Presented at the IEEE International Conference on Robotics and Automation (Nice, France, May 1992)

Model-driven Pose Correction – Techniques and Experiments*

Yuval Roth, Annie S. Wu, Remzi H. Arpaci,
Terry Weymouth, and Ramesh Jain
The Artificial Intelligence Laboratory
Electrical Engineering and Computer Science Department
The University of Michigan
Ann Arbor, MI 48109-2110

September 10, 1991

Abstract

A mobile system should take advantage of a priori knowledge of its surroundings. This knowledge can be used to correct the dead-reckoning based self position and orientation. This paper will describe a system in which different levels of models are used to guide the sensory interpretation, and to correct the pose expectations. In this system simulated images are used to analyse the real images and to correct the pose parameters. The reported techniques have been implemented and experiments with real images in a real environment have been performed. Along with the formulations and the algorithms, we shall provide experimental results which we have obtained with an indoor mobile robot system.

1 Introduction

A perception system, whether biological or silicon, has to recover information about the real world from a projection of the world. In most cases, this projection is many to one, resulting in irrecoverable information. This lost information can only be recovered if strong assumptions are made about the environment in which the system has to perform.

*This work is partially sponsored by the Department of Energy Grant No. DE-FG02-86NE37969, the State of Michigan Research Excellence Funds, and the Rackham Predoctoral Fellowship

For example, a general system to detect tomatoes in a scene would be hard to implement. On the other hand, if the system knows that it is in a greenhouse with tomato plants, simple detection schemes based on color could be employed. The introduction of such assumptions has been referred to as *controlled hallucination*. In controlled hallucination the system synthesizes expected sensory information based on local models to aid in the perceptual analysis task. It is an *hallucination* process since internal information is added to the sensory data, and it is *controlled* by expectations from the existing models. For

Figure 1: Model-driven synthesized view (right) is registered with the real camera view (left) after the pose correction has been applied. Detectable features from the model are superimposed and highlighted on the real image. These can be used to qualitatively evaluate the quality of the pose determination

example, Figure 1 shows a comparison between a real sensory image (on the left) and a simulated image (on the right). Such synthesized sensory data is used to simplify the perceptual analysis of the real data. This approach, clearly, places a strong emphasis on the available models, their management and updating.

The problem addressed by this paper is pose determination for robot navigation. The problem is to maintain the system's instantaneous percept of its position and orientation in space for performing various tasks. The matching of pose descriptions to reality is easiest (and perhaps only possible) when there is a small deviation between the expected pose and the actual pose. Dead-reckoning from sensors such as wheel encoders can be relatively precise for short time intervals. In long range navigation, however, (due to uneven terrain, slippage, variations in wheel size, digitization errors, etc.,) the dead-reckoning estimation deviates from the real position. This estimation, if not constantly corrected, will deteriorate with time. Global positioning systems (e.g. satellite based) are also available, but unless constructed specifically for a particular task and restricted environment (for example with laser beams at known loci), these methods have a limited

resolution. Therefore, additional local sensory feedback is necessary to enhance and correct the dead-reckoning and global positioning information.

The controlled hallucination paradigm can be particularly beneficial for robot navigation and pose determination. In this paradigm, the problem is converted into a pose correction problem where an instantaneous expectation drives the analysis, and bounds the solution space. The expectations are driven by internal models at different levels of descriptiveness: generic models, and instantiated models. Instantiated models, which describe the particular details of an existing environment, are used for direct comparison with the sensory data. Generic models, which specify a generic type of environment (for example a 3 lane highway, a hippodrome, a cotton field, etc), are used to suggest specific interpretation procedures. Both specific and generic models are incorporated in our approach.

The problem we address involves two types of potential unknowns: extrinsic variables, namely position and orientation of the platform relative to some fixed coordinate frame, and intrinsic variables, namely the parameters and sensor characteristics instrumental in obtaining the extrinsic values. For example, for a system with a mounted CCD camera, the geometry, optical characteristics of the lens, and size of the CCD array are necessary to interpret the information this camera is collecting. Evidently the intrinsic unknowns need not be evaluated as frequently as the extrinsic unknowns.

The techniques described in this paper are part of the low level navigation module of a mobile robot system. They consist of new techniques as well as enhancements of previously reported techniques. The techniques are not only theoretically sound but have also resulted in successful experiments in a real environment. We shall present the following 3 components:

1. The central technique is for position and orientation correction based on matching model features with vision sensory data. A short interval dead-reckoning expectation is used to constrain the solution space.

2. The second component is for using a vanishing point to correct orientation. This technique is a necessary preprocessing stage for the pose determination. The pose determination relies on the expectation being close enough to the reality. If the potential deviation is large, the search space is harder to constrain, and the procedure slows or even breaks down. Here, a fast and crude correction scheme is applied to the orientation angles which are a major source of deviation from the expectation in the 2D space.

3. The third component is a technique for using the a priori model information to calibrate the camera. These camera parameters (intrinsic unknowns) are instrumental for the first two components. These parameters will change whenever the camera itself is replaced or when the optical system is adjusted.

The model-based approach for navigation has been followed by other researchers as well. Fennema *et al.* [5] report on a similar approach addressing the problem of pose determination. The differences between our technique and theirs is that they use 2D matching in the image plane, whereas we perform the matching practically in the 3D space. Other differences are the different formulation we use based on quaternions for representing and solving the rotation matrix, and the modeling or knowledge management employed. Beveridge *et al.* [3, 4] present the 2D matching technique used by the group at the University of Massachusetts. Kumar [11] presents a technique used for pose determination, and he compares this method, which combines the solution for the rotation and translation under a single constraint, to solving for the rotation first and then solving for the translation. This method is enhanced by Kumar and Hanson [12] with robustification by the use of a least-median technique instead of least-squares.

Arun *et al.* [1] also address the problem of pose determination. They present a least-squares solution for the transformation and rotation matrices based on on a singular value decomposition of a 3×3 matrix. Liu *et al.* [16] introduce linearization methods for this nonlinear problem. However, this linearized method sacrifices small data size for simplicity, and requires a larger number of matched data points. The matching here is based on line rather than point correspondences. Linnainmaa *et al.* [14] address a related problem of determining the pose of a 3D object. They use a generalized Hough transform to estimate the rotation and translation parameters from the distribution of values determined by matching triples of points on the object to triples of image points. This general problem of object pose determination is also addressed by Lowe [17], here models with arbitrary curves are matched to 2D image features. Besl and Mckay [2] present a general technique for registering 3D shapes including free-form curves with data. The method is based on the Iterative Closest Point (ICP) algorithm which requires finding the closest point on a given geometrical shape to a given point. In solving for the rotation and orientation they follow the approach described by Horn [7] which uses quaternions for representing rotations. The use of quaternions introduces an elegant closed-form solution in which the parameters sought have an intuitive interpretation. In determining the rotation and translation we follow this last approach. We vary, however, from these 3D object pose determination techniques in the fact that by using the dead-reckoning estimation we can introduce reliable expectation based constraints on the solution space. We claim that this expectation guided approach is not only a performance related improvement, but also a necessary direction for dynamic autonomous systems which will have to deal with vast amounts of sensory information and internal knowledge.

Liou and Jain [15] present a reliable technique for road boundary determination for road following using vanishing points. In this paper we also present a technique which assumes that vanishing points in a structured environment will be easily detected. It uses the vanishing points' position in 2D space to determine and correct the rotation matrix for a robot in an indoor environment.

Tsai [21] and Lenz and Tsai [13] present various techniques for calibrating camera parameters. In their work, the intrinsic parameters they are calibrating include effective focal length, radial lens distortion, horizontal scale factor, and image center position. We are using an ideal pin hole camera model, and the intrinsic parameters of interest are the effective focal length and the vertical to horizontal scale factor (e.g. square or rectangular pixels). We use the same model of the environment used for the position correction scheme to perform the calibration procedure. This calibration only needs to be performed whenever the camera or lens are modified in some way. The pose correction modules use these calibration parameters, and therefore the importance of specifying a method that uses the same model-guided framework for estimating these parameters.

Section 2 will provide a general overview of our robotic experimental system. It will briefly present the various components and the Control and Simulation Program for Mobile Platforms (COSIM) which is used to develop and run the various experiments. It will also outline the general architecture philosophy followed by this research effort and the modeling scheme – Context-based Caching (CbC) – used in our system. The following sections will be more technically oriented. Section 3 will detail the model-guided technique for position and orientation correction. Section 4 will detail the vanishing-point based module used for orientation correction, and Section 5 will describe the calibration scheme employed to obtain the camera parameters used by the preceding techniques. For the sake of completeness and notational consistency, appendix A establishes the notation to be used throughout the paper and enumerates and defines the various variables, and appendix B presents an overview of the various transformations and mappings in 3D space between the different coordinate systems and projections to 2D space.

2 System Overview

Our robotic experimental system consists of a mobile platform connected through RS232 to a graphics workstation. The robot platform used in the reported experiments is a HERO Heathkit 2000 robot. The robot has in addition to its native ultrasonic sensors a CCD camera mounted on top. The robot is controlled from our Control and Simulation Program for Mobile Platforms (COSIM) [20]. COSIM is a development tool which provides simulation of the platforms and their various sensing modalities side by side with actual control of the platforms. The general philosophy behind our robotic system can be found in [8].

