Automatic Calibration For A Robot Navigation System

Z. Zhang
R. Weiss
A. Hanson

COINS TR92-70

AUTOMATIC CALIBRATION FOR A ROBOT NAVIGATION SYSTEM[1]

Zhongfei Zhang
Richard Weiss
Allen R. Hanson

Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003
Phone : (413)545-0528
NetAd : zzhang@cs.umass.EDU
October 27, 1992

Abstract

This paper attacks the problem of automatic calibration of a mobile robot in an indoor environment. The problem is that the robot must locate itself accurately with respect to given landmarks, before it starts to navigate. In this paper, we examine the problem in a special case, where the ground plane is assumed to be horizontal and there are two locally parallel side-lines available. This assumption holds in many indoor environments, such as hallways, where the system's success has been demonstrated. The algorithm uses geometric features such as vanishing points and line orientations. Both theoretical analysis and experimental results show that this algorithm works very robustly and accurately.

[1]This work was supported in part by DARPA and TACOM under contract DAAE07-91-C-R035

1. INTRODUCTION

Robot navigation has been the focus of attention of many researchers in recent years. Some of the systems need to be very accurately calibrated in order to perform navigation successfully[9, 3]. What is calibration? We quote the general definition from[2]:

Calibration is the process of determining the relationship of sensor output to the actual value of the input. It is desirable to calibrate the complete sensor system if maximum accuracy is required. One way to calibrate is to measure the values at known points and record the outputs obtained. Then, if these input-output relationships are obtained, we know how to correct the sensor at these points, at least.

In the domain of robot vision, camera calibration can be partitioned into two categories, according to two different camera parameter types: *intrinsic calibration* and *extrinsic calibration*. The former calibrates the intrinsic parameters of a camera, such as focal length, principal point, etc. The latter calibrates the extrinsic parameters like the orientation angles, pose information of the camera focal point with respect to some coordinate system.

Briefly, automatic, extrinsic calibration is any procedure which results in the robot's position and orientation being known to a specified level of accuracy based on sensory data and knowledge of the environment. There are at least two reasons why automatic calibration is needed. First of all, calibration by hand is usually tedious and time-consuming. Secondly, owing to the fact that accuracy requirements may be high, a procedure which relies on human judgement and motor skills may have a bigger error than one which is done by machine automatically. In other words, automatic calibration can achieve greater accuracy than manual.

Automatic calibration can be related to the general problem of robot pose estimation[8, 7, 10]; however, many standard pose estimation algorithms are not necessarily suitable for calibration. Kumar and Hanson[8, 7] proposed a pose estimation and refinement algorithm for computing the transform matrix between the world coordinates and image coordinates. The algorithm works very robustly in the sense that it can handle a large percentage of outliers. However, it requires the correspondence between frames to be done before the algorithm can be applied. Lee and Deng[10] also proposed an algorithm to estimate the pose information and camera parameters in a hallway environment. Like ours, their algorithm also

assumes that the ground plane is flat[2], and that there are some vertical landmarks available. There is no error analysis for their algorithm. Their algorithm also requires three images, in general, to give the pose information each time. Our algorithm, which is homing-based, uses only one image, but it assumes that the target pose information is given in advance. Lowe[11] used viewpoint consistency constraint to match a set of characteristic features against models and achieved relatively good results, although he did not report error measures.

For calibration, accuracy is usually the most important criterion, while other applications may have less stringent requirements for accuracy. For example, pose estimation can be used to determine if a goal in a plan has been satisfied, or a precondition for another part of the plan has been met. In general, these applications may have other requirements that distinguish them from the calibration task. For example, some applications may require pose estimation to be performed at arbitrary times and for arbitrary environments, while the calibration task may only need this to be performed at prespecified times, e.g. when starting the robot, and in structured environments, e.g. special features can be guaranteed to be present. Thus, the requirements for a general purpose pose estimation algorithm are not the same as those for an automatic calibration algorithm.

The technique used in this paper for calibration is based on homing [5, 14]. Homing is a procedure whose goal is to put the robot in a prespecified pose. Homing can also be applied to navigation in many different ways, e.g. as a technique for guiding the robot from the start position to a goal position. Here we are concerned with positioning the robot as accurately as possible, so the requirements for a general purpose homing algorithm are different from those for automatic calibration.

In the literature of robot navigation and calibration, many researchers have used vanishing points as an efficient tool to estimate orientation. Tsuji et al[12] developed an algorithm to estimate and compensate for the camera pitch and roll during the dynamic navigation process. This was based on the motion of the vanishing point of the vertical lines on a Gaussian sphere. Katanani[6] used vanishing points for intrinsic camera calibration. Neither methods involved the estimation of position as well as orientation.

[2] An analysis of the sensitivity of our algorithm to deviations from this assumption is presented in Section 4. In addition, the algorithm can be estended to situations in which the camera axis is not parallel to the ground plane.

The Umass Mobile Robot project uses perceptual servoing in order to maintain a straight path down the hallway and avoid collision with the walls [3, 4]. Although the system initially performed satisfactorily and robustly with manual calibration, it was a time-consuming process. Automatic calibration greatly reduced the amount of work and produced even more accurate results. The coordinate system for the hallway is illustrated in Fig. 1, and calibration is formulated as a homing problem: *Given an unknown starting pose of the robot, we are required to move the robot to the target position precisely and align the camera orientation with the axis determined by the two locally parallel side-lines on the ground plane.*

This paper presents a calibration algorithm that makes use of the geometry of the hallway environment, in particular, the two locally parallel lines that lie on the ground plane and the distance between them. Three degrees of freedom (DOF) in terms of the hallway axis are controlled in this algorithm, that is, *orientation with respect to the axis*, *lateral distance from the axis*, and *depth distance along the axis*. These three DOFs are independently measured by: *vanishing point location*, which is an efficient vehicle to estimate orientations [1, 6, 12]; *orientation of side-lines*, which enables a closed form solution on how to calculate the lateral distance; and *looming*, which is used to estimate depth information [13]. By combining those techniques, we can accurately determine the pose of a robot in this environment. Empirically, the experiments show that the automatic calibration is more precise than that done by hand. Theoretically, the error analyses of the closed form solution used in the algorithm makes it possible to predict the accuracy of results. With a little modification, this algorithm can also be applied as a pose estimation and perceptual servoing algorithm to servo the robot onto the right track during navigation.

2. GEOMETRIC ANALYSIS

In this section, we present three independent geometric measurements, and thus, three different techniques to infer the pose information of the robot during the calibration process. The three measurements which are described in terms of the hallway axis are: *orientation with respect to the axis*, *lateral distance from the axis*, and *depth distance along the axis*. Based on these measurements, three techniques are developed. They are: using the *vanishing point* position to estimate orientation [1, 6, 12]; using the *directional axis* to obtain a closed

form solution for the lateral distance; and using *looming* to estimate depth information [13]. In the following analysis, we assume that the center of the robot is the same as the focal point of the camera, and that the intrinsic parameters are known. Moreover, we assume that the environment contains locally parallel side-lines (in our experiments, we use the hallway base-board lines) and a horizontal ground plane, so that an implicit directional axis parallel to the side-lines is available. Under these assumptions, the robot works in three degrees of freedom (two translational and one rotational). Although our environmental structure and Denning Mobile Robot are not perfect, it can be seen through the theoretical analysis and the experimental results that these assumptions hold to a close approximation.

