CVGIP: IMAGE UNDERSTANDING
Vol. 56, No. 3, November, pp. 271–329, 1992

INVITED PAPER

Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties

AKIO KOSAKA* AND AVINASH C. KAK

Robot Vision Laboratory, 1285 EE Building, Purdue University, West Lafayette, Indiana 47907-1285

Received January 28, 1992; accepted February 4, 1992

The model-based vision system described in this paper allows a mobile robot to navigate indoors at an average speed of 8 to 10 m/min using ordinary laboratory computing hardware of approximately 16 MIPS power. *The navigation capabilities of the robot are not impaired by the presence of stationary or moving obstacles.* The vision system maintains a model of uncertainty and keeps track of the growth of uncertainty as the robot travels toward the goal position. The estimates of uncertainty are then used to predict bounds on the locations and orientations of landmarks expected to be seen in a monocular image. This greatly reduces the search for establishing correspondence between the features visible in the image and the landmarks. Given a sequence of image features and a sequence of landmarks derived from a geometric model of the environment, a special aspect of our vision system is the sequential reduction in the uncertainty as each image feature is matched successfully with a landmark, allowing subsequent features to be matched more easily; this is a natural by-product of the manner in which we use Kalman filter-based updating.

CONTENTS

1. INTRODUCTION

Fast navigation using vision feedback has been a quest of researchers in sensor-based robotics for many years now. Consider the fact that not too many years ago, in 1983, the fastest that a robot could navigate indoors under vision control was 1 m per 15 min [Mor83]. Although considered a pioneering contribution at that time, that speed was simply not fast enough for useful applications.

In this paper, we present a vision-based reasoning and control architecture that allows a robot to navigate indoors at an average speed of 8 to 10 m/min, this speed being maintained in the presence of stationary and moving obstacles detected by ultrasonic sensors. This 120-fold increase in speed is not solely due to advances in affordable computing hardware and software, although that certainly has helped. Much of the gain in speed is owing to the use of model-based reasoning for scene interpretation. While most recent contributions to model-based scene analysis have relied solely on the geometric models of objects for organizing the flow of control, in this paper we show that, for the case of mobile robot navigation, great efficiency in computation is achieved when models of uncertainty are used in conjunction with the geometric models of the environment.

* Akio Kosaka is now with Olympus Optical Co., Ltd., 2-3 Kuboyama-cho, Hachioji-shi, Tokyo 192, Japan.

1049-9660/92 $5.00

The key idea in our system, which we have christened FINALE[1] for ease of reference, is simply that if it is possible to model the uncertainties in the position of the robot and how these uncertainties are transformed by the different motions of the robot, then it should also be possible to place bounds on where one should look for a landmark in the camera image. [Such position-uncertainty models, constructed by analyzing the differences between the actual motions of the robot and the commanded motions, involve the usual uncertainty parameters, such as the mean, the variances, and the correlations, and spatial transformations corresponding to the motions.] Although the precise details of how this is done are discussed later, to motivate the reader Fig. 1a shows the camera image from a certain position of the robot in the hallway. On the other hand, Fig. 1b is an expectation map rendered from a 3D model of the hallway assuming the robot is located at the center of its uncertainty region. Given the uncertainty at this particular position of the robot, with the help of the dashed ellipses we show in Fig. 1c the uncertainties associated with the vertices of the various edges in the expectation of Fig. 1b. Each ellipse represents one unit of Mahalanobis distance, meaning that the probability of finding a corresponding vertex within the ellipse is 39%.[2] In Section 3, we show how we can derive edge uncertainty regions from the vertex uncertainty regions; the search for scene correspondents of model edges can then be limited to these regions in the image and also in Hough space. Now, if we analyze the camera image and process this image only within the uncertainty regions associated with the landmarks whose presence is sought for the self-location exercise, we can quickly find the correspondences between, say, the edges shown in Fig. 1b and their counterparts extracted from Fig. 1a. What aids the establishment of these correspondences is the fact that, through the use of Kalman filtering, each match between a landmark and an image feature reduces both the uncertainty associated with the robot position and the size of the uncertainty regions associated with the subsequent landmarks.[3] After FINALE has computed the location and the orientation of the robot using the data in Figs. 1a, 1b, and 1c, the accuracy of self-location can be tested by reprojecting into the the camera image the expectation map using the estimated values for the position of the robot. Figure 1d shows, with the help of superimposed white lines, such a reprojection for those model edges that FINALE selected for the self-location exercise.

Our FINALE system should be considered in the context of recent contributions by Ayache and Faugeras [AyaFau89] and by Kriegman *et al.* [KriTri89]. Our use of Kalman filtering for both uncertainty reduction and position updating was inspired by the work of Ayache and Faugeras. However, the focus of the work by Ayache and Faugeras is less on autonomous navigation in an already known—known via CAD or other geometric representations—model of the environment and more on building 3D representations of the environment by fusing binocular and trinocular stereo images. On the other hand, while the overall goal of Kriegman *et al.* is similar—but not identical—to ours, their approach is quite different and entails vision processes that are different from ours. Central to the work of Kriegman *et al.* is the fast fusion of binocular images taken from two cameras mounted on the robot. By keeping the optic axes of the cameras horizontal and parallel, Kriegman *et al.* show how it is sufficient to retain from the two camera images thin strips in the middle and how binocular fusion of vertical lines extracted from these thin strips can then be used both for model building and for matching with what are essentially two-dimensional depth models of hallways.

In the work reported here, we are neither interested in stereo processes, nor are we interested in building models. We assume that geometric models of the hallways are already available. [Anything in the scene that is not in the models is treated as an obstacle.] Our focus is more on matching landmarks derived from the geometric models of the hallways with monocular images. A weak analogy would be to a one-eyed person engaged in indoor navigation.[4]

In Section 2 we present a survey of past work related to ours. The main discussion of our system starts in Section 3, where we present an overview of how uncertainty maintenance is carried out in our system and how self-location and navigation are then accomplished. In Section 4, we discuss a framework for the representation and transformation of uncertainties. Section 5 deals with the model representation of hallways and with the rendering of expectation maps as a robot navigates down the hall-

[1] FINALE stands for Fast Indoor Navigation Allowing for Locational Errors.

[2] As will be explained in Section 6, we limit our search regions to two units of Mahalanobis distance. The probability of finding the correspondent of a model feature within this uncertainty region is 86%. For pictorial displays of uncertainty, however, we use only one unit of Mahalanobis distance, since larger distances tend to clutter up the renditions in displays like the one shown in Fig. 1c.

[3] Of course, there may be more than one image feature within the specified uncertainty region for a given landmark. Such multiple possibilities, which occur only when the uncertainties become large, can be disambiguated by backtracking, as will be discussed later.

[4] To some this analogy may seem excessively weak since even a one-eyed human has many depth cues available to him/her. Although, of course, we are not trying to capture the depth cues that may be attributed to phenomena such as shading, our formalism here definitely captures those cues that are generated by comparing a priori spatial expectations with monocular images.

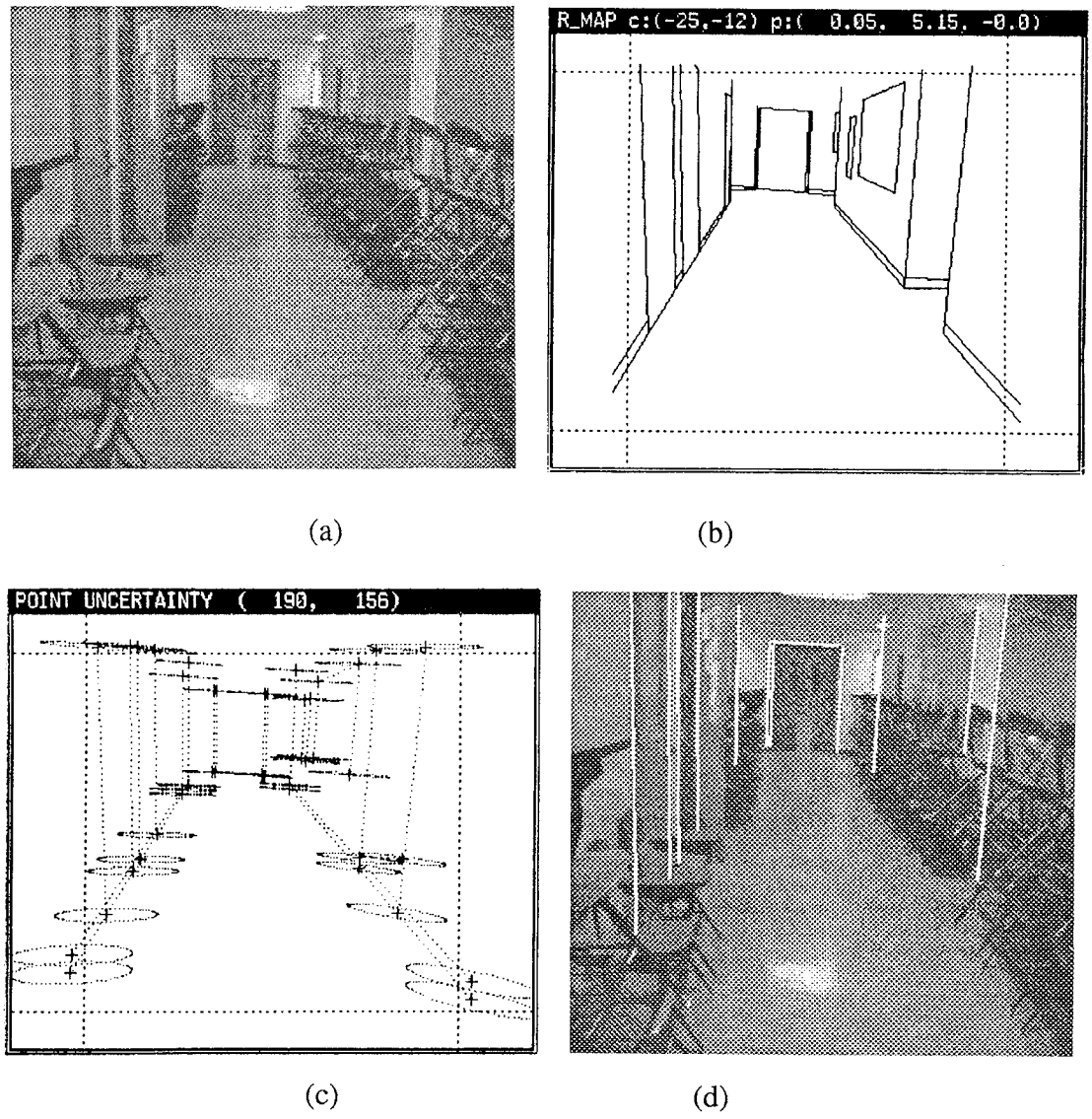


FIG. 1. (a) A camera image taken from a position of the robot while it is engaged in hallway navigation. (b) Scene expectation map rendered from a 3D model of the hallway. (c) The ellipses shown represent the uncertainties associated with the vertices of the edges in (b). From these ellipses one can easily obtain the uncertainties associated with the edges themselves. The FINALE system uses the data in (a), (b), (c) for robot self-location. Shown in (d) is a reprojection of those model edges into the camera image that were used for self-location after the robot determined its position in the hallway.

ways. Although it would be simple to use a commercial-type CAD system for the purpose of modeling, we show that, for the purpose of indoor navigation, all the geometric information that is relevant for scene interpretation can be captured in a simple data structure that lends itself to fast rendering. We next discuss the subject of landmark detection in Section 6 and present a new technique for a quick extraction of vertical lines in an image. This technique is based on our derivation that in a perspective image all the vertical world lines will have the same vanishing point regardless of the position of the robot, assuming that the robot is navigating on a flat floor. Subsequently, in Section 7, we present a formalism that allows landmarks to be matched to image features in a Kalman filter-based sequential scheme in which each match between a landmark and an image feature is used both to update the position of the robot and to estimate the new bounds on the various uncertainties. Section 8 then goes into global path planning and perception planning, both being necessary prerequisites to navigation in the presence of stationary and moving obstacles. Finally, in Section 9 we discuss the experimental results.

2. RELATED WORK

We now provide the reader with a brief review of past research in sensor-based mobile robot navigation. Our review is limited to indoor robots, especially those that are designed for hallway navigation. For mobile robots intended for outdoor applications, the reader is referred to the articles by Thorpe *et al.* [ThoSha87], Davis and Kunshner [DavKun87], Gat *et al.* [Gat90], and Zheng *et al.* [ZheBar91].

Historically, research in indoor mobile robotics originated with the work of Moravec [Mor81, Mor83], who focused on visual perception and control. The platform used by Moravec was remote controlled and used a video camera as its only sensor. To obtain 3D spatial information, a slider stereo vision system was used. In a slider stereo, a set of images taken when a single camera is moved along a track generates the disparity fields, which then can be transformed into depth maps. Next came Hilare, a robot designed by Giralt *et al.* [GirSob79] and Chatila and Laumond [ChaLau85]. This robot, equipped with a suite of sensors (a video camera, a rotating sonar sensor, and a laser range sensor mounted on a two-axis scanning system), uses dynamic world modeling, meaning that as the robot senses objects that it does not know about, it enters them into the world model.

More recently, the Mobi robot developed by Kriegman *et al.* [KriTri89] uses a pair of stereo cameras for 3D vision. The stereo images are used both for building simplified maps of the environment and for subsequent navigation through the environment. As mentioned in the Introduction, in the Mobi system thin horizontal strips are first extracted from the images and the vertical lines are then extracted from these strips for the purpose of binocular fusion. Another recent system, by Ayache and Faugeras [AyaFau89], uses extended Kalman filtering to help a robot acquire robust estimates of visual features in its environment. The function of the Kalman filter is to optimally combine the information generated by a trinocular stereo vision system from different locations. This robot can successfully recognize polygonal objects in the scene. It is useful to note that, as was done by Ayache and Faugeras, empirically interesting object recognition strategies can be devised by exploiting the ability of a mobile robot to examine a scene from different viewpoints.

Another approach to vision-based indoor navigation, by Tsubouchi and Yuta [TsuYut87], reasons over scene expectation maps and extracts from them those trapezoidal regions that the image processor can be expected to segment out from color camera images. The system then attempts to establish a matching between the model trapezoids and the trapezoids actually segmented out from the camera image. Along similar lines, the PSEIKI system developed in our laboratory [AndKak88, KakAnd89] also first renders scene expectation maps from a 3D CAD model of the hallways and represents this expectation map as a hierarchy of geometrical abstractions. The image processor then tries to extract a similar geometric hierarchy from the camera image. Finally, an evidential approach using the Dempster–Shafer theory of evidence is employed to compare the expectation map hierarchy with the image hierarchy. The correspondences thus derived help the robot determine its position. The main advantage of PSEIKI-like approaches is that they seek to carry out an evidential interpretation of the entire image. But since, if the goal is self-location for a mobile robot, it is not necessary to interpret the entire camera image, any system that is designed primarily for image interpretation would in most cases be too slow for navigation purposes. Using comparable computing hardware, it would take PSEIKI minutes to do what takes the FINALE system only seconds.

Another recent contribution to hallway navigation is by Fennema *et al.* [FenHan90]. They use a modification of the gradient descent approach to implement a model-driven grouping of edges extracted from the camera image. The correspondences thus established are then fed into an algorithm developed by Kumar and Hanson [KumHan89] for the calculation of camera position and orientation. The algorithm developed by Kumar and Hanson is an extension, to the case of possibly incorrect correspondences, of the work done by Liu *et al.* [LiuHua88] on the subject of how to determine the position and the aim direction of a camera given a set of correspondences, which may be supplied by a human, between line features in the model and line features in the camera image. Yet another contribution that might interest the reader, by Sugihara [Sug87], presents a solution to the determination of the robot location parameters by first hypothesizing a possible location on the basis of a partial map between the features in a scene and the visible features in the camera image, and then verifying the hypothesis by testing for the appearance in the image of the other features in the model. Sugihara has verified his approach by computer simulation studies using two-dimensional model maps, which could correspond to floor plans of hallways, and what could be called one-dimensional camera images.

3. FRAMEWORK FOR NAVIGATION

Figure 2 depicts the overall reasoning and control architecture of FINALE for vision-based navigation in the presence of stationary and moving obstacles. This framework is motivated by the observation, made by many researchers before us, that it is indeed possible to construct useful models of uncertainties associated with ro-

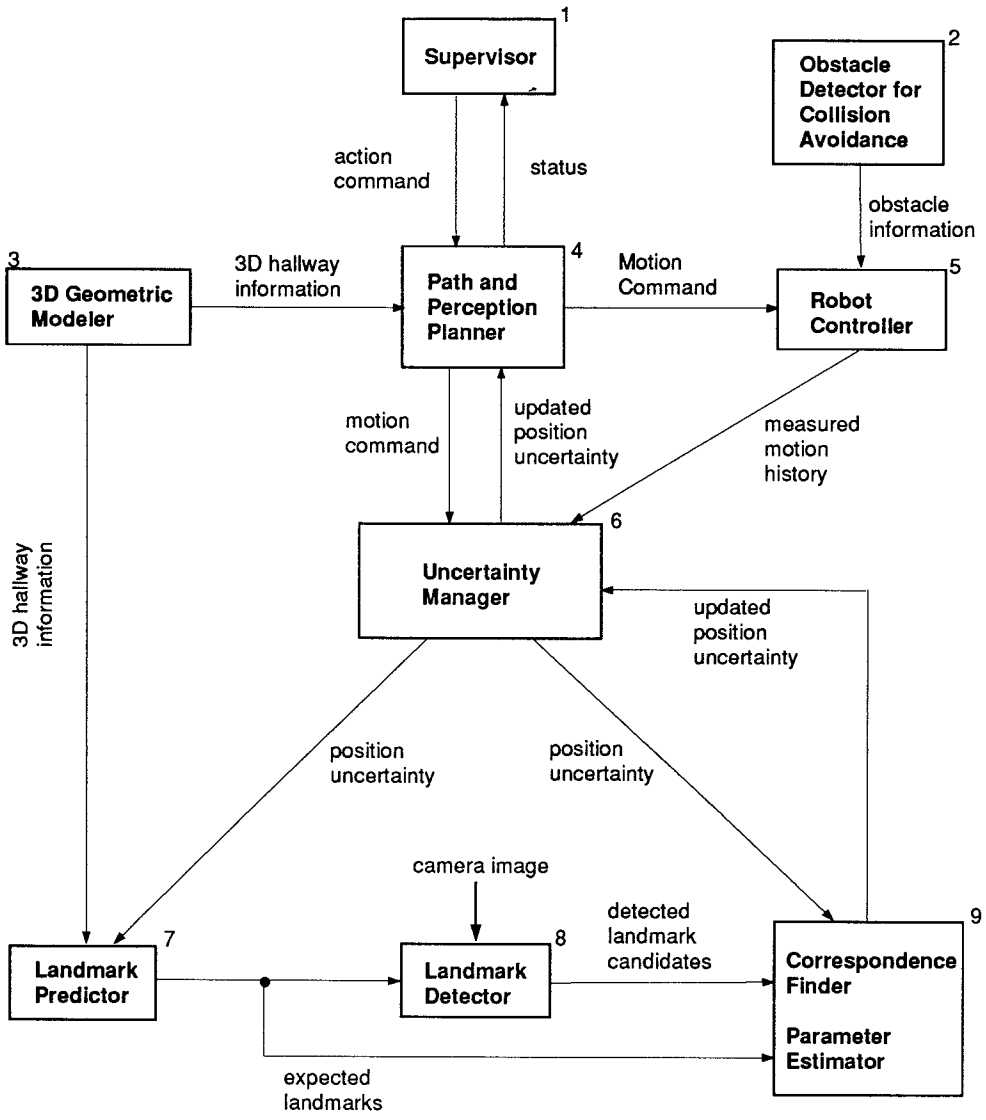


FIG. 2. The FINALE reasoning and control architecture for simultaneous vision-guided navigation and collision avoidance. Module 2, in conjunction with module 5, controls the reactive behavior of the robot, while the rest of the modules, again in conjunction with module 5, control the deliberative behavior.

bot motion. A key aspect of FINALE is that the estimated uncertainties are used to limit the region of a camera image that must be analyzed and to severely constrain the choices of image features as candidates for the landmarks expected to be seen by the robot. In the rest of this section, we briefly touch on the functions of each of the modules shown in Fig. 2, concentrating mainly on the information that flows in the directed pathways between the modules.

