weightage to all points irrespective of their location in the image.

E_1, E_2, E_3, E_R, E_{p1} and E_{p2} are all minimized by modifiying the same basic non-linear technique. Therefore, only the technique for minimizing E_1 is presented. E_T is minimized by a straight-forward linear least squares algorithm. Appendix A discusses various representations for rotation and motivates our particular choice of quaternions for large rotations and the 3D rotation vector for small angles.

3.1 Non-linear Technique for "R_and_T"

To minimize "E_1", we adapt an iterative technique formulated by Horn [13] to solve the problem of Relative Orientation. It needs an initial estimate for both "R" and "T". The technique linearizes the error term about the current estimate for "R" and "T". It is then possible to determine how the overall error is affected by small changes in rotation and translation. This allows one to make iterative adjustments to the rotation and translation that reduce "E_1". These iterations are continued until the algorithm converges to a minimum. Note that the algorithm, like all such descent algorithms, does not guarantee a global minimum.

Assume we have a current estimate "R" for rotation. The coordinates p_i' of a rotated 3D point is given by $p_i' = R(p_i)$. We add an incremental rotation vector $\delta\omega$ to the rotation estimate "R". The direction of this incremental vector is parallel to the axis of rotation, while its magnitude is the angle of rotation. This incremental rotation takes p_i' to p_i''

$$p_i'' = p_i' + \delta\omega \times p_i' \tag{16}$$

This follows from Rodrigue's formula [13] for the rotation of a vector "r" by angle "θ" about axis "ω" :

$$r' = r(cos\theta) + sin\theta(w \times r) + (1 - cos\theta)(\omega \cdot r)\omega \tag{17}$$

where $\theta = ||\delta\omega||$ and $\omega = \delta\omega/||\delta\omega||$

Let ΔT represent a small translation added to the current translation estimate T.

Thus, the linearized energy function about the current estimate "R" and "T" becomes :

$$E = \sum_{i=1}^{2n} (N_i \cdot (p_i' + \delta\omega \times p_i' + T + \Delta T))^2 \tag{18}$$

Let $b_i = p_i' \times N_i$. Using the chain rule of triple scalar product for vectors, differentiating the objective function with respect to ΔT and $\delta\omega$ respectively, and setting the results equal to 0, after some manipulation the following two equations are obtained :

$$\sum_{i=1}^{2n} (N_i \cdot \Delta T + \delta\omega \cdot b_i) N_i = -\sum_{i=1}^{2n} (N_i \cdot (p_i' + T)) N_i \tag{19}$$

$$\sum_{i=1}^{2n} (N_i \cdot \Delta T + \delta\omega \cdot b_i) b_i = -\sum_{i=1}^{2n} (N_i \cdot (p_i' + T)) b_i \tag{20}$$

Together, the above two vector equations constitute 6 linear scalar equations in the 6 unknown components of ΔT and $\delta\omega$. We can rewrite them in the more compact matrix form:

$$\begin{aligned} C\Delta T + F\delta\omega &= -\bar{c} \\ F^T \Delta T + D\delta\omega &= -\bar{d} \end{aligned} \tag{21}$$

where $C = \sum_{i=1}^{2n} N_i N_i^T \quad D = \sum_{i=1}^{2n} b_i b_i^T \quad F = \sum_{i=1}^{2n} N_i b_i^T$

while $\bar{c} = \sum_{i=1}^{2n} (N_i \cdot (p_i' + T)) N_i$ and $\bar{d} = \sum_{i=1}^{2n} (N_i \cdot (p_i' + T)) b_i$.

Solving the above set of 6 linear equations gives a way of finding small changes in rotation and translation that reduce the overall objective function. The algorithm can therefore be expressed in the following four steps.

Step 1 Guess an initial estimate for rotation "R" and translation "T".

Step 2 Compute the coefficients of the matrices in equation (21). Solve the linear system for ΔT and $\delta\omega$.

Step 3 Compose $\delta\omega$ with the current estimate R of rotation to get the new estimate. Add ΔT to T to get the next estimate for translation.

Step 4 Stop if the algorithm has converged or has exceeded a maximum number of iterations, else go back to Step 2.

We compose the $\delta\omega$ and ΔT to the current estimate and iterate. We stop iterating when either a maximum number of iterations is exceeded or when the difference in the result between two successive iterations is less than a pre-specified minimum. The algorithm seems to converge for initial estimates which differ considerably from the correct solution. Incremental adjustments cannot be computed if the six-by-six coefficient matrix becomes singular. This will happen when we have a situation for which there are an infinite number of solutions. To compose the $\delta\omega$ with the current estimate of the Rotation R, the current rotation is represented as a quaternion [13].

3.2 Initial Estimates of Rotation and Translation

The non-linear algorithm in the above subsection needs the user to specify an initial estimate for translation and rotation. How close must the initial estimate be to the final estimate, depends on the particular data set. We have experimented with data sets where starting from almost anywhere, the algorithm will converge to the correct solution. On the other hand for some data sets, the initial rotation estimate must be within 40 deg. for all the three Euler angles representing the rotation. Generally, the rotation estimate is more important then the translation estimate.

For some applications, the user may have no good way to specify an initial estimate for rotation and translation. In that case, the user can sample the rotation space and use each of the samples as an initial estimate for the rotation estimation part of "R_then_T". *The solution* from each of these runs which has the lowest error is used to estimate the translation. This rotation and translation are then used as initial estimates for "R_and_T". We have successfully tried this procedure with 16 uniform samples of the rotation space. In the above procedure, the user could also bypass the "R_then_T" step and use the initial rotation samples with a fixed translation sample as inputs for different runs to "R_and_T".

Horn [13] presents a uniform way of sampling the rotation space. His sampling is based on the rotation groups of regular polyhedra. We present below from his paper two sets of samplings of the rotation space (in quaternion form). The components of the unit quaternions may take on the values 0 and 1, as well as the following :

$$b = \frac{1}{2} \qquad c = \frac{1}{\sqrt{2}} \tag{22}$$

In the first set, there are 12 unit quaternion samples and they form the twelve elements of the rotation group of the tetrahedron:

$$\begin{array}{llll}
(1,\ 0,\ 0,\ 0) & (0,\ 1,\ 0,\ 0) & (0,\ 0,\ 1,\ 0) & (0,\ 0,\ 0,\ 1) \\
(b,\ b,\ b,\ b) & (b,\ b,\ b,\ -b) & (b,\ b,\ -b,\ b) & (b,\ b,\ -b,\ -b) \\
(b,\ -b,\ b,\ b) & (b,\ -b,\ b,\ -b) & (b,\ -b,\ -b,\ b) & (b,\ -b,\ -b,\ -b)
\end{array}$$

In the second set, there are 24 unit quaternion samples and they form the twenty four elements of the rotation group of the octahedron and the hexahedron(cube):

$$\begin{array}{llll}
(1,\ 0,\ 0,\ 0) & (0,\ 1,\ 0,\ 0) & (0,\ 0,\ 1,\ 0) & (0,\ 0,\ 0,\ 1) \\
(0,\ 0,\ c,\ c) & (0,\ 0,\ c,\ -c) & (0,\ c,\ 0,\ c) & (0,\ c,\ 0,\ -c) \\
(0,\ c,\ c,\ 0) & (0,\ c,\ -c,\ 0) & (c,\ 0,\ 0,\ c) & (c,\ 0,\ 0,\ -c) \\
(c,\ 0,\ c,\ 0) & (c,\ 0,\ -c,\ 0) & (c,\ c,\ 0,\ 0) & (c,\ -c,\ 0,\ 0) \\
(b,\ b,\ b,\ b) & (b,\ b,\ b,\ -b) & (b,\ b,\ -b,\ b) & (b,\ b,\ -b,\ -b) \\
(b,\ -b,\ b,\ b) & (b,\ -b,\ b,\ -b) & (b,\ -b,\ -b,\ b) & (b,\ -b,\ -b,\ -b)
\end{array}$$

Horn in his paper also gives the 64 rotation samples for the rotation groups of the icosahedron and the dodecahedron.

4 Least Median of Squares (LMS) technique

The robust algorithm "Med_R_and_T" we develop to handle data with outliers is adapted from Rousseeeuw [28]. It is based on minimizing the median over all lines of the square of the error measure or finding the LMS (least median of squares) estimate. The error measure we use is the same as that used for the "R_and_T" algorithm. Therefore in "Med_R_and_T" the following error measure "E_m" is minimized :

$$E_m = \underline{Median_i}(N_i \cdot (R(p_i) + T))^2 \tag{23}$$

Since the median is not a differentiable function, E_m has to be minimized by a combinatorial algorithm. We have developed two versions of this algorithm. In the first algorithm the combinatorics is applied to the set of "n" input lines while in the second it is applied to the set of "m" landmarks where each landmark is a set of lines. The "camera determination and location problem" needs a minimum of 3 input lines. In the line version of the algorithm, we use every 3 line subset of the data to estimate a pose. For each of these poses, the squared error for all lines is found and then the median error across all lines for each pose estimated. The pose which has the minimum of these medians is chosen as the output. In the landmark version, we use every 3 landmark subset (i.e. the union of the lines in the 3 landmarks) to estimate a pose and median error. If we expected less than 3 outliers then we could use all C^m_{m-1} or C^m_{m-2} subsets for the 1 or 2 outlier landmark cases respectively.