2.1 Vanishing Point

The vanishing point is defined to be the intersection point of the images of the two parallel side-lines in the environment, as shown in Fig. 2. Here side-lines are defined as two lines in an environment that are parallel with each other, and the plane formed must be parallel to the ground plane. Now, we have the following property about the vanishing point:

Property 1 *The vanishing point is located at the principal point*[3] *of the image plane, no matter where the robot is positioned, if and only if the camera's axis is parallel to the side-lines; Moreover, if the camera is not parallel to the side-lines, the locus of the vanishing point must be in the horizon line of the image plane, and the horizon line passes through the principal point of the image. The offset angle away from the side-line orientation (denoted as θ) is determined as:*

$$\theta = \arctan \frac{X_v - X_c}{S_X} \tag{1}$$

where X_v is the X coordinate of the vanishing point in the image, X_c is the X coordinate of the principal point of the image, and S_X is the scale factor of the focal length along the X axis of the image plane.

In other words, this property states that the vanishing point in the image is *only relevant* to the *orientation* of the camera, and *irrelevant* to the *position* of the robot.

[3]There is exact one point, such that the ray through focal point is perpendicular to image plane. This is called the principal point or center point of the image.

It is easily seen that this property is true. As illustrated in Fig. 3(a), assuming the robot *is positioned* at an arbitrary location, as long as the camera axis is aligned with the side-line direction, the optical axis of the camera is parallel to the side-lines. Thus, it is actually the intersection line of the projection planes of the two side-lines. If the image plane is perpendicular to the optical axis and also perpendicular to the ground plane, the intersection point of the optical axis of the camera and the image plane is exactly the intersection point of the two side-lines in the image. Therefore, this point is always located at the principal point of the image plane. Even if the camera is tilted forward or backward, the X-Coordinates will be equal.

However, if the camera is oriented at an offset angle θ away from the direction of the side-lines, as illustrated in Fig. 3(b), this angle is exactly the angle between the intersection line of the two projection planes of the two side-lines and the optical axis of the camera. Since the plane formed by the intersection line of the two projection planes of the side-lines and the optical axis (say, plane P) is parallel to the ground plane, and since the image plane is perpendicular to the ground plane, the locus of the vanishing point is the intersection line between plane P and the image plane. Thus it is the horizon bisector line of the image plane. Since the optical axis of the camera is perpendicular to the image plane, from the obvious trigonometric relationship, Eq. 1 is easily obtained.

2.2 Directional Axis

The directional axis here is referred to as the line parallel to the side-lines where the given robot target pose (or called home pose) is located, as shown in Fig. 1. Note that this line does not explicitly exist in the environment, i.e. it does not have to be marked on the floor. It is, however, implicitly existing once the starting target position is given, i.e., once we know the distances from this line to the two sides, a, and b.

Now we have a dual property with respect to **property 1**:

property 2 *The image of the directional axis is a vertical line in the image plane, no matter what direction the camera is oriented, if and only if the robot is positioned on the directional axis. Moreover, if the robot is not positioned on the implicit target directional axis, the lateral distance from the directional axis can be calculated as:*

$$d = -\frac{b \sin \beta}{\tan(\gamma + \beta) \sin \gamma} \tag{2}$$

where α and β are the directional angles of the two side-lines in the image plane, a and b are the distances from the directional axis to the two sides, respectively, as illustrated in Fig. 1 and Fig. 5, and γ is determined by:

$$\gamma = \arctan \frac{b \sin \beta \sin(\alpha + \beta)}{a \sin \alpha - b \sin \beta \cos(\alpha + \beta)} \tag{3}$$

In other words, this property states that the orientation of the directional axis in the image is *only relevant* to the *position* of the robot, and *irrelevant* to the *orientation* of the camera.

Here, the angle $\beta + \gamma$ indicates the orientation of the directional axis in the image. As shown in Fig. 5, if $\beta + \gamma < \frac{\pi}{2}$, d is negative, which means that the robot is on the right side of the directional axis with $-d$ lateral distance away from it; if $\beta + \gamma > \frac{\pi}{2}$, d is positive, which means that the robot is on the left side of the directional axis with d lateral distance away from it; if $\beta + \gamma = \frac{\pi}{2}$, $d = 0$, which means that the robot is exactly on the directional axis.

To see that **Property 2** is true, look at Fig. 4. First we consider the case when there is no offset angle involved. i.e., $\theta = 0$. In this case, the intersection line formed by the two projection planes of the side-lines coincides with the optical axis of the camera, and passes through the focal point, as shown in Fig. 4(a). Now, we term *directional axis* as the implicit line on the ground plane parallel to the side-lines and where the robot is located. Thus, the current directional axis *in the image plane* is formed by the intersection of the projection plane of the current directional axis on the ground and the image plane. Since both this projection plane and the image plane are perpendicular to the ground plane, their intersection is also perpendicular to the ground plane. That is, the current directional axis in the image is always *the implicit vertical line* passing through the vanishing point in the image. Similarly, the target directional axis in the image is formed by the intersection of the projection plane of the target directional axis on the ground and the image plane, which is another line passing through the vanishing point in the image, but not necessarily vertical. Therefore, we can determine if the robot is located at the target directional axis, in which

case the current directional axis coincides with the target one. Since the target directional axis in the image can be determined from the 3D distance from the axis on the ground to the two side-lines, we can figure out how far the current robot location is away from the target directional axis by calculating the offset angle between the target directional axis and the vertical line.

Now we go on further to see how to get the lateral distance information. Refer to Fig. 5, where O denotes the vanishing point of the two side-lines, OA and OB are the side-lines with directional angle α and β, respectively. AB is an arbitrary horizontal line so that a triangle OAB is formed.

Let OD be the implicit target directional axis in the image whose distance a and b from the side-lines are known, and OE be the vertical line passing through the vanishing point O, indicating the current *position* of the robot. Then the distance $\|DE\| = d$ is what we want to find.

Now, in $\triangle OAD$, we have

$$\frac{\|OD\|}{\sin\alpha} = \frac{a}{\angle AOD}$$

Since $\angle AOD = \pi - \alpha - \beta - \gamma$, we have

$$\|OD\| = \frac{\sin\alpha}{\sin(\alpha+\beta+\gamma)}a$$

Similarly, in $\triangle ODB$, we have

$$\|OD\| = \frac{\sin\beta}{\sin\gamma}b$$

Thus, we have

$$\frac{a\sin\alpha}{\sin(\alpha+\beta+\gamma)} = \frac{b\sin\beta}{\sin\gamma}$$

By solving this equation, we can get γ in terms of Eq. 3.

Now, in $\triangle ODE$, since

$$\angle DOE = \gamma + \beta - \frac{\pi}{2}$$

we immediately have

$$d = \|OD\| \tan \angle DOE$$

By substituting $\|OD\|$ and $\angle DOE$, we have Eq. 2.

Now we show that the orientation of the directional axis in an image is only relevant to the position of the robot in the environment, and has nothing to do with the orientation of the camera. To see that, let us assume that the robot now is located at an arbitrary position, say d lateral distance away from the target directional axis, and with θ offset camera orientation angle, as shown in Fig. 4(b). Since in this case, both the projection plane of the current directional axis and the image plane are still perpendicular to the ground plane, their intersection is also perpendicular to the ground plane. In other words, the current directional axis in the image is still always the vertical line passing through the vanishing point in the image, except that in this case, the vanishing point is not at the center of the image. Similarly, the target directional axis in the image is a line passing through the vanishing point also. Thus, we have the same conclusion: if and only if the robot is located at the target directional axis, will the target directional axis in the image be vertical.