The Supervisor (module 1) is the human. His/her function is to supply the robot with the coordinates of the goal position. Another important function of the Supervisor is to quickly shut down the robot in case the navigation process fails, especially when it seems that the robot is headed for any unwary bystander.[5] The Supervisor also must supply the robot with the robot's initial position. The initial position specification is rather loose. For example, the supplied floor coordinates of the robot can be off by as much as half a meter and the supplied orienta-

[5] The navigation experiments succeed roughly 90% of the time. This number was arrived at by conducting a large number of experiments over a run of 50 m of hallways with two right-angled turns at both ends. We believe the failures are caused by the wheels occasionally slipping on the floor in a manner that is not consistent with the mathematical model.

tion can be off by as much as 15°. Before the robot starts navigating, it uses its vision faculty to eliminate these uncertainties in the supplied initial position.

The Path and Perception Planner (module 4) takes the initial and the goal positions from the Supervisor and plans out a path in the 3D geometric model of the hallways that resides in module 3. Path planning in hallways is a rather simple exercise. The details of how it is done in our system are given in Section 8. The Planner first calculates a sequence of motions, in terms of straight line traversals and rotations, that would take the robot from its current position to the goal position. The straight line traversals are broken into segments so that the maximum expected uncertainty at the end of each segment will not exceed a threshold that itself is a function of the environment. As a result, in tight spaces the straight line traversals are decomposed into shorter segments. The Perception Planner in module 4 plays a critical role during obstacle avoidance when the robot must of necessity deviate from the originally planned path. The Perception Planner is aware, on a real-time basis, of the growth in the position-uncertainty as the robot maneuvers around the obstacle and, when the uncertainty exceeds a prespecified threshold, the Perception Planner brings the robot to a halt for another exercise at self-location through vision.

The Robot Controller (module 5) is the place where each motion, a translation or a rotation, received from the Path Planner is interpreted as a motion command. These motion commands can be overridden by module 2 for Obstacle Detection and Collision Avoidance.[6] Yet another important function of the Robot Controller is to supply odometry information to the Uncertainty Manager (module 6), whose function is to keep track of the uncertainty in the position of the robot. The parameters of the motion dependency of the uncertainty in the position of the robot are stored in the Uncertainty Manager module. These parameters, in conjunction with the odometry information received from the Robot Controller, help the Uncertainty Manager compute the latest positional uncertainty associated with the robot, specified by a mean value and a covariance matrix, as explained in Section 4.

[6] This override mechanism, while smacking of the Brooksian notions [Bro86, Con87], is actually in reverse of what would be implemented in a subsumption architecture. In a truly subsumption architecture, vision input would occupy a "higher layer" than the proximity signals corresponding to obstacle detection. However, the type of vision needed in a subsumption architecture would have to be fast enough for influencing real-time behavior. In other words, the vision processes in a subsumption architecture would have to be much faster than what is possible today. The FINALE reasoning and control architecture tries to make the best of what is possible today with vision sensing for what may be referred to as deliberative behavior and uses ultrasonic sensing for the reactive behavior for collision avoidance.

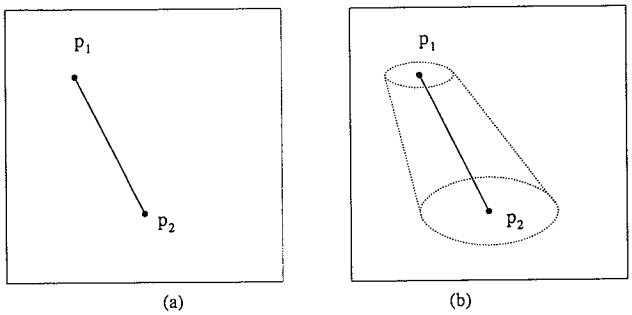


FIG. 3. This figure illustrates how Landmark Predictor (module 7 of Fig. 2) creates the uncertainty region for a single line segment. A single line segment constructed from the two endpoints p_1 and p_2 is extracted from the expectation map as shown in (a). Landmark Predictor then outputs the mean vectors and the covariance matrices associated with the two endpoints, forming the uncertainty ellipses shown in (b). A convex hull, also shown in (b), of the two ellipses constitutes the uncertainty region for the line segment.

In addition to supplying the positional uncertainty parameters to module 4, the Uncertainty Manager also supplies this information to the Landmark Predictor whose job is to decide where in the image one should look for a given landmark. The means and covariances of the different parameters of the robot motion are also needed in module 9 for the operation of the Kalman filter, whose function is to update the means and the covariances of the robot position after each match between a landmark and its corresponding image feature.

For each attempt at self-location, the first action of Landmark Predictor is to render an expectation map, like the one shown in Fig. 1b, given the current mean position and orientation of the robot. By using the various transformations discussed in Section 4 and by using the means and covariances of the parameters of robot position, the Landmark Predictor then calculates the mean and the variances associated with the image features corresponding to the various landmarks. Just to explain the function of module 7, imagine there is a single line, as shown in Fig. 3a, in the expectation map. For this line, Landmark Predictor will output the mean vectors and the covariances associated with the two end points of the line, as shown pictorially in Fig. 3b. The Landmark Detector will then search for an image line in a region formed by the convex hull of the two uncertainty ellipses.[7]

The job of module 9 is to match the landmarks in the image rendered from the geometric model with the features extracted from the image taken by the camera; more specifically, it matches the lines in the expectation

[7] For computational efficiency, the search carried out by Landmark Detector actually takes place in the smallest rectangular enclosure whose sides are parallel to the image rows and columns and which contains the two uncertainty ellipses.

map shown in Fig. 1b with the edges extracted from the appropriate uncertainty regions of the camera image. Module 9 treats all the edges extracted from an uncertainty region as candidates for matching with a landmark, the choice of these candidates being constrained by edge orientation considerations in the Hough space. Matching of each candidate image edge with a model edge is followed, via Kalman filtering, of recomputation of the uncertainties associated with robot position. In general, each match will reduce the uncertainties and further constrain the choice of candidate image edges as candidates for subsequent model edges. If, despite the pruning of the search space affected by the reduced uncertainties owing to the prior matches, there still exist multiple image feature candidates for a given landmark, module 9 will select one on a greatest likelihood basis, as explained in Section 7.

4. UNCERTAINTY REPRESENTATION

In this section, the uncertainties involved in vision-based mobile robotics are discussed in a systematic manner. We first discuss the transformations in the parameters of positional uncertainty as the robot undergoes translational and rotational motions. We then show how the uncertainty in the position of the robot translates into uncertainty in the location and orientation of the image features corresponding to a landmark in the scene.

4.1. Parameters of Robot Position

We assume the robot is navigating on a perfectly flat floor that exists in the xy-plane of a world coordinate frame (x_w, y_w, z_w), as shown in Fig. 4. Local to the robot will be a robot-centered coordinate frame (x_r, y_r, z_r). We assume that the (x_r, y_r) plane in the local frame is coplanar with the (x_w, y_w) plane of the world frame. What that implies is that the vertical axes z_w and z_r will always be parallel.

As the robot travels down the hallways, its position is defined as the position of the robot-centered frame with respect to the world frame; the translational and the rotational components of this position are all packaged in the 3-vector $\mathbf{p} = (p_x, p_y, \phi)$, where p_x and p_y describe the translations of the robot and ϕ describes the rotation, the latter defined as an angle measured counterclockwise from the x_w axis of the world frame to the x_r axis of the robot frame.

Assume for the purpose of argumentation, although in reality this is never the case, that initially the location and the orientation of the robot are known without error. As the robot travels down the hallway, its actual location and orientation will always differ from the location and the orientation that the robot is commanded to occupy. This discrepancy between the commanded and the actual

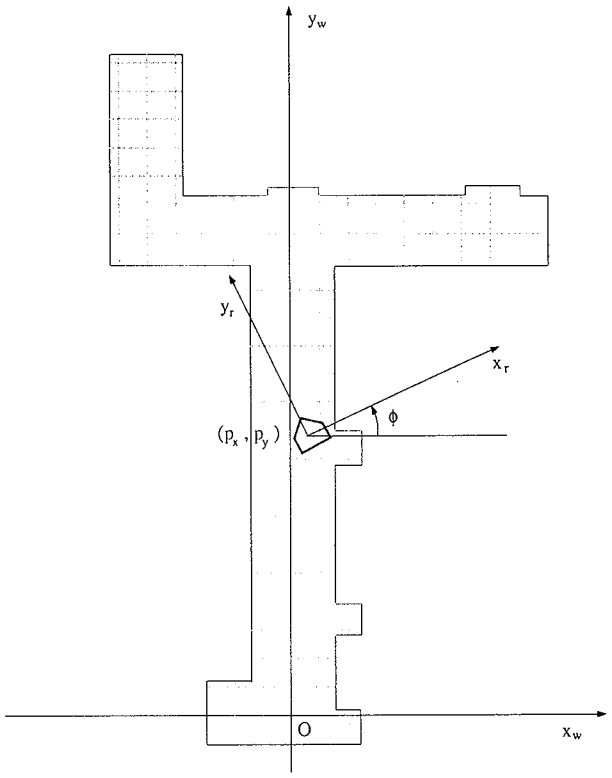


FIG. 4. A hallway with the robot located at world coordinates (p_x, p_y) and with the orientation ϕ. The axes marked x_r and y_r show the robot-centered coordinate frame.

locations and orientations is presumably caused by the differential slippage in the wheels, which in turn is a function of the weight distribution on the base of the robot, the condition of the rubber on the wheels, the condition of the floor, etc. We have noticed through experiments that when the robot is commanded to travel perfectly straight down the long hallway in Fig. 4, it eventually veers off to one side or the other and eventually bumps into one of the walls.[8] Of course, the distance between the initial start point and the location where the robot would bump into a wall is also dependent on the accuracy with which the initial orientation of the robot is known. However, after much experimentation and through repeated trials, we have observed that under the best possible alignment between the robot and the length of the hallways, the robot would eventually bump into one of the walls within about 15 m.

As the robot continues to travel down a hallway in a dead reckoning mode, its positional uncertainty in-

[8] This is called navigation by dead reckoning, meaning navigation in which there is no sensory feedback regarding the current position of the robot. Statistical analysis of the performance of the robot while it is commanded to navigate by dead reckoning yields the parameters of the uncertainty model that are then used for vision-driven navigation.

creases. How should this uncertainty be modeled? While there are no clear-cut answers to this question—at least no answers that are founded soundly on physical phenomena underlying the causes of uncertainty—we follow our predecessors [SmiChe86, AyaFau89, FriTri89] and assume the uncertainty can be described by a Gaussian distribution whose mean $\bar{\mathbf{p}}$ and covariances, denoted by the matrix Σ_p, are given by

$$\bar{\mathbf{p}} = E[\mathbf{p}] \tag{4.1}$$

$$\Sigma_p = E[(\mathbf{p} - \bar{\mathbf{p}})(\mathbf{p} - \bar{\mathbf{p}})^{\mathrm{T}}]. \tag{4.2}$$

The pair $(\bar{\mathbf{p}}, \Sigma_p)$ will be called the *robot position uncertainty*.

Although there is ample precedence for using Gaussian distributions [SmiChe86], other possible ways for representing uncertainty in the robotic context do exist. For example, as argued by Brooks [Bro85] for the general case and as used by Gottschlich and Kak [GotKak91a] for the case of assembly path planning, uncertainties can be modeled by tolerance regions surrounding the objects and obstacles. Since tolerance regions do not lend themselves to Kalman filter type schemes for estimating and updating uncertainty parameters, we have not used them in the work reported here.

4.2. Transformation of Uncertainty Parameters by Robot Motion

In general, the functional relationship between the position $\mathbf{p} = (p_x, p_y, \phi)$ before the start of a commanded motion and the position $\mathbf{p}' = (p_x', p_y', \phi')$ afterward can be written as

$$\mathbf{p}' = \mathbf{h}(\mathbf{p}). \tag{4.3}$$

It is well known that when the transformation function $\mathbf{h}$ can be considered to be linear, the uncertainty parameters $(\bar{\mathbf{p}}, \Sigma_p)$ and $(\overline{\mathbf{p}'}, \Sigma_p')$ for before and after a commanded motion, respectively, are related by [SmiChe86, Dur87]

$$\overline{\mathbf{p}'} = \mathbf{h}(\bar{\mathbf{p}}) \tag{4.4}$$

$$\Sigma_p' = \left(\frac{\delta \mathbf{h}}{\delta \mathbf{p}}\right) \Sigma_p \left(\frac{\delta \mathbf{h}}{\delta \mathbf{p}}\right)^{\mathrm{T}}. \tag{4.5}$$

The derivative $\delta\mathbf{h}/\delta\mathbf{p}$ is the Jacobian of the transformation $\mathbf{h}$. These equations form the foundation of how the positional uncertainty can be propagated as the robot travels, assuming that the transformations corresponding to the function $\mathbf{h}$ are linear, or approximately so.

As mentioned above, we assume that the positional uncertainty can be modeled by a Gaussian distribution. For such distributions, a useful metric for comparing a position vector $\mathbf{p}$ with its mean $\bar{\mathbf{p}}$ is the Mahalanobis distance given by

$$\mathrm{dist}_{\bar{\mathbf{p}},\Sigma_p}(\mathbf{p}) = [(\mathbf{p} - \bar{\mathbf{p}})^{\mathrm{T}} \Sigma_p^{-1} (\mathbf{p} - \bar{\mathbf{p}})]^{1/2}. \tag{4.6}$$

Based on this assumption, we can consider a set of points in R^n such that

$$A(d) = \{\mathbf{p} \in R^n \mid \mathrm{dist}_{\bar{\mathbf{p}},\Sigma_p}(\mathbf{p}) \leq d\}. \tag{4.7}$$

$A(d)$ creates a closed convex set in the 3-dimensional (p_x, p_y, ϕ) space, this set being a hyperellipsoid. For any given value of d, the surface of this hyperellipsoid is completely specified by the covariance matrix Σ_p. The set of points $A(d)$ will be called the *uncertainty region* induced by the Mahalanobis distance d.[9]

When, as before, the function $\mathbf{h}$ corresponds to the transformation introduced by a commanded motion, the shape of the uncertainty region undergoes a change, but it continues to remain a hyperellipsoid under the assumption that $\mathbf{h}$ is linear. Therefore, since a hyperellipsoid is completely specified by the covariance matrix, in order to compute the new uncertainty region all we need to do is to keep track of the covariance matrix Σ_p, along with the mean vector. As we show later, the Kalman filter plays a central role in the calculation of the new $\bar{\mathbf{p}}$'s and Σ_p's as the robot travels down the hallway.

The Case of Translations

So far in this section, we have addressed the transformation $\mathbf{h}$ induced by a commanded motion in a rather abstract sense. We will now be more specific and consider the case in which the commanded motion is a pure translation. Recall that in our robot the translations and rotations are two distinct motion commands.

Due to the manner in which the world and the robot-centered coordinate frames were defined, a purely translation motion of the robot can occur only along the y_{r} direction. [Of course, a rotation of the robot will alter the y_{r} direction vis-a-vis the coordinate directions in the world frame.]

Assume that the robot, located initially at O in Fig. 5a, is commanded to move a distance d_0 along y_{r}. In actuality, due to the aforementioned phenomena, at the end of the commanded motion the robot will occupy a position at some radial distance d from the starting point; also, the location of the robot will subtend an angle α with the original y_{r} direction. Furthermore, the new y_{r} axis will point in a direction different from the old y_{r} axis; the

[9] If the reader is not familiar with this approach to specifying uncertainty regions, it might help if we pointed out that, for the one-dimensional case, $A(d)$ is the set of points within d standard deviations from the mean.

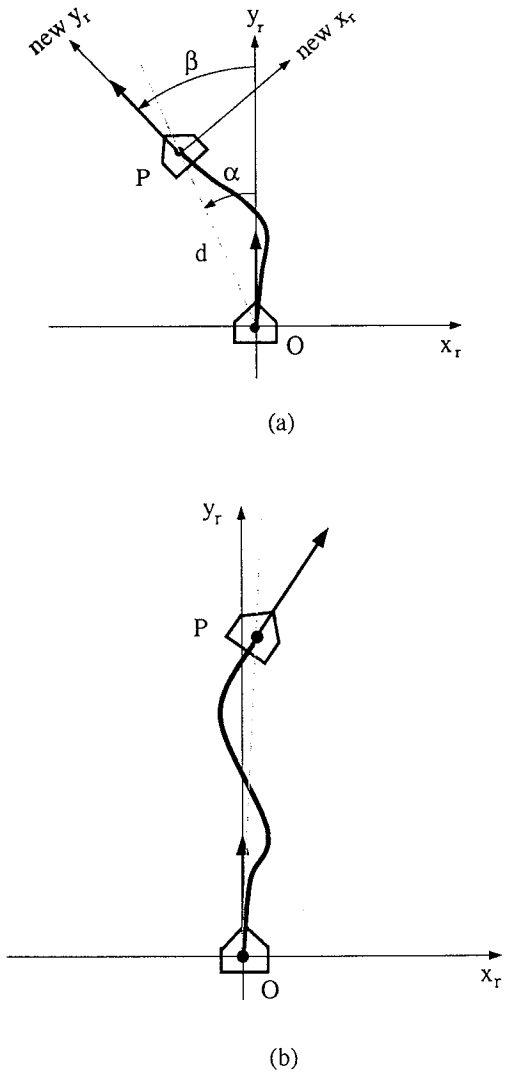


FIG. 5. To understand this figure, the reader should recall that the robot can execute translational moves only along its own y_r axis. When the robot is commanded to translate by d_0 units, the differential slippage in the wheels causes the robot to occupy the position shown in (a). To facilitate uncertainty characterization, this position at the end of the commanded travel is characterized by three parameters d, α, and β. Note that β represents the angle between the old y_r axis and the new y_r axis. (b) The point being made here is that when the commanded d_0 is long, the randomness of the differential slippage may cause the robot to zig and zag, the result being the final value of α close to zero.

angle between the two will be denoted by β. Although the phenomena that result in the angle α are essentially the same as the phenomena that result in β, the two angles will not in general be identical. To make this point clearer, consider a commanded translational motion for a large value of d_0. It is entirely possible that in this case the robot will zig and zag a little, as shown in Fig. 5b, but when it comes to rest it will most probably point in a direction different from the one at the start point. In this case, the value of α will be close to zero, while for β the value may be any value within a range that is experimentally significant.