Another measure of robust statistical procedures is their "relative efficiency" [16,28]. It is defined in Kim's et. al. paper [16] as the "ratio between the lowest achievable variance for the estimated parameters (the Cramer- Rao bound) and the actual variance provided by the given method", so that the best possible value is 1. Kim et. al. also note that "the least mean square estimator in the presence of gaussian noise has an asympotic (large sample) efficiency of 1 while the median's efficiency is only 0.637" [16,25]. To improve the efficiency of our algorithm, we use the minimum median pose to detect and remove outliers from the data and then run the least squares algorithm "R_and_T" on the remaining lines.

The algorithm "Med_R_and_T" can be summarized by the following steps.

Step 1 Find all 3-line or 3-landmark subsets of the input data.

Step 2 For each subset, determine the pose by running algorithm "R_and_T"; estimate the residual error for all "n" lines given this pose and find the median square error.

Step 3 Select the pose which gives the minimum median error. Filter out lines as outliers whose square of the residual error for that pose is greater than a certain threshold.

Step 4 Run algorithm "R_and_T" on the remaining lines and return the estimated translation and rotation as the final output.

The line version of the above algorithm has a time complexity of $O(n^4 log(n))$, where there are "n" input lines. There are $O(n^3)$ subsets and for each subset, we sort and find the median which is $O(nlog(n))$. The time complexity of the landmark version of the algorithm is $O(m^3 nlog(n))$ where there are "m" landmarks. Typically in our experiments there are of the order of 15 - 20 lines and 6 - 10 landmarks in each image. The landmark version is therefore much more efficient. The 2D line matcher algorithm, developed by Beveridge et. al. [27] used to provide the model-data line correspondences, and has been applied to matching each landmark independently. Therefore, if an outlier is present then often the entire landmark is an outlier. Thus, we employ the landmark version of the algorithm when coupled with the 2D line matcher. The complexity can be reduced further by using probablistic methods. For example, Rousseeuw [28] states that if we want the correct answer with 99 % probablity and expect no more than 30 % outliers, then only 37 out of 1140 subsets (for a set of 20 lines) need to be randomly chosen. Thus, using Monte Carlo methods and choosing a tolerable probablity of failure, the number of subsets to be explored can be drastically reduced. Finally, some subsets will lead to degenerate solutions because all lines are parallel etc. This can be detected before any further processing of the subset is done. A simple method to detect degenerate subsets in the case of three lines is to threshold on the determinant of the matrix whose rows are the unit direction vectors of the 3D lines.

5 Uncertainty Analysis

The analysis in this section is valid for both the algorithms "R_and_T" and "Med_R_and_T". In the case of "Med_R_and_T", it is assumed that the outliers have been detected and removed and the uncertainty analysis is done only over the remaining lines. Thus, all assumptions of noise stated below are only for the non-outlier lines.

Noise is assumed to be in the image data only; and the 3D model data is assumed to be accurate. In this section, closed form expressions are developed for the variance of the error in the output parameters (rotation and translation) as a function of the input data and variance of the noise and the output translation and rotation values. In the analysis, as in all such statistical analyses [26,3], the basic assumption is that the returned output parameter is the true or correct output parameter and the uncertainty region is centered around it. The analysis is local in nature. We assume our solution is at the global minimum and near the true solution. This condition could be violated by our algorithm if the initial estimate was poorly chosen. In our domain of robot navigation using landmarks, this is unlikely to happen because the "camera location determination" step is used to refine the position of the robot, which has moved a few feet from its previous known position.

The image data for lines can be specified by two parameters ρ_i and θ_i as in equation (5). For the analysis, we assume the noise for both ρ_i and θ_i is Gaussian distributed, zero mean and uncorrelated with variance $\sigma^2_{\rho_i}$ and $\sigma^2_{\theta_i}$, respectively. Instead of assuming zero-mean gaussian noise for the (ρ_i,θ_i) parameters of the noise, the assumption could be made that the endpoints of the line are zero-mean gaussian. The following derivation can be easily modified for that case.

The error in translation ΔT and rotation $\delta\omega$ is expressed as a function of the input data, output translation and rotation values and input noise. Note that the translation and rotation are in camera coordinates (1). The objective function is linearized around the computed Rotation"R" and Translation "T". Minimizing the linearized energy function enables us to express ΔT and $\delta\omega$ as linear functions of Δ_{ρ_i} and Δ_{θ_i}, the error in the input data.

The variances of ΔT and $\delta\omega$ are computed using the linear functions. We also compute the expected value of the objective function. Finally, we check if the linearization is valid by determining whether or not if the actual objective function value is of the order of the computed expected value (as predicted by linearizing). If not, we disregard the output variances we have computed.

5.1 Error expression for normals of the projection planes

The image data in the rotation and translation constraints appears in the form of the normals of the projection planes formed by the image lines and the focal point. Therefore, we represent small errors in the ρ, θ specifications of the image lines as errors in the normals of the projection plane. The normal vector is given in equation (7). Let us consider the errors in ρ, θ to be small and given by $\Delta\rho, \Delta\theta$ respectively. Equation (7) then becomes :

$$N_i^{\prime} = (s_x \cos(\theta_i + \Delta\theta_i), s_y \sin(\theta_i + \Delta\theta_i), -\rho_i - \Delta\rho_i) \tag{24}$$

$N_i^{\prime}$ is to be normalized. After normalizing the error in the normal vector, ΔN_i can be expressed as follows :

$$\Delta N_i = 1/M_i(-s_x \sin(\theta_i)\Delta\theta_i, s_y \cos(\theta_i)\Delta\theta_i, -\Delta\rho_i) \tag{25}$$

where "M_i" is the magnitude of $N_i^{\prime}$. We approximate "M_i" by :

$$M_i \approx ((s_x \cos\theta_i)^2 + (s_y \sin\theta_i)^2 + \rho_i^2)^{1/2} \tag{26}$$

Two observations can be made :

1. The components of ΔN_i are scaled by M_i. One of the terms in M_i is the square of ρ_i. The larger the ρ_i of a line, the smaller the components of ΔN_i will be. Thus, our algorithms would be more tolerant to noise in lines, which have a larger ρ_i. The effects of this are not too significant, because the sum of the other terms in M_i will be larger than the square of ρ_i. Nevertheless, it is an outcome of forming normals of projection planes from image lines.

2. Rotation of image lines due to noise is more harmful than translation of image lines. The rotation terms involving $\Delta\theta$ are scaled by s_x and s_y. The translation term $\Delta\rho$ will be generally smaller. This is borne out by our experiments, as can be seen in Table 1 in the result section for the 5 line case.

5.2 Variance in output parameters

Algorithm "R_and_T" minimizes the energy term E_1 given by equation (10). Algorithm "Med_R_and_T" minimzes the same energy function after outliers have been removed. Let us assume that N_i, R, T and p_i are the correct or true normals, rotation, translation and 3D points respectively. If we substitute them into equation (10), E_1 should exactly be equal to zero. Now, we add noise ΔN_i to the normals N_i . We wish to find the expressions which relate the noise in the output parameters, $\delta\omega$ for rotation and ΔT for translation to the input noise. We assume the error, at least for rotation, is small, that is, less than 20 deg. around each axis. The energy term E_1 can be rewritten in terms of ΔN_i, $\delta\omega$ and ΔT. N_i, R, T and p_i are assumed to be constant and represent the true values. Given ΔN_i, we can solve for $\delta\omega$ and ΔT by minimizing the new energy term E'. Based on equation (16) and (18) the energy term E' can be rewritten as :

$$E' = \sum_{i=1}^{2n}((N_i + \Delta N_i) \cdot (p'_i + \delta\omega \times p'_i + T + \Delta T))^2 \qquad (27)$$

where, as before $p'_i = R(p_i)$. Now, from our above assumptions $N_i \cdot (p'_i + T) = 0$. Therefore ignoring second order terms, E' now becomes:

$$E' \approx \sum_{i=1}^{2n} (\Delta N_i \cdot (p'_i + T) + N_i \cdot (\delta\omega \times p'_i + \Delta T))^2 \tag{28}$$

Ignoring the second order terms in (27) basically means assuming that $N_i \cdot (\delta\omega \times p'_i + \Delta T)$ is approximately equal to $(N_i + \Delta N_i) \cdot (\delta\omega \times p'_i + \Delta T)$.

After differentiating E' in the above equation with $\delta\omega$ and ΔT respectively and setting the result equal to zero, we get the following two equations:

$$\sum_{i=1}^{2n} ((N_i \cdot \Delta T) + (\delta\omega \cdot b_i) + \Delta N_i \cdot (p'_i + T))N_i = 0 \tag{29}$$

$$\sum_{i=1}^{2n} ((N_i \cdot \Delta T) + (\delta\omega \cdot b_i) + \Delta N_i \cdot (p'_i + T))b_i = 0 \tag{30}$$

where $b_i = p'_i \times N_i$.

Rearranging terms, the above two equations can be written in matrix form:

$$C\Delta T + F\delta\omega = -\hat{c} \tag{31}$$

$$F^T\Delta T + D\delta\omega = -\hat{d} \tag{32}$$

Here C, F and D are 3 by 3 matrices defined just as before.