The environment modeling scheme we use is based on the Context-based Caching (CbC) paradigm [20]. CbC is a mechanism for knowledge management in which the knowledge is divided into several *context* elements. CbC dynamically maintains a smaller set of *local contexts* which are potentially necessary for the operation of the system at a given time. This smaller set of contexts is modified based on multi-dimensional

connections between the context and *activation tests* which are associated with these connections. The activation tests which evaluate the state of the system are constantly performed by the knowledge management unit (see Figure 2). The activation tests are

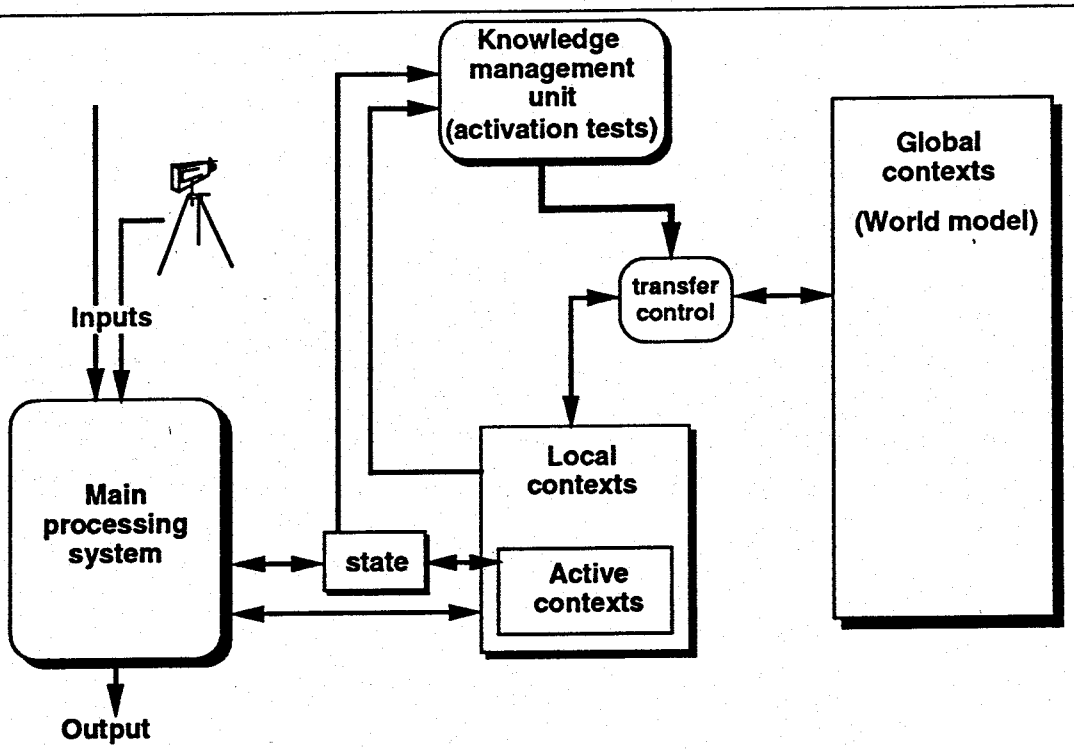

Figure 2: Knowledge Management with Context-based Caching. The processing system uses external (sensory) and internal (model) information. Internal information (knowledge) is segmented into interconnected context elements. All the contexts are stored in the World model. The internal information available to the processing elements is restricted to the Local contexts module. Contexts are transferred into the Local contexts set based on activation tests performed on the dynamic state of the system.

defined within the knowledge base and relate to a subset of the local contexts referred to as the *active contexts*, which are the set of contexts currently being used by the system. The argument of the activation tests is a dynamic state vector which indicates the local state of the system. The activation tests use this state vector to indicate a necessary modification of the active contexts. The knowledge management operations are transparent to the processing units that use the information and can therefore be decoupled from them. We follow the object-oriented paradigm for defining the various context objects. This allows us to accommodate in our models any type of data along with the operations that can be performed with it or on it.

In the technique reported in this paper we use the CbC mechanism to provide suggested local features to be looked for in the real data. When the robot moves, this set of features dynamically changes. In COSIM the graphics engine is used as an *internal visualization* (in addition to the regular on screen external visualization) to synthesize expectations for the projection of the local features in the real data. These expectations are used to constrain the search space. The role of the reported technique is to maintain and correct a set of extrinsic unknowns. These unknowns, namely 3D position and orientation of the platform, are (except in the initialization phase) confined to a small envelop around an expected set of values based primarily on dead-reckoning from the system's motion encoders. The size of this uncertainty envelop varies based mostly on time – longer time lapses between sensory verification will generate larger uncertainty envelopes.

A system using these correction techniques is schematically shown in Figure 3. The correction module receives a time stamped set of sensory data, and the expected state of the main system at that given point in time. It then, independently of any other operation the system is performing, evaluates the sensory data based on the expectation and provides a correction for it. While the correction mechanism is in the evaluation phase, the system continues its regular operation and maintains, in addition to the global state, a history of the dead-reckoning changes since the sensory data collection transmitted to the correction mechanism. Once the correction mechanism generates the corrected state, the local history is applied to it to produce an updated internal state. This scheme allows for an asynchronous relation between the main system and the correction module. Even if the correction module needs longer periods of time (or even varying lengths) relative to the sensing rate or the dead-reckoning input rate, the system need not slow down. Roth and Weymouth [19] present a similar approach which is based on an a priori model but uses ultrasonic sensor data to correct the expectation. The use of visual data with its superior spatial resolution provides a higher degree of potential precision.

3 Position and Orientation Correction

The pose determination technique has three components:

- Matching 2D image points to model features given an image and a 3D model.
- Reverse projection of image data points given a set of pairs of matched 3D model points to 2D image points.
- Determining the registration vector (rotation and translation) given a set of pairs of matched 3D model points to 3D data-based points.

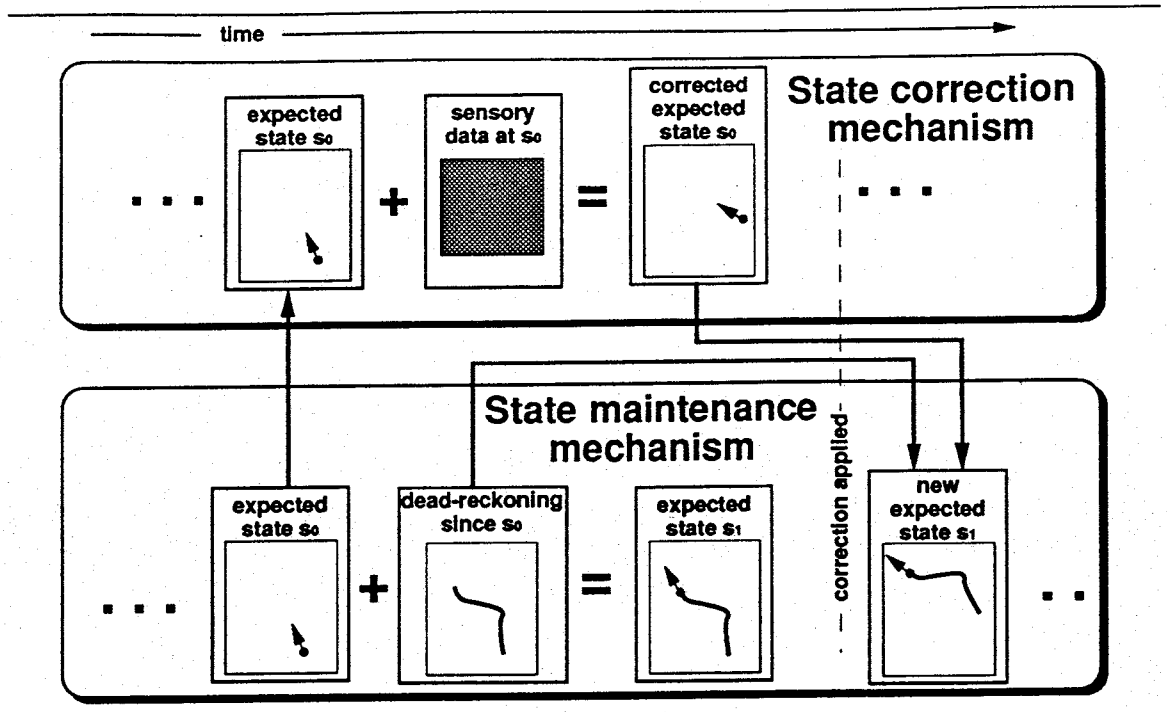

Figure 3: Correcting extrinsic variables. The state correction mechanism receives an expected state and the instantaneous sensory information associated with that state. Based on the external (sensory) information, the internal state is corrected. The state maintenance module, meanwhile, continuously maintains the latest expected state. This is done based on dead-reckoning until a correction is available from the correction mechanism. At that point the corrected expectation for the old state (s_0) is adopted, a recorded dead-reckoning history is applied to it to generate a corrected current state (s_1) , and the dead-reckoning history is cleared.