Clearly, for a given d_0, we may think of d, α, β as random variables, with mean values of $\bar{d}$, $\bar{\alpha}$, and $\bar{\beta}$, and the covariance matrix

$$\Sigma(d, \alpha, \beta) = \begin{bmatrix} \sigma_d^2 & \rho_{d\alpha}\,\sigma_d\,\sigma_\alpha & \rho_{d\beta}\,\sigma_d\,\sigma_\beta \\ \rho_{d\alpha}\,\sigma_d\,\sigma_\alpha & \sigma_\alpha^2 & \rho_{\alpha\beta}\,\sigma_\alpha\,\sigma_\beta \\ \rho_{d\beta}\,\sigma_d\,\sigma_\beta & \rho_{\alpha\beta}\,\sigma_\alpha\,\sigma_\beta & \sigma_\beta^2 \end{bmatrix}, \tag{4.8}$$

where $\rho_{d\alpha}$, $\rho_{d\beta}$, and $\rho_{\alpha\beta}$ represent the correlation coefficients. Because what we really seek are the means and the covariances of the vector $\mathbf{p}$, we need to establish a relationship between the random entities d, α, β, and the components p_x, p_y, ϕ. To establish this relationship, let us assume that at the moment the command to travel straight through a distance d_0 is issued, the position vector of the robot is (p_x, p_y, ϕ). Let us further assume that after the execution of the commanded motion, the new position vector is (p_x', p_y', ϕ'). The two position vectors are related by

$$\begin{bmatrix} p_x' \\ p_y' \\ \phi' \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ \phi \end{bmatrix} + \begin{bmatrix} -d\,\sin(\phi + \alpha) \\ d\,\cos(\phi + \alpha) \\ \beta \end{bmatrix}. \tag{4.9}$$

This shows that the components of the position vector at the termination of the commanded motion are nonlinear functions of the parameters of the new position of the robot. For example, p_x' is a nonlinear function of p_x, d, α, and β. Since the applicability of the covariance propagation formulas, such as the one shown in Eq. (4.5), is limited to the case of linear relationships between the old and the new positions of the robot, we must use a linear approximation to Eq. (4.9). Since from Eq. (4.5) it is clear that for the case of linear transformations we only need to know the Jacobian of the transformation, the question then becomes one of finding the Jacobian corresponding to a linear approximation to Eq. (4.9). This Jacobian may be found by taking differentials of both sides in Eq. (4.9) and evaluating the coefficients of the differentials at the mean values of the various parameters

$$\begin{bmatrix} \delta p_x' \\ \delta p_y' \\ \phi' \end{bmatrix} = \mathrm{J}_{S_1} \begin{bmatrix} \delta p_x \\ \delta p_y \\ \delta\phi \end{bmatrix} + \mathrm{J}_{S_2} \begin{bmatrix} \delta d \\ \delta\alpha \\ \delta\beta \end{bmatrix}, \tag{4.10}$$

where J_{S_1} and J_{S_2} are the Jacobians, these being given by

$$J_{S_1} = \begin{bmatrix} 1 & 0 & -\bar{d}\cos(\bar{\phi}_1 + \bar{\alpha}) \\ 0 & 1 & -\bar{d}\sin(\bar{\phi}_1 + \bar{\alpha}) \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and}$$

$$J_{S_2} = \begin{bmatrix} -\sin(\bar{\phi} + \bar{\alpha}) & -\bar{d}\cos(\bar{\phi} + \bar{\alpha}) & 0 \\ \cos(\bar{\phi} + \bar{\alpha}) & -\bar{d}\sin(\bar{\phi} + \bar{\alpha}) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.11)$$

Assuming that the initial position vector (p_x, p_y, ϕ) and the parameters of the change in position, (d, α, β), are statistically independent, we obtain the following relationships for the mean vector and the covariance matrix at the end of the commanded motion:

$$\begin{bmatrix} \bar{p}'_x \\ \bar{p}'_y \\ \bar{\phi}' \end{bmatrix} = \begin{bmatrix} \bar{p}_x \\ \bar{p}_y \\ \bar{\phi} \end{bmatrix} + \begin{bmatrix} -\bar{d}\sin(\bar{\phi} + \bar{\alpha}) \\ -\bar{d}\cos(\bar{\phi} + \bar{\alpha}) \\ \bar{\beta} \end{bmatrix} \quad (4.12)$$

$$\Sigma(p'_x, p'_y, \phi') = J_{S_1}\,\Sigma(p_x, p_y, \phi)\,J_{S_1}^{T} + J_{S_2}\,\Sigma(d, \alpha, \beta)\,J_{S_2}^{T}. \quad (4.13)$$

The derivation of the Jacobians above should cast light on what is meant by a linear approximation to the transformation that takes the robot from the initial position to the destination position at the end of the commanded motion. Clearly, the approximation would be valid if the initial position is sufficiently close to the mean vector $(\bar{p}_x, \bar{p}_y, \bar{\phi})$ and the parameters of the change are sufficiently close to their mean values $(\bar{d}, \bar{\alpha}, \bar{\beta})$.

Equations (4.12) and (4.13) tell us how to propagate the means and the covariances of the position vector **p** as the robot travels down the hallway.

The Case of Rotations

By definition of the robot-centered coordinate frame, the robot is always "facing" the positive y_r direction, meaning the "face" of the robot is always pointing in the positive y_r direction. A commanded rotation, denoted θ_0, means that the robot should turn counterclockwise so that its new y_r axis subtends an angle of θ_0 with the old y_r direction.

In response to a commanded rotation, the differential slippage in the wheels will cause the robot to undergo some translation in addition to the rotation. Let the change in the position of the robot in response to the commanded rotation be given by (u, v, θ), where u is the displacement in the direction of old x_r, v the displacement in the direction of old y_r, and θ the angle of rotation between the old y_r and the new y_r, the designations old and new referring, respectively, to the positions of the robots before and after the commanded rotation (see Fig. 6).

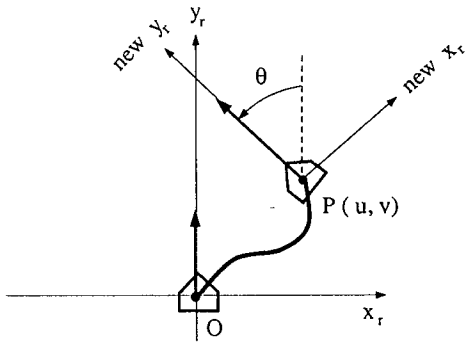


FIG. 6. When the robot is commanded to execute a purely rotational motion, the differential slippage in the wheels will cause the robot to occupy a position like the one shown here. To facilitate the characterization of the uncertainty associated with this final position, we again use three parameters, u, v, and θ. Note that u and v are translations along the old x_r and y_r axes.

For a given commanded rotation through θ_0, the parameters u, v, and θ need to be treated as random variables with mean values $\bar{u}$, $\bar{v}$, and $\bar{\theta}$, and covariances given by the following matrix:

$$\Sigma(u, v, \theta) = \begin{bmatrix} \sigma_u^2 & \rho_{uv}\,\sigma_u\,\sigma_v & \rho_{u\theta}\,\sigma_u\,\sigma_\theta \\ \rho_{uv}\,\sigma_u\,\sigma_v & \sigma_v^2 & \rho_{\theta u}\,\sigma_v\,\sigma_\theta \\ \rho_{u\theta}\,\sigma_v\,\sigma_\theta & \rho_{v\theta}\,\sigma_v\,\sigma_\theta & \sigma_\theta^2 \end{bmatrix}, \quad (4.14)$$

where ρ_{uv}, $\rho_{u\theta}$, and $\rho_{v\theta}$ represent the correlation coefficients.

Since our main interest is in the mean values and the covariances of the components (p_x, p_y, ϕ) of the position vector, we need to make explicit the relationship between the **p** vectors at the beginning and at the end of the commanded rotation. As before, if (p'_x, p'_y, ϕ') are the components of the position vector after the commanded rotation and (p_x, p_y, ϕ) the components before, their relationships are given by

$$\begin{bmatrix} p'_x \\ p'_y \\ \phi' \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ \phi \end{bmatrix} + \begin{bmatrix} u\cos\phi - v\sin\phi \\ u\sin\phi + v\cos\phi \\ \theta \end{bmatrix}. \quad (4.15)$$

As is the case with the translational motion, the relationships between the old and the new positions are nonlinear. For example, the component p'_x is a nonlinear function of p_x, u, v, and ϕ. Again, as before, we are only interested in linear approximations to these relationships, linear in the vicinity of their average values. Moreover, we are less interested in the exact forms of these linear approximations and more in the Jacobians implied by

them. Using derivational steps parallel to the case of the translational motion, we obtain

$$\begin{bmatrix} \delta p_x' \\ \delta p_y' \\ \phi' \end{bmatrix} = \mathrm{J}_{R_1} \begin{bmatrix} \delta p_x \\ \delta p_y \\ \delta\phi \end{bmatrix} + \mathrm{J}_{R_2} \begin{bmatrix} \delta u \\ \delta v \\ \delta\theta \end{bmatrix}, \tag{4.16}$$

where the Jacobians J_{R_1} and J_{R_2} are given by

$$\mathrm{J}_{R_1} = \begin{bmatrix} 1 & 0 & -\bar{u}\sin\bar{\phi} - \bar{v}\cos\bar{\phi} \\ 0 & 1 & \bar{u}\cos\bar{\phi} - \bar{v}\sin\bar{\phi} \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and}$$

$$\mathrm{J}_{R_2} = \begin{bmatrix} \cos\bar{\phi} & -\sin\bar{\phi} & 0 \\ \sin\bar{\phi} & \cos\bar{\phi} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.17}$$

We then obtain the mean vector and the covariance matrix as

$$\begin{bmatrix} \overline{p_x'} \\ \overline{p_y'} \\ \overline{\phi'} \end{bmatrix} = \begin{bmatrix} \bar{p}_x \\ \bar{p}_y \\ \bar{\phi} \end{bmatrix} + \begin{bmatrix} \bar{u}\cos\bar{\phi} - \bar{v}\sin\bar{\phi} \\ \bar{u}\sin\bar{\phi} + \bar{v}\cos\bar{\phi} \\ \bar{\theta} \end{bmatrix}. \tag{4.18}$$

$$\Sigma(p_x', p_y', \phi') = \mathrm{J}_{R_1}\,\Sigma(p_x, p_y, \phi)\,\mathrm{J}_{R_1}^{\mathrm{T}} + \mathrm{J}_{R_2}\,\Sigma(u, v, \phi)\,\mathrm{J}_{R_2}^{\mathrm{T}}. \tag{4.19}$$

Equations (4.12), (4.13), (4.18), and (4.19) give us the necessary tools for updating the position uncertainty of the robot after each commanded motion, the first two applying to the case of commanded translations and the latter two to the case of commanded rotations. For these equations to be useful, we must find, through experimentation, the values for the various variances and correlation coefficients that appear in the matrices displayed in Eqs. (4.8) and (4.9). As the reader should be able to infer quickly from the preceding discussion, these variances and correlation coefficients are functions of d_0 for commanded translations, and of θ_0 for commanded rotations.

The functional dependence of the variances and the correlation coefficients on d_0 and θ_0 is an important issue in its own right and is discussed further in Section 9. There we show that the functional dependences, as observed empirically, tend to be smooth functions and therefore easily amenable to either interpolation or extrapolation.

Before concluding this subsection, we would like to draw the reader's attention to the fact that all the calculations in Eqs. (4.12), (4.13), (4.18), and (4.19) for the propagation of the means and variances of the position vector **p** are simple matrix multiplications and therefore can be carried out quickly.

4.3. Landmark Projection Uncertainty

Since the camera on the robot is calibrated in the robot-centered coordinate frame, if there is no uncertainty in the position of the robot in the world frame, it goes without saying that we should be able to predict without error the locations of the various hallway landmarks in the camera image. But, since commanded motions will always cause the position of the robot to become uncertain, we are naturally interested in analyzing the relationship between the uncertainty in the robot position and the resulting uncertainty in the location of a hallway landmark in the camera image.

Before studying the uncertainties associated with the projection of a hallway landmark in the camera image, one must specify a model for the formation of the camera image, the choice of the model playing a central role in the selection of the camera calibration procedure used. Various camera models are available for this purpose. For high-quality cameras with good lenses—most solid-state cameras fall in this category—it usually suffices to use what is frequently called the pinhole model. In this model, for all objects located within the depth of focus of the lens used and when sufficient illumination is available so that a small aperture can be used, the camera is assumed to be a pinhole located one focal length in front of the sensor plane. When these assumptions cannot be satisfied, which is frequently the case with the older vidicon camera and for very high-precision work even with modern cameras, one must take a recourse to other models, such as one that takes radial distortion into account [Tsa87]. Another possibility when a lens cannot be assumed to be a thin lens is to use the two-plane method for camera calibration [MarBir81]. Should the reader be interested in pursuing further the subject of camera modeling, we believe the best recent comparison of the different camera models, with regard to accuracy and the ease with which the physical parameters of a camera can be calculated, appears in [LopKak89].

In the work report here, we have used the pinhole model for image formation. This assumption, which as we mentioned above appears well justified for modern solid-state cameras, especially so in the context of mobile robotics, was also used in [AyaFau89]. This assumption implies that if (x_r, y_r, z_r) are the coordinates of a scene point in the robot-centered coordinate frame and if (X, Y) are the coordinates of the image pixel corresponding to the scene point, then, using homogeneous coordinates, we have the relationship

$$\begin{bmatrix} X\,W \\ Y\,W \\ W \end{bmatrix} = T \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix}, \tag{4.20}$$

where T is the calibration matrix associated with the camera. Since the focus of this exposition is not on camera calibration, the various components of T, which depend on such physical parameters of the camera as focal length, sampling rates, the location of the lens center, etc. [KakNof85], will be left unspecified. When needed, these components will simply be referred to as $\{t_{ij} \mid i = 1, 2, 3, \; j = 1, 2, 3, 4\}$.

Consider a point (x, y, z) in the world frame. Let the robot-centered coordinates of the same point be (x_r, y_r, z_r). If we invoke the assumption stated earlier that the robot is confined to the $z = 0$ plane of the world frame—more accurately stated, the origin of the robot-centered frame is restricted to lie in the xy plane of the world frame—we can write the following relationship between the two sets of coordinates:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & 0 & -p_x\cos\phi - p_y\sin\phi \\ -\sin\phi & \cos\phi & 0 & p_x\sin\phi - p_y\cos\phi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{4.21}$$

where, as usual, (p_x, p_y, ϕ) are the components of the position vector of the robot in the world frame. Let

$$H = \begin{bmatrix} \cos\phi & \sin\phi & 0 & -p_x\cos\phi - p_y\sin\phi \\ -\sin\phi & \cos\phi & 0 & p_x\sin\phi - p_y\cos\phi \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.22}$$

Then we can write the following matrix–vector relationship between points in the world frame and the pixels in the image:

$$\begin{bmatrix} X\,W \\ Y\,W \\ W \end{bmatrix} = T\,H \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{4.23}$$

We now show that this relationship can be used directly for predicting the uncertainty in the camera image associated with a landmark consisting of a single point, such as a hallway corner on the floor. Let us assume the uncertainty associated with the robot at position $\mathbf{p}$ is characterized by the parameters $(\bar{\mathbf{p}}, \Sigma_p)$, where $\bar{\mathbf{p}}$ is the mean vector and Σ_p the covariance matrix for the position vector. As a result of the uncertainty in the position of the robot, the pixel coordinates X and Y will become random variables, and so will the factor W, which captures the perspective effects of imaging through a pinhole. The functional dependence of these random variables on the components (p_x, p_y, ϕ) of the robot position vector is given by

$$W = [t_{31}\; t_{32}\; t_{33}\; t_{34}] \times \begin{bmatrix} x\cos\phi + y\sin\phi - p_x\cos\phi - p_y\sin\phi \\ -x\sin\phi + y\cos\phi + p_x\sin\phi - p_y\cos\phi \\ z \\ 1 \end{bmatrix} \tag{4.24}$$

$$X = \frac{1}{W}[t_{11}\; t_{12}\; t_{13}\; t_{14}] \times \begin{bmatrix} x\cos\phi + y\sin\phi - p_x\cos\phi - p_y\sin\phi \\ -x\sin\phi + y\cos\phi + p_x\sin\phi - p_y\cos\phi \\ z \\ 1 \end{bmatrix} \tag{4.25}$$

$$Y = \frac{1}{W}[t_{21}\; t_{22}\; t_{23}\; t_{24}] \times \begin{bmatrix} x\cos\phi + y\sin\phi - p_x\cos\phi - p_y\sin\phi \\ -x\sin\phi + y\cos\phi + p_x\sin\phi - p_y\cos\phi \\ z \\ 1 \end{bmatrix}. \tag{4.26}$$

Let $\overline{X}$, $\overline{Y}$, and $\overline{W}$ be the mean values of these random variables. Evidently, these mean values are given by

$$\overline{W} = [t_{31}\; t_{32}\; t_{33}\; t_{34}] \times \begin{bmatrix} x\cos\bar{\phi} + y\sin\bar{\phi} - \bar{p}_x\cos\bar{\phi} - \bar{p}_y\sin\bar{\phi} \\ -x\sin\bar{\phi} + y\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi} \\ z \\ 1 \end{bmatrix} \tag{4.27}$$

$$\bar{X} = \frac{1}{\bar{W}} [t_{11}\ t_{12}\ t_{13}\ t_{14}] \times$$

$$\begin{bmatrix} x\cos\bar{\phi} + y\sin\bar{\phi} - \bar{p}_x\cos\bar{\phi} - \bar{p}_y\sin\bar{\phi} \\ -x\sin\bar{\phi} + y\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi} \\ z \\ 1 \end{bmatrix} \quad (4.28)$$

$$\bar{Y} = \frac{1}{\bar{W}} [t_{21}\ t_{22}\ t_{23}\ t_{24}] \times$$

$$\begin{bmatrix} x\cos\bar{\phi} + y\sin\bar{\phi} - \bar{p}_x\cos\bar{\phi} - \bar{p}_y\sin\bar{\phi} \\ -x\sin\bar{\phi} + y\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi} \\ z \\ 1 \end{bmatrix}. \quad (4.29)$$

As is clear from Eqs. (4.24), (4.25), and (4.26), for any given point landmark at a fixed location (x, y, z) in the world frame, the relationship between the pixel coordinates (X, Y) and the robot position (p_x, p_y, ϕ) is nonlinear. As with the case of transformations corresponding to robot motions, we will only be interested in the relationships shown in Eqs. (4.24), (4.25), and (4.26) when all the random variables are in the vicinity of their mean values. This assumption allows us to limit ourselves to linearized approximations to the relations between W, X, and Y on the one hand and p_x, p_y, and ϕ on the other. Since our interest is limited to the propagation of uncertainties from the (p_x, p_y, ϕ) parameter space to the image space, it is not necessary to make explicit this linearized approximation; it is sufficient if we know the Jacobian associated with the approximation. The Jacobian, by definition, relates the differentials of X and Y with the differentials of p_x, p_y, and ϕ in the following manner:

$$\delta X = \begin{bmatrix} \delta X \\ \delta Y \end{bmatrix} = J(\bar{\mathbf{X}}, \bar{\mathbf{p}})\ \delta\mathbf{p}, \quad (4.30)$$

where the Jacobian J is evaluated at the mean values of the random variables involved. Taking the differentials of both sides of the relations in Eqs. (4.24), (4.25), and (4.26), we get for the Jacobian

$$J(\bar{\mathbf{X}}, \bar{\mathbf{p}}) = \begin{bmatrix} \frac{1}{\bar{W}} & 0 & -\frac{\bar{X}}{\bar{W}} \\ 0 & \frac{1}{\bar{W}} & -\frac{\bar{Y}}{\bar{W}} \end{bmatrix} \times T \times$$

$$\begin{bmatrix} -\cos\bar{\phi} & -\sin\bar{\phi} & -x\sin\bar{\phi} + y\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi} \\ \sin\bar{\phi} & -\cos\bar{\phi} & -x\cos\bar{\phi} - y\sin\bar{\phi} + \bar{p}_x\cos\bar{\phi} + \bar{p}_y\sin\bar{\phi} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (4.31)$$

The covariance matrix associated with the pixel coordinates of a single point landmark in the scene may now be written as

$$\Sigma_X = E[\delta\mathbf{X}\ \delta\mathbf{X}^{\mathrm{T}}] = J(\bar{\mathbf{X}}, \bar{\mathbf{p}})\ \Sigma_p\ J(\bar{\mathbf{X}}, \bar{\mathbf{p}})^{\mathrm{T}}, \quad (4.32)$$

where Σ_p is the covariance associated with the robot position vector $\mathbf{p}$.