To get the lateral distance information in this general case, we have a figure similar to Fig. 5, as shown in Fig. 4(b), with here a, b, d replaced by a', b', d', as illustrated in Fig. 6. Note we have

$$a = a' \cos\theta$$

$$b = b' \cos\theta$$

$$d = d' \cos\theta$$

By substituting them into Eq. 3 and Eq. 2, we can see that we have the same formulae for γ and d! That is, no matter what the orientation of the camera is, the orientation of the target directional axis in the image is only relevant to the current robot position. Thus, we proved **Property 2**.

2.3 Looming Distance

The problem of finding the looming distance here refers to the determination of robot position along the directional axis from the size of image features. This assumes that the

robot has been positioned on the directional axis with the correct orientation as shown in Fig. 7(a).

There are many ways to calculate the looming distance. Here we use the "conventional" approach. As illustrated in Fig. 7(b), assume there is a landmark in front of the robot camera, with height h. The image heights of the landmark when the robot is at the given target pose and when the robot is at the off-target pose are denoted as δY_t and δY_i, respectively. Let l be the looming distance, and let S_Y be the focal scale factor along the Y axis. Then, by simple triangular relationship, we have

$$l = hS_Y(\frac{1}{\delta Y_t} - \frac{1}{\delta Y_i}) \tag{4}$$

Note that this approach requires the landmark should have a height large enough. Otherwise, there would be a very large error. Sec. 4. shows this point. Fortunately, in the usual environment, there are landmarks such as doors, as shown in the Sec. 5..

3. CALIBRATION ALGORITHM

3.1 Basic Algorithm

From Fig. 1 and the discussion of the previous Section, it is straightforward to find an algorithm to solve the calibration problem in three steps:

- find the orientation offset angle and rotate that angle;
- find the offset distance to the target directional axis and move on to the target line;
- find the looming distance and move to the target pose;

However, using **Property 1** and **Property 2**, i.e., that the vanishing point and the directional axis are independent, we can combine the first two steps into one. This is much more efficient than separate executions of the two steps. Thus, the actual implementation of the algorithm takes only two loops:

Basic Algorithm

1. **Loop 1** take an image; calculate the orientation angle and the lateral distance with respect to the directional axis; reduce these quantities by rotating and/or moving the robot; repeat until these two quantities are both within their tolerated error thresholds, respectively;

2. **Loop 2** take an image; calculate the looming distance; repeat until the distance with respect to the given target pose is within an tolerated threshold; move toward the target pose by perceptual servoing;

Note that in the second loop, we use the perceptual servoing facility of our navigation system [3, 4]. This guarantees that the robot will not move off the target directional axis during its moving toward the target pose.

Since the quantities of camera orientation angle, lateral distance with respect to the target directional axis, and looming distance along the axis are calculated in closed-form solutions, the computation is very straightforward and takes little time. The basic operation that takes time is the base-board line pixel grouping for an image. Hence, if this algorithm takes n loops in the first step, and m loops in the second step, the convergence rate is in $O(n+m)$.

3.2 More Efficient Algorithm

By **Property 1** and **Property 2**, we know that given an arbitrary image, we can compute the orientation angle and the lateral distance with respect to the target directional axis simultaneously. That is the basis of the algorithm presented in the previous subsection. We can, however, go further to simplify the algorithm by noticing some more geometric constraints implied in the environment and our system. Specifically, we can compute the three geometric quantities: orientation angle, lateral distance, and looming distance from a single image!

As shown in Fig. 8(a), given an arbitrary pose of the robot, we have an image which is illustrated as the solid-lined plane in Fig. 8(a). The landmark, which has height h, projects in the image plane to a line with length $\delta Y_i'$. Assume that the orientation angle of the camera is θ, which can be calculated by Eq. 1. If we rotate the robot by $-\theta$ so that there would be no offset for the orientation angle as shown as the dashed image plane in Fig. 8(a), we could

have a new image of the projected landmark denoted as δY_i. If we know δY_i, by applying Eq. 4, we can compute the looming distance l immediately.

Now, let us extract the drawing of the geometric relationship on the plane formed by the landmark and the focal point, as illustrated in Fig. 8(b). It is clear to see that

$$\frac{\delta Y_i}{\delta Y_I'} = \frac{S_Y}{\frac{S_Y}{\cos\theta}}$$

Hence, we have the following simple relation:

$$\delta Y_i = \delta Y_i' \cos\theta \tag{5}$$

Therefore, by only one image, we can compute all the three geometric quantities. Thus, we can combine the two steps of the basic algorithm into only one step.

3.3 Summary of the Algorithms

The simplified algorithm is obviously more efficient than the basic algorithm. However, according to our experimental results, it produces more error than the basic one, although the error is still within the tolerated limit. That makes sense because the simplified algorithm uses less information (i.e., takes fewer images) than the basic algorithm, and most of the parameters are calculated *indirectly* as opposed to *direct* measurement in the basic algorithm. Since the image quantization error and the robot mechanical error can be accumulated during a series of indirect calculations, it is not surprising that the simplified algorithm produces more error than the basic one.

4. ERROR ANALYSIS

In this Section, we investigate the stability of the the closed-form solutions for the geometric properties used in our algorithm, with respect to noise or measurement error. The purpose of this is to show that the solutions obtained in this paper are reliable in the presence of noise. This is also confirmed by the experimental results.

4.1 Orientation Angle with respect to the Directional Axis

Eq. 1 is obtained under the assumption that the ground plane is flat. Owing to the real situation that the ground plane may be locally uneven, the vanishing point might not be on the horizontal scan line through the principal point. Taking that into account, the offset orientation angle θ is determined by:

$$\theta = \arctan \frac{\sqrt{(X_v - X_c)^2 + (Y_v - Y_c)^2}}{f} \tag{6}$$

where (X_v, Y_v) and (X_c, Y_c) are the coordinates of the current vanishing point and the principal point, respectively, and f is the focal length of the camera.

Now, we have

$$\triangle\theta = \frac{\partial\theta}{\partial X_v}\triangle X_v + \frac{\partial\theta}{\partial Y_v}\triangle Y_v \tag{7}$$

where

$$\frac{\partial\theta}{\partial X_v} = \frac{f}{f^2 + (X_v - X_c)^2 + (Y_v - Y_c)^2} \frac{X_v - X_c}{\sqrt{(X_v - X_c)^2 + (Y_v - Y_c)^2}} \tag{8}$$

$$\frac{\partial\theta}{\partial Y_v} = \frac{f}{f^2 + (X_v - X_c)^2 + (Y_v - Y_c)^2} \frac{Y_v - Y_c}{\sqrt{(X_v - X_c)^2 + (Y_v - Y_c)^2}} \tag{9}$$

In our system, $|X_v - X_c| \simeq 10^2, |Y_v - Y_c| \simeq 1$. Thus, $|X_v - X_c| \gg |Y_v - Y_c|$, and $f \simeq S_X = 990.970 \simeq 10^3$.