We shall present these components in reverse order so that the final goal of determining the registration vector is presented first followed by the other necessary modules that support it.

3.1 Registration vector determination

The solution outlined here is primarily based on the solution presented in [2]. Notice that the notation, however, is slightly modified for consistency.
Given 2 sets of matched 3D points our goal is to find the correction vector $\vec{q}$ which specifies the rotation and translation between the matching set of points. The correction

vector $\vec{q}$ is made up of two vectors:

$$\vec{q} = [\vec{q}_R \mid \vec{q}_T]$$

where $\vec{q}_R = [q_0 \;\; q_1 \;\; q_2 \;\; q_3]^t$, $q_0 \geq 0$, $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ is the unit rotation quaternion, and $\vec{q}_T = [q_4 \;\; q_5 \;\; q_6]$ is the translation vector. The following technique first calculates $\vec{q}_R$ and then calculates $\vec{q}_T$ from $\vec{q}_R$. For techniques that compute both translation and rotation using a combined constraint (referred to as R_and_T or Med_R_and_T) see [11] and [12]. To simplify the interpretation of the solution we perform the operations in the camera centered coordinate frame. This way transformations and rotations relate directly to the position and orientation of the camera. The set of model points were therefore converted to the camera coordinate frame.

Let $\{\vec{C}_{m_i}\}_{i=1..n}$ be the set of 3D points from the model, and let $\{\vec{C}_{d_i}\}_{i=1..n}$ be the set of matching points established from the data, where n is the number of matched pairs.

The first step is to find the center of mass of each set of points. Let $\vec{\mu}_d$ be the center of mass for the data point set, and $\vec{\mu}_m$ be the center of mass for the corresponding model point set:

$$\vec{\mu}_d = \frac{1}{n}\sum_{1=1}^{n} \vec{C}_{d_i} \qquad \vec{\mu}_m = \frac{1}{n}\sum_{1=1}^{n} \vec{C}_{m_i}$$

Next we calculate the cross-covariance matrix Σ_{md} of the data and model points:

$$\Sigma_{md} = \frac{1}{n}\sum_{i=1}^{n} \left[\vec{C}_{m_i}\vec{C}_{d_i}^t\right] - \vec{\mu}_m\vec{\mu}_d^t$$

Next, define the antisymmetric matrix $A_{ij} = (\Sigma_{md} - \Sigma_{md}^t)_{ij}$, and define the vector $\Delta = [A_{23} \;\; A_{31} \;\; A_{12}]^t$. This is used to form the symmetric 4×4 matrix $Q(\Sigma_{md})$:

$$Q(\Sigma_{md}) = \begin{bmatrix} \mathrm{tr}(\Sigma_{md}) & \Delta^t \\ \Delta & \Sigma_{md} + \Sigma_{md}^t - \mathrm{tr}(\Sigma_{md})\mathbf{I}_3 \end{bmatrix}$$

where $\mathbf{I}_3$ is the 3×3 identity matrix and $\mathrm{tr}()$ is the trace of a matrix.

The eigenvector that corresponds to the maximum eigenvalue of the matrix $Q(\Sigma_{md})$ is the desired vector $\vec{q}_R = [q_0 \;\; q_1 \;\; q_2 \;\; q_3]$. Techniques for finding eigenvalues and eigenvectors can be found in chapter 11 of [18]. In our experiments we used the Jacobi method.

Given $\vec{q}_R$, we can calculate $\vec{q}_T$ as follows:

$$\vec{q}_T = \vec{\mu}_d - \mathbf{R}'(\vec{q}_R)\vec{\mu}_m$$

where

$$\mathbf{R}'(\vec{q}_R) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}$$

The vector $\vec{q}_R$ describes a rotation in terms of angle of rotation ψ (swing) and an axis of rotation defined by the unit vector $\vec{\omega} = [\omega_x \ \ \omega_y \ \ \omega_z]$. The elements of $\vec{q}_R$ are defined by the following equations:

$$q_0 = \cos\frac{\psi}{2} \quad , \quad q_1 = \omega_x \sin\frac{\psi}{2}$$
$$q_2 = \omega_y \sin\frac{\psi}{2} \quad , \quad q_3 = \omega_z \sin\frac{\psi}{2}$$

Each element of $\vec{q}_T = [q_4 \ \ q_5 \ \ q_6]$ represents the needed displacement on the x, y, and z axes respectively.

Given $\mathbf{R}'(\vec{q}_R)$ and $\vec{q}_T$ we can correct the position estimation as follows:
Let $\mathbf{R}_e$ and $\mathbf{T}_e$ be the estimated transformation parameters, and let $\vec{Cp}_{e_i}$ be the expected position of point i (prior to correction) in the camera coordinate frame, then the corrected position in camera centered coordinates will be:

$$\vec{Cp}_i = \mathbf{R}'(\vec{q}_R)\vec{Cp}_{e_i} + \vec{q}_T = \mathbf{R}'(\vec{q}_R)\mathbf{R}_e(\vec{P}_i + \mathbf{T}_e) + \vec{q}_T =$$

$$\mathbf{R}'(\vec{q}_R)\mathbf{R}_e(\vec{P}_i + (\mathbf{T}_e + \mathbf{R}_e^{-1}\vec{q}_T))$$

From which we can see that

$$\mathbf{R} = \mathbf{R}'(\vec{q}_R)\mathbf{R}_e \quad \text{and} \quad \mathbf{T} = \mathbf{T}_e + \mathbf{R}_e^{-1}\vec{q}_T$$

The rotation angles can then be found by decomposing $\mathbf{R}$ (see appendix B).

3.2 Reverse projection

In this section we present the rigidity constraint, our contribution to the pose determination technique, used to establish a 3D point from a 2D projection based on the model information.
Given a set of image points $\{I_{d_i}\}_{i=1..n}$ matching the set $\{C_{m_i}\}_{i=1..n}$ of model points in camera centered coordinates (see Figure 4), we first apply a reverse rasterization and origin transformation:

$$x_{d_i} = \frac{Ix_{d_i} - Ix_0}{k_x} \quad \text{and} \quad y_{d_i} = \frac{Iy_{d_i} - Iy_0}{k_y}$$

where Ix_{d_i} and Iy_{d_i} are the x and the y pixel coordinates respectively of point i, x_{d_i} and y_{d_i} are the x and the y coordinates in image plane and in regular units of point i (prior to rasterization – see Figure 13), and k_x and k_y are the horizontal and vertical single pixel dimensions respectively (see appendix A).

The problem is now to transform projected 2D points $\{p_{d_i}\}_{i=1..n}$ to 3D space. The 3D points must lie on the line connecting the projected points on the image plane, and

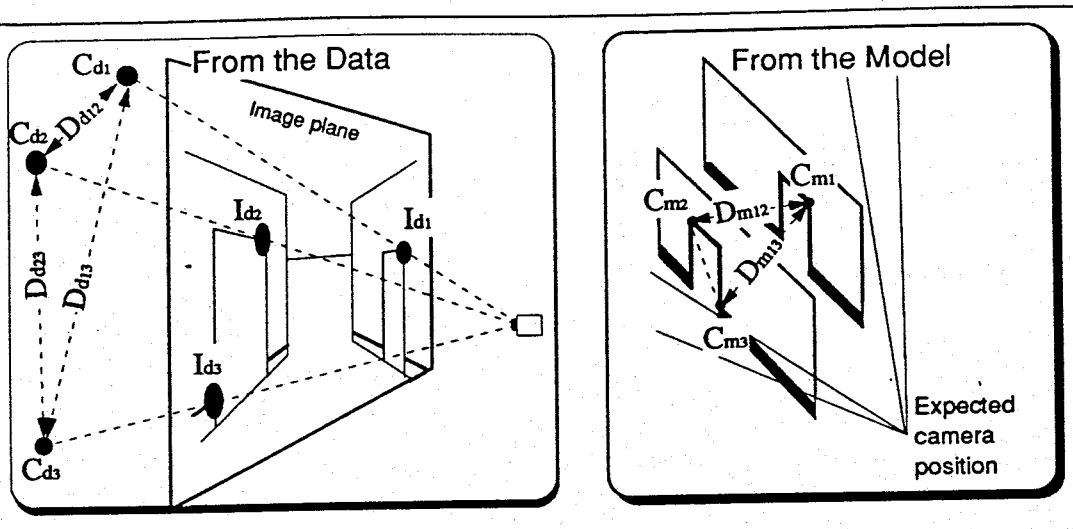

Figure 4: The model and data parameters used in the reverse projection

the focal point of the camera. The only constraint that can be imposed on the reverse projected points (as the observer's position is unknown) is a relative constraint between the 3D points. We introduce the rigidity constraint in which we try to minimize the difference between 3D distances of pairs of estimated reverse projected data points to the 3D distance between the matching 3D model points. The idea is to find a set of points which is a reverse projection of the set of the 2D points Ix_{d_i} , and which also maintains the same relative positions between the points as the set of 3D model points.