Figures 7, 8, and 9 demonstrate how the uncertainty of the image pixels corresponding to point landmarks in the hallway is used. Assume the robot to be located at the origin of the world frame with its orientation ϕ equal to zero. Actually, this will be the mean position of robot in its uncertainty description while the robot is situated at the origin of the world frame. Therefore,

$$\bar{\mathbf{p}} = \begin{bmatrix} 0.0\ \mathrm{m} \\ 0.0\ \mathrm{m} \\ 0.0^\circ \end{bmatrix}. \quad (4.33)$$

Further assume the covariance Σ_p at this position of the robot is given by

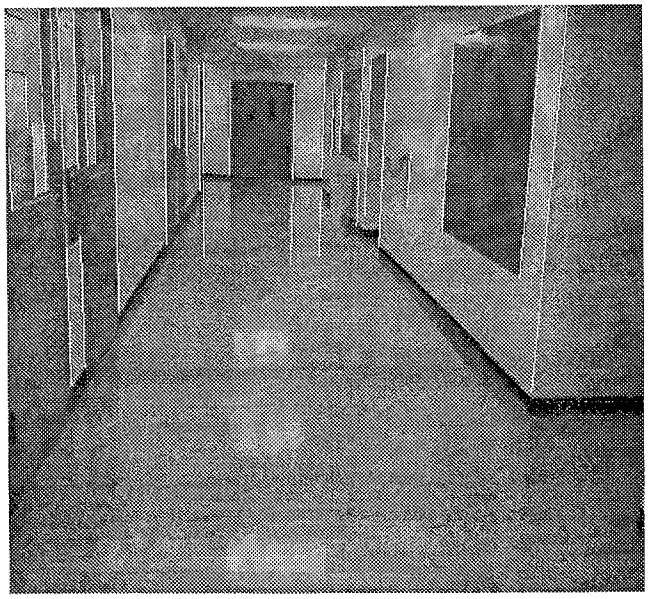


FIG. 7. An image taken by the mobile robot positioned at the origin of the world frame marked O in Fig. 4 and when the orientation of the robot is 0°.

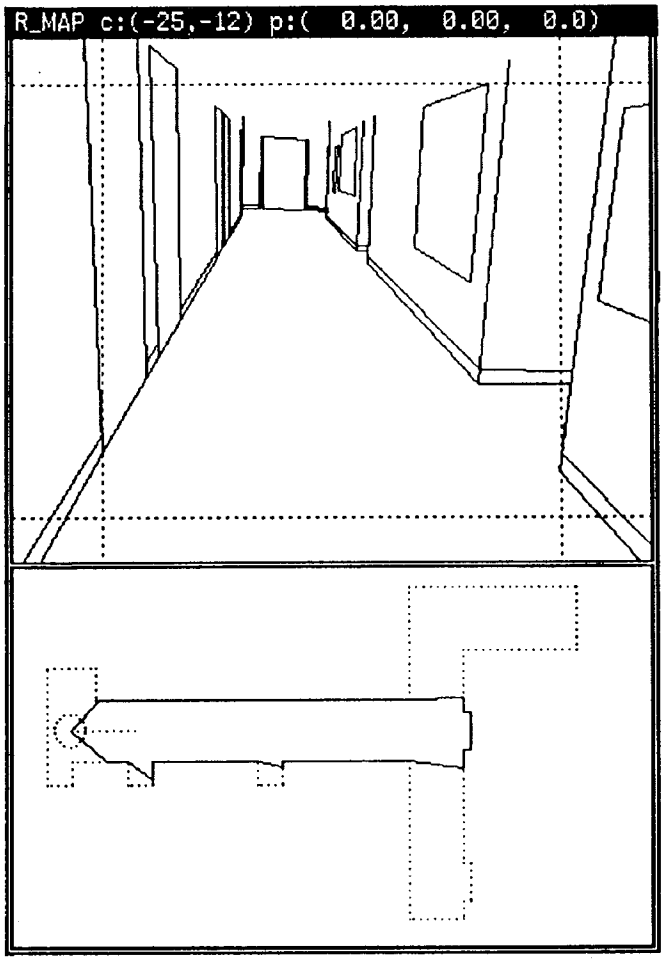


FIG. 8. This figure shows an expectation map when the robot position parameters are the same as in Fig. 7. (During actual navigation, as will be explaining in Section 7, due to uncertainties the position of the robot from where the camera image is taken will not coincide with the position of the robot that is used for the construction of the expectation map.) The upper part of the figure illustrates a rendering from a geometric model of the hallway. The lower part of the figure graphically illustrates the location of the robot in the hallway structure and the portion of the hallway visible to the robot (thick lines).

$$\Sigma_p = \begin{bmatrix} (0.5\text{ m})^2 & 0 & 0 \\ 0 & (0.5\text{ m})^2 & 0 \\ 0 & 0 & (10°)^2 \end{bmatrix}. \qquad (4.34)$$

This uncertainty could be a consequence of the prior motions of the robot. Figure 7 is an actual camera image of the hallway taken when the robot is physically located at the origin of the world frame with its orientation ϕ set to zero. The upper part of Fig. 8 illustrates a rendering from a geometric model of the hallway, assuming the robot to be positioned at the same location as for the image of Fig. 7. Using a 2D floor plan, the lower part of Fig. 8 illustrates graphically the location of the robot in the hallway structure—the origin of the robot frame is at the center of the circle—and the solid line is a depiction of what the robot should see from this position. Treating each corner in the expectation map in the upper part of Fig. 8 as a distinct point landmark, we have shown in Fig. 9 the uncertainty regions corresponding to each corner of the hallway that is expected to be visible. Also shown are the uncertainty ellipses corresponding to the ceiling counterparts of these floor points. In keeping with our explanation in Section 4.2, each ellipse in Fig. 9 corresponds to a Mahalanobis distance of $d = 1$.

All formulas we have derived so far for the uncertainties associated with the image pixels corresponding to the point landmarks can be extended to more extended landmarks, such as lines formed by the junctions of walls, by the junctions of walls with either floors or ceilings, or by the edges of doors and windows, etc. In order to find the location of the line feature in the image let (x_1, y_1, z_1) and (x_2, y_2, z_2) be the two endpoints of a line landmark in the 3D space of the hallway. From Eqs. (4.27), (4.28), (4.29), and (4.32), we obtain the location uncertainty associated with the pixel coordinates corresponding to each of the endpoints. The search for the image line corresponding to the line landmark can now be confined to the convex hull of the two uncertainty regions, as shown in Fig. 10. In other words, to find in the image the pixels that correspond to the line landmark joining the world points (x_1, y_1, z_1) and (x_2, y_2, z_2), we need not analyze the image outside the convex hull shown in Fig. 10.

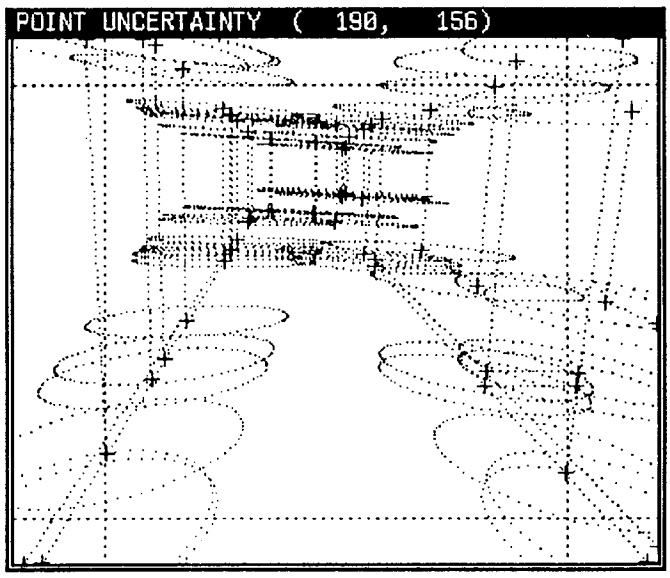


FIG. 9. The uncertainty ellipses for the vertices of the edges in the expectation map of Fig. 8 are shown here; these ellipses correspond to the covariance matrix of Eq. (4.34). The Mahalanobis distance $d = 1$ is used to construct the uncertainty ellipses.

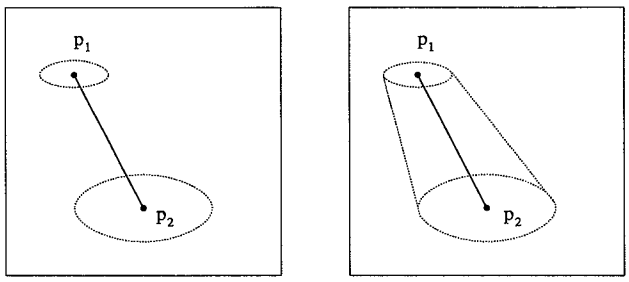


FIG. 10. The left figure depicts the uncertainty ellipses corresponding to one unit of Mahalanobis distance for the endpoints p_1 and p_2. The convex hull, shown in the right figure, of the two end-point uncertainty ellipses specifies the uncertainty region for the line segment.

4.4. Feature Space Uncertainty

The preceding discussion showed how, if we know the uncertainty in the position of the robot, we can limit the region of the camera image in which we must search for the pixels corresponding to a landmark. For obvious reasons, this has the desired effect of making more efficient the establishment of correspondences between the landmarks and the image features.

For those landmarks whose corresponding image features can be easily parameterized, a further speedup in computations can be achieved by constraining the parameter space in the manner discussed here.

Consider, for example, the case of line landmarks. Regardless of the location of the robot, all the straight lines in the hallway will manifest themselves as straight lines in the camera image. And, each straight line in the camera image may be represented by a pair of parameters in the Hough space. Let ρ and γ be used as shown in Fig. 11 to characterize a line feature in the image. The Hough space accumulator will then be spanned by two axes, one corresponding to ρ and the other to γ. So, going back to the case of a line landmark shown on the left in Fig. 10, to find the image line feature corresponding to this landmark, we need to construct a Hough space representation of only those pixels that are in the uncertainty region shown on the right in Fig. 10.

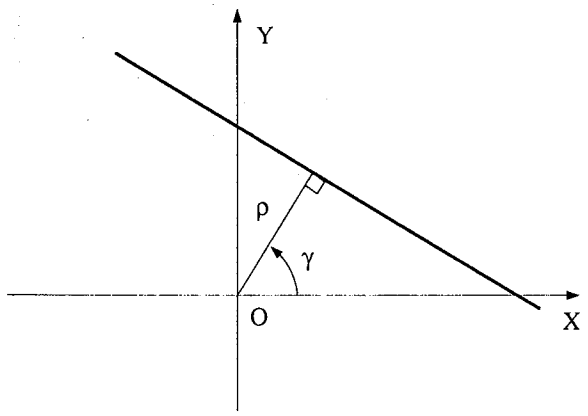


FIG. 11. The two parameters, ρ and γ, that are used for parameterizing a line in the image for its detection by Hough transform.

Filling up the Hough accumulator array can be computationally expensive, even when we limit processing to only those pixels that are in a specified uncertainty region of the image. To get around this difficulty, as shown in this subsection, we propagate the uncertainty from the robot position directly into the Hough space. We can then quickly rule out most of the cells of the accumulator, cells that are outside the uncertainty region. In other words, what we did for the case of line landmarks in the preceding subsection can be pushed one step further by computing the mean values and the covariances of the ρ, γ parameters associated with the image lines corresponding to the line landmarks, and then limiting search in the Hough space to a small region defined by the estimated means and covariances of ρ and γ.

To find the mean and the covariance of where an image line corresponding to a line landmark might fall in the Hough space, consider first a line landmark in the hallways defined by the endpoints (x_1, y_1, z_1) and (x_2, y_2, z_2) in the world frame. Let (X_1, Y_1) and (X_2, Y_2) be the image pixels corresponding to these endpoints.[10]

From this point on we will assume that the origin of the image plane is located at the center of the image plane; in other words, all the pixel coordinates will be measured with respect to the center of the image plane. In actual digitization of images, the pixel coordinates are usually measured with respect to one of the corners; but, for the purpose of using the formulas shown in this section, it is a rather simple matter to enforce the assumption of the origin being at the center by appropriate subtractions.

Under the assumption that $Y_1 \leq Y_2$, we may now write the following expressions for the Hough space parameters of the image line joining (X_1, Y_1) and (X_2, Y_2):[11]

$$\begin{aligned}\rho &= X_1 \cos\gamma + Y_1 \sin\gamma \\ &= \frac{X_1(Y_2 - Y_1)}{\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}} \\ &\quad - \frac{Y_1(X_2 - X_1)}{\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}}\end{aligned} \tag{4.35}$$

[10] For the derivation here, it is not necessary that the points (X_1, Y_1) and (X_2, Y_2) be visible in the image. The final result of this derivation will be in terms of the Hough space parameters of the image line corresponding to the world line joining the endpoints (x_1, y_1, z_1) and (x_2, y_2, z_2).

[11] These formulas for ρ and γ will yield a positive value for the ρ parameter whenever the image line intersects the positive X-axis in the image plane; otherwise, the value of ρ will be negative. (Note that the angle γ is measured positive counterclockwise from the X-axis.) This expedient makes ρ differentiable at the point $\rho = 0$. Another advantage is that the placement of the origin at the center halves the range of ρ, which is significant from a computational standpoint.

$$\gamma = \tan^{-1}\frac{Y_2 - Y_1}{X_2 - X_1} + \frac{\pi}{2}. \tag{4.36}$$

If we now substitute Eqs. (4.25) and (4.26) into the equations shown here, we obtain ρ and γ as functions of the robot position variables p_x, p_y, and ϕ. Since the uncertainty in robot position implies that p_x, p_y, and ϕ are random variables, the above equations tell us that, for a given line landmark in the hallway, ρ and γ will also be random variables; furthermore, these equations show us the functional dependence of the ρ, γ random variables on the p_x, p_y, ϕ random variables. The functional dependences yield the following expressions for the mean values of the random variables ρ and γ:

$$\bar{\rho} = \frac{\bar{X}_1(\bar{Y}_2 - \bar{Y}_1)}{\sqrt{(\bar{X}_2 - \bar{X}_1)^2 + (\bar{Y}_2 - \bar{Y}_1)^2}} - \frac{\bar{Y}_1(\bar{X}_2 - \bar{X}_1)}{\sqrt{(\bar{X}_2 - \bar{X}_1)^2 + (\bar{Y}_2 - \bar{Y}_1)^2}} \tag{4.37}$$

$$\bar{\gamma} = \tan^{-1}\frac{\bar{Y}_2 - \bar{Y}_1}{\bar{X}_2 - \bar{X}_1} + \frac{\pi}{2}, \tag{4.38}$$

where the mean values of the intermediate random variables, $\bar{X}_1$, $\bar{Y}_1$, $\bar{X}_2$, $\bar{Y}_2$ are to be obtained from Eqs. (4.28) and (4.29).

The functional relationships between ρ and γ on the one hand and p_x, p_y, and ϕ on the other are nonlinear. As was the case before with such relationships, we are only interested in linear approximations to how these relationships behave in the vicinity of the mean values of all the random variables. Of course, as before, we are not interested in explicating the precise analytical forms of these linear approximations, but only in deriving the relevant Jacobian, since a knowledge of the Jacobian would then help us propagate the robot position uncertainty into the Hough space. As in the preceding subsection, the Jacobian associated with the linear approximation relates the following differentials:

$$\begin{bmatrix}\delta\rho\\ \delta\gamma\end{bmatrix} = J_{XY}\begin{bmatrix}\delta X_1\\ \delta Y_1\\ \delta X_2\\ \delta Y_2\end{bmatrix}, \tag{4.39}$$

the Jacobian to be evaluated at the mean values of the random variables. Differentiating both sides of Eqs. (4.35) and (4.36) and evaluating the coefficients of the partial derivatives at the mean values of the random variables, we obtain

$$J_{XY} = \begin{bmatrix}\left(1 - \frac{C}{R}\right)\cos\bar{\gamma} & \left(1 - \frac{C}{R}\right)\sin\bar{\gamma} & \frac{C}{R}\cos\bar{\gamma} & \frac{C}{R}\sin\bar{\gamma}\\ \frac{1}{R}\cos\bar{\gamma} & \frac{1}{R}\sin\bar{\gamma} & -\frac{1}{R}\cos\bar{\gamma} & -\frac{1}{R}\sin\bar{\gamma}\end{bmatrix}, \tag{4.40}$$

where

$$R = \sqrt{(\bar{X}_2 - \bar{X}_1)^2 + (\bar{Y}_2 - \bar{Y}_1)^2} \tag{4.41}$$

$$C = \bar{X}_1\sin\bar{\gamma} - \bar{Y}_1\cos\bar{\gamma} \tag{4.42}$$

$$\sin\bar{\gamma} = -\frac{\bar{X}_2 - \bar{X}_1}{R} \quad \text{and} \quad \cos\bar{\gamma} = \frac{\bar{Y}_2 - \bar{Y}_1}{R}. \tag{4.43}$$

Now from Eq. (4.30), we already know

$$\begin{bmatrix}\delta X_1\\ \delta Y_1\end{bmatrix} = J(\bar{\mathbf{X}}_1, \bar{\mathbf{p}})\begin{bmatrix}\delta p_x\\ \delta p_y\\ \delta\phi\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}\delta X_2\\ \delta Y_2\end{bmatrix} = J(\bar{\mathbf{X}}_2, \bar{\mathbf{p}})\begin{bmatrix}\delta p_x\\ \delta p_y\\ \delta\phi\end{bmatrix}, \tag{4.44}$$

where

$$\bar{W}^1 = [t_{31}\ t_{32}\ t_{33}\ t_{34}] \times \begin{bmatrix}x_1\cos\bar{\phi} + y_1\sin\bar{\phi} - \bar{p}_x\cos\bar{\phi} - \bar{p}_y\sin\bar{\phi}\\ -x_1\sin\bar{\phi} + y_1\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi}\\ z_1\\ 1\end{bmatrix} \tag{4.45}$$

$$\bar{X}_1 = \frac{1}{\bar{W}_1}[t_{11}\ t_{12}\ t_{13}\ t_{14}] \times \begin{bmatrix}x_1\cos\bar{\phi} + y_1\sin\bar{\phi} - \bar{p}_x\cos\bar{\phi} - \bar{p}_y\sin\bar{\phi}\\ -x_1\sin\bar{\phi} + y_1\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi}\\ z_1\\ 1\end{bmatrix} \tag{4.46}$$

$$\bar{Y}_1 = \frac{1}{\bar{W}_1}[t_{21}\ t_{22}\ t_{23}\ t_{24}] \times \begin{bmatrix}x_1\cos\bar{\phi} + y_1\sin\bar{\phi} - \bar{p}_x\cos\bar{\phi} - \bar{p}_y\sin\bar{\phi}\\ -x_1\sin\bar{\phi} + y_1\cos\bar{\phi} + \bar{p}_x\sin\bar{\phi} - \bar{p}_y\cos\bar{\phi}\\ z_1\\ 1\end{bmatrix} \tag{4.47}$$

$$\bar{W}_2 = [t_{31}\ t_{32}\ t_{33}\ t_{34}] \times \begin{bmatrix} x_2 \cos\bar{\phi} + y_2 \sin\bar{\phi} - \bar{p}_x \cos\bar{\phi} - \bar{p}_y \sin\bar{\phi} \\ -x_2 \sin\bar{\phi} + y_2 \cos\bar{\phi} + \bar{p}_x \sin\bar{\phi} - \bar{p}_y \cos\bar{\phi} \\ z_2 \\ 1 \end{bmatrix} \tag{4.48}$$

$$\bar{X}_2 = \frac{1}{\bar{W}_2} [t_{11}\ t_{12}\ t_{13}\ t_{14}] \times \begin{bmatrix} x_2 \cos\bar{\phi} + y_2 \sin\bar{\phi} - \bar{p}_x \cos\bar{\phi} - \bar{p}_y \sin\bar{\phi} \\ -x_2 \sin\bar{\phi} + y_2 \cos\bar{\phi} + \bar{p}_x \sin\bar{\phi} - \bar{p}_y \cos\bar{\phi} \\ z_2 \\ 1 \end{bmatrix} \tag{4.49}$$

$$\bar{Y}_2 = \frac{1}{\bar{W}_2} [t_{21}\ t_{22}\ t_{23}\ t_{24}] \times \begin{bmatrix} x_2 \cos\bar{\phi} + y_2 \sin\bar{\phi} - \bar{p}_x \cos\bar{\phi} - \bar{p}_y \sin\bar{\phi} \\ -x_2 \sin\bar{\phi} + y_2 \cos\bar{\phi} + \bar{p}_x \sin\bar{\phi} - \bar{p}_y \cos\bar{\phi} \\ z_2 \\ 1 \end{bmatrix}. \tag{4.50}$$

Substituting Eqs. (4.44) in Eq. (4.39), we obtain

$$\begin{bmatrix} \delta\rho \\ \delta\gamma \end{bmatrix} = J_{XY} \begin{bmatrix} J(\bar{\mathbf{X}}_1, \bar{\mathbf{p}}) & 0 \\ 0 & J(\bar{\mathbf{X}}_2, \bar{\mathbf{p}}) \end{bmatrix} \begin{bmatrix} \delta p_x \\ \delta p_y \\ \delta\phi \end{bmatrix}. \tag{4.51}$$

Therefore, the covariance matrix associated with the uncertainty in the Hough space is given by

$$\Sigma_{\rho\gamma} = J_{XY} \begin{bmatrix} J(\bar{\mathbf{X}}_1, \bar{\mathbf{p}}) & 0 \\ 0 & J(\bar{\mathbf{X}}_2, \bar{\mathbf{p}}) \end{bmatrix} \times \Sigma_p \begin{bmatrix} J(\bar{\mathbf{X}}_1, \bar{\mathbf{p}}) & 0 \\ 0 & J(\bar{\mathbf{X}}_2, \bar{\mathbf{p}}) \end{bmatrix}^{\mathrm{T}} J_{XY}{}^{\mathrm{T}}. \tag{4.52}$$

To recapitulate what we have accomplished in this section, given a line landmark in the hallway and given estimates of robot uncertainty, we now know two things that have a great bearing on the efficiency of computations:

(i) we now know how to limit the region of the image which must be processed for the detection of the pixels corresponding to the landmark; and

(ii) we also know how to gain a further speedup in computations by limiting the region of the Hough space for the detection of the corresponding image line within the already delimited piece of the image.