Thus, $\frac{\partial\theta}{\partial X_v} \simeq 10^{-3}$, and $\frac{\partial\theta}{\partial Y_v} \simeq 10^{-5}$. That implies that $\frac{\partial\theta}{\partial X_v}$ plays a dominant role in determining the accuracy of the estimation of θ. Thus, we can safely apply Eq.1 to estimate the value of θ even with the actual case of uneven ground plane. Moreover, with every one pixel error of X_v and Y_v, i.e., $\triangle X_v = 1, \triangle Y_v = 1$, the resulted error of the calculated orientation angle $\triangle\theta$ is still on the order of 10^{-3} radians.

4.2 Lateral Distance from the Directional Axis

From Eq. 3 and Eq. 2, we know that the lateral distance d is determined by the two variables α and β. Therefore, we have

$$\triangle d = \frac{\partial d}{\partial \alpha}\triangle\alpha + \frac{\partial d}{\partial \beta}\triangle\beta \tag{10}$$

where

$$\frac{\partial d}{\partial \alpha} = \frac{\cos\gamma\sin(\gamma+\beta)\cos(\gamma+\beta)+\sin\gamma}{\sin^2\gamma\sin^2(\gamma+\beta)} b\sin\beta\frac{\partial\gamma}{\partial\alpha} \tag{11}$$

$$\frac{\partial\gamma}{\partial\alpha} = \frac{b\sin\beta\cos 2(\alpha+\beta)-a\sin\beta}{[a\sin\alpha - b\sin\beta\cos(\alpha+\beta)]^2 + b^2\sin^2\beta\sin^2(\alpha+\beta)} b\sin\beta \tag{12}$$

$$\frac{\partial d}{\partial\beta} = \frac{\sin\beta\cos\gamma\sin(\gamma+\beta)\cos(\gamma+\beta)\frac{\partial\gamma}{\partial\beta}+\sin\beta\sin\gamma\frac{\partial\gamma}{\partial\beta}+\sin\beta\sin\gamma-\sin\gamma\cos\beta\sin(\gamma+\beta)\cos(\gamma+\beta)}{\sin^2\gamma\sin^2(\gamma+\beta)} l \tag{13}$$

$$\frac{\partial\gamma}{\partial\beta} = \frac{ab\sin\alpha\sin(\alpha+2\beta)-b^2\sin\beta\sin(2\alpha+3\beta)}{[a\sin\alpha - b\sin\beta\cos(\alpha+\beta)]^2 + b^2\sin^2\beta\sin^2(\alpha+\beta)} \tag{14}$$

For the typical data in our experiments, $\frac{\partial d}{\partial\alpha}$ and $\frac{\partial d}{\partial\beta}$ are in the range of $[10^{-3}, 1]$. If the calculation of α and β have 1° error, respectively[4], i.e. $\triangle\alpha$ and $\triangle\beta$ are on the order of 10^{-2} Radian, the resulted range of error of the lateral distance from the target directional axis will be in the range of $[10^{-5}, 10^{-2}]$ ft. In Sec. 5., we can see that the actual error of the distance never exceeds 0.1 ft.

4.3 Looming Distance along the Directional Axis

From Eq. 4, we know that h, S_Y, and δY_i are constants. Thus, the looming distance l is only the function of the variable δY_i. Since

[4]Since we used Least Mean Square method to group the line pixels, usually, the errors of these directional angles of the base-board lines are no more than 1°

$$\frac{dl}{dY_i} = \frac{hS_Y}{(\delta Y_i)^2}$$

We have

$$\triangle l = \frac{hS_Y}{(\delta Y_i)^2} \triangle(\delta Y_i) \tag{15}$$

In our experiments, we used the hallway door as the reference landmark, (see Sec. 5.). The height of the door is $h = 8.05$ ft. The camera focal scale factor along the Y axis is $S_Y = 1209.658$, in terms of the number of pixels. In the experiments, δY_i is usually around 300 pixels. Thus, the typical value for $\frac{dl}{d\delta Y_i}$ is around 10^{-1}. For one pixel error in the measurement of δY_i, i.e., $\triangle(\delta Y_i) = 1$, the resulted error of the looming distance could be up to 10^{-1} ft.

4.4 Summary of the Error Analysis

From the analysis in the previous subsections, it can be seen that the calculation of the orientation angle θ has the highest error stability, whereas the calculation of the looming distance l has the lowest, with the calculation of the lateral distance d lying at the middle. The experimental results show that the calculation of θ and d are very accurate actually. Only the estimation of l has noticeable error, usually 0.1 ft. or so. But even so, the relative deviation is still within 1% (see Sec. 5.).

5. EXPERIMENTAL RESULTS

The algorithms described above are implemented in LISP code on a Sun IV SPARC workstation.

The automatic calibration system works as follows. Initially, we have to teach the robot what the environment should look like. This step is absolutely necessary because there is no other knowledge that can tell the robot what the correct calibration is. This is achieved by moving the robot to a given pose in our navigation environment and calibrating it. This very first calibration has to be done by hand. Then we let the robot take an image. The

robot is actually only required to remember the vanishing point coordinates, and the height of a landmark (e.g. door in the hallway) of the "standard" image (We call it target image). Thus, whenever we want to do an experiment, we just have to leave the robot at an arbitrary pose in the environment. By running the algorithm presented in this paper, the robot is able to move to the target pose and calibrate itself accurately, as long as it is working in the same environment as it was first taught.

At present, the automatic calibration has been tested in our hallway environment. Fig. 9 shows model image used in our tests. Note that the cross black/white sign at the door is not used during the operation of the automatic calibration. It *is* used to check if the algorithm has worked correctly after the robot has automatically been calibrated. In other words, if the alogrithm works correctly, once the robot has been calibrated, the orientation angle should be zero. Thus, the cross center of the sign should be on the center vertical line in the image. Table 1 lists those geometric parameters used in our algorithm for this image, as well as the thresholds used to terminate the algorithm. In all the experiments, we used an CCD camera with intrinsic parameters $S_X = 990.970$ and $S_Y = 1209.658$. The principal point was measured at the very first manual calibration as $(255.540, 240.240)$ for an 512 by 480 image.

To implement the algorithm, we used some low-level processing tools to extract the two base-board lines in the image and compute the vanishing point coordinates. There are several algorithms available to compute vanishing points very robustly and accurately [1, 6]. However, since we want the real-time performance, we keep the low-level processing part simple. Fortunately, the base-board lines in a hallway image have very strong contrasts. Therefore, we applied simple Roberts' operator to detect edge pixels, and then used Least-Mean-Square method (i.e. Linear Regression) to fit a line to the edge pixels, and then to get the vanishing point. We compared the vanishing point obtained by using this simple method with that by using Collins *et al*[1]'s algorithms, and found that the differences were always within half pixel. That means the simple method can work fine in this particular environment.

Historically, we first implemented the basic version of the algorithm, which uses two frames. Table 2 records a test run of this algorithm. The robot was left at a pose in the

hallway about 4° off the correct orientation, about 1.4 ft. lateral distance from the target directional axis, and about 5 ft. away from the target pose along the target directional axis. Fig. 10 shows the image taken at that moment. After the termination of the algorithm, the robot's camera is pointing exactly at the target (i.e. the bilevel cross sign on the door, which is 41.1 ft. away from the camera location); the final actual lateral distance and the final actual looming distance were both within 0.05 ft. of the estimated values calculated by the algorithm, as shown in Table 2.[5] The symbol "–" means those quantities are not calculated yet at the current loop because they are irrelevant to the current action.