Formally the constraint can be presented as follows: let $D_{m_{ij}}$ be the 3D distance between two model points i, j, and let $D_{d_{ij}}$ be the 3D distance between the two reverse projected matching data points (see Figure 4). The objective function E to minimize is:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{n} \left| D_{m_{ij}} - D_{d_{ij}} \right|$$

To simplify and shorten the computation we define a different objective function E_1 which is the one we are actually minimizing. Here we are minimizing the sum of the differences between the square of the distances. The advantage is in the fact that the square roots need not be computed. Also the derivatives of the objective function E_1 are considerably simpler to compute. The difference in the summation indices indicate only that every distance is used once rather than twice.

$$E_1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (D_{m_{ij}}^2 - D_{d_{ij}}^2)^2$$

We formulate the problem as a multidimensional minimization problem in the n-space of the Cz_{d_i} (z values of the 3D data points).
We first compute the constant model distances:

$$D^2_{m_{ij}} = (Cx_{m_i} - Cx_{m_j})^2 + (Cy_{m_i} - Cy_{m_j})^2 + (Cz_{m_i} - Cz_{m_j})^2$$

and we define the data distances as functions of the z values (Cz_{d_i}):

$$D^2_{d_{ij}} = \frac{1}{F^2}\left[(Cz_{d_i}x_{d_i} - Cz_{d_j}x_j)^2 + (Cz_{d_i}y_{d_i} - Cz_{d_j}y_j)^2 + (Cz_{d_i}F - Cz_{d_j}F)^2\right]$$

The objective function is:

$$E_1 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}$$

$$\left\{D^2_{m_{ij}} - \frac{1}{F^2}\left[(Cz_{d_i}x_{d_i} - Cz_{d_j}x_j)^2 + (Cz_{d_i}y_{d_i} - Cz_{d_j}y_j)^2 + (Cz_{d_i}F - Cz_{d_j}F)^2\right]\right\}^2$$

and its partial derivatives are:

$$\frac{\partial E_1}{\partial Cz_{d_i}} = -\frac{4}{F^2}\sum_{j=1}^{n}$$

$$\left\{D^2_{m_{ij}} - \frac{1}{F^2}\left[(Cz_{d_i}x_{d_i} - Cz_{d_j}x_j)^2 + (Cz_{d_i}y_{d_i} - Cz_{d_j}y_j)^2 + (Cz_{d_i}F - Cz_{d_j}F)^2\right]\right\} \cdot$$

$$[(Cz_{d_i}x_{d_i} - Cz_{d_j}x_j)x_{d_i} + (Cz_{d_i}y_{d_i} - Cz_{d_j}y_j)y_{d_i} + (Cz_{d_i}F - Cz_{d_j}F)F]$$

To solve this multidimensional minimization problem we employ the Polak-Ribiere variant of the conjugative gradient methods in multi-dimensions (see [18] chapter 10.6). As this is an iterative technique we need to define an initial guess for the z values. Since we assume that the expected values will be close to the correct values, we use the model values $\{Cz_{m_i}\}_{i=1..n}$ as the initial guess.

3.3 Matching image features to model features

The components described in the previous 2 sections will work precisely if the image based matching is reliable. Small errors due to noise or digitization do not have any drastic effect on the results. On the other hand, since the technique is least-squares based, outliers can introduce intolerable errors in the results.

Our matching technique was initially based on searching through small regions in the image space for particular oriented features (lines), and their intersections. The features to

be looked for are picked exclusively from the local model, and are projected on the image plane where they are expected to appear. The features of interest are indicated by the generic object models as potentially detectable features. A local oriented-line-detection scheme is then employed to detect these features. The weakness of this technique lies in the fact that it is based on local information only. If the expectation deviates considerably from the reality, outliers are introduced and the solution is not as reliable any more (see Figure 7). In particular, relatively small rotation errors generate large translations in the image plane. This creates problems for our expectation based feature search in the image plane.

Clearly the local matching technique needs to be enhanced. We suggest 3 potential solutions to this problem:

1. Introduce a global consistency check in the 2 dimensional space. This will eliminate gross matching errors. The global consistency can be obtained by using the available model at different levels:

   (a) By using the detailed a priori model, one could generate an expected 2D primitive image (e.g. a stick figure or line drawing) which could be matched in the 2D image. Such a technique is described in [4].

   (b) By using a generic model of the environment, one could employ a model-based image interpretation technique to look for features of interest, then build a primitive 3D model based on the detected features and use it as a ruler to detect *local inconsistencies*. For example, if the system knows it is in a particular environment, such as a building, a *door detection* technique could be employed. We have implemented a simple such technique for hypothesizing door positions in corridors in our building to compare with the available a priori detailed model. A similar approach is described in [9, 10].

2. Introduce simple but less precise pose correction techniques prior to the pose correction technique described above. In particular, correcting the expected camera orientation will help in establishing correct matches. We have implemented a simple orientation correction technique based on vanishing points in the image plane. This simple technique, which is described in section 4, could be used in structured environments where dominant orientations, such as inside of buildings, corridors, streets or even roads and highways, exist and if good models of those are available.

3. Introduce consistency tests at the next phases of reverse projection and pose determination. The idea is to use robust approaches such as median based filters instead of least-squares for the reverse projection and pose determination. We have not implemented this, but such a technique is described in [12].

3.4 The position and orientation correction algorithm

The complete algorithm we have implemented can be described as follows:

1. For a set of model-based feature points in 3D space $\{\vec{C}_{m_i}\}_{i=1..n}$ detect the corresponding 2D image feature points $\{\vec{I}_{d_i}\}_{i=1..n}$. The search for these points is constrained to small regions and particular 2D features of interest.

2. Apply the reverse projection on this set of points by using the rigidity constraint and the minimization procedure described. This produces the set $\{\vec{C}_{d_i}\}_{i=1..n}$ of 3D points.

3. Use the set of matching pairs of data and model points to determine the registration vector $\vec{q}$ from which all the transformation parameters can be extracted as described.

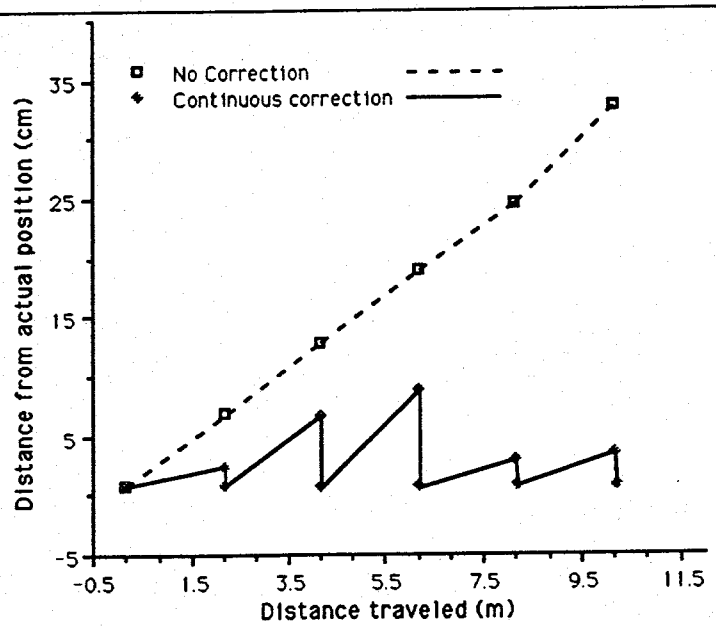

Figure 5: Instantaneous corrected errors versus accumulated errors between robot's expected position (internal model) and the real position.

We have implemented this algorithm for both manual and automatic point matching (for step 1). Figure 1 shows an example with a real camera image on the left, and a simulated view from the expected position on the right. This picture shows the registration after a pose correction. Feature lines from the model image are superimposed on the real image to visually examine the quality of the match. Problems with this technique arise

only in the automatic point matching when the expectation considerably differs from reality. One solution to this problem is introduced in the following section.

Figure 5 shows a graph of the distance deviation between the correct and expected positions for accumulated error and for a constantly corrected error. The data was collected over a run of equal type and length. This graph demonstrates how the pose correction technique keeps the deviation in a tight envelope. Without the constant correction, the deviation for the same motion sequence deteriorates linearly. With the large final deviation for the uncorrected motion it would have been much more difficult to use the expectations to restrict the 2D search space.