The delineations of both the image region and the Hough space accumulator cells are carried out by putting thresholds on the Mahalanobis distance, defined in terms of the covariance matrix, from the mean. For example, if we limit our search in the image plane to a region delineated by two units of Mahalanobis distance, we are guaranteed, with a probability of 86%, that we will find the image line corresponding to the landmark line. Similarly, if, in the Hough space, we only fill those accumulator cells that are within two units of Mahalanobis distance from the cell corresponding to the mean, we are again guaranteed, with 86% probability, that we will detect our landmark line. Figure 12 shows the Hough space uncertainty derived from the robot position uncertainty. The horizontal axis in the figure measures γ and the vertical axis ρ. The ellipses shown correspond to a unit Mahalanobis distance.

Although our discussion has involved only point and linear landmarks, the exact same arguments could be made for any other type of landmark. The only difference would be that instead of a (ρ, γ) parameterization, one will have to use generalized versions of the Hough transform [Bal81]. Other ways of generalizing our formalism include propagating the robot position uncertainty to infer the uncertainties associated with the distance be-

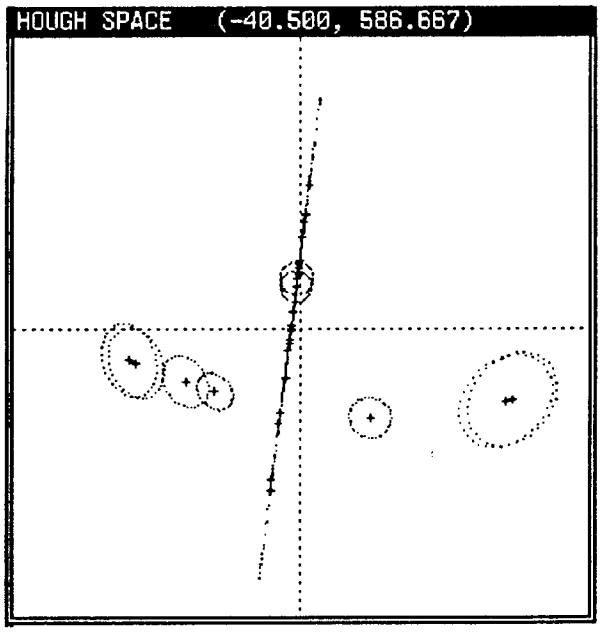


FIG. 12. The Hough-space uncertainty ellipses for the straight lines in the expectation map of Fig. 8. The horizontal axis is the γ parameter depicted in Fig. 11, and the vertical axis the ρ parameter. Again, the ellipses correspond to one unit of Mahalanobis distance.

tween objects in the scene or the angle between the lines joining the centroids of the objects.

5. LANDMARK REPRESENTATION AND ENVIRONMENT MODELING

How the hallways are modeled affects the computational efficiency of the entire system. Perhaps the most straightforward approach to hallway modeling would consist of using the traditional tools of 3D computer graphics. We could, for example, use a CSG (constructive solid geometry) based representation, as was done in the work reported in [KakAnd89], and model a hallway as consisting of a Boolean combination of a small number of volumetric primitives. Or, we could model a hallway by a faceted boundary representation.

The generality and versatility of these traditional approaches to 3D object representation, while certainly a virtue if the aim is to model complex industrial objects, makes them ill-suited for the task at hand. For mobile robot navigation, especially when navigation must be carried out in a dynamic environment created by obstacles in a state of motion, it is imperative that a hallway modeling system allow fast rendering of what would be visible to the camera from any given viewpoint. More precisely, a good hallway modeling scheme will trade off the generality of the more traditional CAD systems for the speed in rendering.

In this section, we present a data structure for hallway modeling. This data structure, while simple, contains all the information that a rendering algorithm would need to construct expectation maps for a vast majority of cases. In contrast with, say, a CSG-based representation, the shortcomings of our data structure is that it must be entirely hand-compiled at the model building time.[12] However, its main advantage is that after the data structure is compiled, for any position of the robot a scene can be rendered in less than a couple of seconds on a run-of-the-mill 16 MIPS serial processor.

5.1. The Data Structure

Fundamental to the hallway data structure is the notion of a *basic face*. It is assumed that a basic face is a vertical planar entity of unspecified height bounded by two vertical lines. (The height information becomes explicit eventually through a pointer to one of the lines; this will be explained later.) Therefore, a face is represented by the following 3-tuple:

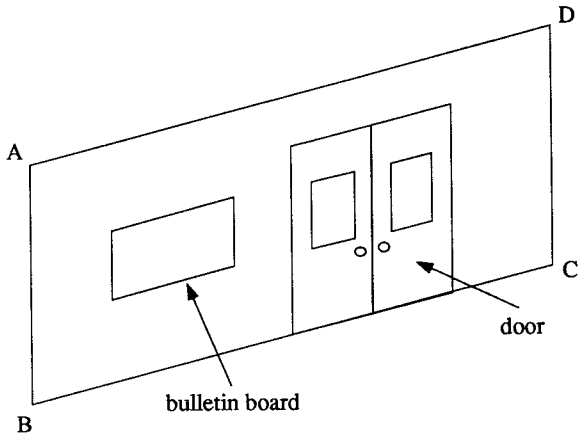


FIG. 13. A wall section in the hallway.

$$(\mathrm{BF}, (x_1, y_1), (x_2, y_2)), \tag{5.1}$$

where the first element, BF, is the symbolic name of the basic face, and the next two elements are the world coordinates of the line formed by the projection of the basic face onto the xy-plane. For example, in the wall section shown in Fig. 13, the geometric entity $ABCD$ is a basic face defined by the xy-coordinates of the vertices B and C in the world coordinate frame.

A basic face points to all the lines that are deemed by a human to be significant from the standpoint of scene interpretation. For example, the basic face delineated by dotted lines in Fig. 14 will point to the 13 lines shown in the figure. While some of these lines help define the face itself in 3D, the other lines are for visually interesting

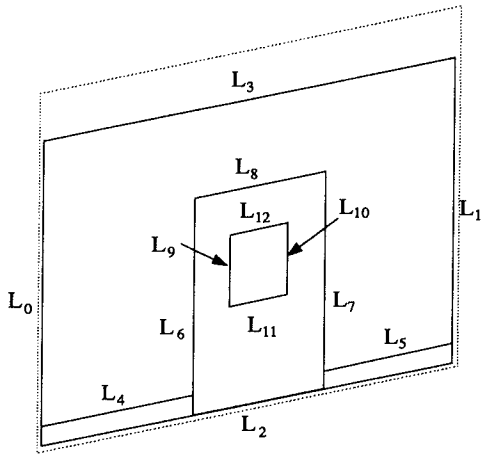


FIG. 14. The node for this basic face will point to the 13 visually significant lines, L_0 through L_{12}.

[12] It is certainly possible to *automatically* extract all the simple landmarks, such as the linear features formed by the junctions of faces, from a CSG-based representation. However, such extraction requires what is called *boundary evaluation*, which for a given viewpoint is the determination of the edges of a Boolean combination of solids whose edges are already known [Til80]. Boundary evaluation, which usually involves ray tracing along the edges of the solids participating in the Boolean combination, would be too time consuming for our application.

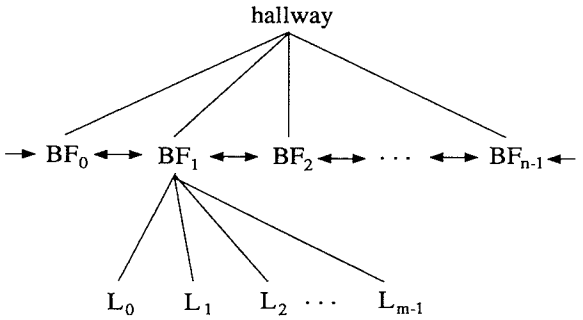


FIG. 15. A complete hallway is represented by a threaded-tree data structure. The hallway consists of basic faces BF_0 through BF_{n-1}. Each basic-face node points to the visually significant line segments that it contains.

features contained in the basic face; for example, lines formed by the outline of a door or by features on the door. Each line is represented by a 4-tuple:

$$(L_i, BF_j, (x_1, y_1, z_1), (x_2, y_2, z_2)), \tag{5.2}$$

where L_i is the symbolic name of the line, BF_j a pointer to the face which contains the line, (x_1, y_1, z_1) and (x_2, y_2, z_2) the two end points of the line.

A complete hallway may therefore be represented by the tree data structure shown in Fig. 15. A more precise description of the data structure is that it is a threaded tree, since each basic face node contains a pointer to the next basic-face node in a clockwise traversal of the hallway when the observer is facing in the positive y direction in the world coordinate frame. The geometry corresponding to just the basic-face level nodes in Fig. 15 is shown in Fig. 16 for the hallway for which we will report experimental results in Section 9.[13] Each small square shown in the figure represents a square 1 m on each side.

Obviously, this data structure will not be capable of representing visually interesting features such as fire-extinguishers, indoor plants, and flower pots. But, then, a majority of hallways, for the most part, are devoid of such objects; at least that is the case in our university. It is important to realize that a nonrepresentation of isolated objects is not really a shortcoming of our system, since all objects not in the model will be treated as obstacles by the system. Therefore, it is not critical that all that can be seen is captured in the model, only that sufficiently large portion of what can be used for matching with features extracted from an image be present in the model. The question of what constitutes sufficiency in the modeling processes will not be pursued at this time, except to say that if the final uncertainty in the robot position calculated on the basis of the available modeling information proves excessive, the robot turns and tries a different viewpoint.

[13] The alcoves formed by the protruding face clusters in Fig. 16, like the cluster $\{BF_{19}, BF_{20}, BF_{21}\}$, lead to doors to the laboratories on the right side of the main hallway. The height of the ceiling in this alcove is lower than the height of the ceiling in the main hallway. A consequence of our simple data structure is that the horizontal line formed by the lower ceiling would not be representable in the model. However, this is not a matter of serious consequence, as explained later in the section.

5.2. Rendering Expectation Maps from the Model

As stated before, by definition an expectation map is a map of what the robot expects to see from a given position. Since in our current implementation all the visually interesting features are of the linear variety, the expectation map is obtained by rendering an edge image from the data structure of the hallway model described in the previous subsection. We will now describe a particularly simple algorithm that can quickly construct an expectation map given the robot position and the calibration matrix of the camera in the robot coordinate frame.

The rendering algorithm consists of two stages. In the first stage, we categorize all the basic faces with regard to whether they are fully visible, fully invisible, or only par-

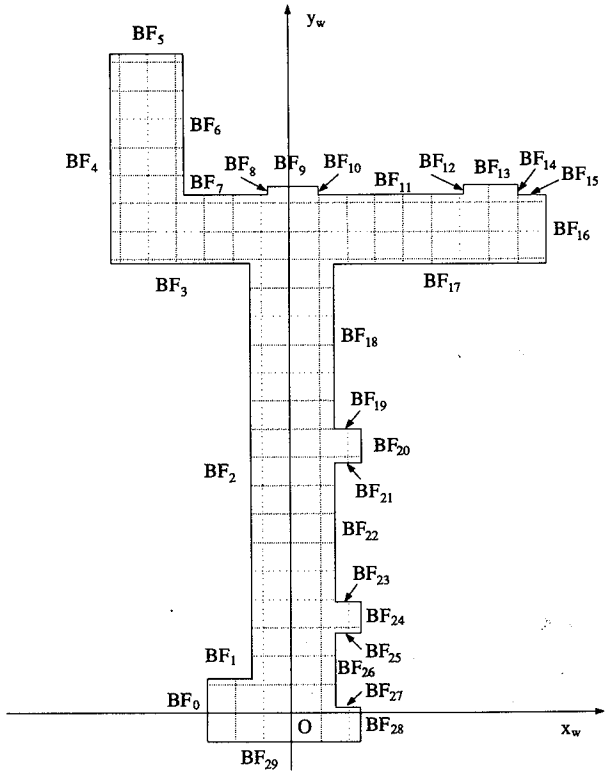


FIG. 16. The geometry corresponding to the hallway data structure, a generic example of which was shown in Fig. 15. Essentially this figure shows the spatial arrangement, vis-à-vis the actual layout of the hallway, of the basic face nodes BF_0 through BF_{29} in the data structure of Fig. 15.

tially visible. This determination is made by analyzing only the basic-face level nodes in the data structure of Fig. 15. What that means is that analysis is confined to the two-dimensional basic-face representation shown in Fig. 16. After the determination of those basic-faces that are fully or partially visible has been made, in the second stage the line features hanging from the corresponding basic-face level nodes in Fig. 15 are then projected into the camera plane using the calibration matrix of the camera. This two-stage approach is more fully explained with the help of the following steps:

Step 1: Specify the Frustrum

We associate a viewing frustrum with the camera in the robot coordinate frame (Fig. 17). In the pinhole model the frustrum is formed by projecting the sensor plane in the camera out through the pinhole that is located one focal length away. Only what is inside the frustrum will be imaged onto the sensor plane.

The near and the far planes of the frustrum correspond to the depth of field of the camera lens, which in turn is a function of the spatial resolution of the sensor [KakNof85]. The frustrum is also characterized by the angles Ψ_h and Ψ_v shown in Fig. 17. Evidently (made obvious by the geometry shown in the figure) these angles are directly a function of the size of the sensor plane. For the camera we use on our mobile robot, these angles are $\Psi_h = 52°$, $\Psi_v = 35°$.

We will assume that all the relevant parameters of the camera are already known, meaning that in the robot coordinate frame we know the locations of the near and far planes of the viewing frustrum, the calibration matrix which incorporates information on parameters such as the location of the camera lens center and the pan, tilt, and swing angles associated with the direction of the optic axis of the camera.

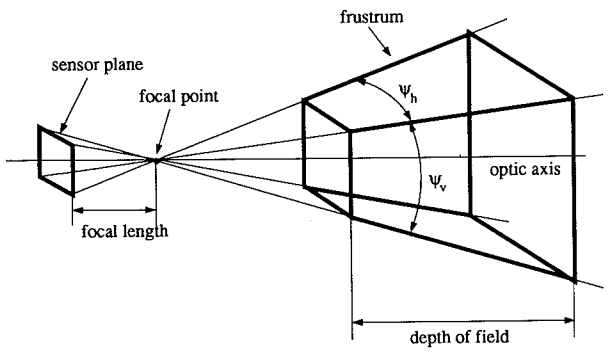


FIG. 17. The geometry of the viewing frustrum for the pinhole model.

Step 2: Construct line_of_sight_space

The *line_of_sight_space* is the set of all the points in the hallway that can be connected by straight lines with the (x, y) location of the robot without interference from any other parts of the hallway. It is important to note that the *line_of_sight_space* does not take into account the orientation of the robot.

For the construction of the *line_of_sight_space*, we now assume that the tilt and the swing angles of the camera are zero. It is important to realize that despite this assumption we obtain correct expectation maps even when the tilt angle is nonzero.[14] That is so because eventually the actual contents of the expectation map are found by taking into account the full calibration matrix of the camera, which implies using the actual tilt angle. Note that in Step 2, we only want to make quick and coarse determination of which basic faces are visible and to what extent. By assuming zero tilt and zero swing, we can carry out a fast two-dimensional analysis of just the mid-level nodes in the data structure of Fig. 15 to make this determination.

To explain this two-dimensional analysis of the geometry implied by the mid-level nodes in Fig. 15, assume the robot is located at position P shown in Fig. 18. Using the lateral pointers at the basic face level, the system now extracts from the threaded tree shown in Fig. 15 an ordered list of vertices defining the basic faces. This list, shown in Fig. 19, is processed simultaneously to yield

(i) the vertex that is closest to the current (x, y) location of the robot; this vertex is called the distinguished vertex and denoted v^*,

(ii) the visibility status of each vertex, meaning whether or not a vertex would be visible from the current location of the robot *regardless of its orientation.*

In Fig. 18, we have displayed the distinguished vertex v^* and marked with heavy dots the visible vertices for the location of the robot shown there. While finding the closest vertex is simple as it only requires that the Euclidean distances from each of the vertices to the current robot location be computed, the computation of visibility is, relatively speaking, slightly more cumbersome since we must determine whether or not the line joining the vertex with the robot location intersects any of the other baselines.[15] The computational complexity of this computa-

[14] For the type of research reported here, there is almost never a need for a nonzero swing angle, unless one is experimenting with active vision (now more frequently referred to as animate vision) where the cameras must be free to point in almost any direction within a large range. In all our experimentation, the swing angle is always kept at zero.

[15] For a basic face defined by the tuple (BF, (x_1, y_1), (x_2, y_2), the baseline is the line joining the points (x_1, y_1) and (x_2, y_2) in the xy-plane of the world coordinate frame.