To show the efficiency of our simplified version of the algorithm, which uses one frame, we put the robot to a similar pose to the previous experiment, and ran the simplified algorithm. The operation process of the algorithm is recoreded in Table 3. The actual error is similar to that of the previous one.

6. CONCLUSION

In this paper, we presented an algorithm based on geometric properties for automatic robot calibration. The whole system has been implemented on a Denning Mobile Robot. Theoretical analysis also proves that those geometric properties used in the algorithm are robust enough for calibration. Experimental results show that the automatic calibration performed by this algorithm works more accurately than previous hand-based calibration.

The algorithm assumes that the environment contains two locally parallel side lines. It was originally designed for automatic calibration. However, it can also be applied to perceptual servoing during navigation on a road. Currently, the algorithm also assumes that the two side-lines are locally parallel and the ground plane is flat. Those assumptions are not necessary, and the algorithm can be extended to account for the angle between the ground plane and the camera axis. Therefore, the algorithm could potentially be used for outside road navigation. Future work will be to apply this algorithm in the upcoming UMass UGV Project.

[5]Since the robot is not a perfect round object, it is difficult to measure the coordinate of its center up to the precision of within 0.05 ft.

Acknowledgement

The first author would like to thank his other advisor Prof. Edward M. Riseman for his support and encouragement. Special thanks also go to Prof. Claude Fennema for his help on how to get started on his navigation system. We would also like to acknowledge J. R. Beveridge for his useful conversation with us, and Bruce Draper for his comments on this paper. Finally, We are grateful to Robert Heller, Val Conti, and Jonathan Lim for their technical help.

REFERENCES

[1] R. T. Collins and R. S. Weiss, "Vanishing Point Calculation as a Statistical Inference on the Unit Sphere", *Proc. ICCV*, Osaka, Japan, Dec., 1990.

[2] A. J. Critchlow, *Introduction to Robotics*, Macmillan Publishing Company, New York, 1985.

[3] C. Fennema and A. Hanson, "Experiments in Autonomous Navigation", *DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September, 1990, pp 772 – 781.

[4] C. Fennema, A. R. Hanson, E. M. Risemanm, J. R. Beveridge, and R. Kumar, "Towards Autonomous Navigation", *IEEE SMC*, 1990.

[5] J-W Hong, X-N Tan, B. Pinette, R. Weiss, and E. M. Riseman, "Image-Based Navigation Using 360°", *DARPA Image Understanding Workshop*, Pittsburgh, Pennsylvania, September, 1990.

[6] K. Kanatani and Y. Onodera, "Anatomy of Camera Calibration Using Vanishing Points", *IEICE Trans. Infor. Sys.*, 74-10, 1991, 3369-3378.

[7] R. Kumar and A. R. Hanson, "Pose Refinement: Application to Model Extension and Sensitivity to Camera Parameters", COINS TR90-112.

[8] R. Kumar and A. R. Hanson, "Robust Estimation of Camera Location and Orientation from Noisy Data Having Outliers", *Proc. Workshop on Interpretation of 3D Scenes*, Austin, TX, Nov. 1989.

[9] D. J. Kriegman, E. Triendl, and T. O. Binford, "Stereo Vision and Navigaiton in Buildings for Mobile Robots", *IEEE Trans. Robotics and Automation*, Vol.5, No.6, Dec. 1989.

[10] H-J Lee and C-T Deng, "Camera Model Determination Using Multiple Frames", *Proc. CVPR*, Lahaina, Hawaii, June 1991.

[11] D. G. Lowe, "The Viewpoint Consistency Constraint", *IJCV*, Vol.1, No.1, 1987.

[12] S. Tsuji, Y. Yagi, M. Asada, "Dynamic Scene Analysis for a Mobile Robot in a Man-Made Environment", *Proc. IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, March, 1985.

[13] L. R. Williams and A. R. Hanson, "Translating Optical Flow into Token Matches and Depth from Looming," *Proc. of the 2nd Intl. Conf. on Computer Vision*, Clearwater, Fl., Dec. 1988.

[14] Z. Zhang, R. Weiss, and E. M. Riseman, "Feature Matching in 360° Waveforms for Robot Navigation", *Proc. CVPR*, Lahaina, Hawaii, June 1991.

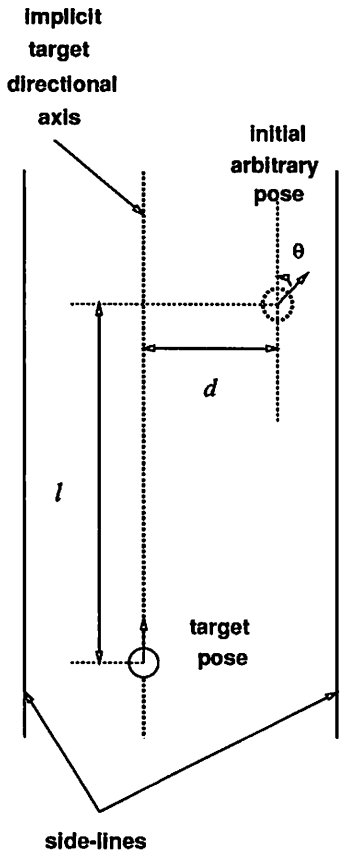


Figure 1: The calibration problem addressed in this paper. The dashed circle denotes the initial pose of the robot. The solid one is the given target pose.

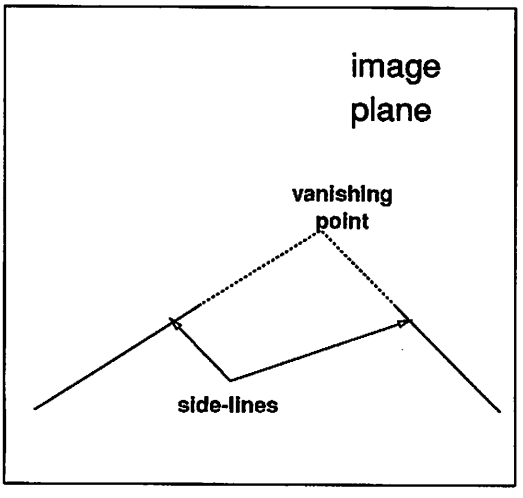
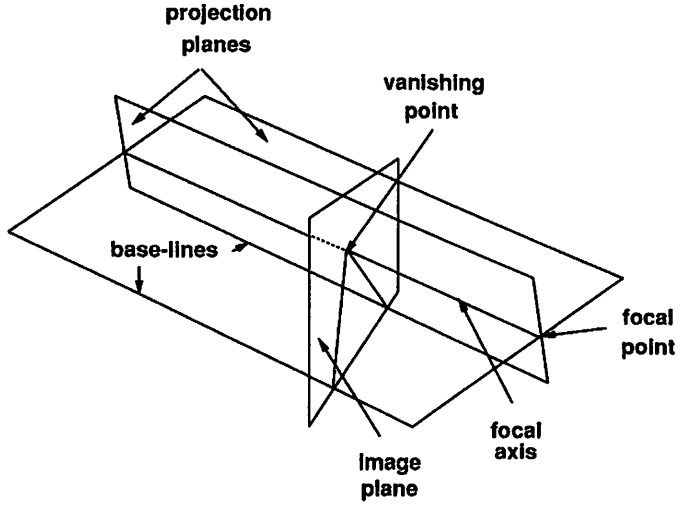
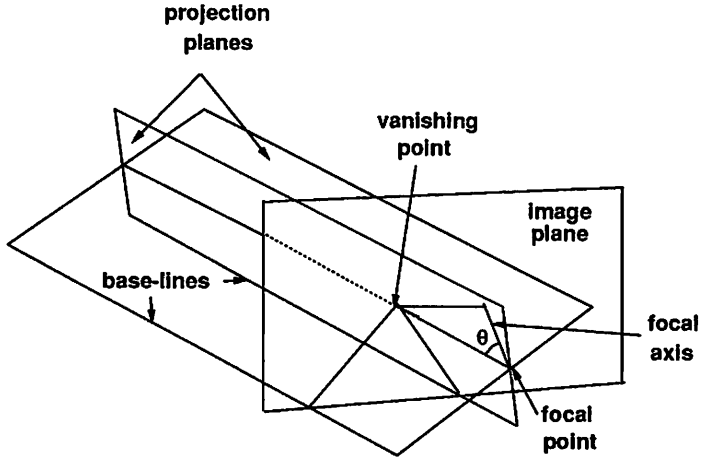