4 Vanishing Point Analysis for Fast Orientation Correction

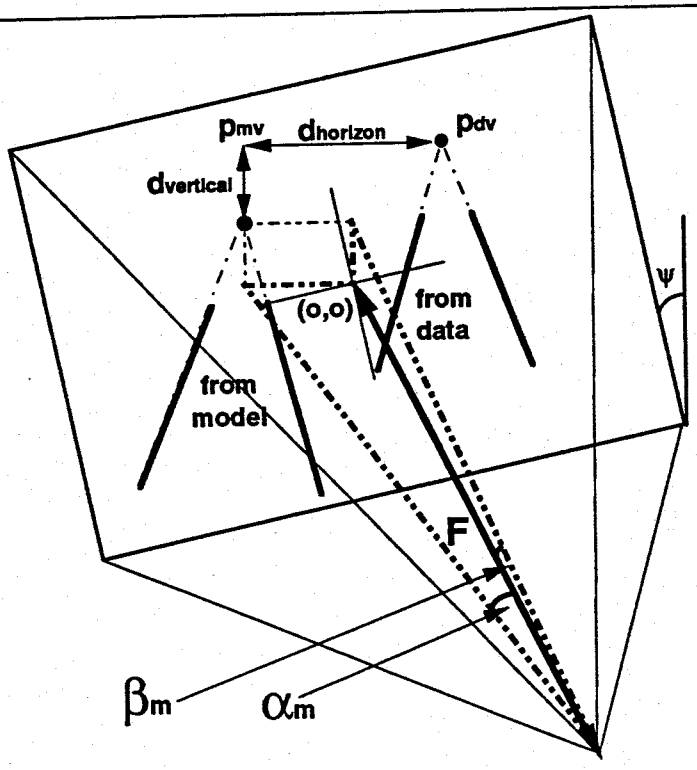

Figure 6: Correcting pan (θ) and tilt (ϕ) angles using the deviation of vanishing point position in the image plane. The horizontal and vertical distances between the projected point and the expected projected point provides the angular displacements for the pan and tilt.

This component is used in our system as a preprocessing stage to the correction scheme described in the previous section. It provides a faster yet cruder technique for orientation angle estimation, allows the system to restrict the search space in the feature matching phase, and helps to prevent local inconsistencies. This component uses the deviation of the position of a vanishing point in the image from its expected position to correct the pan and tilt angles. It assumes that the expected position and the swing angle are close to their actual values. These are reasonable assumptions for the type of system we are using it for. This technique also assumes that there are dominant features (preferably parallel lines) for which a vanishing point can be found.

For the following description we shall assume that there exists a pair (or more) of dominant parallel lines in the environment. The technique, nevertheless, does not depend on that particular set of features, rather on the existence of a point in 2D space representing an infinitely far point in 3D. Similarly a far enough distinguishable real landmark point could be used as well.

Given a set of dominant parallel features in the 3D model we project (based on the expected camera position) the model lines onto the image plane. Let I_{m_v} represent the 2D intersection point of the lines in the image plane (in pixel units). This point is converted to the 2D point p_{m_v} in the image plane (in regular units centered around the focal axis, see appendix B).

Based on the projection of the 3D lines, a simple oriented line search is applied to the image and the dominant features are detected. Let I_{d_v} represent the 2D intersection point of the data detected lines in the image plane. This point is converted to the 2D point p_{d_v} in the image plane in regular units.

Let $dx = x_{d_v} - x_{m_v}$ and $dy = y_{d_v} - y_{m_v}$. Since the image may be tilted by ψ this is converted to horizontal and vertical deviations (see figure 6).
Let $d_{\text{horizon}} = dx\cos\psi - dy\sin\psi$ and $d_{\text{vertical}} = dx\sin\psi + dy\cos\psi$.

Let α_m be the expected horizontal angle to the vanishing point p_{m_v} (see Figure 6).

$$\alpha_m = \tan^{-1}\left[\frac{-(x_{m_v}\cos\psi - y_{m_v}\sin\psi)}{F}\right]$$

Similarly

$$\alpha_d = \tan^{-1}\left[\frac{-(x_{m_v}\cos\psi - y_{m_v}\sin\psi + d_{\text{horizon}})}{F}\right]$$

The expected pan angle is corrected as follows:

$$\theta = \theta_e - (\alpha_d - \alpha_m)$$

where θ_e is the expected pan angle.

We perform a similar operation for the tilt angle. Notice, however, the difference in signs in the definition of the angles α and β, due to the way these angles have been defined for consistency with the definitions of θ and ϕ.
Let β_m be the expected vertical angle to the vanishing point p_{m_v}

$$\beta_m = \tan^{-1}\left[\frac{(x_{m_v}\sin\psi + y_{m_v}\cos\psi)}{F}\right]$$

Similarly

$$\beta_d = \tan^{-1}\left[\frac{(x_{m_v}\sin\psi + y_{m_v}\cos\psi + d_{\text{vertical}})}{F}\right]$$

The expected tilt angle is corrected as follows:

$$\phi = \phi_e - (\beta_d - \beta_m)$$

where ϕ_e is the expected tilt angle.

This technique was found out to be very reliable in performing initial orientation angle correction. The platform we were using has a damping mechanism which affects the tilt of the camera considerably. With this technique we were able to overcome these problems. Figure 7 shows an example of a registration before this correction. It shows the superimposed dominant features from the model projected onto the real image.

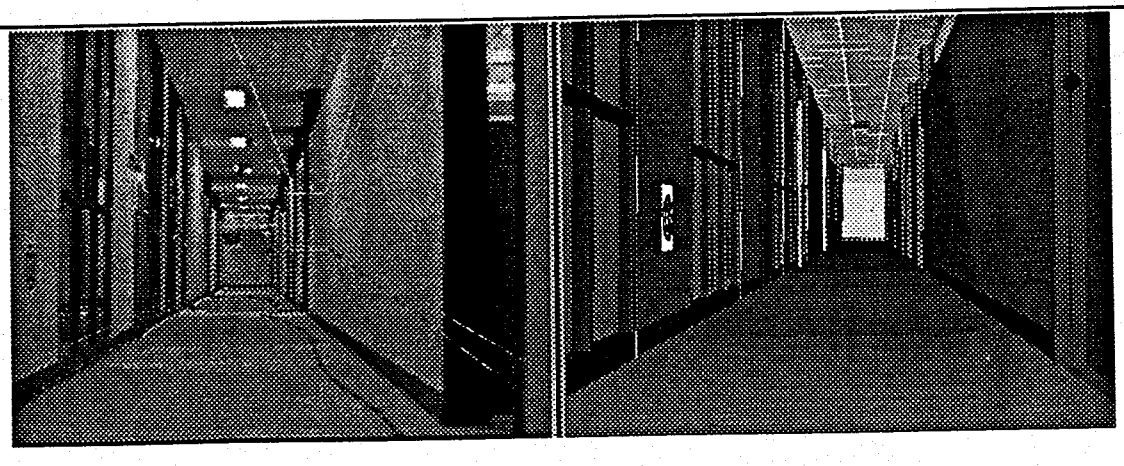

Figure 7: Real data and model view before orientation correction. The superimposed lines from the model on the image show the deviation in the 2D space due to this angular discrepancy.

Figure 8 shows a graph of the angular deviation for the constantly corrected case and for the uncorrected case. This graph demonstrates the fact that the instantaneous angular error is large enough to make the expectation based matching more difficult at every step

if the angular error is not corrected first. For example with a lens with focal length of 25mm and a 640×480 pixels image, a deviation of 1.0 degree for the tilt angle results in a translation on the order of over 20 pixels in the image plane. It is also interesting to note that the angular errors (without correction) do not accumulate as badly as the linear distance errors. This is particularly true for our system in which, due to the mechanics of the robot, the tilt angle could deviate relatively drastically for short movements, but was restricted to a definite envelope of values at any given instant.

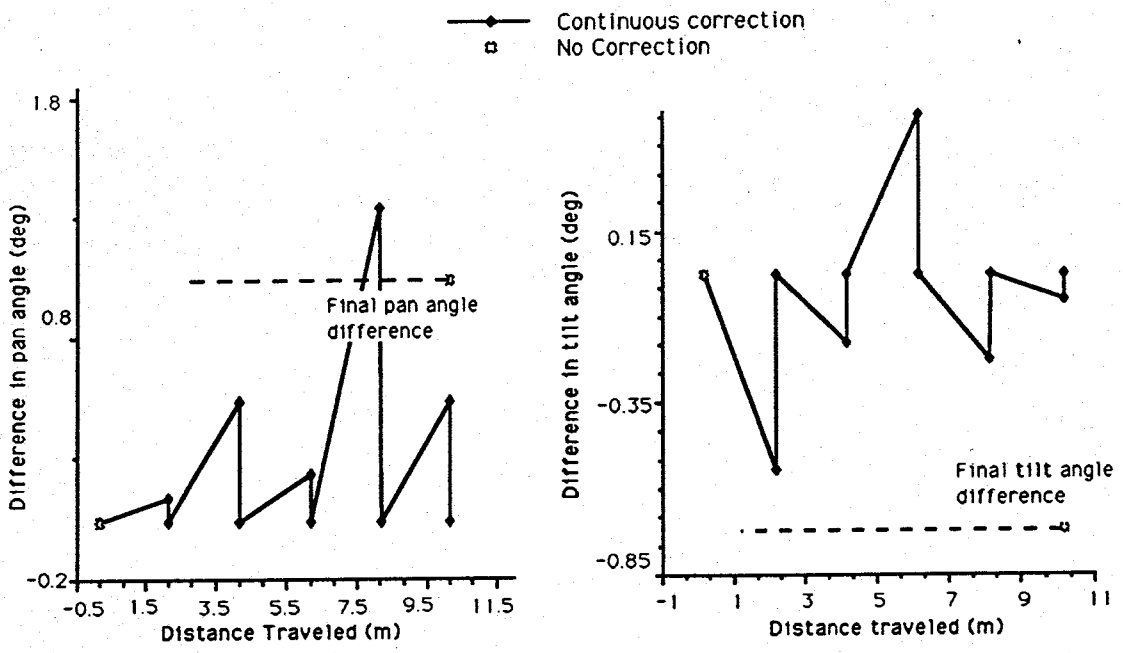

Figure 8: Instantaneous corrected errors versus accumulated errors between robot's expected orientation (internal model) and the real orientation (for the pan and tilt angles).