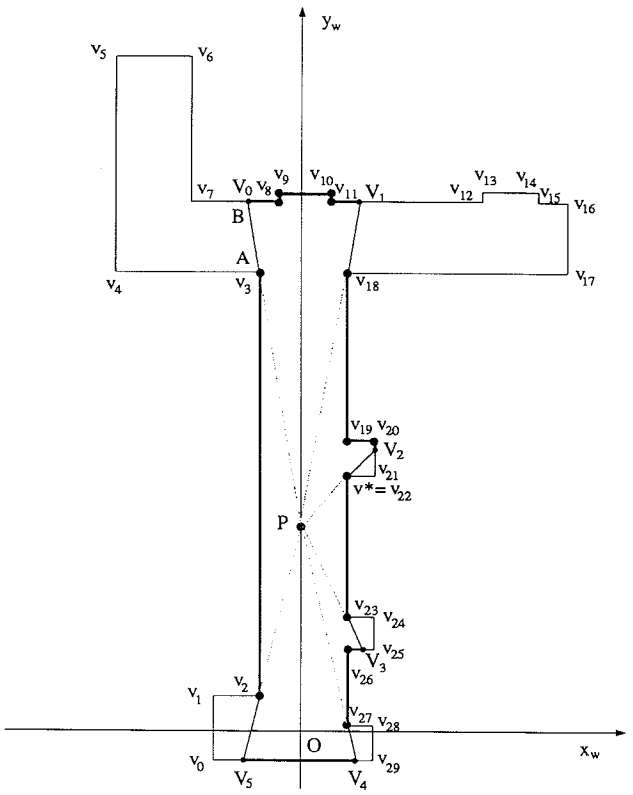


FIG. 18. The line_of_sight_space for the position of the robot. It is the set of all points on the hallway floor that would be visible from the center of the robot regardless of the orientation of the robot.

tion is $O(N^2)$, where N is the number of basic faces in the hallway.

Starting from the distinguished vertex v^* and the visibility status of each vertex in the linked list of Fig. 19, the system then constructs the *line_of_sight_space* by using the following steps:

(a) Scan the list of the visible vertices, starting from the distinguished vertex v^*.

(b) If two consecutive visible vertices belong to the same basic face, as is the case with the vertices v_{22} and v_{23} in Fig. 18, declare that part of the hallway that is enclosed by this basic face and the lines joining its vertices to the (x, y) location of the robot as belonging to the visible space.

(c) If two consecutive visible vertices do not belong to a single basic face, as is the case with the vertices v_{23} and

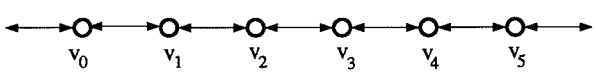


FIG. 19. The ordered list consisting of all the vertices that define the basic faces of the hallway.

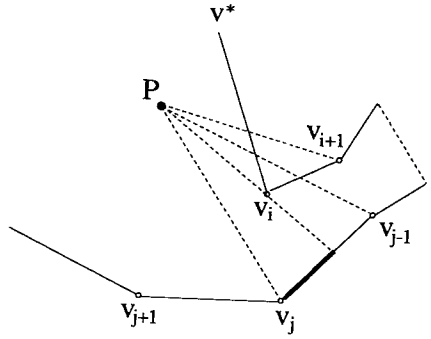


FIG. 20. Case 1 for the construction of the line_of_sight_space.

v_{26} in Fig. 18, a portion of the hallway bounded by these two vertices and the other vertices that are between them in the ordered list of Fig. 19 must be partially invisible from the current (x, y) location of the robot. Similar situations would arise with the visible pair of vertices v_{27} and v_2 and with the visible pair v_{20} and v_{22}. The following three rules are used to delineate the visible fraction of the hallway when any of these situations arise:

Case 1. This case, corresponding to the situation at the (v_{23}, v_{26}) pair of visible vertices in Fig. 18, is depicted more generally in Fig. 20. Let v_i and v_j be two consecutive visible, but nonadjacent, vertices with $i < j$. Let $\angle v_i$ denote the angle $\angle Pv_iv^*$, the angles being measured clockwise from the line Pv^*. The test of Case 1 is that $\angle v_{j-1} < \angle v_j$. If this condition is satisfied, a part of the basic face defined by the vertices v_{j-1}, v_j must be visible. This part is extracted by extending the line Pv_i to the baseline v_{j-1}, v_j.[16]

Case 2. This case, corresponding to the situation at the (v_{20}, v_{22}) pair of visible vertices in Fig. 18, is depicted more generally in Fig. 21. Again let v_i and v_j be two consecutive visible, but nonadjacent, vertices with $i < j$. The test of Case 2 is $\angle v_{i+1} > \angle v_i$. If this condition is satisfied, a part of the basic face defined by the vertices v_i, v_{i+1} must be visible. This fraction of the basic face is extracted by extending the line Pv_j to the baseline v_i, v_{i+1}.

Case 3. If the consecutive but nonadjacent pair of vertices v_i and v_j, $i < j$, does not satisfy either of the two cases above, then Case 3 must apply. This case, which corresponds to the visually consecutive vertices v_{27}, v_2 in

[16] Strictly speaking, this would only give use the visible part of the physical hallway line corresponding to the base line v_{j-1}, v_j. What we are doing is tantamount to assuming that the visible and the invisible parts of a basic face are divided by a vertical line passing through the visible/invisible transition on the baseline v_{j-1}, v_j. This assumption is strictly true only when the tilt angle of the camera is zero. For nonzero tilt angles, this assumption is capable of injecting tiny fragments of some of the invisible edges into the frustrum. Since these extra edges are visually insignificant, they have never caused any problems.

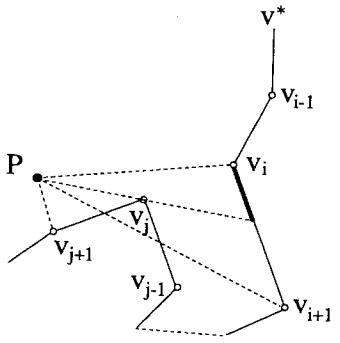


FIG. 21. Case 2 for the construction of the line_of_sight_space.

Fig. 18, implies that the lines joining the robot center P with the vertices v_i and v_j be extended until they both intersect one of the baselines. The intersection points thus obtained give us the visible region that must then be included with the rest of the visible space (Fig. 22).

This is the end of Step 2. At the end of Step 2, we have the *line_of_sight_space* for the current location of the robot regardless of its orientation. In Fig. 18, we have shown this space for the location of the robot marked there. In the computer memory, the data structure for representing the *line_of_sight_space* consists of ordered list of vertices:

$$(v^*, v_{i_1}, \ldots, V_{j_1}, V_{j_2}, \ldots, v_{k_1}, v_{k_2}, \ldots, V_{l_1}, \ldots), \tag{5.3}$$

where the v_i nodes are the visible nodes corresponding to the basic faces in the data structure of Fig. 18, and the V_j nodes are the new nodes corresponding to the intersection of the lines of sight with the baselines of the partially visible basic faces (Fig. 18). To decide quickly whether or not a given consecutive pair of vertices in the list shown above corresponds to an actual basic face or to a segment like AB in Fig. 18, we also store, in parallel with the list of vertices shown above, a list of status flags, each entry in this list being either a 1 or a 0.

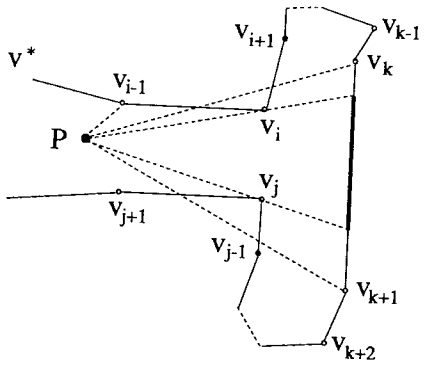


FIG. 22. Case 3 for the construction of the line_of_sight_space.

Step 3: Construct visible_space

Recall that the *line_of_sight_space* gives us the set of all the points that would be visible from the current (x, y) location of the robot *without taking into account the robot orientation*. We now extract that portion of the *line_of_sight_space* which would be visible for a specific orientation ϕ of the robot. This latter space will be called the *visible_space*. Clearly, the *visible_space* is the intersection of the *line_of_sight_space* with the frustrum of the camera.

To compute this intersection, the system simply scans the vertex data structure in Eq. (5.3) and compares the angle $\angle v^* P v_i$ or the angle $\angle v^* P V_j$, as the case may be, with the angles defining the projection of the frustrum on the xy-plane. Basically, we want to make sure that the angles corresponding to v_i or V_j are within the Ψ_h angle of the direction in which the robot is facing.

Displayed in Fig. 23 is the *visible_space* for the position and the orientation of the robot shown there.

Step 4: Construct Expectation Map

The expectation map is constructed by scanning from left to right the *visible_space* data structure shown in Eq. (5.3) and addressing the following cases:

(i) If the status flag associated with the current vertex and the next vertex is 0, then this pair of vertices corresponds to a line like *AB* in Fig. 18. Such delineations of the *visible_space* do not contain any landmarks. So nothing needs to be done.

(ii) If the status flag associated with the current vertex and the next vertex is 1, this pair of vertices defines either a fully visible or a partially visible basic face.[17] We will now discuss these cases separately.

Case 1. If the basic face is fully visible, we simply project all the leaf nodes that hang from this basic face node (Fig. 15) into the camera plane. Recall that the leaf nodes are the visually significant straight-line features contained in the basic face. Of course, the projections of the linear features onto the camera plane must be followed by clipping to remove fragments that are not within the frame defining the image.

Case 2. If the basic face is only partially visible, as is the case with the basic face BF_2 in Fig. 23, we must

[17] As was mentioned before, the full and the partial visibilities are only with respect to the horizontal extent of the frustrum. Any clipping due to the vertical extent of the frustrum will be taken into account automatically via the camera calibration matrix during the projection of the landmarks onto the camera plane.

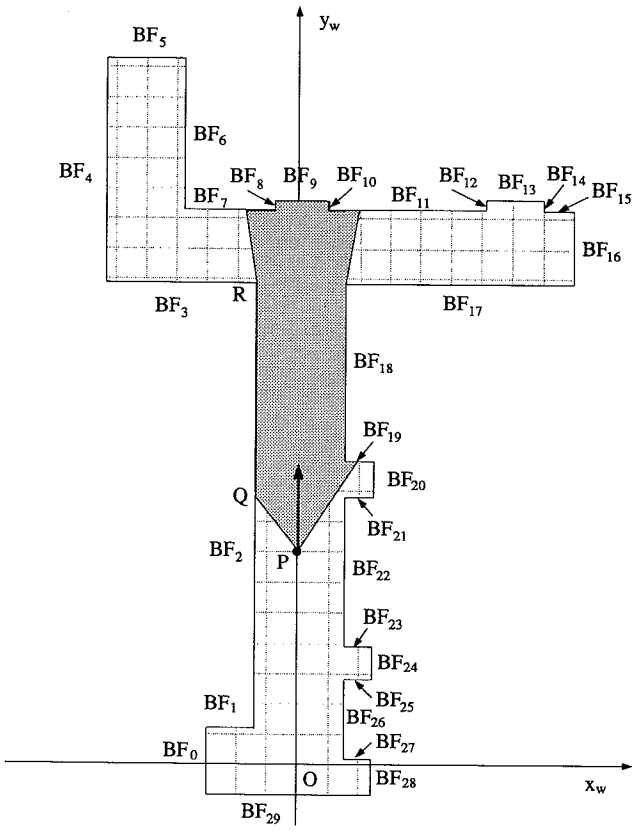


FIG. 23. Visible_space for the location and the orientation of the robot at point P. While the line_of_sight_space in Fig. 18 does not take into account the orientation of the robot, the visible_space does. Visible_space also takes into account the frustrum of the camera.

extract only those segments of the linear features that are within the *visible_space*. Consider the example of single basic face shown in Fig. 23, where the segment QR on the floor describes the extent of what is visible in this basic face. To extract the visible parts of the linear features, we first parameterize the baseline in the manner:

$$x = x_1 + t(x_2 - x_1) \tag{5.4}$$

$$y = y_1 + t(y_2 - y_1). \tag{5.5}$$

Evidently, the point $t = 0$ corresponds to one end of the baseline and the point $t = 1$ to the other. The same parameterization is then used for all the other nonvertical lines in the basic face. What we mean by that is the $t = 0$ for any line feature will correspond to the intersection of the line with the vertical boundary at (x_1, y_1), and $t = 1$ to the vertical boundary at (x_2, y_2). This simple expedient allows us to quickly extract just the visible parts of all the nonvertical lines in the basic face. For a vertical line, we simply examine its projection on the baseline; only if the projection falls within the visible portion is it then projected into the camera plane.

Figure 24 shows an expectation map made by this algorithm for the position of the robot shown in Fig. 23.

6. A GEOMETRICAL PROPERTY THAT FURTHER SPEEDS UP EXTRACTION OF VERTICAL HALLWAY LINES FROM CAMERA IMAGES

We discussed in Section 4 two key points that expedite the extraction of the image features required for matching with the landmark features present in the expectation map:

1. For each linear feature in the expectation map, the delineation of the image region to which we may limit our processing.

2. The delineation of a small region of the Hough space to which we may confine the accumulation of pixel counts.

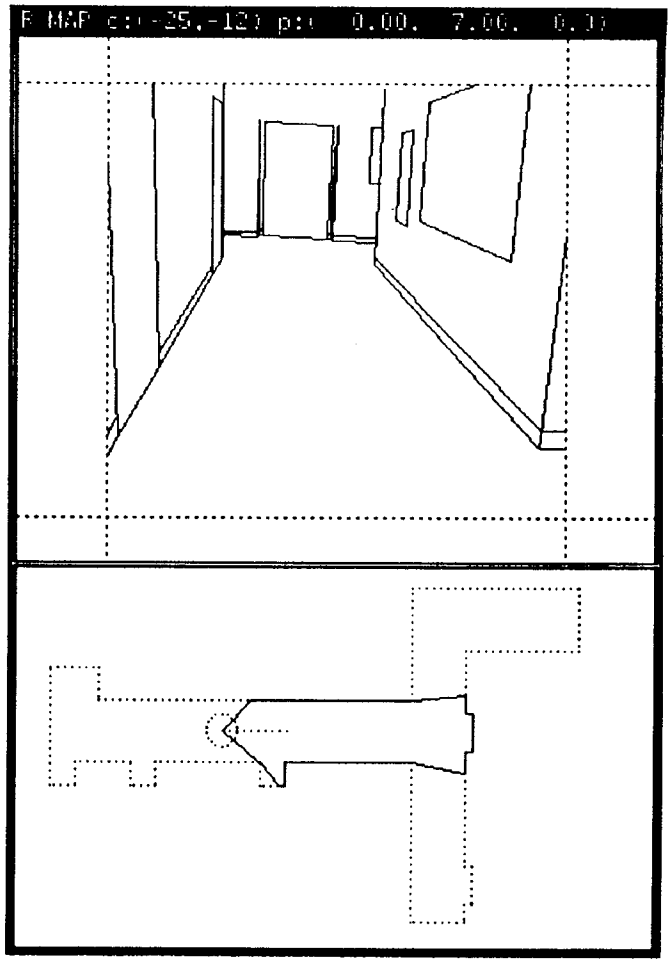


FIG. 24. The combined result of Steps 1 through 3 for the construction of an expectation map when the robot is located and oriented as shown in Fig. 23. While the upper part shows the expectation map, the lower depicts the visible_space.

Both these delineations were a result of our propagating the uncertainty in robot position into the camera image and then into the Hough space.

In this section, we demonstrate that a further reduction in the computation burden can be achieved when the landmark features in the expectation maps correspond to the vertical hallway edges. Vertical hallway edges obey a special property that makes it possible to further confine the search in the Hough space. We state this property with the help of the following lemma:

LEMMA. *There exists a fixed vanishing point* (X_F, Y_F) *in the image plane such that the image lines corresponding to all vertical world lines will pass through* (X_F, Y_F). *Furthermore, the coordinates* (X_F, Y_F) *of the vanishing point in the image plane are independent of the location and orientation of the robot as long as the assumption of a flat floor is satisfied.*

Proof. Let l_v be an arbitrary vertical line in the world coordinate system. l_v can be described as

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \tag{6.1}$$

where (x_1, y_1, z_1) is a point on the line and u is a real parameter. Then the image of the hallway line l_v in the camera frame is given by

$$\begin{bmatrix} X\,W \\ Y\,W \\ W \end{bmatrix} = T\,H \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u + T\,H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}, \tag{6.2}$$

where X and Y are the coordinates of the pixels that fall on the image of the hallway line and W the perspective effect factor. T, the same as in Section 4, is the camera calibration matrix in the robot-centered coordinate frame and H, as given by Eq. (4.22), is the 4×4 homogeneous transformation matrix that for a given point in the world coordinate frame gives us its coordinates in the robot-centered coordinate frame.

Let (X_1, Y_1) be the image pixel corresponding to the point (x_1, y_1, z_1) in the world coordinate system. Then

$$\begin{bmatrix} X_1\,W_1 \\ Y_1\,W_1 \\ W_1 \end{bmatrix} = T\,H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}. \tag{6.3}$$

Subtracting Eq. (6.5) from Eq. (6.4), we get

$$\begin{bmatrix} X\,W \\ Y\,W \\ W \end{bmatrix} - \begin{bmatrix} X_1\,W_1 \\ Y_1\,W_1 \\ W_1 \end{bmatrix} = T\,H \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u = T \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u = \begin{bmatrix} t_{13} \\ t_{23} \\ t_{33} \end{bmatrix} u, \tag{6.4}$$

where the second equality follows from the structure of H shown in Eq. (4.22). The last equality above makes explicit the elements of the third row of the 3×4 T matrix; the precise values of these elements are not important for the proof here. Therefore,

$$X\,W - X_1\,W_1 = t_{13}\,u \tag{6.5}$$

$$Y\,W - Y_1\,W_1 = t_{23}\,u \tag{6.6}$$

$$W - W_1 = t_{33}\,u. \tag{6.7}$$

Substituting the value of W from Eq. (6.7) in Eqs. (6.5) and (6.6), we get the following pair of equations:

$$X = \frac{t_{13}}{t_{33}} + \frac{W_1}{t_{33}\,u + W_1}\left[X_1 - \frac{t_{13}}{t_{33}}\right] \tag{6.8}$$

$$Y = \frac{t_{23}}{t_{33}} + \frac{W_1}{t_{33}\,u + W_1}\left[Y_1 - \frac{t_{23}}{t_{33}}\right]. \tag{6.9}$$

Since u is arbitrary, we can define a new parameter s as

$$s = \frac{W_1}{t_{33}\,u + W_1}. \tag{6.10}$$

In terms of s, Eqs. (6.8) and (6.9) may be written as

$$X = \frac{t_{13}}{t_{33}} + s\left[X_1 - \frac{t_{13}}{t_{33}}\right] \tag{6.11}$$

$$Y = \frac{t_{23}}{t_{33}} + s\left[Y_1 - \frac{t_{23}}{t_{33}}\right]. \tag{6.12}$$

Let

$$X_F = \frac{t_{13}}{t_{33}} \quad \text{and} \quad Y_F = \frac{t_{23}}{t_{33}}. \tag{6.13}$$

Equations (6.11) and (6.12) tell us that the point (X_F, Y_F) will be on the image of the hallway line l_v regardless of the choice of the world point (x_1, y_1, z_1) and regardless of the position of the robot since the expressions for X_F and Y_F are independent of the elements of the matrix H. Therefore, we may say that the point (X_F, Y_F) is a vanishing point for the images of the vertical hallway lines in the

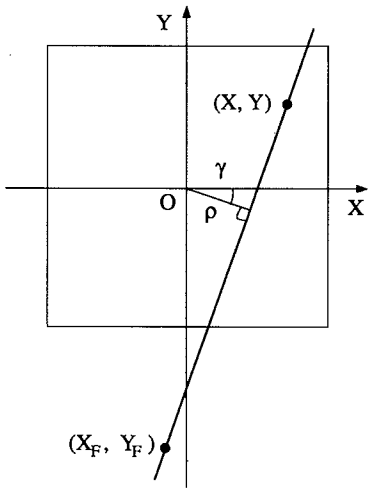


FIG. 25. (X_F, Y_F) designates the vanishing point, meaning that the image lines corresponding to all the vertical lines in the hallway will pass through this point in the image. (X, Y) is an arbitrary pixel on an image line that is the perspective projection of a vertical hallway line.

camera image, since the images of all the vertical lines, regardless of the position of the robot, will pass through this point in the image plane. Q.E.D.