$C = \sum_{i=1}^{2n} N_i^T N_i \qquad D = \sum_{i=1}^{2n} b_i^T b_i \qquad F = \sum_{i=1}^{2n} N_i^T b_i$ while

$\hat{c} = \sum_{i=1}^{2n} (\Delta N_i \cdot (p'_i + T))N_i = \sum_{i=1}^{2n} (\Delta N_i \cdot p_{ci})N_i$

$\hat{d} = \sum_{i=1}^{2n} (\Delta N_i \cdot (p'_i + T))b_i = \sum_{i=1}^{2n} (\Delta N_i \cdot p_{ci})b_i$

where $p_{ci} = (R(p_i) + T)$. Note p_{ci} is the i'th point in camera coordinates.

Let "B" be a 6 by 6 matrix given by : $B = \begin{bmatrix} C & F \\ F^T & D \end{bmatrix}$

We represent B^{-1} the inverse of B as follows : $B^{-1} = \begin{bmatrix} G1 & G2 \\ G2^T & G4 \end{bmatrix}$

where from [1] we know, G1, G2 and G4 are :

$G1 = (C^{-1} - C^{-1}FG2^T) \quad G2 = -(C^{-1}FG4) \quad G4 = (D - F^TC^{-1}F)^{-1}$

Using the above expression for the inverse of B, we can rewrite equation (32) :

$$\begin{bmatrix} \Delta T \\ \delta\omega \end{bmatrix} = - \begin{bmatrix} G1 & G2 \\ G2^T & G4 \end{bmatrix} \begin{bmatrix} \hat{c} \\ \hat{d} \end{bmatrix} \tag{33}$$

or

$$\Delta T = -G1\hat{c} - G2\hat{d} \tag{34}$$

$$\delta\omega = -G2^T\hat{c} - G4\hat{d} \tag{35}$$

We introduce two more symbols U_i and V_i.

$U_i = G1_{i1}N_{ix} + G1_{i2}N_{iy} + G1_{i3}N_{iz} + G2_{i1}b_{ix} + G2_{i2}b_{iy} + G2_{i3}b_{iz}$

$V_i = G2^T_{i1}N_{ix} + G2^T_{i2}N_{iy} + G2^T_{i3}N_{iz} + G4_{i1}b_{ix} + G4_{i2}b_{iy} + G4_{i3}b_{iz}$

Using the above two expressions for U_i and V_i, equation (35) and the expansions for $\hat{c}$ and $\hat{d}$ we can write expressions for ΔT and $\delta\omega$ in terms of $\Delta\rho_i$ and $\Delta\theta_i$.

$$\Delta T_i = \sum_{i=1}^{2n}(\frac{U_i}{M_i}(p_{cix}s_x \sin\theta_i - p_{ciy}s_y \cos\theta_i)\Delta\theta_i + p_{ciz}\Delta\rho_i)) \tag{36}$$

$$\delta\omega_i = \sum_{i=1}^{2n}(\frac{V_i}{M_i}(p_{cix}s_x \sin\theta_i - p_{ciy}s_y \cos\theta_i)\Delta\theta_i + p_{ciz}\Delta\rho_i) \tag{37}$$

From statistics [3], we know that the variance of a parameter "z" which is a function of input parameters "y_i" is given by

$$\sigma_z^2 = \sum\left[\sigma_{y_i}^2(\frac{\delta z}{\delta y_i})^2\right] \tag{38}$$

Note, that each line will contribute two terms, one for each point, to the summation in the two equations (36,37). Let us denote the two points we use for a line to be p_i^R and p_i^L. For both these 3D points, the corresponding projection plane normals would be the same and so would ΔT_i and $\delta\omega_i$. Using the above equations (36), (37) and (38) we can write the following closed form expressions for the variance in the output parameters.

$$\sigma^2_{\Delta Ti} = \sum_{i=1}^{n}(\quad ((U_i^L p_{cix}^L + U_i^R p_{cix}^R)s_x \sin\theta_i - (U_i^L p_{ciy}^L + U_i^R p_{ciy}^R)s_y \cos\theta_i))^2 \frac{\sigma^2_{\Delta\theta_i}}{M_i^2} + (U_i^L p_{ciz}^L + U_i^R p_{ciz}^R)^2 \frac{\sigma^2_{\Delta\rho_i}}{M_i^2}) \tag{39}$$

$$\sigma^2_{\delta\omega i} = \sum_{i=1}^{n}(\quad ((V_i^L p_{cix}^L + V_i^R p_{cix}^R)s_x \sin\theta_i - (V_i^L p_{ciy}^L + V_i^R p_{ciy}^R)s_y \cos\theta_i))^2 \frac{\sigma^2_{\Delta\theta_i}}{M_i^2} + (V_i^L p_{ciz}^L + V_i^R p_{ciz}^R)^2 \frac{\sigma^2_{\Delta\rho_i}}{M_i^2}) \tag{40}$$

Note $\sigma_{\Delta T_i}$ for i = 1..3 corresponds to $\sigma_{\Delta T_x}$, $\sigma_{\Delta T_y}$ and $\sigma_{\Delta T_z}$ respectively, and $\sigma_{\delta\omega_i}$ for i = 1..3 corresponds to $\sigma_{\delta\omega_x}$, $\sigma_{\delta\omega_y}$ and $\sigma_{\delta\omega_z}$ respectively. These are the variances of the rotation and translation in camera coordinates. To compute the variances of the rotation (R^T) and translation (T_w) in world coordinates we use the covariance matrices of the above rotation and translation parameters and equation (3).

From equations (39,40), it can be seen that the squares of both σ_{Ti} and $\sigma_{\delta\omega i}$ have a quadratic dependence on p_c^R and p_c^L, the location of the 3D endpoints in camera coordinates. Therefore, 3D lines which are closer to the camera will contribute less to the error variance in the output parameters. This is to be expected, since the projections of these lines changes the most for a small change in translation. Lines parallel to the x-axis will not constrain the translation along the x-axis or rotation about it. Similiarly, lines parrallel to the y-axis and z-axis will not constrain the output parameters along the y-axis and z-axis, respectively.

6 Results and Discussion

The development of the algorithms presented here are part of a larger effort to have the UMASS robot "Harvey" navigating the sidewalks of the UMASS campus [6]. We present results for both indoor corridor images and outdoor sidewalk images. Fig. 4a and Fig. 7 are examples of the indoor corridor images. Figs. 2a, 3a, 5a, 6 and 8 are examples of the outdoor images.

The indoor model was built by measuring distances with tape measure and is accurate to 0.1 feet [6]. The outdoor 3D model was built over two passes. In the first pass, it was built using blueprints of the campus. These blueprints are drawn to a scale of 40 feet to an inch. We found errors of up to 10 feet in this 3D model. In the second pass, we surveyed the landmarks using theodolites. We believe most of our 3D model is now accurate to within 0.3 feet. Some landmarks, such as poles and posts, are difficult to position accurately, because of their cylindrical shape and no good distinguishing points. Accuracy of the 3D model is very important for our present experiments. An error of 1 foot in the location of a 3D landmark, 50 feet away from the camera, can cause its projection to be displaced by 24 pixels in the image.

The images were acquired using a SONY B/W camera, model AVC-D1 mounted on the robot vehicle. Linked to a GOULD frame grabber, 512 by 484 size images are obtained, with field of view of 24.0 deg. by 23.0 deg.. In the introduction, we had mentioned that knowledge of the intrinsic parameters of the camera was extremely important for our real data experiments. Perturbations of some intrinsic camera parameters does not effect significantly some output parameters. For instance, we found experimentally that uncertainty in the location of the center of the image does not affect the location of the camera in world coordinates (translation in world coordinates). This may be intuitively understood by the fact that displacing the center of the image by order of 10 pixels or so moves the origin of the camera coord. system by about a tenth of a millimeter. For our outdoor experiments we assumed the center to be (270, 256) while for the indoor experiments we assumed it to be (256, 256).

Experiments were conducted with both real image data and simulated data with noise added to it. For outdoors, the landmarks used were the 3D lines forming the visible corner of the building, window lines, lampposts, telephone poles and one sidewalk line. The experiments for both synthetic and real data were conducted with the camera being about 300 feet distant from the building in Fig. 2a. The synthetic data experiments were conducted with projections of 3D lines from the model. The camera was assumed to be placed at the same location as the first frame of the real data experiments. For that frame, there was one telephone pole 50 feet away from the camera. The rest were in the range of 150 to 300 feet away.

6.1 Synthetic Data Experiments

The synthetic data experiments were conducted for both algorithms "R_and_t" and "R_then_t" using the Outdoor 3D model. The two algorithms were run with four different sets of lines, each set being perturbed by at least two different amounts of noise. Zero mean uniform noise was added to the ρ and θ of each image line. In Tables 1 and 2 the noise for each simulation is specified in the ρ and θ columns. One pixel noise in ρ means that to the ρ of each line, we added a $\Delta\rho$, which was a random number anywhere in the range [-1,+1]. Similarly, one deg. noise in θ means that to the θ of each line, we added a $\Delta\theta$, which was a random number anywhere in the range [-1,+1]. We did our simulations for 1 deg. or 5 deg. noise in θ and 1 pixel or 5 pixel noise in ρ. For each set of lines and each specification of input noise, we created 100 data samples, by starting our random number generator each time with a different seed point. The results presented in the Tables are the average absolute error of the computed rotation and translation over these 100 data samples, for each set of lines and each noise specification. The results for the "R_then_T" and "R_and_T" algorithm are in Table 2 and Table 1, respectively. Rotations and translation errors in the Tables for synthetic data are specified with respect to the camera coordinate system (Fig 1). The rotation errors are specified in terms of error in degrees of the axis-angle 3D rotation vector. ΔT_x corresponds to error in translation in the direction of the rows in the image plane (in camera coordinates). ΔT_y corresponds to error in translation in the vertical direction in camera coordinates. ΔT_z corresponds to error in translation along the direction of the optical axis in camera coordinates.