5 Camera Calibration

In the components outlined above we have had to use the following camera parameters: F – the effective focal length, and k_x and k_y – the x and y pixel dimensions. This section describes the component which uses the same a priori geometric model to calibrate for these camera parameters. When this component is activated, the system chooses a set of 3D lines and projects them onto a simulated camera view. The lines are not necessarily physical lines in the environment, but rather lines that connect feature points. The system chooses the lines which will project the longest 2D lines on the image. This improves the

quality of the results. The corresponding lines are then indicated on the real image and the parameters are computed. This routine is invoked at the initialization, or whenever the optical system or the camera have been modified in some way. This component assumes that the position of the sensor is known reliably. It could be performed manually (for picking the proper lines in the real image) or automatically by employing 2D matching techniques that can deal with scaled objects. We have currently implemented only the manual matching technique.

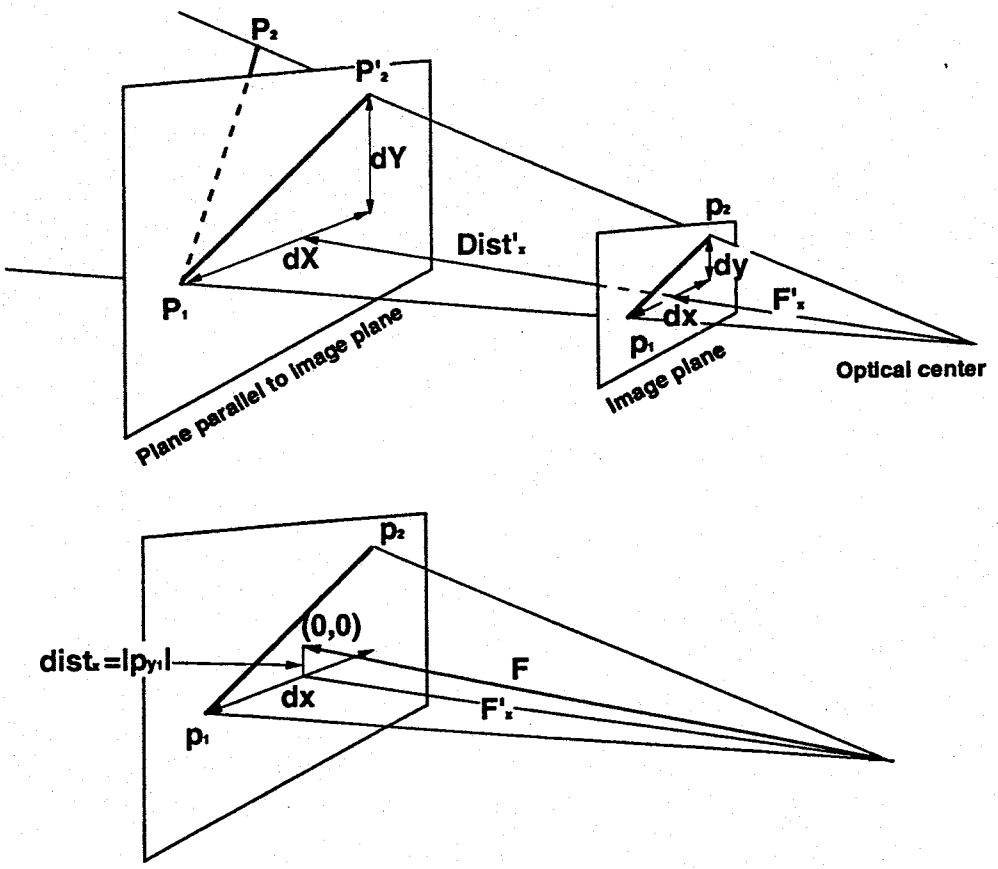

Figure 9: Calibration by using model line segments. A projected segment is compared to the parallel corresponding 3D segment. To solve for the different scaling in the x and y image dimensions, the segments are broken into their x and y components.

It is assumed that the swing angle ψ is 0, or is negligible. In any event, the calibration will exhibit graceful degradation with respect to deviations in ψ which will cause errors in the determination of k_x and k_y, but will not affect the calculation of F. Since most camera systems attempt to provide close to square pixels ($k_x = k_y$), this error can be shown to be insignificant for even larger than reasonable deviations in ψ.

Given a set of matched segments (2 points) $\vec{P}_{m_i}, \vec{P}_{m_j}$ obtained from the model, and the corresponding segments in the image plane $\vec{I}_{d_i}, \vec{I}_{d_j}$, we perform the following operations: Consider one line segment $[\vec{P}_1, \ \vec{P}_2]$ as shown in Figure 9. We first obtain the point $\vec{P'}_2$, which is the intersection point of the line from the camera position $\vec{P}_c$ to point $\vec{P}_2$ with the plane parallel to the image plane passing through point $\vec{P}_1$ (i.e. $\vec{P}_1\vec{P'}_2$||imageplane).

To find point $\vec{P}'_2$, let $A \cdot X + B \cdot Y + C \cdot Z + D = 0$ be the equation of the plane passing through $\vec{P}_1$. $[A \ \ B \ \ C]^t$ is the normal vector to the plane, and can be determined from the pan and tilt angles as follows (refer to Figure 12 for the definition of the angles):
$A = \cos\theta\cos\phi$
$B = \sin\phi$
$C = -\sin\theta\cos\phi$
and D is found by $D = -(A \cdot X_1 + B \cdot Y_1 + C \cdot Z_1)$.

The line $\vec{P}_2$ to $\vec{P}_c$ can be defined in parametric form:

$$\begin{cases} X = X_c + (X_2 - X_c) \cdot t \\ Y = Y_c + (Y_2 - Y_c) \cdot t \\ Z = Z_c + (Z_2 - Z_c) \cdot t \end{cases}$$

We solve for the parameter t:

$$t = \frac{-(AX_c + BY_c + CZ_c)}{A(X_2 - X_c) + B(Y_2 - Y_c) + C(Z_2 - Z_c)}$$

and we find $\vec{P}'_2 = [X(t) \ \ Y(t) \ \ Z(t)]^t$.

Next the problem will be separated for the x and the y components as shown in Figure 9. Let dX be the x distance component between $\vec{P}_1$ and $\vec{P}'_2$, and dY be the y component.
Following is a derivation for the x component only:
Let Dist'_x be the distance between the segment dX (from $\vec{P}_1$), and the optical center (camera position) $\vec{P}_c$. Let dx be the projection of dX on the image plane, and let F'_x be the distance between the line segment dx to the optical center. Let dist_x be the distance on the image plane between the center of the image and the dx segment.

From similar triangles we can deduce that:

$$F'_x = \frac{dx \cdot \mathrm{Dist}'_x}{dX}$$

following:

$$F^2 = (F'_x)^2 - \mathrm{dist}_x^2 = \left(\frac{dx \cdot \mathrm{Dist}'_x}{dX}\right)^2 - y_1^2 =$$

$$= \left(\frac{\mathrm{Dist}'_x}{dX}\right)^2 \cdot \frac{(Ix_2 - Ix_1)^2}{k_x^2} - \frac{(Iy_1 - Iy_0)^2}{k_y^2}$$

This is a linear equation of the three unknowns F^2, $1/k_x^2$, and $1/k_y^2$. All the rest of the parameters can be computed from the given data: Ix_1, Ix_2, Iy_1, and Iy_2 are given as inputs to the problem. $dX^2 = (X'_2 - X_1)^2 + (Z'_2 - Z_1)^2$, and Dist'_x is computed as the shortest distance between a line and a point as follows:
The equation of the line segment dX in parametric form is

$$\begin{cases} X' = X_1 + (X'_2 - X_1)t' \\ Y' = Y_1 \\ Z' = Z_1 + (Z'_2 - Z_1)t' \end{cases}$$

we shall minimize the distance between the point $\vec{P}_c$ and any point on the line segment with respect to t' and find that:

$$t' = \frac{(X_c - X_1)(X'_2 - X_1) + (Z_c - Z_1)(Z'_2 - Z_1)}{(X'_2 - X_1)^2 + (Z'_2 - Z_1)^2}$$

We use t' to find X', Y', and Z' from the line equation, and

$$(\mathrm{Dist}'_x)^2 = (X'(t') - X_c)^2 + (Y_1 - Y_c)^2 + (Z'(t') - Z_c)^2$$

A similar approach can be derived around the y dimension to obtain another linear equation of the three unknowns:

$$F^2 = \left(\frac{\mathrm{Dist}'_y}{dY}\right)^2 \cdot \frac{(Iy_2 - Iy_1)^2}{k_y^2} - \frac{(Ix_2 - Ix_0)^2}{k_x^2}$$

This technique requires at least two such different line segments to obtain 3 or more linear equations of the unknowns necessary for the solution. We have obtained good results by using exactly three independent equations. One could however use as many equations as can be derived, and use a least-squares (or median least-squares) techniques for an over-constrained set of linear equations. Figure 10 shows the the real and the model images prior to calibration with wrong parameters. Figure 11 shows the the real and the model images after the calibration parameters have been detected and the simulated projection corrected.