Figure 2: Definition of the vanishing point

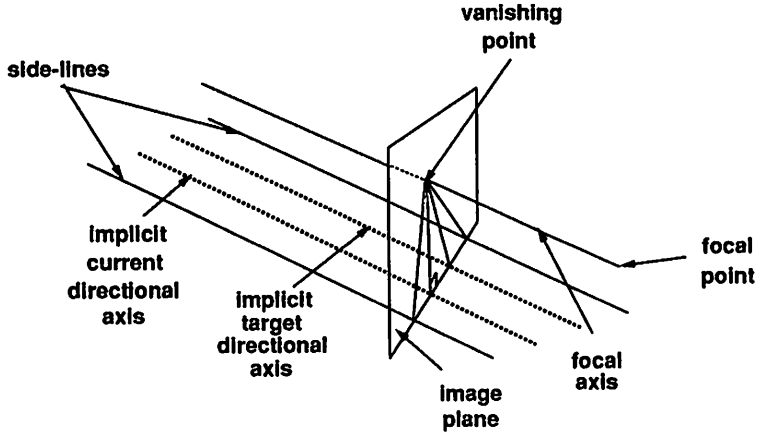


(a)

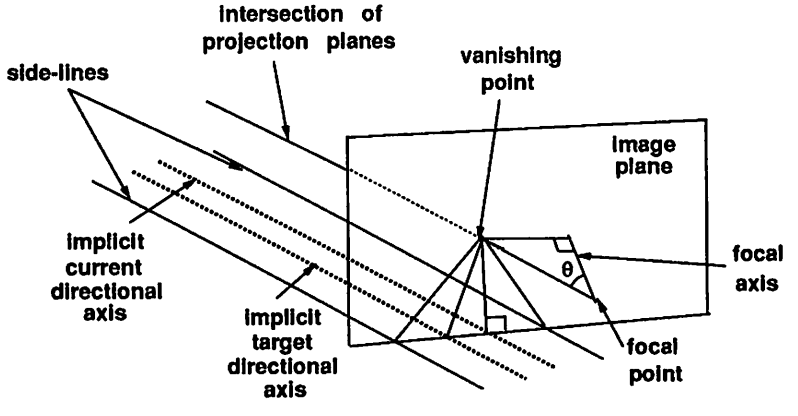


(b)

Figure 3: The figures used to prove Property 1. (a) when the camera orientation angle is 0, i.e., the focal axis is coincided with the interseciton line of the two projection planes of the side-lines; (b) when the camera orientation angle is not 0; Note the plane formed by the camera focal axis and the intersection line of the projection planes is referred to as plane P in the text.



(a)



(b)

Figure 4: The figures used to prove Property 2. Note, to be visually concise, we do not draw the projection planes of the implicit target directional axis and the implicit current directional axis, as well as those of the two side-lines. (a) when the camera orientation angle is 0. (b) when the camera orientation angle is not 0.

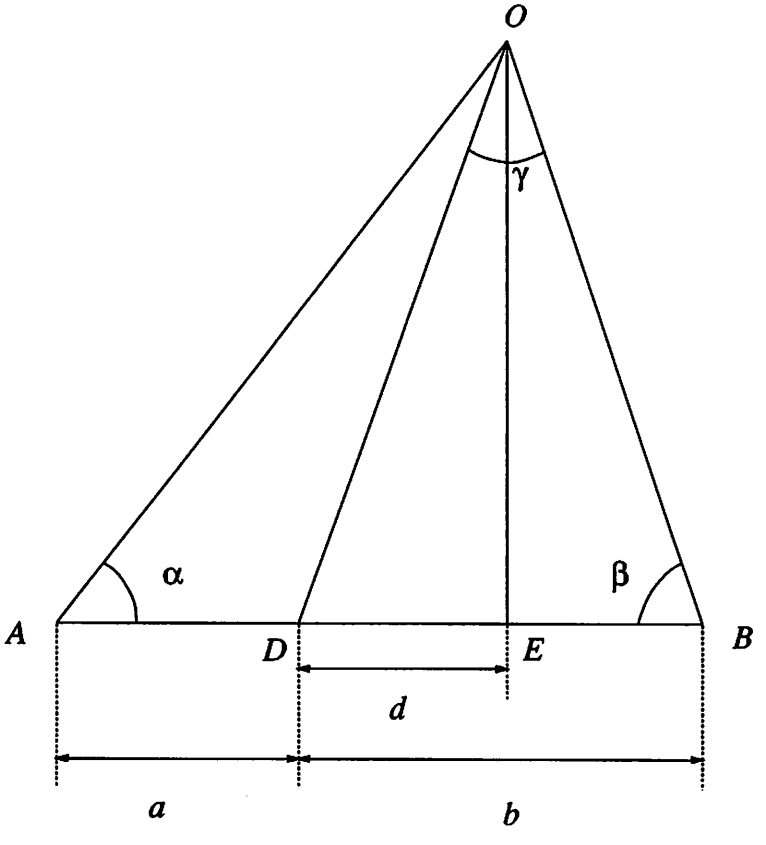


Figure 5: The figure used to derive the formula to calculate the lateral distance from the implicit target directional axis