The first set of 5 lines consisted of the 4 corner edges of the building visible in Fig. 2 and one window line in that same building. The second set of 10 lines consisted of the 4 corner edges of the building, as above, and 6 lampposts and telephone pole lines. The third set of 14 lines consisted of these 10 lines plus three more lines on the building and one side walk line. The fourth set of 30 lines consisted of the above 14 lines. Plus, we assumed we had been able to identify 6 vertices, e.g. the corner of the building and drew virtual lines between these vertices if they were not already joined.

As can be seen by comparing the Tables, the results of "R_and_T" algorithm in Table 1 are much better then the results of "R_then_T" algorithm in Table 2. With zero noise specified, both algorithms gave the correct result. For each set of lines and each specification of noise, "R_and_T" performs much better than "R_then_T". The results for "R_then_T" are particularly bad for the 5 and 10 line simulations. This can be explained by the observation that, in the 5 line case, 3 of the lines form a trihedral junction. As noted before, for trihedral junctions we can have an infinite number of translations. Thus, the translation is pinned from this infinite set by just the remaining two lines, both of which are vertical and not too far from each other. With noise therefore, we would expect large errors in translation. Similarly, in the 10 line simulation, most of the lines are vertical. Vertical lines do not disamiguate rotations about the x-axis and translations along the y-axis. This problem is even more compounded when the rotation stage is separated from the translation stage.

In the Tables for both algorithms, we notice that the error decreases appreciably, decreases as the number of lines increases. In the "R_and_T" Table, we give results of experiments with the 5 line data set for two extra cases. If we look in Table 1 at the results of the 5 line data sets, we find appreciably larger error when the noise in θ is 5 deg.. However, when the noise in ρ is 5 pixels and the noise in θ 1 deg., the errors are much smaller. This demonstrates what we had predicted via the uncertainty analysis section: noise in θ for lines is much more harmful than noise in ρ. Finally, in all experiments, the error in ΔT_y was found to be often larger than the errors in ΔT_x and ΔT_z. This is because of the fact that the majority of our 3D lines are vertical.

6.2 Real Data Results for "R_and_T"

A sequence of 6 outdoor frames and 2 indoor frames was used for this experiment. For the outdoor sequence, the camera was moved in an approximate translatory motion 25 feet along the walkway. Each subsequent frame was taken after a movement of 5 feet down the walkway. The sidewalk line is close to parallel to the x-axis in the world coordinate system. The z-axis is the vertical axis in the world coordinate system. The 2D images lines were taken from the output of a 2D line matching system [27], this is part of our mobile robot project. For each frame, column 2 in table 3 gives the number of lines the 2D line matcher was able to correctly match. Figs. 2a and 5a show frames 4 and 6 of the outdoor image sequence with the corresponding input 2D lines as returned by the line matcher. These were used as input to our algorithm along with the 3D model.

Table 3 gives the estimated error of our algorithm for translation in world coordinates. In most cases, the robot is located to within a foot. The errors in table 3 are approximate to 0.5 feet. The precise location of the camera is not known. It is better to judge the performance of the algorithm by looking at the projections of the 3D landmarks

Table 1: **Average Absolute Error of Translation and Rotation in camera coordinates for algorithm "R_and_T"** The average for each experiment is taken over 100 samples of uniform noise.

NOISE			ROTATION ERROR			TRANSLATION ERROR		
No. Lines	θ deg.	ρ pixels	$\delta\omega_x$ deg.	$\delta\omega_y$ deg.	$\delta\omega_z$ deg.	ΔT_x feet	ΔT_y feet	ΔT_z feet
Correct			0.00	0.00	0.00	0.00	0.00	0.00
5	1.0	1.0	0.24	0.15	0.04	0.21	2.03	1.16
5	5.0	5.0	1.20	0.79	0.19	1.08	10.14	6.20
5	1.0	5.0	0.24	0.16	0.04	0.21	2.04	1.18
5	5.0	1.0	1.19	0.78	0.19	1.08	10.14	6.20
10	1.0	1.0	0.21	0.08	0.05	0.02	1.73	0.08
10	5.0	5.0	0.72	0.27	0.31	0.18	6.33	0.48
14	1.0	1.0	0.07	0.06	0.08	0.03	0.77	0.02
14	5.0	5.0	0.34	0.30	0.39	0.17	3.80	0.12
30	1.0	1.0	0.03	0.05	0.06	0.06	0.48	0.06
30	5.0	5.0	0.16	0.24	0.31	0.32	2.39	0.32

Table 2: **Average Absolute Error of Translation and Rotation in camera coordinates for algorithm "R_then_T"** The average for each experiment is taken over 100 samples of uniform noise.

NOISE			ROTATION ERROR			TRANSLATION ERROR		
No. Lines	θ deg.	ρ pixels	$\delta\omega_x$ deg.	$\delta\omega_y$ deg.	$\delta\omega_z$ deg.	ΔT_x feet	ΔT_y feet	ΔT_z feet
Correct			0.00	0.00	0.00	0.00	0.00	0.00
5	1.0	1.0	1.08	5.06	0.62	11.44	13.96	51.16
5	5.0	5.0	3.19	14.65	1.62	32.69	39.85	149.40
10	1.0	1.0	0.50	2.26	0.31	9.24	8.83	8.84
10	5.0	5.0	2.44	10.45	1.28	40.83	40.03	38.65
14	1.0	1.0	0.29	0.29	0.18	0.35	2.37	0.23
14	5.0	5.0	1.50	1.56	0.91	1.92	12.44	1.27
30	1.0	1.0	0.09	0.10	0.13	0.40	1.01	0.36
30	5.0	5.0	0.45	0.50	0.66	2.09	5.05	1.82

Table 3: **Estimated Errors of Translation in world coordinates for algorithm "R_and_T".** Real Data results for Outdoor and Indoor frames without outliers.

Frame No.	Num. Lines	TRANSLATION ERROR ΔT_x feet	ΔT_y feet	ΔT_z feet
Outdoor Frames				
1	17	0.10	0.06	0.03
2	15	0.38	-0.25	0.13
3	12	-1.1	0.3	0.10
4	7	0.57	0.86	0.65
5	13	1.60	0.72	0.54
6	13	1.87	1.10	0.72
Indoor Frames				
1	19	0.15	0.08	0.03
2	19	0.10	0.01	0.04
3	18	0.30	0.02	0.02
4	17	0.04	0.08	0.04
5	13	0.27	0.68	0.88
6	10	0.07	0.42	0.22

Table 4: **Estimated Errors of Translation in world coordinates for algorithms "R_and_T" & "Med_R_and_T".** Data with outliers.

			"Med_R_and_T"			"R_and_T"		
Frame No.	Num. Lines	Num. Out-Liers	TRANSLATION ERROR ΔT_x feet	ΔT_y feet	ΔT_z feet	TRANSLATION ERROR ΔT_x feet	ΔT_y feet	ΔT_z feet
Outdoor 1	17	5	0.80	-0.23	0.20	2.90	0.15	1.10
Outdoor 3	17	3	-1.10	0.14	0.10	34.74	7.34	2.3
Indoor 1	19	8	0.25	0.07	0.10	5.75	0.40	0.60
Indoor 7	46	8	0.2	0.1	0.1	0.3	0.1	0.4

on the image after the pose has been estimated. Figs. 2b, 5b and 6 are the projections of the 3D landmark lines from the poses returned by the algorithm "R_and_T" for outdoor frames 4, 6 and 5 respectively.

Table 3 also gives the results of algorithm "R_and_T" on six indoor corridor frames. The camera location for these frames ranged from 40 feet to 23 feet from the door. The results in Table 3 are with no outliers in the input data. Fig. 4a shows the first frame. The correspondences for the last four frames were obtained from a motion line tracking system [18]. As can be observed from the table, the results from the indoor data are much better than the outdoor data. This is to be expected for two reasons : (1) The landmarks are much closer, (2) The 3D model is probably much more accurate.

The results for the outdoors can be improved by the following four measures: (1) Use more lines, some of which can be obtained by drawing virtual lines from the corner of the building to the tops of lampposts. (2) Use closer landmarks. The most accurate result is for frame 1; this is probably because it is the only frame in which there is a telephone pole 50 feet away. (3) We may not know all our intrinsic camera parameters accurately. Work is underway to calibrate the camera very precisely. (4) Improve the 3D positioning of the lampposts and poles.

Finally, the algorithm "R_and_T" as it stands is not able to handle outliers. Sometimes the 2D line matcher errs and matches a wrong set of telephone lines. In the case of the telephone poles this introduces an error of 50 pixels or so. Our algorithm cannot recover from this by just looking at the residue errors for each line, from a least mean square analysis.

6.3 Results for "Med_R_and_T"

The algorithm "Med_R_and_T" was tested over both the indoor and outdoor frames. Table 4 gives results for both "Med_R_and_T" and "R_and_T" on outlier data for 4 frames . Figs.

3a, 4a, 8 and 10 show the input image line data with outliers for outdoor frame 3, indoor frame 1, outdoor frame 1 and indoor frame 3 resp..