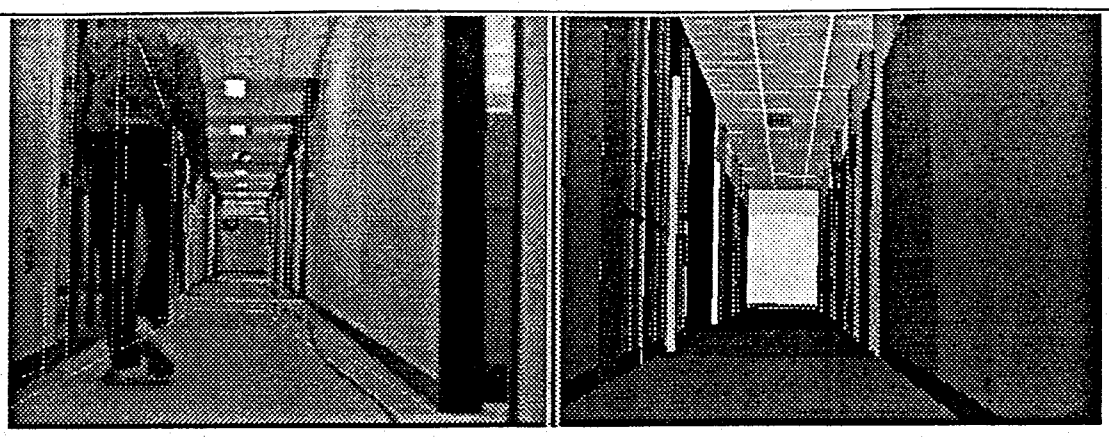

Figure 10: Real data and model view before calibration with erroneous camera parameters

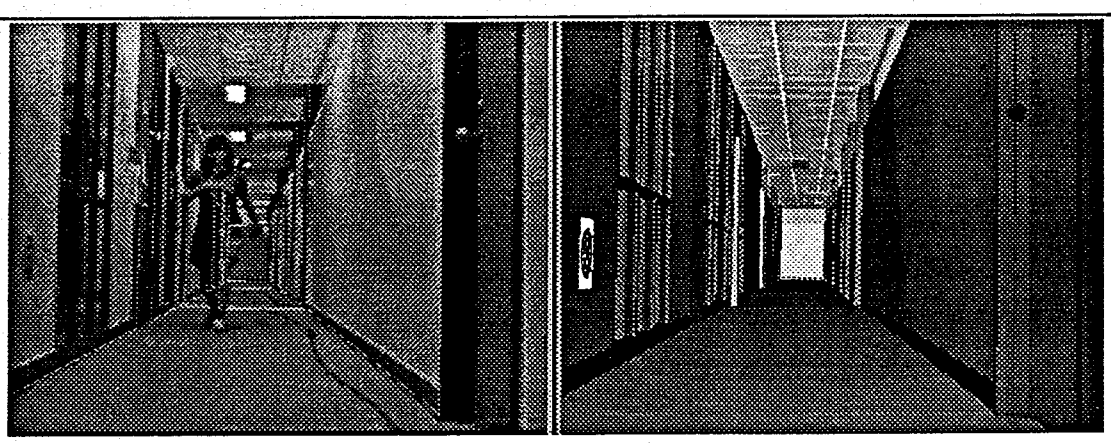

Figure 11: Real data and model view after calibration of the camera parameters

6 Conclusion

We have presented a technique for pose correction based on an a priori model of an environment and a lower accuracy expectation for the pose. Along with this technique, we presented a faster preprocessing mechanism for orientation angle correction based on the displacement of vanishing points in the image plane. This mechanism is used to correct the expectation and help in constraining the search space for matching model features to image features. We also presented a mechanism for the calibration of the necessary camera parameters used by the above routines. We have implemented and experimented with all these techniques within COSIM – a simulation and control program for mobile platforms. The models of the environment are represented in the CbC (Context-based Caching) framework which dynamically maintains the 'local' (necessary) information.

Using this knowledge representation framework, the techniques described in this paper always had access to the observable features, whereas the non-local features were 'hidden' from it.

Our intention in this paper was to provide a self contained document describing our approach and this technique. We have obtained good experimental results, and we believe that other practitioners could benefit from this document as well.

References

[1] K. S. Arun, T.S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(5):698–700, September 1987.

[2] Paul J. Besl and Neil D. Mckay. A method for registration of 3-d shapes. Accepted for publications in IEEE Trans. Pattern Analysis and Machine Intelligence, 1991.

[3] J. Ross Beveridge, R. Weiss, and E. M. Riseman. Optimization of 2-dimensional model matching. In *Proc. DARPA Image Understanding Workshop*, pages 815–830, May 1989.

[4] J. Ross Beveridge, R. Weiss, and E. M. Riseman. Combinatorial optimization applied to variable scale 2d model matching. In *10th International Conference on Pattern Recognition*, pages 18–23. IEEE, June 1990.

[5] C Fennema, A. Hanson, E. Riseman, J. R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. Technical Report COINS TR 90-42, Department of Computer and Infromation Science, University of Massachuaetts at Amherst, 1990.

[6] Sundaram Ganapathy. Decomposition of transformation matrices for robot vision. In *Proc. IEEE International Conference on Robotics*, pages 130–139, March 1984.

[7] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, April 1987.

[8] Ramesh Jain and Yuval Roth. Towards integrated autonomous systems. In *International Symposium on Optical Engineering and Photonics in Aerospace Sensing, Proc. Applications of Artificial Intelligence IX,SPIE Vol. 1468 Orlando, FL*, pages 188 – 201, April 1991.

[9] David J. Kriegman and Thomas O. Binford. Generic models for robot navigation. In *IEEE Int Conf. on Robotics and Automation*, pages 746–751, 1988.

[10] David J. Kriegman, Ernst Triendl, and Thomas O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Trans. Robotics and Automation*, 5(6):792–803, December 1989.

[11] Rakesh Kumar. Determination of camera location and orientation. In *Proc. DARPA Image Understanding Workshop*, pages 870–879, May 1989.

[12] Rakesh Kumar and Allen R. Hanson. Robust estimation of camera location and orientation from noisy data having outliers. In *Workshop on Interpretation of 3D Scenes*, pages 52–60. IEEE, November 1989.

[13] Reimar K. Lenz and Roger 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(5):713–720, September 1988.

[14] Seppo Linnainmaa, David Harwood, and Larry S. Davis. Pose determination of a three-dimensional object using triangle pairs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(5):634–647, 1988.

[15] Shih-Ping Liou and Ramesh C. Jain. Road following using vanishing points. *Computer Vision, Graphics and Image Processing*, (39):116–130, 1987.

[16] Yuncai Liu, Thomas S. Huang, and O. D. Faugeras. Determination of camera location from 2d to 3d line and point correspondences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 82–88, June 1988.

[17] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[18] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[19] Y. Roth and T. Weymouth. Using and generating environment models for indoor mobile robots. In *MVA '90 IAPR Int. Wksp. on Machine Vision Applications, Tokyo, Japan*, pages 343–346, November 1990.

[20] Yuval Roth and Ramesh Jain. Context-based caching for control and simulation in mobile platforms. In *SimTec'91, Simulation Technology International Conference*. The Society for Computer Simulation (SCS), October 1991.

[21] Roger Y. Tsai. An efficient and accurate camera callibration technique for 3d machine vision. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 364–374, June 1986.

Appendices

A Notation

Throughout this document we consistently use right handed coordinate systems. The image plane is defined as x-axis: horizontal and y-axis: vertical. In 3D the y-axis remains the vertical axis, and the xz-plane is the horizontal plane.