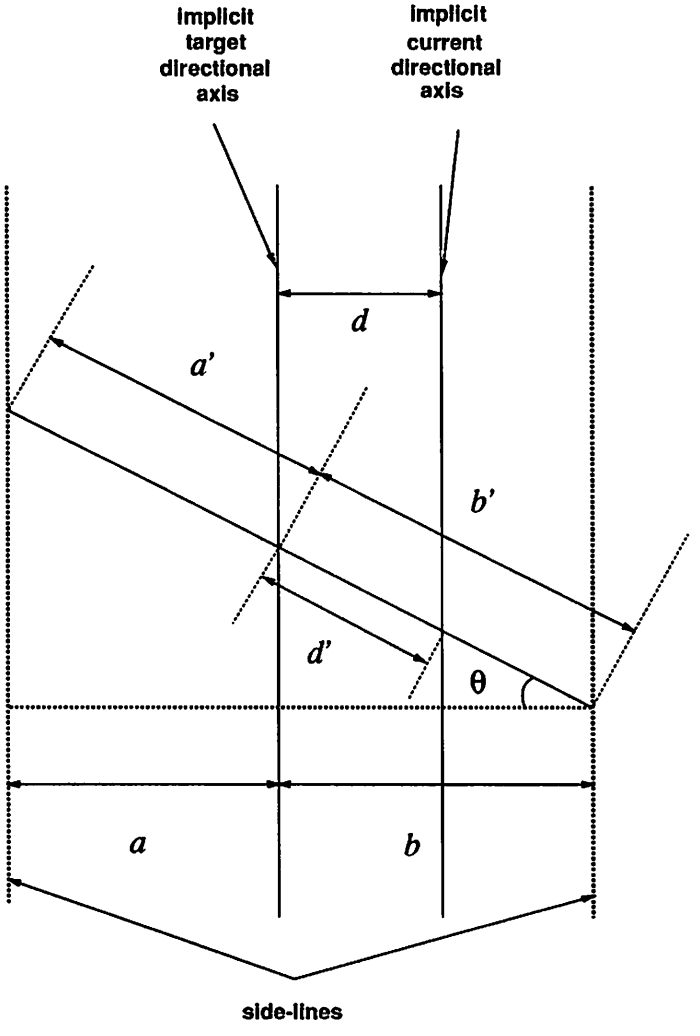


Figure 6: The figure used to derive the formula for the lateral distance from the implicit target directional axis in the general case

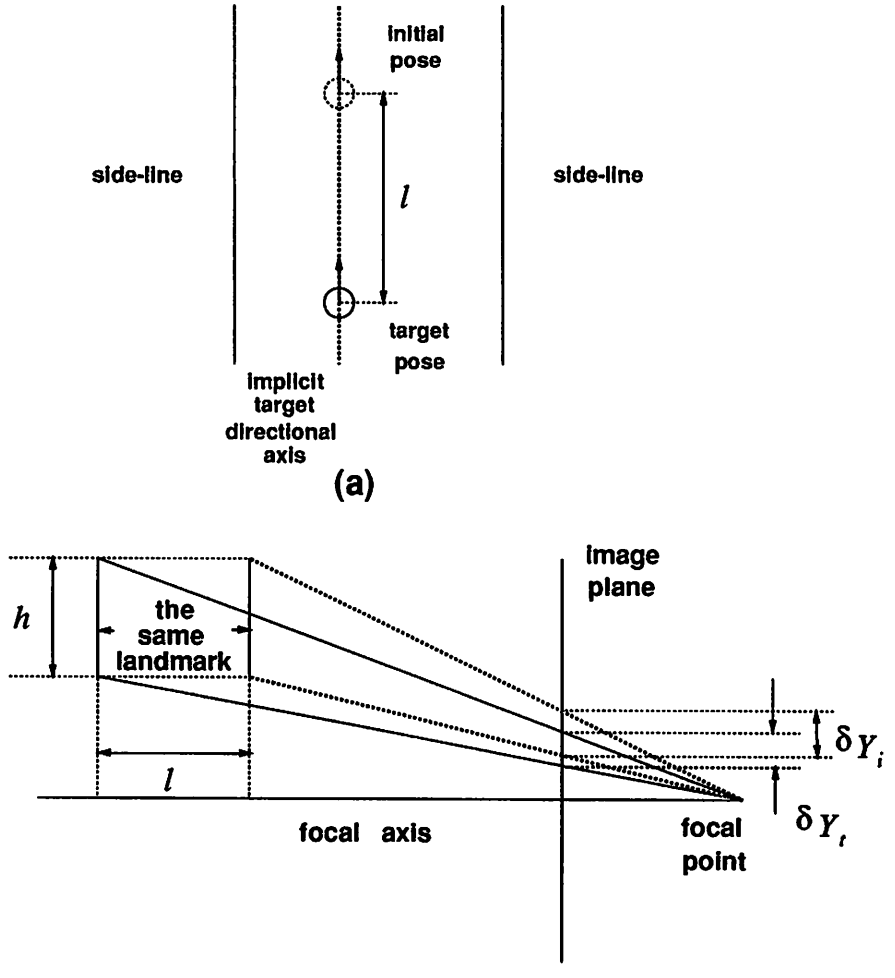


Figure 7: (a) The figure to address the looming problem, where the dashed circle denotes the initial pose, and the solid circle denotes the target pose. The dashed straight line is the target directional axis which does not explicitly exist in the environment, or in the image, either. (b) The figure used to derive the formula to calculate the looming distance

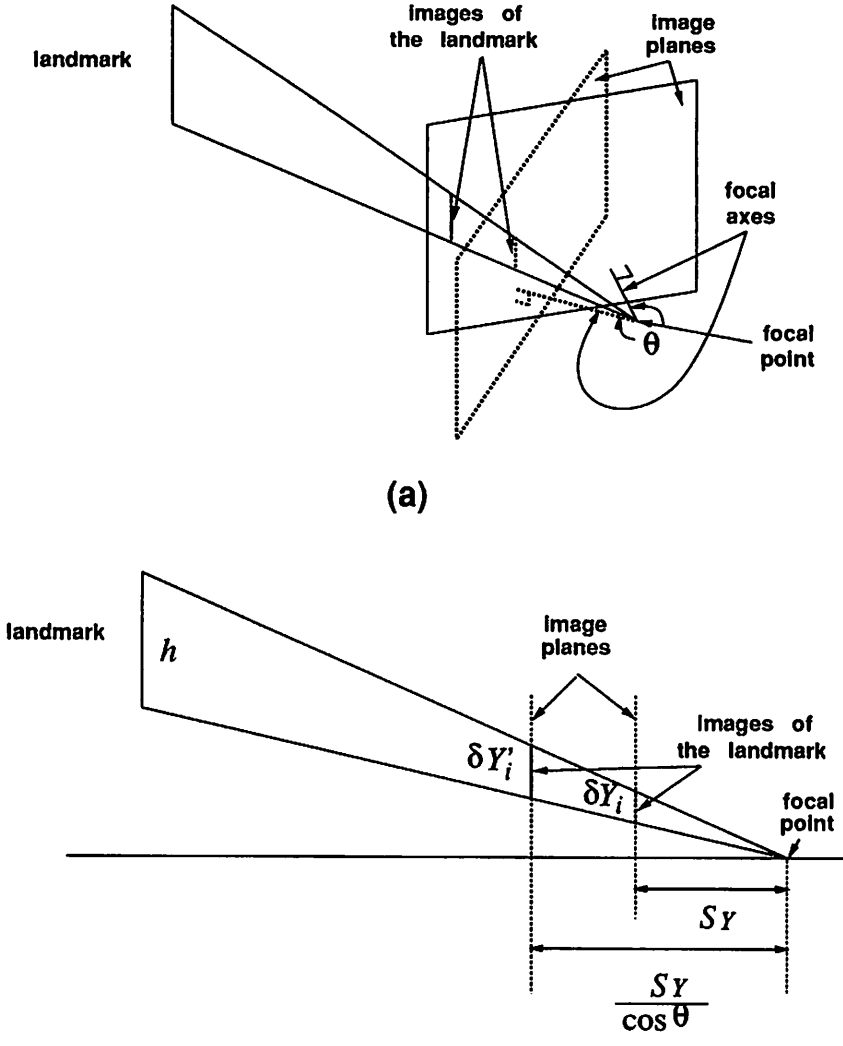


Figure 8: The figure used to derive the simplified algorithm. (a) The imaging system when given an arbitrary pose. The dashed plane is the imaginary image plane after the robot has rotated the angle θ. (b) The relationship between the landmark in the current arbitry image plane and that in the imaginary image plane. Here S_Y stands for the focal length of the camera.