Fig. 3a represents a classic example of a mismatch from the 2D line matcher. The telephone pole has been wrongly matched to be the street light. Thus, there is one landmark outlier or 3 lines of a set of 17 (approx. 17 %) are outliers. As can be observed from the Outdoor frame 3 entry in Table 4, the least squares solution ("R_and_T") is off by 34 feet. Basically, the least square solution for this data is meaningless. We ran the landmark version of alogorithm "Med_R_and_T" over this data set. It is able to correctly identify the outlier. There were 6 landmarks and thus the algorithm explored 20 different subsets. The output projection after estimation of pose by "Med_R_and_T" is given in Fig. 3b. The location of the camera returned by the "Med_R_and_T" is off by 1.2 feet along the x-axis (parallel to the walkway), 0.14 feet along the y-axis (perpenidcular to the walkway) and 0.1 feet along the z-axis (vertical direction). The effect of an outlier depends significantly on its relationship to other data items. In Fig. 3a the mismatched telephone pole has a large effect on the least squares solution because of its location at the right extremity of the image.

Fig. 8 is another example of outliers in outdoor data (Frame 1). The matched image lines have been randomly perturbed to make outliers. Of the 17 lines, 5 are outliers (approx. 29 %). Fig. 9 shows the projection of the 3D lines after estimation of pose by algorithm "Med_R_and_T" on this data. The outliers were correctly detected. Note, from table 4 we can see that the least squares solution of "R_and_T" in this case wasn't as bad as the outdoor frame 3 case. This may be because of the presence in Fig. 8 of the telephone pole (left extreme of image) only 50 feet away from the camera.

Fig. 4a is an example of outliers in indoor frame 1. Here again, the matched image lines have been randomly perturbed to make outliers. Of the 19 lines, 8 (approx. 42 %) are made into outliers. Fig. 4b shows the output of the "R_and_T" on this data. From Table

4, it can be seen the camera location as returned by "R_and_T" is off by 5.75 feet. We ran the line version of "Med_R_and_T" on this data. It was again correctly able to identify all outliers. Fig. 4c shows the output projection after estimation of pose by "Med_R_and_T" on this data set. The location of the camera returned by "Med_R_and_T" was off by 0.25 feet along the x-axis (along the corridor), 0.07 feet along the y-axis (perpendicular to the corridor) and 0.1 feet along the z-axis (vertical direction).

Fig. 10 showing indoor frame 7 is another example of input data with outliers from the 2D matching algorithm [27]. In this case, from Table 4 we can see that the "R_and_T" pose is fairly close to the "Med_R_and_T" pose estimation. However the projection after outlier detection (Fig. 12) is much better than the projection using the pose estimated by "R_and_T" (Fig. 11). All outliers were correctly detected. In this frame the 2D line matcher detected three of the lines in depth correctly. We find that for the indoor hallway scenes these lines (roughly parallel to the optical axis) play an important role in determining the pose accurately.

There are some cases when "Med_R_and_T" fails. This happens when the outlier is very small. For example, on a run on the image in Fig. 3a the 2D line matcher matched the building a little off to the lower right. It did this by matching the corner vertical line of the building to the first window's rightmost line and the far corner vertical line to the near corner line. The error due to this mismatch causes the robot to be placed 5 feet forward on the walkway by "R_and_T". The least squares solution "R_and_T" is not able to negate the effect of the above outlier, because beside the building lines most other lines are vertical. The resulting residual errors are very small and do not get detected by "Med_R_and_T".

Acknowledgements

A lot of this paper was made possible by discussions and debates with Harpreet Singh Sawhney, Raghavan Manmatha, Brian Burns, John Oliensis, Rich Weiss, Chris Connolly and Ross Beveridge. Ed Riseman was very supportive of the work and provided challenging ideas. Renee Stephens helped in making the figures. Matt Easely helped in collecting the outdoor 3D model data. Ross Beveridge and Claude Fennema provided data for the experiments.

References

[1] G. Adiv, *Interpreting Optical Flow* PhD thesis, COINS Tech. Report 85-35, Univ. Of Mass. at Amherst, MA., 1985.

[2] P. J. Besl, J. B. Birch and L. T. Watson, "Robust Window Operators", *Proceedings of the Second International Conference on Computer Vision,* Dec. 1988, Tampa, Florida, 591-600.

[3] P. R. Bevington, *Data Reduction and Error Analysis for the Physical Sciences,* McGraw-Hill, NY, 1969.

[4] R.Manmatha, R.Dutta, E.M. Riseman and M.Snyder, "Issues in Extracting Motion Parameters and Depth from Approximate Translational Motion", *IEEE Workshop on Visual Motion - Proceedings,* March 1989, pgs 264-272.

[5] O.D. Faugeras and G.Toscani, "Camera Calibration for 3D Computer Vision", *Proceedings International Workshop on Machine Vision and Machine Intelligence,* Tokyo,Japan, Feb 2-5,1987.

[6] C. Fennema, A. Hanson, and E. Riseman, "Towards Autonomous Mobile Robot Navigation", to appear *Proc. DARPA Image Understanding Workshop,* Morgan Kaufman Publishers, Palo Alto, CA, May 1989.

[7] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM,* vol. 24, pp. 381-395, 1981.

[8] W. Forstner, "Reliability Analysis of Parameter Estimation in Linear Models with Applications to Mensuration Problems in Computer Vision," *Computer Vision, Graphics, and Image Processing,* 40, pp. 273-310, 1987.

[9] S. Ganapathy, "Decomposition of transformation matrices for robot vision," in *Proc. 1st IEEE Conf. Robotics,* pp. 130-139, 1984.

[10] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw and W. A. Stahel, *Robust Statistics, The approach based on Influence Functions,* John Wiley & Sons, N.Y. 1986.

[11] R. M. Haralick and H. Joo, "2D-3D pose estimation," *Proceedings of the 9th International Conference on Pattern Recognition,* Rome, Italy, pp. 385-391, Nov. 1988.

[12] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quarternions," *J. Opt. Soc. A.* vol. 4, pp. 629-642, 1987.

[13] B. K. P. Horn, "Relative Orientation," *Proceedings: Image Understanding Workshop,* vol. 2, pp. 826-837, 1988.

[14] R. Horaud, B. Conio, O. Laboullex and B. Lacolle, "An analytic solution for the persepective 4-point problem," *Computer Vision, Graphics and Image Processing,* vol. 47, No. 1, pp. 33-45, July 1989.

[15] P. J. Huber, *Robust Statistics,* John Wiley & Sons, N.Y. 1981.

[16] D. Y. Kim, J. J. Kim, P. Meer, D. Mintz and A. Rosenfeld, "Robust Computer Vision: A Least Median of Squares Based Approach," *Proc. DARPA Image Understanding Workshop,* Morgan Kaufman Publishers, Palo Alto, CA, May 1989.

[17] R. Kumar, "Determination of Camera Location and Orientation," *Proc. DARPA Image Understanding Workshop,* Morgan Kaufman Publishers, Palo Alto, CA, May 1989.

[18] L. R. Williams and A. R. Hanson, "Translating Optical Flow into Token Matches and Depth from Looming", *Second Int. Conf. on Computer Vision,* pp. 441-448, 1989.

[19] R.K. Lenz and R.Y.Tsai, "Techniques for calibiration of the scale factor and image center for high accuracy 3-D machine vision metrology," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 10 # 5, pp. 713-719, 1988.

[20] E.J. Konopinski, *Classical Descriptions of Motion,* Ch. 9, pp. 234-280, W. H. Freeman and Co., San Francisco.

[21] G. Li, "Robust Regression," *Exploring Data Tables, Trends and Shapes. D.C. Hoaglin, F. Mosteller and J. W. Tukey (eds.),* John Wiley & Sons, 281-343, 1985.

[22] S. Linnainmaa, D. Harwood and L.S. Davis, "Pose determination of a three-dimensional object using triangle pairs," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 10 # 5, pp. 634-647, 1988.

[23] Y. Liu, T. S. Huang and O. D. Faugeras, "Determination of camera location from 2D to 3D line and point correspondences," *IEEE Int. Conf. Computer Vision and Pattern Recognition,* pp. 82-88, 1988.

[24] D. G. Lowe, *Perceptual Organization and Visual Recognition,* Kluwer Academic Publishers, Hingham, MA, 1985.

[25] F. Mosteller and J. W. Tukey, *Data Analysis and Regression,* Addison-Wesley, Reading, MA., 1977.

[26] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipies,* Cambridge University Press, Cambridge, MA, 1986.

[27] J. Ross Beveridge, R. Weiss and E. Riseman, "Optimization of 2-Dimensional Model Matching," *Proc. DARPA Image Understanding Workshop,* Morgan Kaufman Publishers, Palo Alto, CA, May 1989.

[28] P. J. Rousseeuw and A. M. Leroy, *Robust Regression & Outlier Detection,* John Wiley & Sons, N.Y., 1987.

[29] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *SIAM Review,* Vol. 6, No. 4, pp. 422-430, Oct. 1964.

[30] R. Y. Tsai, "An Efficient and Accurate Camera Caliberation Technique for 3D Machine Vision," *IEEE Int. Conf. Computer Vision and Pattern Recognition,* pp. 364-374, 1986.

[31] P. R. Wolf, *Elements of Photogrammetry,* McGraw Hill, New York, 1974.