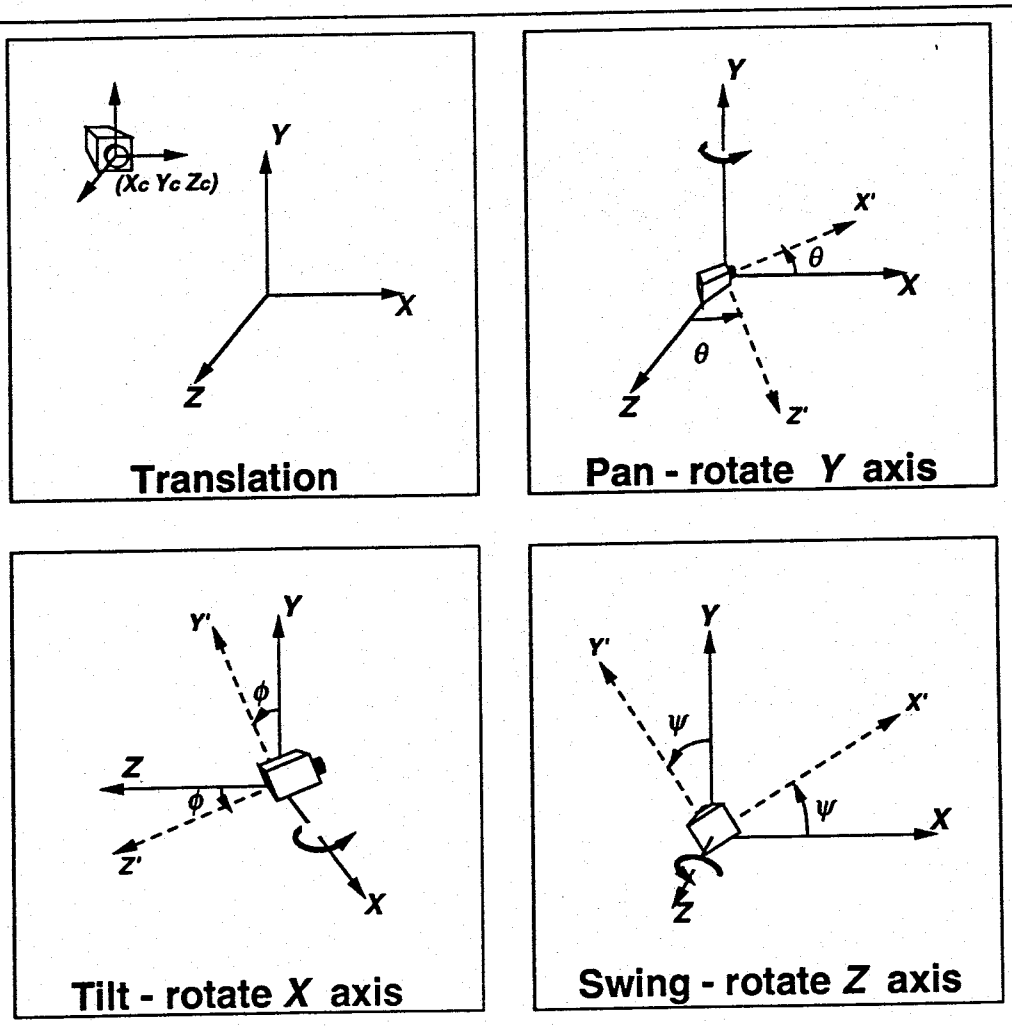

Figure 12: Rotation angles definition, and transformations to camera centered coordinate system

In the notation subscripts $_i$ or $_j$ represent an index number. We use $\vec{P_i} = [X_i \;\; Y_i \;\; Z_i]^t$

to represent points in 3D space in global coordinates. Specifically $\vec{P}_c = [X_c \;\; Y_c \;\; Z_c]^t$ is the camera position in the global coordinate system. The camera position is defined as the position of the optical center of the camera. $\vec{Cp}_i = [Cx_i \;\; Cy_i \;\; Cz_i]^t$ represents points in 3D space in camera centered coordinates. We use θ, ϕ, and ψ to represent the *pan, tilt,* and *swing* angles respectively (see Figure 12).

We assume an ideal pin hole camera model (perspective projection). F represents the effective focal length of the camera in regular units. $\vec{p}_i = [x_i \;\; y_i]^t$ represents points in 2D space on the image plane in the same unit as the 3D points (prior to rasterization). $\vec{I}_i = [Ix_i \;\; Iy_i]^t$ represents points in 2D image space in image units (pixels) relative to the image origin (bottom left pixel). *XSIZE* is the number of image pixels in x dimension. Similarly, *YSIZE* is the number of image pixels in y dimension. Pixels on the image plane are indexed from 0, and the coordinate center is at the bottom left of the image. k_x is the size of the x dimension and k_y is the size of the y dimension of a single pixel in the regular units. $\vec{I}_0 = [Ix_0 \;\; Iy_0]^t$ represents the 2D intersection point between the optical axis and the image plane in the image coordinate system. We assume the ideal case where the optical axis of the camera system intersects the image at the center of the image i.e.: $Ix_0 = (XSIZE/2.0 - 0.5)$ and $Iy_0 = (YSIZE/2.0 - 0.5)$.

We use the additional subscripts $_d$ and $_m$ to distinguish data derived points from model derived points whenever such a distinction is necessary.

B Transformations and Projections

A good reference for decomposition of transformation matrices can be found in Ganapathy [6]. Nevertheless, we shall define the necessary transformations here for the sake of completeness and consistency with the rest of our notation.

B.1 Transformation to camera centered coordinate system

Translation of coordinate system

$$\mathbf{T} = -\vec{P}_c$$

where $\mathbf{T}$ is the translation vector.

Rotation around camera centered coordinate system From the definition of the rotation angles, as defined in Figure 12, the rotation matrix $\mathbf{R}$ can be developed (applying pan, tilt, and then swing):

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

$$\begin{bmatrix} \cos\theta\cos\phi + \sin\theta\sin\phi\sin\psi & \cos\phi\sin\psi & -\sin\theta\cos\psi + \cos\theta\sin\phi\sin\psi \\ -\cos\theta\sin\psi + \sin\theta\sin\phi\cos\psi & \cos\phi\cos\psi & \sin\theta\sin\psi + \cos\theta\sin\phi\cos\psi \\ \sin\theta\cos\phi & -\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

and

$$\vec{Cp}_i = \mathbf{R}(\vec{P}_i + \mathbf{T})$$

The reverse transformation can be easily developed as well:

$$\mathbf{R}^{-1} =$$

$$\begin{bmatrix} \cos\theta\cos\phi + \sin\theta\sin\phi\sin\psi & -\cos\theta\sin\psi + \sin\theta\sin\phi\cos\psi & \sin\theta\cos\phi \\ \cos\phi\sin\psi & \cos\phi\cos\psi & -\sin\phi \\ -\sin\theta\cos\psi + \cos\theta\sin\phi\sin\psi & \sin\theta\sin\psi + \cos\theta\sin\phi\cos\psi & \cos\theta\cos\phi \end{bmatrix}$$

and

$$\vec{P}_i = \mathbf{R}^{-1}\vec{Cp}_i - \mathbf{T}$$

B.2 Projection to image plane

The 3D points are projected to the image plane based on a perspective projection:

$$x_i = \frac{Cx_i F}{-Cz_i} \quad \text{and} \quad y_i = \frac{Cy_i F}{-Cz_i}$$

Rasterization and origin transformation

Finally image coordinates in regular units are rasterized to pixel space and translated according to pixel coordinate system (left bottom of image to maintain a right handed coordinate system) – see Figure 13:

$$Ix_i = Ix_0 + k_x x_i \quad \text{and} \quad Iy_i = Iy_0 + k_y y_i$$

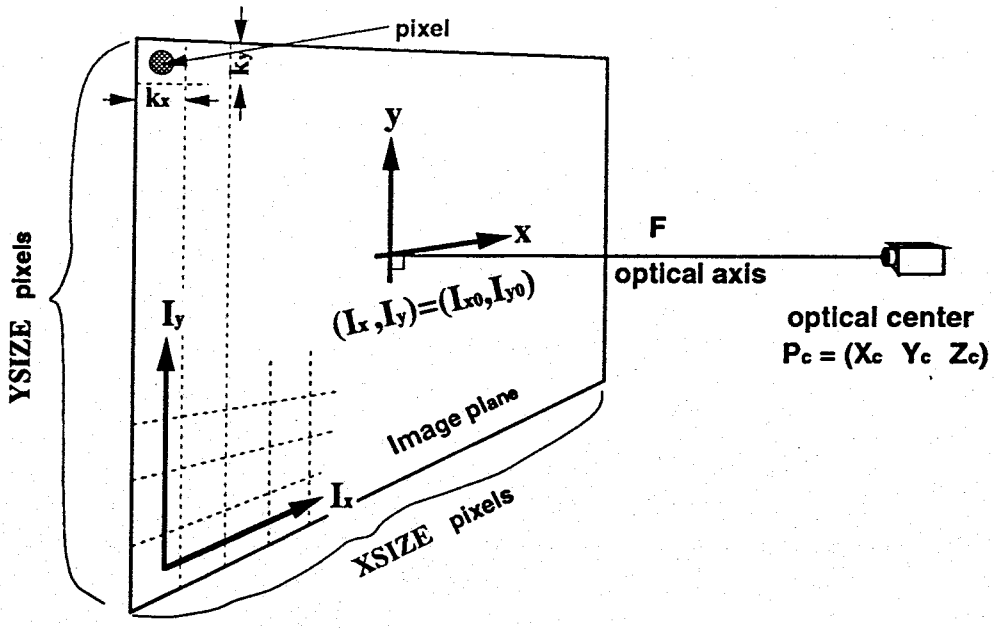

Figure 13: Rasterization and image coordinate origin translation

B.3 Decomposition of the rotation matrix R

This decomposition is useful if the rotation matrix **R** is available and the rotation angles need to be extracted from it.

Note that the tilt angle ϕ is in the range $(-\pi/2, \pi/2)$. We proceed as follows:
$\sin\phi = -r_{32}$ since the range of ϕ is limited (above) we can find ϕ.
$\sin\theta = r_{31}/\cos\phi$ and $\cos\theta = r_{33}/\cos\phi \Rightarrow \theta$
$\sin\psi = r_{12}/\cos\phi$ and $\cos\psi = r_{22}/\cos\phi \Rightarrow \psi$