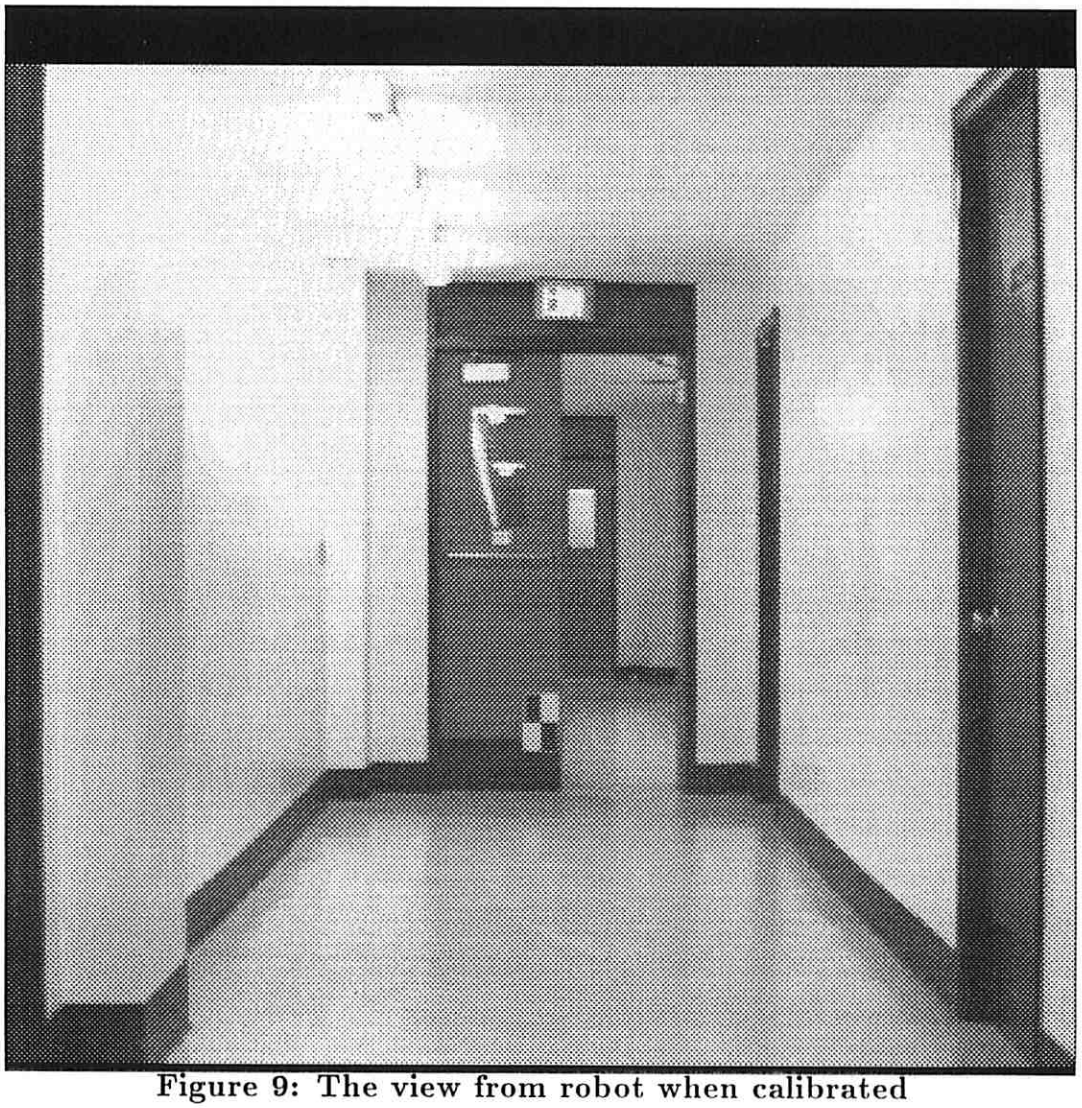


Figure 9: The view from robot when calibrated

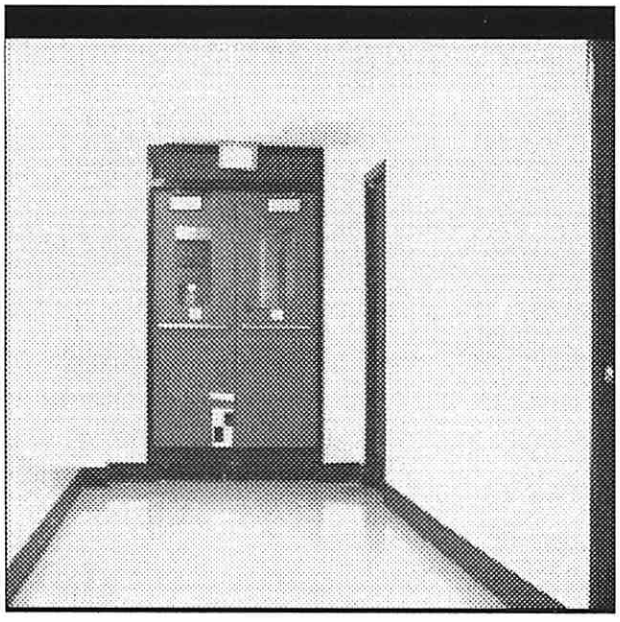


Figure 10: A hallway view from robot in an experiment

Figure 11: Another hallway view from robot

Table 1 – Parameters of Target Image and Thresholds

Parameters	X_c	Y_c	d	δY_t
	255.540	240.240	0.0	245.0
Thresholds	θ		d	l
	0.02°		0.1 ft.	0.1 ft.

Table 2 – A Recording of an Experiment Using Basic Algorithm

loop #	X_c	θ	d	l	Action
1	187.740	−3.92°	–	–	Turn left 3.92°
2	256.416	0.05°	-1.372 ft.	–	Move right 1.372 ft.
3	293.736	2.208°	–	–	Turn right 2.208°
4	254.354	−0.0686°	0.057	-5.006 ft.	Move backward 5.006 ft.
5	249.611	−0.343°	–	-0.163 ft.	Move backward 0.163 ft.
6	249.995	−0.321°	–	–	Turn left 0.343°
7	255.118	−0.024°	–	–	Turn left 0.024°
8	255.357	−0.010°	0.0418 ft.	0.0	Stop

Table 3 – A Recording of an Experiment Using the Efficient Algorithm

loop #	X_c	θ	d	l	Action
1	145.778	−6.278°	-1.226 ft.	-5.499 ft.	Turn left 6.278°, Move Right 1.226 ft., and Move Backward 5.499 ft.
2	252.242	−0.191°	0.033 ft.	-0.163 ft.	Move Backward 0.163 ft.
3	253.776	−0.102°	0.042 ft.	0.0	Turn Left 0.191°
4	255.693	0.009°	0.043 ft.	0.0	Stop