[32] A. D. Worrall, K. D. Baker and G. D. Sullivan, "Model based perspective inversion," *Image and Vision Computing,* Vol. 7, No. 1, pp. 17-23, Feb. 1989.

A Different ways of representing 3D rotation

The rotation operator in equation (10) can be expressed in many ways [20,29]. Each gives rise to different non-linear expressions for "E". Some ways would require non linear equality constraints to be satisfied. Some are unconstrained but give rise to complex objective functions, which must be minimized. A minimum of 3 parameters is needed to represent rotation, but this leads to non-unique representations. It has been proven that to represent 3D rotation uniquely at least 6 parameters are needed [29]. We now describe some of the common representations and build the motivation for our final choice on representing rotation.

Orthonormal Matrix : This is one of the most common ways of representing 3D rotation. Here rotation is specified by a 3 × 3 orthonormal matrix.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{41}$$

To be an orthonormal matrix, the following equality constraints must be satisfied.

$$\sum_{j=1}^{3} r_{ij}^2 = 1.0 \tag{42}$$

$$\sum_{j=1}^{3} r_{ij} r_{kj} = 0.0 \qquad where \ \ i \neq k. \tag{43}$$

$$det(R) = 1.0 \tag{44}$$

Note, any matrix "R" which satisfies the first two constraints will have its determinant to be +1.0 or -1.0. The determinant of "R" must be +1.0 to represent a rotation operator. When the determinant of "R" is -1.0 it represents a reflection. In our solution process, the determinant constraint need not be explicitly observed. If we obtain a solution where the determinant of the "R" matrix is -1.0, all the elements of "R" and "T" in equation (10) can be mutliplied by -1.0 to obtain a solution where "R" represents a rotation. Thus, if we represent "R" as an orthonormal matrix, our objective function given by equation (10) becomes a quadratic function of 12 unknowns with 6 quadratic equality constraints to be satisfied. The rotation matrix gives us 9 of the 12 unknowns. The translation vector is the remaining 3.

Axis and Angle : The rotation of a vector "p" by angle "θ" about axis "ω" is given by Rodrigue's formula :

$$p' = (cos\theta)p + sin\theta(\omega \times p) + (1 - cos\theta)(\omega \cdot p)\omega \tag{45}$$

Here, "ω" must be a unit vector. So, we have a quadratic equality constraint. Therefore, if we represent "R" by axis and angle, we have an objective function given by equation (10) which contains trignometric terms of power 2 and higher and a quadratic equality constraint must be satisfied.

There are two variations of the above method for representing rotations. As described above, we require 4 terms to represent the rotation. Both of the following variations require only 3. We could respresent the axis by two spherical coordinates, in which case, we would have a high order trignometic objective function but no constraints. The three terms would be the two spherical coordinate angles representing the axis and the angle of rotation about the axis. We could also remove the constraint, that the axis vector is a unit vector. The magnitude of the axis vector would then be the angle of rotation about the axis. This too, is an unconstrained representation of rotation. It also leads to a complicated trignometric objective function. However, we use this representation for small rotations in our formulation. For small representations, we will see that this leads to a linear operator for rotation. Note, in the axis angle representation, negating both the axis and the angle represents the same rotation.

Euler angles : Here, we represent 3D rotation by 3 succesive rotations about each of the 3 coordinate axis, resp.. The order of the rotations must be specified. We rotate first by angle ψ about the z-axis, then by angle ϕ about the y-axis and finally by angle θ about the x-axis. Each of these rotations can be specified by a 3×3 matrix, whose elements are trignometric functions of the angle of rotation. Finally, we would have a 3×3 matrix whose elements are combinations of trignometric functions of the three angles. The final rotation matrix is given by the following equation:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} cos(\phi) & 0 & -sin(\phi) \\ 0 & 1 & 0 \\ sin(\phi) & 0 & cos(\phi) \end{bmatrix} \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{46}$$

Representing rotations by Euler angles makes our minimization problem to be an unconstrained one, however, our objective function becomes an extremely complicated trignometric function. Euler angles are often used in engineering applications as they can be measured by instruments and also are easy to picture.

Quaternions : Here we represent rotations by a 4D vector q. The quaternion "q" can be thought of as a scalar q_o and a vector q_v. It is related to the axis ω and angle θ representation in the following manner:

$$q_o = cos(\theta/2) \qquad q_v = sin(\theta/2)\omega \tag{47}$$

Rotation of a vector p using quaternions is given by $p' = q \circ p \circ q^*$. Where "o" denotes quaternion multiplication. The complex conjugate of q is q^* where $q^* = (q_o, -q_v)$. Quaternion multiplication of two quaternions is given by the following equations:

$$(p \circ q)_o = (p_o q_o - p_{v1} q_{v1} - p_{v2} q_{v2} - p_{v3} q_{v3})$$

$$(p \circ q)_{v1} = (p_o q_{v1} + p_{v1} q_o + p_{v2} q_{v3} - p_{v3} q_{v2})$$

$$(p \circ q)_{v2} = (p_o q_{v2} - p_{v1} q_{v3} + p_{v2} q_o + p_{v3} q_{v1})$$

$$(p \circ q)_{v3} = (p_o q_{v3} + p_{v1} q_{v2} - p_{v2} q_{v1} + p_{v3} q_o)$$

(q_o, q_v) and $(-q_o, -q_v)$ represent the same rotation. While (q_o, q_v) and $(q_o, -q_v)$ represent opposite rotations. For a 4-tuple to represent rotation it must be a unit vector, i.e., the following constraint must be satisfied:

$$q_o^2 + q_{v1}^2 + q_{v2}^2 + q_{v3}^2 = 1.0 \tag{48}$$

Representing rotations by quaternion, we have an objective function with poloynomial terms of degree 4. Our solution also has to satisfy a quadratic equality constraint. With quaternions [12] however, it is easy to compose rotations. Another advantage of using quaternions is that given a 4D vector which does not satisfy the unit magnitude constraint, it is easy to find the closest unit quaternion to it. This is useful for us. In our formulation, we represent large rotations by quaternions. At each iteration in our search for the minimum of the objective function, we represent small rotations by a 3D vector for rotation axis and angle. We form quaternions from these 3D vectors and compose these to the current estimates for the rotation. This allows us to have an unconstrained mininization problem. Our objective function at each stage in our iteration can also be linearized and is easy to minimize.

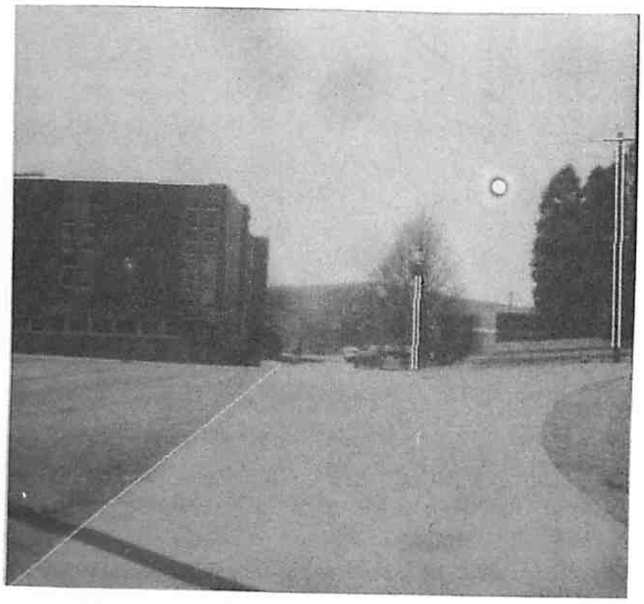

Fig. 2A: Input 7 lines from 2D line matcher to algorithm "R_AND_T" for outdoor Frame 4.

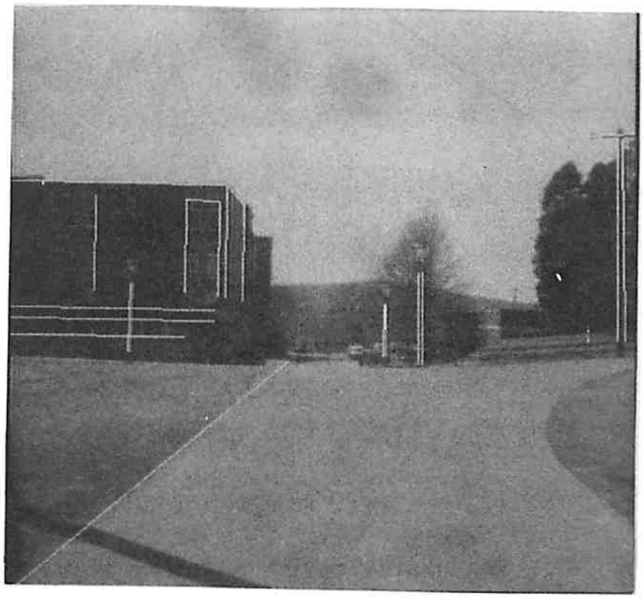

Fig. 2B: Projected lines after estimation of pose by algorithm "R_and_T" for outdoor Frame 4.