We will now show how the above lemma may be used to expedite the recovery of the vertical hallway lines from an image. Let (X, Y) be an arbitrary pixel in the image and assume for a moment that the pixel falls on the image of a vertical hallway line. We note from the lemma that the image line through (X, Y) must pass also through (X_F, Y_F). The Hough space characterization of this line is given by (Fig. 25)

$$\rho = X_F \cos\gamma + Y_F \sin\gamma \tag{6.14}$$

$$\gamma = \tan^{-1}\frac{Y - Y_F}{X - X_F} + \frac{\pi}{2}. \tag{6.15}$$

As expected, all the points on the image line passing through (X, Y) and (X_F, Y_F) will be characterized by the same values of ρ and γ; therefore, all those pixels will contribute to the cell located at this (ρ, γ) point in the Hough space.

Now consider all the vertical hallway lines simultaneously. Since the vanishing point is unique, the images of these lines will all pass through the same point, the point (X_F, Y_F), as shown in Fig. 26. Although for all these lines in the image, both ρ and γ will be different and given by Eqs. (6.14) and (6.15), the value of γ will stay close to 0. Since γ is close to 0, we may use the following approximations to those two equations:

$$\gamma \approx \tan\gamma = -\frac{X - X_F}{Y - Y_F} \tag{6.16}$$

$$\rho = X_F \cos\gamma + Y_F \sin\gamma \approx X_F + Y_F\,\gamma. \tag{6.17}$$

Although these approximations imply that we may locate all the hallway vertical lines in the Hough space by confining our search to a straight line in the Hough space, we have chosen to follow a different procedure, which is based on the precise forms in Eqs. (6.14) and (6.15). This procedure is explained next.

Algorithm for the Extraction of Vertical Hallway Lines

The actual procedure used by us for the extraction of vertical hallway lines is displayed in Fig. 27. We explain this procedure under the assumption that we are interested in extracting all the vertical hallway lines simultaneously from the image. The reader must realize that in actual robot navigation, that would never be done. In actual robot navigation, for each vertical line in the expectation map the processing of the image will be confined to the uncertainty regions delineated according to the arguments advanced in Section 4.

We first define a histogram, ρ_hist, in just the ρ space. Lines 1 through 3 initialize this histogram. We then apply an oriented edge detector to the image. This edge detector, denoted diff_$x(X, Y)$, is simply the x-component of the Sobel operator. Since the angular sensitivity of both components of the Sobel operator is not that sharp, the diff_x operator will respond almost uniformly to the images of all the hallway vertical lines regardless of their values of γ. As stated by line 7 of the procedure, if at a given pixel the absolute value of the response to diff_x exceeds a certain threshold, compute γ and ρ for that pixel using Eqs. (6.14) and (6.15). Subsequently, in line 9, the appropriate cell of the histogram is incremented.

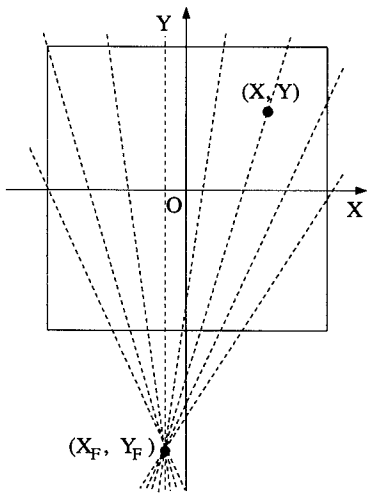


FIG. 26. If we consider all the vertical hallway lines simultaneously, the value of γ, defined in Fig. 25, will stay to close to zero. Note that the images of all the vertical hallway lines will pass through the point (X_F, Y_F) regardless of the location and the orientation of the robot.

Algorithm for Vertical Line Detection:

```
1: for i := -Max to Max do begin
2:    ρ_hist(i) := 0;
3: end;

4: for Y := Row_min to Row_max do begin
5:    for X := Column_min to Column_max do begin
6:       z := diff_x (X,Y);
7:       if (|z| > threshold) then begin
8:          γ = tan^-1 (Y - Y_F)/(X - X_F) + π/2;
9:          ρ = X_F cos γ + Y_F sin γ;
10:         ρ_hist(ρ) := ρ_hist(ρ) + 1;
11:      end;
12:   end;
13: end;

14: Extract peaks in ρ_hist(i);
```

FIG. 27. The algorithm for the detection of the hallway vertical lines. A special feature of this algorithm is that it uses only a singly indexed accumulator even though the perspective projections of vertical lines possess different slopes in the image.

The end result of this procedure is a one-dimensional histogram in the ρ space. All the vertical lines may now be detected by thresholding this histogram. In other words, we have reduced the complexity of vertical line detection by eliminating the need to search in two dimensions of the (ρ, γ) space; now we need search in only one dimension, the ρ dimension. The method works because the angle γ is confined to a small interval around the point $\gamma = 0$. Therefore, it is possible to actually construct a one-dimensional histogram *along* the arc traced by the curve defined by Eqs. (6.14) and (6.15) in the two-dimensional (ρ, γ) space.

Figure 28 shows the values in the different cells of the ρ_hist histogram for the image of Fig. 7. The count corresponding to each peak in this histogram is the number of pixels along a vertical line in the image. Since we are not interested in vertical lines that are too short—many of these short vertical lines correspond to glare reflections of door edges and hallway corners in the linoleum floor—a low threshold is first applied to the histogram to eliminate the low ripple that is visible in Fig. 28. In much of our experimentation, a threshold count of 30 has eliminated most small edges. We then count the remaining number of peaks and apply a new threshold to make sure that the final number of peaks does not exceed twice the number of the vertical lines in the expectation map. In that sense, the threshold selection is dynamic and always maintains approximately the same ratio between the same number of candidate image lines and number of hallway vertical lines. Note that it is not possible to fine-tune the threshold selection to the extent that the number of lines extracted from the image would equal exactly the number of vertical lines in the expectation map; there are many instances when the distortions in imaging and in image processing cause the appearance of more than one line at the output of the edge detector even when there should be only one. Another important phenomenon that can lead to the number of lines in the image being larger or smaller vis-à-vis the number of lines in the expectation map is the occlusion created by the misregistration between the actual location of the robot and the computer's estimate of where the robot is. Obstacles can be yet another source of additional lines in an image.

This dynamic thresholding of the ρ_hist histogram yields the values of ρ corresponding to all the visually significant vertical lines. For each such ρ, we now calculate the associated γ from the following equation derived from Eqs. (6.14) and (6.15):

$$\gamma = \sin^{-1} \frac{\rho}{\sqrt{X_F^2 + Y_F^2}} - \alpha, \tag{6.18}$$

where the value of α is given by

$$\sin \alpha = \frac{X_F}{\sqrt{X_F^2 + Y_F^2}} \quad \text{and} \quad \cos \alpha = \frac{Y_F}{\sqrt{X_F^2 + Y_F^2}}. \tag{6.19}$$

Although it would seem that the solutions to these equations first for α and then for γ would be multivalued, in practice that difficulty does not arise because of the con-

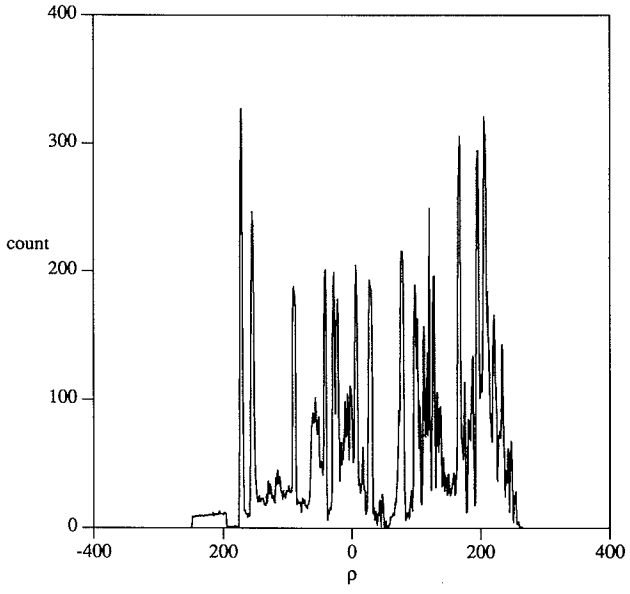


FIG. 28. This figure shows the ρ histogram for the image shown in Fig. 7.

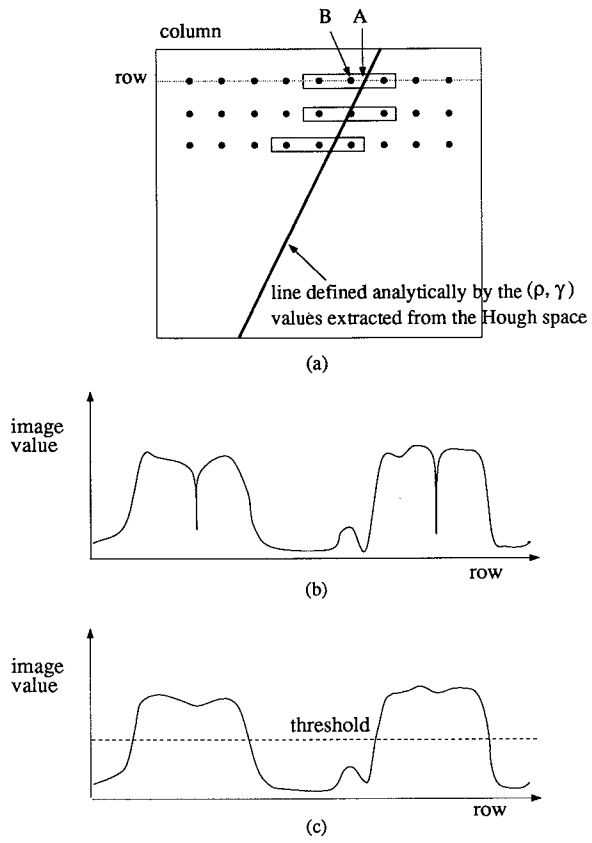


FIG. 29. After the parameters ρ and γ are extracted from the Hough space, the steps used for the extraction of image pixels corresponding to the line defined by these parameters are shown here. (a) First in each row we find the pixel closest to the line defined analytically by the parameters. Next, we find the max gray value within a small window around pixels such as B in each row. A plot of these max gray levels as a function of row index is shown in (b). This function is then smoothed to yield what is shown in (c). Thresholding of this smoothed function yields the straight line segments in the image.

straint that the selected value for γ must be in the vicinity of 0.

Each pair of ρ, γ thus extracted defines an analytical line in the image. To cope with the quantization effects caused by image sampling, we adopt the following procedure to actually identify the pixels that correspond to this ρ, γ analytical line. The intersection of this line with each row is calculated to yield a point such as the point A shown in Fig. 29a. The image pixel nearest to this point is then found by a simple calculation which rounds off the floating point value of the X-coordinate of A to the integer coordinate of the nearest pixel, this pixel being B for the example shown in the figure. Subsequently, in the edge image we look at the values within a 1-neighborhood of the pixel, such as B, and retain the maximum value to construct a one-dimensional array, an example of which is shown in Fig. 29b. This array is then smoothed with a three-element window to yield the array in Fig. 29c, which is then thresholded to identify the white pixels shown in Fig. 30.

Algorithm for the Extraction of Nonvertical Lines

Lest the reader think that we only extract and match vertical hallway lines, we will now quickly show the algorithm for the extraction of the nonvertical lines from the camera image and show results comparable to those just presented for vertical lines. Of course, since nonvertical lines do not obey the nice geometrical property presented earlier—for example, the horizontal hallway lines do not result in a unique vanishing point in the image plane that would be independent of the position of the robot—it is not possible to present a one-dimensional search algorithm for their extraction.

Since the procedure for the extraction of nonvertical hallway lines shares many of the latter steps of the procedure described above for the hallway vertical lines, we will only discuss the initial steps here that are different.

Figure 31 shows a procedure that constructs a two-dimensional histogram in the (ρ, γ) space for the extraction of the nonvertical hallway lines. As explained in Section 4, in line 2 we first compute the means and the covariances associated with the endpoints of the line in the expectation map. These means and covariances delineate the uncertainty regions in which the endpoints of the hallway line would be located with high probability; how high this probability would be depends on how many units of the Mahalanobis distance are used in the defini-

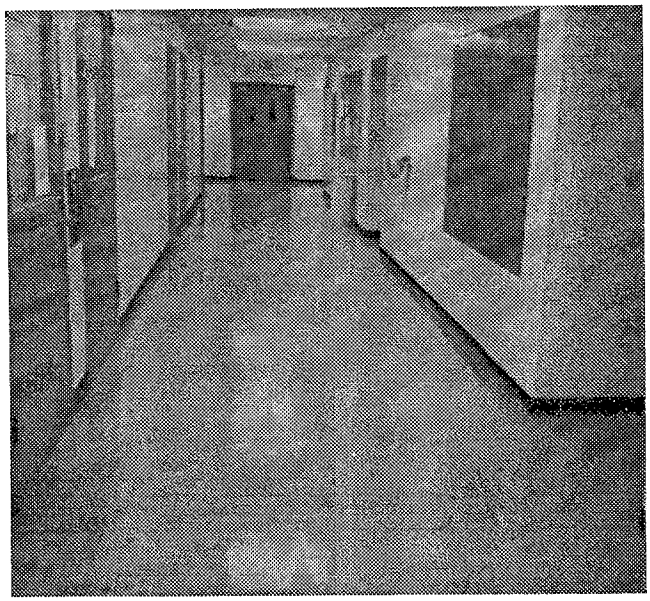


FIG. 30. The superimposed white lines show vertical hallway lines extracted from the image of Fig. 7 by using the algorithm presented in Fig. 27. Note the straight line segments on the glare reflections in the floor of some of the vertical lines.

Algorithm for Non-vertical Line Detection:

```
1: for each linear landmark L do:
2:     compute the uncertainty regions of the two endpoints in the camera frame
           induced by the n-unit Mahalanobis distance;
3:     approximate the uncertainty region of L in the camera frame by a rectangular
           region: [X_min, X_max] in column and [Y_min, Y_max] in row;
4:     compute the uncertainty region of L in the Hough space induced by the
           n-unit Mahalanobis distance;
5:     approximate the uncertainty region of L in the Hough space by a rectangular
           region: [ρ_min, ρ_max] in ρ and [γ_min, γ_max] in γ;
6:     for Y := Y_min to Y_max do begin
7:         for X := X_min to X_max do begin
8:             apply to the neighborhood of (X, Y) a differential operator diff_γ
                   which is especially sensitive to the edge orientation in the range
                   [γ_min, γ_max], producing z := diff_γ(X, Y);
9:             if |z| > threshold then begin
10:                for γ := γ_min to γ_max do begin
11:                    ρ := X cos γ + Y sin γ;
12:                    ργ_hist(ρ, γ) := ργ_hist(ρ, γ) + 1;
13:                end;
14:            end;
15:        end;
16:    end;
17:    extract peaks in ργ_hist;
18: end;
```

FIG. 31. A procedure for the extraction of nonvertical hallway lines.

tion of the uncertainty regions. In line 3, for computational ease a rectangular enclosure is calculated for the convex hull of the two endpoint uncertainty regions. What lines 2 and 3 do in the image space, lines 4 and 5 accomplish in the Hough space. Finally, lines 6 through 16 apply an oriented edge operator to the region of the image delineated by the rectangular enclosure produced by line 3; the output of the oriented edge operator is then used to fill up the accumulator cells in the fragment of the Hough space defined by the enclosure generated in line 5. Figure 32 shows the histogram obtained for one of the

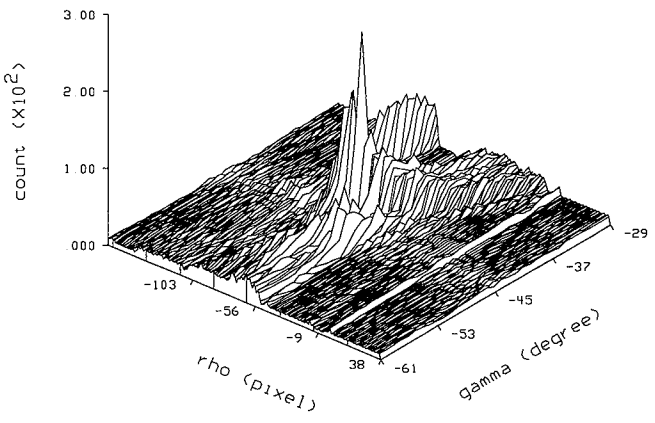


FIG. 32. A three-dimensional histogram in the (ρ, γ) space for the extraction of a nonvertical hallway line.

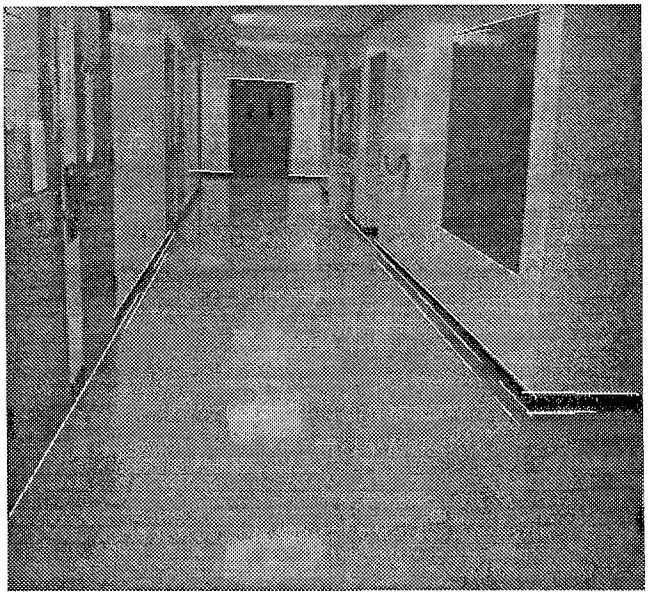


FIG. 33. The superimposed white lines shows the nonvertical lines extracted from the image of Fig. 7.

nonvertical lines in Fig. 7. The peaks in the Hough space, as extracted in line 17, then define the lines in the image along which the relevant pixels must lie. After the construction of the $\rho\gamma$_hist, the thresholding operations are very nearly the same as for the case of vertical lines, except that these operations now take place in two dimensions as opposed to one. After the extraction of ρ and γ corresponding to visually significant nonvertical lines, actual identification of the associated pixels in the image space takes place in a manner very nearly the same as before, the only point of departure being in the construction of 1D neighborhoods shown in Fig. 29 for the case of vertical hallway lines. When the analytical lines are at angles that are closer to Y-axis than to X-axis, the 1D neighborhoods become vertical for obvious reasons.

The superimposed white lines in Fig. 33 show the nonvertical lines extracted in this manner from the image of Fig. 7.