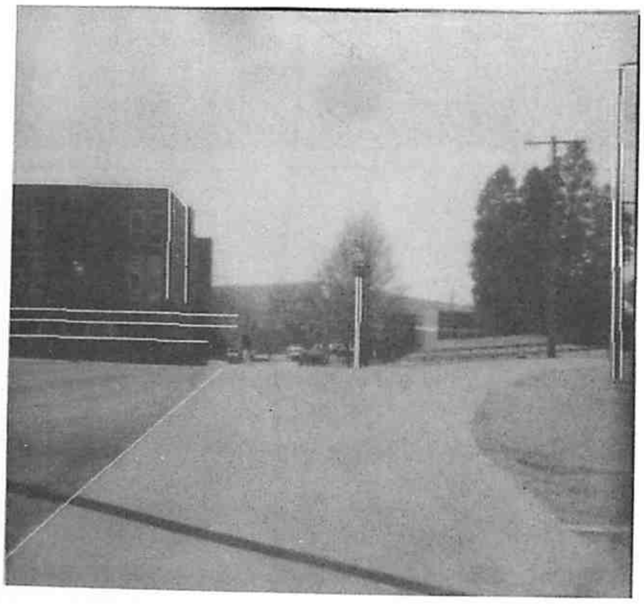

Fig. 3A: Input 14 lines from 2D line matcher to algorithm "MED_R_AND_T" for outdoor frame 3. (NOTE: outlier, telephone pole (3 lines)) has been matched incorrectly to street light.

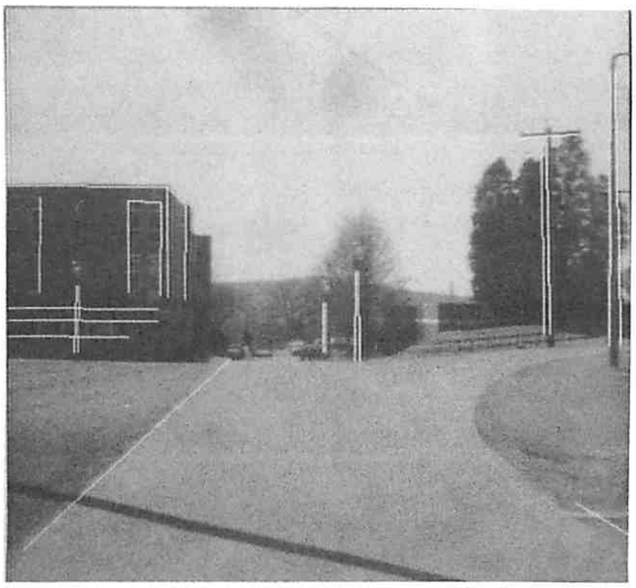

Fig. 3B: Projected lines after estimation of pose by algorithm "MED_R_AND_T" for outdoor Frame 3. The outlier (telephone pole, 3 lines) was correctly detected.

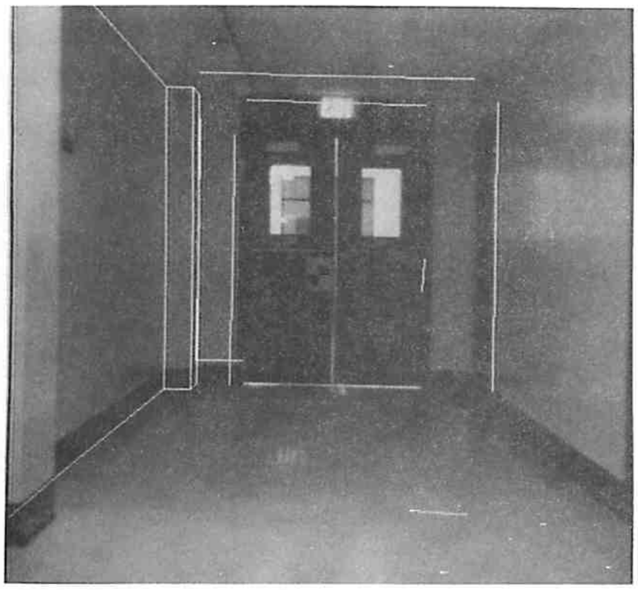

Fig. 4A: Input 19 lines to algorithms "MED_R_AND_T" and "R_AND_T" for indoor Frame 1. 8 of 19 lines are outliers.

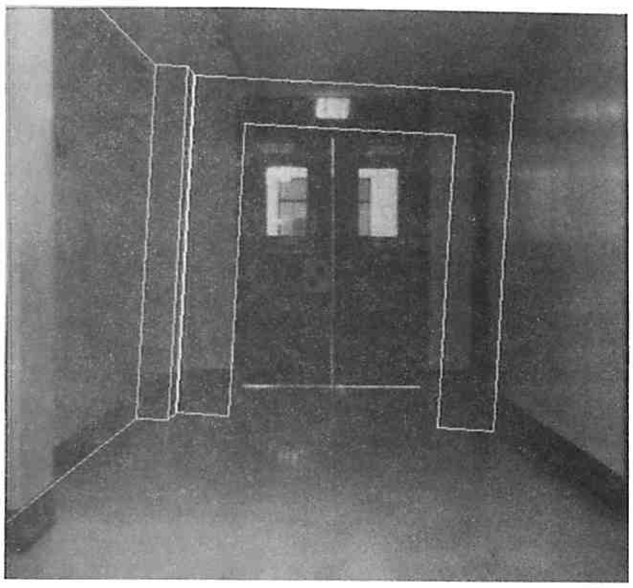

Fig. 4B: Projected lines after estimation of pose by algorithm "R_AND_T" for indoor Frame 1. Least squares algorithm fails when outliers are present.

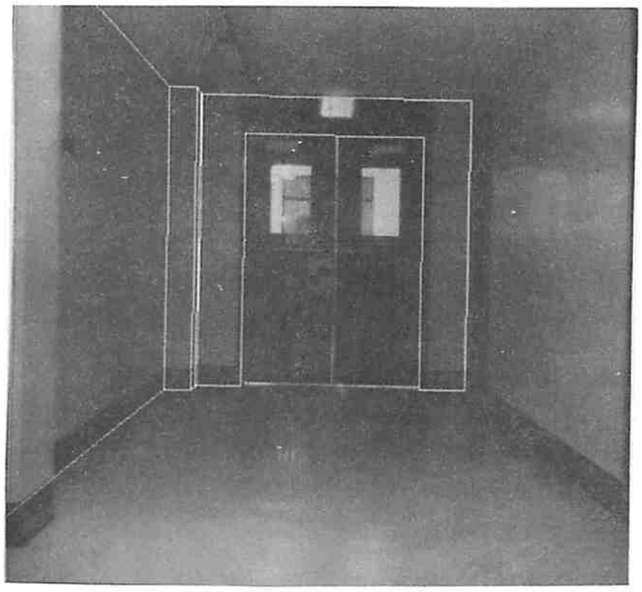

Fig. 4C: Projected lines after estimation of pose by algorithm "MED_R_AND_T" for indoor Frame 1. Outliers are correctly detected. This is also output of "R_AND_T" when data without outliers is given to it.

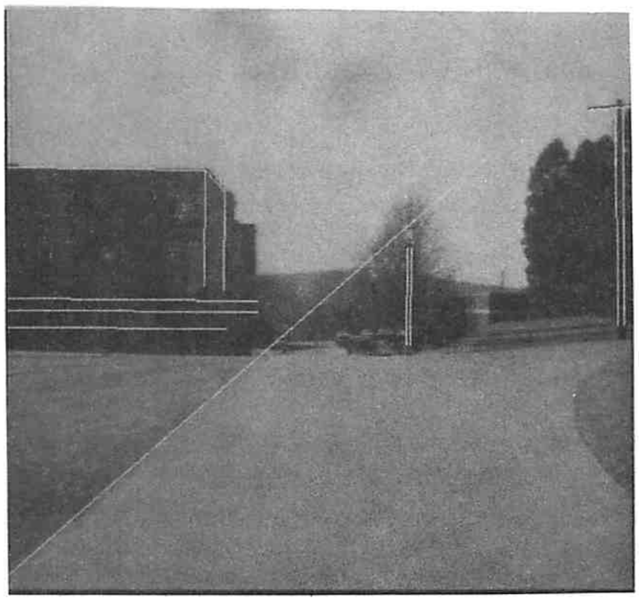

Fig. 5a: Input 13 Lines to algorithm "R_and_T" for Outdoor Frame 6.

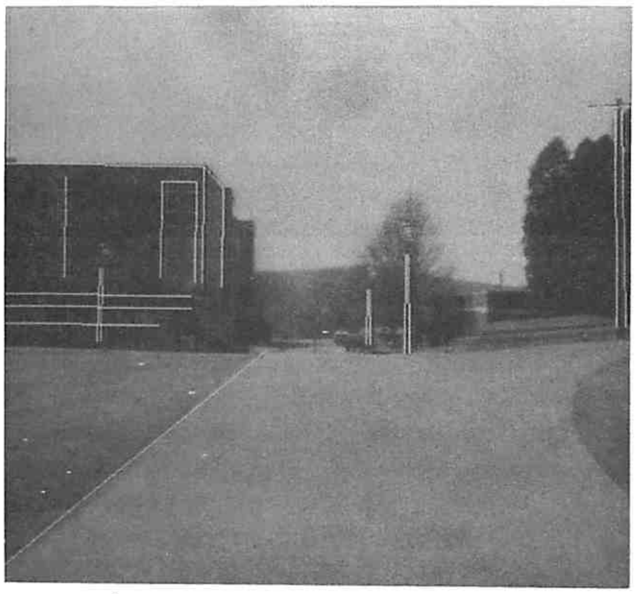

Fig. 5b: Projected lines after estimation of pose by algorithm "R_and_T" for Outdoor Frame 6.

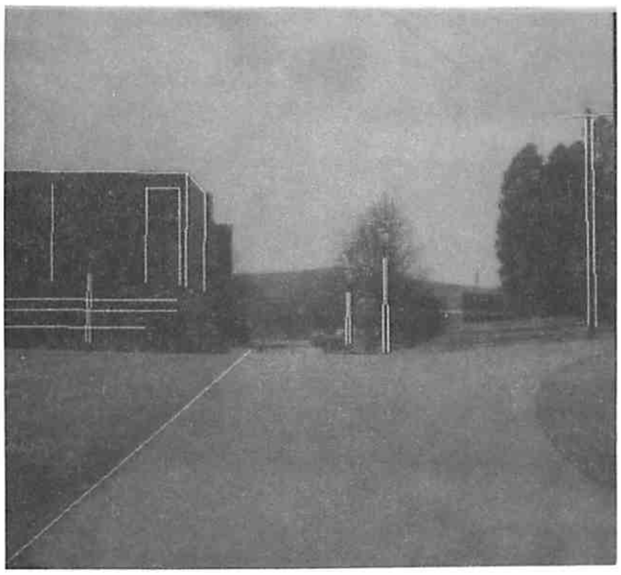

Fig. 6: Projected lines after estimation of pose by algorithm "R_and_T" for Outdoor Frame 5.

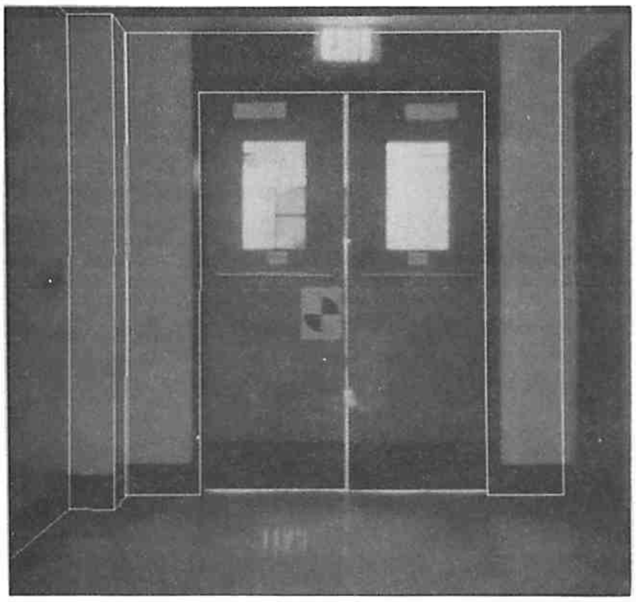

Fig. 7: Projected lines after estimation of pose by algorithm "R_and_T" for Indoor Frame 2.

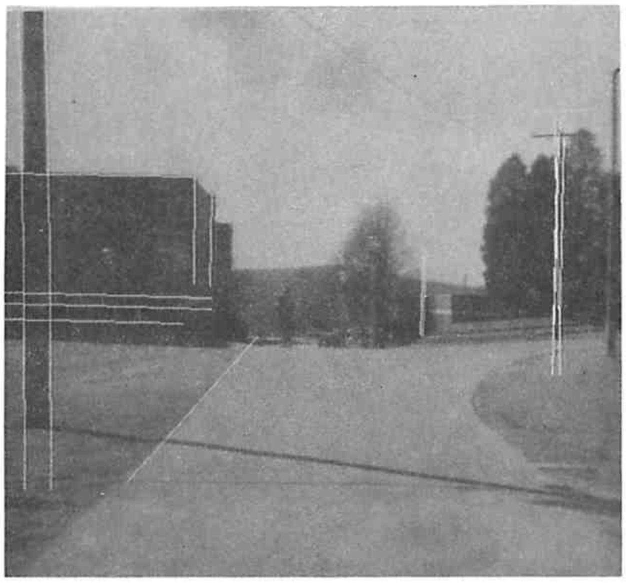

Fig. 8: Input 17 lines to algorithm "Med_R_and_T" for Outdoor Frame 1. 5 of 17 lines are outliers (Street Light and Lamp Post).

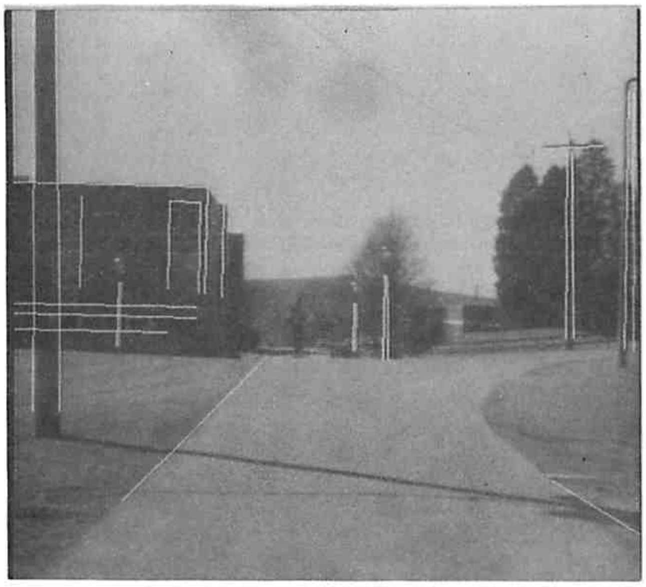

Fig. 9: Projected lines after estimation of pose by algorithm "Med_R_and_T" for Outdoor Frame 1. Outliers were correctly detected.

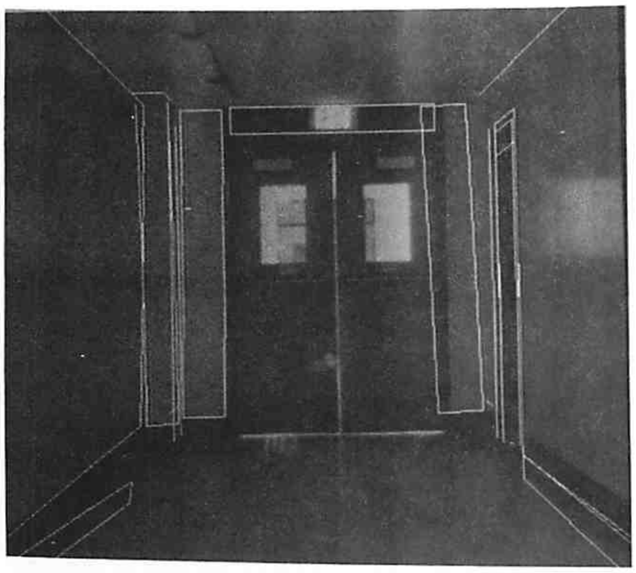

Fig. 10: Input lines to algorithms "MED_R_AND_T" and "R_AND_T" for indoor Frame 7.

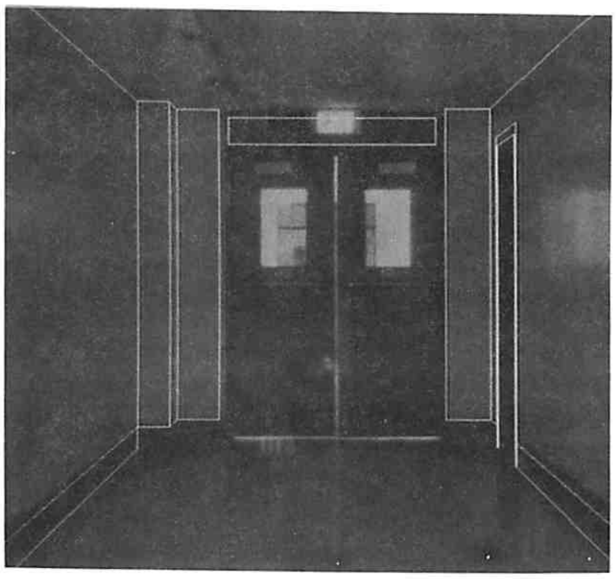

Fig. 11: Projected lines after estimation of pose by algorithm "R_AND_T" for indoor Frame 7.

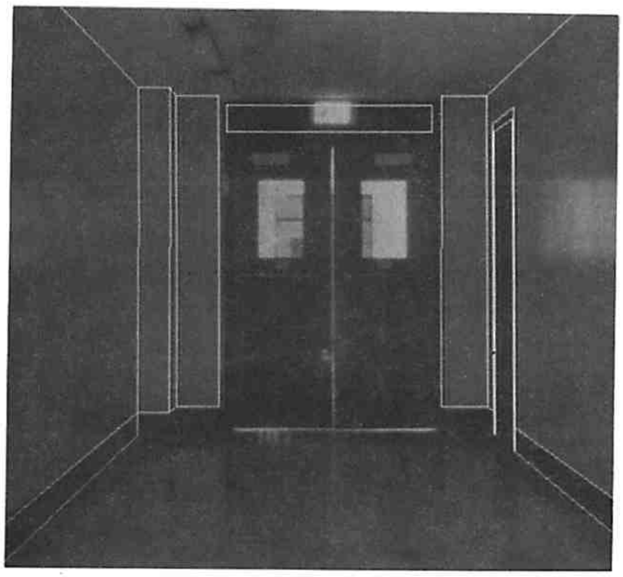

Fig.12: Projected lines after estimation of pose by algorithm "MED_R_AND_T" for indoor Frame 7. Outliers are correctly detected.