7. CORRESPONDENCE FINDING AND PARAMETER ESTIMATION

After the extraction of image lines from the uncertainty regions associated with each of the visually significant hallway lines, our next challenge is to establish a correspondence between the hallway lines and the image lines. Ideally, we would like a one–one mapping between the image lines and the lines in the expectation map. Any attempts at doing so are complicated by the following factors:

1. there may be more than one line extracted from any given uncertainty region,

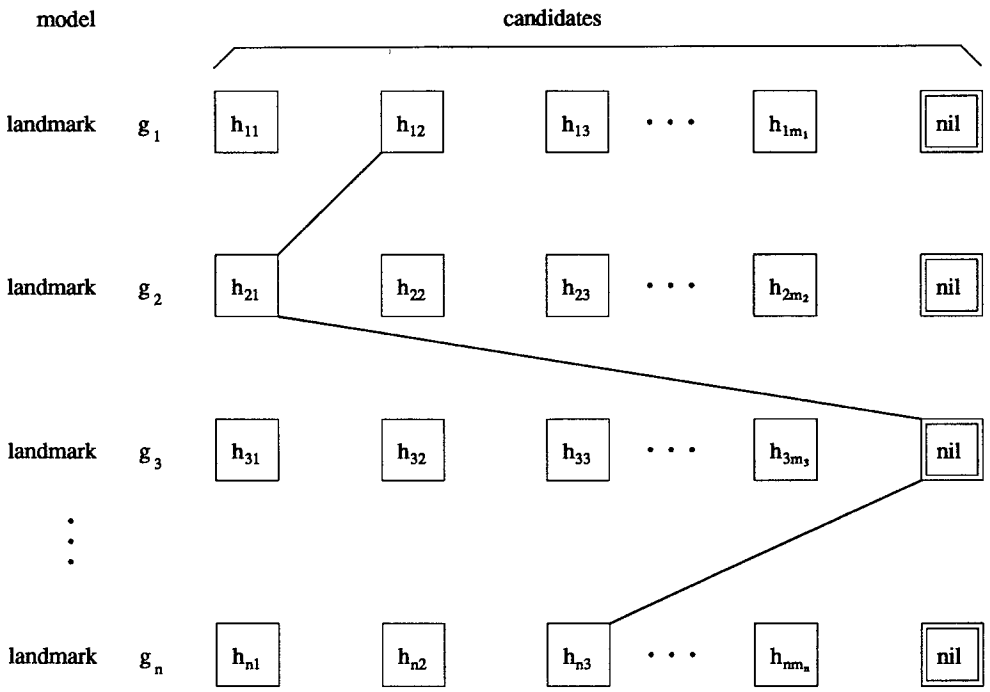


FIG. 34. Each landmark g_i is a visually significant feature from the hallway. In our case, g_i's are the hallway lines, both vertical and nonvertical. For each landmark, there is a pool of candidate features extracted from the image. The problem is to construct a mapping function by pairing up each landmark with a member from its pool. Added to each pool is a nil to account for occlusions and other effects.

2. the misregistration between the actual position of the robot and the position as inferred from the wheel-encoder reports would cause the occlusion effects in the camera image to be different from those in the expectation map,

3. some of the hallway lines may become occluded by the obstacles,

4. the obstacles may give rise to additional lines in the image.

The problem at hand is, we believe, well illustrated by Fig. 34, where for each visually significant line from the hallway in a given expectation map we have shown potential candidates extracted from the corresponding uncertainty regions. For example, for the landmark feature g_n, its potential image candidates are marked h_{n1} through h_{nm_n}. Note that we have also added the *nil* symbol to the list of potential candidates for each landmark line. This addition label helps us assign a nil to a hallway line when the latter may not be visible due to either the second or the third of the difficulties listed above. The first and the fourth difficulties are taken care of automatically by the maximization of an objective function that we will define in this section.[18]

[18] The process of adding *nils* to the list of potential candidates during the construction of a mapping function is referred to as *nilmapping*. For some earlier cases where nilmapping was found indispensable, the reader is referred to [BoyKak88, Gri90].

The goal of this section is to show how we might construct a mapping function that would take us from the expectation map to the linear features extracted from the image. This mapping function should be *one–one* and *into* and must allow for nilmapping. In other words, the mapping from the model space to the image space must be injective while allowing for nil possibilities.

In order to derive such a mapping function, two criteria must be specified:

1. We need a criterion that tells us how well an image line "matches" a landmark line. More precisely, we wish to know how well an image line can be considered to be a projection of a hallway line, given our knowledge of the camera calibration matrix. This criterion is evidently local in nature.

2. Given a knowledge of how well the various lines in the hallway match, on an individual basis, the various lines extracted from the image, we need an evaluation function that would be a measure of the quality of the mapping function.

In the next subsection, we will present what we believe is a unique constraint equation whose satisfaction immediately tells us whether or not an image line is a projection of a hallway line. The extent to which this equation is satisfied will then serve as a measure of local "match" between a given hallway line and a given image line. In a

subsequent subsection, we will then present an evaluation function that will give us the probability that a given mapping function constructed in the search space of Fig. 34 is a valid mapping function. Then, by choosing that mapping function which maximizes this probability, we obtain the best possible assignment of image lines to the hallway lines—best in the sense that what we obtain is a maximum-likelihood mapping function.

As some readers might recall, structural matching in computer vision [ShaHar81], for example structural matching in binocular stereopsis [BoyKak88], also uses two criteria: a local criterion, usually referred to as local compatibility, and a relational criterion. While our first criterion seeks to achieve the same result as the local compatibility in structural stereopsis, the reader will notice that our second criterion will not bear any resemblance to a relational match criterion. In the traditional approach to enforcing relational constraints, the spatial relations between the primitives must be made explicit and must be measured directly in the image. In our formulation, the mechanism of seeking the image correspondent of a hallway feature from only a particular uncertainty region in the image is an indirect, albeit powerful, enforcement of the relational constraints. So, while our second criterion will make no explicit comparisons of the relational type, it nevertheless has the capacity to guarantee that the spatial relations between the lines in the expectation map and the image lines accepted for the mapping function are the same—not in a deterministic sense but in a maximum-likelihood sense—as the spatial relations between the lines in the expectation map.

7.1. A Constraint Equation

Any straight line in the world coordinate frame passing through a designated point (x_1, y_1, z_1) and of orientation (a, b, c), a, b, and c being the direction cosines associated with the direction of the hallway line, can be represented by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} u + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \tag{7.1}$$

where u is a scalar parameter. Similarly, any image line that is described by the Hough parameters (see Fig. 11) ρ and γ can be represented by

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} -\sin\gamma \\ \cos\gamma \end{bmatrix} s + \begin{bmatrix} \rho\cos\gamma \\ \rho\sin\gamma \end{bmatrix}, \tag{7.2}$$

where, in accordance with our discussion in Section 6, X and Y are the two coordinate axes in the image frame.

Now, the image line corresponding to the line of Eq. (7.1) will be given by Eq. (4.23). If we substitute Eqs. (7.1) and (7.2) in Eq. (4.23), we get

$$\begin{bmatrix} -\sin\gamma & \rho\cos\gamma \\ \cos\gamma & \rho\sin\gamma \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ W \end{bmatrix} = T\,H \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} u + T\,H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}. \tag{7.3}$$

Recall that T is the 3×4 calibration matrix of the camera in the robot-centered coordinate frame and H a 4×4 transformation matrix that takes a point from the robot-centered coordinate frame to the world coordinate frame; H is given by Eq. (4.22). Akin to what is done in Gaussian elimination for the diagonalization of matrices, in order to force one of the equations in the above matrix–vector form to equal zero on the right side, we will now multiply both sides of Eq. (7.3) by

$$\begin{bmatrix} 1 & 0 & -\rho\cos\gamma \\ \cos\gamma & \sin\gamma & -\rho \\ 0 & 0 & 1 \end{bmatrix}$$

the result becoming

$$\begin{bmatrix} -\sin\gamma & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\rho\cos\gamma \\ \cos\gamma & \sin\gamma & -\rho \\ 0 & 0 & 1 \end{bmatrix} T\,H \left(\begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} u + \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \right). \tag{7.5}$$

From the equality of the second row elements from both sides, we get

$$[\cos\gamma\ \sin\gamma\ -\rho]\ T\,H \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} u + [\cos\gamma\ \sin\gamma\ -\rho]\ T\,H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = 0. \tag{7.6}$$

Since different values of the parameter u correspond to the different points on the hallway line, and since this equation must hold for an arbitrary u, the two coefficients above must be zero separately. This fact will be expressed in the following manner:

$$\mathbf{f}(a, b, c, x_1, y_1, z_1, \rho, \gamma, p_x, p_y, \phi) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \mathbf{0}, \quad (7.7)$$

where

$$f_1 \equiv [\cos\gamma \ \sin\gamma \ -\rho]\, T\, H \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = 0 \quad (7.8)$$

$$f_2 \equiv [\cos\gamma \ \sin\gamma \ -\rho]\, T\, H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = 0. \quad (7.9)$$

The dependence of f_1 and f_2, and therefore of $\mathbf{f}$, on the robot position parameters, p_x, p_y, ϕ, is induced by the transformation matrix H. The constraint equation Eq. (7.7) will be represented more compactly as

$$\mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{p}) = \mathbf{0}, \quad (7.10)$$

where the arguments of $\mathbf{f}$ are grouped as

$$\mathbf{x} = [a, b, c, x_1, y_1, z_1]^T, \quad \mathbf{z} = [\rho, \gamma]^T, \quad \mathbf{p} = [p_x, p_y, \phi]^T. \quad (7.11)$$

The $\mathbf{x}$ parameters are about the hallway line in question, the $\mathbf{z}$ about the camera image of the hallway line, and the $\mathbf{p}$ about the robot position.

7.2. A Maximum Likelihood Estimate of the Mapping Function

Let $\mathbf{p}$ be a random vector which specifies the robot position in the world coordinate frame. Given a hallway line g_i and an image line h_j characterized by the parameter vector $\mathbf{z}_j$, we use the following as the probability that g_i corresponds to h_j:

$$\text{prob}[g_i \rightarrow h_j \mid \mathbf{z}_j, \mathbf{p}], \quad (7.12)$$

where the symbol $g_i \rightarrow h_j$ denotes the mapping of the model line g_i to the image line h_j.

Assume for a moment we have only two landmark features in the expectation map, denoted by g_1 and g_2. Let $h_j, j = 1, 2, 3, \ldots$ be the lines extracted from the image. To construct the best possible mapping function in the maximum-likelihood sense, we must choose the image lines h_i and h_j such that the probability

$$\text{prob}[g_1 \rightarrow h_i \text{ and } g_2 \rightarrow h_j \mid \mathbf{z}_i, \mathbf{z}_j, \mathbf{p}]$$

is maximized. However, this joint probability may be decomposed in the following manner:

$$\text{prob}[g_1 \rightarrow h_i \text{ and } g_2 \rightarrow h_j \mid \mathbf{z}_i, \mathbf{z}_j, \mathbf{p}] = \text{prob}[g_1 \rightarrow h_i \mid \mathbf{z}_i, \mathbf{p}] \times \text{prob}[g_2 \rightarrow h_j \mid g_1 \rightarrow h_i, \mathbf{z}_i, \mathbf{z}_j, \mathbf{p}]. \quad (7.13)$$

Let us now focus on the conditioning event

$$g_1 \rightarrow h_i, \mathbf{z}_i, \mathbf{z}_j, \mathbf{p}$$

in the second factor on the right in Eq. (7.13). A primary consequence of any landmark to image feature match, such as $g_1 \rightarrow h_i$, is that the uncertainty in the robot position random variable $\mathbf{p}$ will be reduced and consequently the uncertainties in any $\mathbf{z}_j$ vectors considered subsequently will also be reduced. In other words, every match like $g_i \rightarrow h_m$ transforms the random variable $\mathbf{p}$, of statistics $(\bar{\mathbf{p}}, \Sigma_p)$ into a different random variable $\mathbf{p}'$ of statistics $(\bar{\mathbf{p}}^{(1)}, \Sigma_p^{(1)})$, where the superscript (1) signifies the revised statistics of the robot position random variable. Therefore, the conditioning event $g_1 \rightarrow h_i$ in the first factor on the right side in Eq. (7.13) can be interpreted as updating of the random variable $\mathbf{p}$. At the risk of sounding too repetitious, what we are trying to say is that if the position of the robot was $\mathbf{p}$ prior to the match $g_1 \rightarrow h_i$ becoming available, and if the new position of the robot after the match $g_1 \rightarrow h_i$ is designated $\mathbf{p}^{(1)}$, then the decomposition in Eq. (7.13) may be expressed as

$$\text{prob}[g_1 \rightarrow h_i \text{ and } g_2 \rightarrow h_j \mid \mathbf{z}_i, \mathbf{z}_j, \mathbf{p}] = \text{prob}[g_1 \rightarrow h_i \mid \mathbf{z}_i, \mathbf{p}] \times \text{prob}[g_2 \rightarrow h_j \mid \mathbf{z}_j^{(1)}, \mathbf{p}^{(1)}] \quad (7.14)$$

where $\mathbf{z}_j^{(1)}$ is the random variable $\mathbf{z}_j$ transformed by the match $g_1 \rightarrow h_i$.[19]

[19] At this point some readers might ask: Do the conditioning events $\mathbf{z}$ and $\mathbf{p}$ really contain any independent information? After all, in Section 4, we said that the uncertainty in $\mathbf{z}$ was to be derived from the uncertainty from $\mathbf{p}$. The problem that we will soon be facing in this section is that in actual practice the Hough space parameter vector $\mathbf{z}$ contains uncertainties above and beyond those derived in Section 4. The additional uncertainties are due to the sampling and quantization effects and have a direct bearing on the measurement of $\mathbf{z}$. These additional uncertainties will be important to the Kalman filter based updating scheme to be derived in this section for the position vector $\mathbf{p}$. Note that it is in the nature of the perspective geometry of camera imaging that even a small error in $\mathbf{z}$ can cause significant problems with the establishment of correspondences between the image lines and the hallway landmarks.

Let us now generalize the above development to the case of arbitrary number of hallway lines given by $g_1, g_2, g_3, \ldots, g_n$ and an arbitrary number of image lines given by $h_1, h_2, h_3, \ldots, h_m$. Generalizing the arguments that led to Eq. (7.14), we can write for the probability of any given mapping function

$$\begin{aligned}&\text{prob}[g_1 \rightarrow h_{j_1}, g_2 \rightarrow h_{j_2}, \ldots, \\ &\quad g_n \rightarrow h_{j_n} \mid \mathbf{z}_{j_1}, \mathbf{z}_{j_2}, \ldots, \mathbf{z}_{j_n}, \mathbf{p}] \\ &\quad = \text{prob}[g_1 \rightarrow h_{j_1} \mid \mathbf{z}_{j_1}, \mathbf{p}] \\ &\qquad \text{prob}[g_2 \rightarrow h_{j_2} \mid \mathbf{z}_{j_2}^{(1)}, \mathbf{p}^{(1)}] \\ &\qquad \cdots \\ &\qquad \text{prob}[g_n \rightarrow h_{j_n} \mid \mathbf{z}_{j_n}^{(n-1)}, \mathbf{p}^{(n-1)}].\end{aligned} \tag{7.15}$$

For computational reasons, it is preferable to take the logarithm of both sides of this decomposition, the result expressed as

$$\begin{aligned}D_0 = &\log \text{prob}[g_1 \rightarrow h_{j_1} \mid \mathbf{z}_{j_1}, \mathbf{p}] \\ &+ \log \text{prob}[g_2 \rightarrow h_{j_2} \mid \mathbf{z}_{j_2}^{(1)}, \mathbf{p}^{(1)}] \\ &\cdots \\ &+ \log \text{prob}[g_n \rightarrow h_{j_n} \mid \mathbf{z}_{j_n}^{(n-1)}, \mathbf{p}^{(n-1)}],\end{aligned} \tag{7.16}$$

where D_0 may be thought of as an objective function that measures the likelihood of a mapping assignment from the hallway landmarks to the image lines. Our goal evidently is to select that mapping assignment which maximizes D_0.

The decomposition shown above lends itself naturally to a sequential calculation. The realization that the computations can be carried out in a sequential framework requires a fundamental shift in how the assumptions outlined at the beginning of this subsection are to be interpreted. For example, at the beginning we assumed that we have available to us model lines $g_1, g_2, \ldots, g_n$ and the image lines $h_1, h_2, \ldots, h_m$. Our statement there implied a certain simultaneity with regard to the availability of the image lines. The sequential model of computation developed here makes that unnecessary. At the beginning we need to know only one image line, say the line h_{j_1}; its match with the model line g_1 will generate a revised robot position vector $\mathbf{p}^{(1)}$ from the originally known position vector $\mathbf{p}$. This new position vector, with its associated mean and covariance, will then lead us, nominally through the propagation equation in Eqs. (4.37) (4.38), and (4.52) to a new vector $\mathbf{z}_{j_2}$. In other words, the statistics of $\mathbf{z}_{j_2}$ reflect nominally the projection of the hallway line g_2 into the camera frame given the uncertainty of the revised position vector $\mathbf{p}^{(1)}$.[20]

Therefore, in a strictly sequential computational framework in which each match between a hallway line and an image line is used to update the statistics for the next match, we can rewrite Eq. (7.16) as

$$\begin{aligned}D_0 = &\log \text{prob}[g_1 \rightarrow h_{j_1} \mid \mathbf{z}_{j_1}, \mathbf{p}] \\ &+ \log \text{prob}[g_2 \rightarrow h_{j_2} \mid \mathbf{z}_{j_2}, \mathbf{p}^{(1)}] \\ &\cdots \\ &+ \log \text{prob}[g_n \rightarrow h_{j_n} \mid \mathbf{z}_{j_n}, \mathbf{p}^{(n-1)}].\end{aligned} \tag{7.17}$$

The main difference between Eqs. (7.16) and (7.17) is that we have dropped the superscripts on the $\mathbf{z}$ variables because we now recognize that $\mathbf{z}_{j_2}$ comes into existence only after the first match, $g_1 \rightarrow h_{j_1}$ has taken place. The statistics of $\mathbf{z}_{j_2}$, derived by projecting the image line g_2 into the camera frame through the uncertainty in the position of the robot, will then delineate the image and Hough space regions where we will look for the image line h_{j_2}. This interpretation of the link between the model lines and the uncertainty regions in which we must look for the image lines that correspond to the model lines is made more transparent if we rewrite Eq. (7.17) as

$$\begin{aligned}D_0 = &\log \text{prob}[g_1 \rightarrow h_{j_1} \mid \mathbf{z}_1, \mathbf{p}] \\ &+ \log \text{prob}[g_2 \rightarrow h_{j_2} \mid \mathbf{z}_2, \mathbf{p}^{(1)}] \\ &\cdots \\ &+ \log \text{prob}[g_n \rightarrow h_{j_n} \mid \mathbf{z}_n, \mathbf{p}^{(n-1)}].\end{aligned} \tag{7.18}$$

Examining the first factor on the right hand side, this equation tells us that the image line h_{j_1} as a possible match for the model line g_1 must be found in the uncertainty region $\mathbf{z}_1$, the uncertainty region being obtained by projecting the model line g_1 into the camera frame through the uncertainty associated with the position vector $\mathbf{p}$. Similarly, for the next factor, the image counterpart of the model line g_2 is to found from the uncertainty region defined by $\mathbf{z}_2$, this uncertainty region being obtained by projecting g_2 into the camera frame though the uncertainty associated with the vector $\mathbf{p}^{(1)}$, where, as will be discussed in the next section, $\mathbf{p}^{(1)}$ incorporates Kalman filter revisions in the statistics of $\mathbf{p}$.

The sequential approach still leaves open the question that if more than one image line is present in the image region corresponding to the model line g_i, which one should be selected for h_{j_i}. This question will be addressed later when we address the issue of backtracking for the maximization of the objective function D_0.

7.3. A Kalman Filter-Based Update Scheme for the Position Vector

Now we must address the question of how the computation called for by each term on the right in Eq. (7.18) must be carried out. In other words, we now address how a match between a given hallway line g_i and a given image line h_{j_i} changes the statistics of the robot position vector and generates, taking into account the hallway line g_{i+1},

[20] We have used the word *nominally* because we have not yet introduced the effects of observation noise in the parameter vectors $\mathbf{z}$.

the statistics of $\mathbf{z}_{i+1}$, which will then tell us where we should look for the image line corresponding to g_{i+1}. This we will do with the help of the Kalman filter ideas that have recently been injected into mobile robotics by Ayache and Faugeras [AyaFau89]. It is entirely possible that a given image line—the image line currently being considered a correspondent of the hallway line g_{i+1}—may not be the correct line. This will happen particularly when more than one line is extracted from the uncertainty region in the Hough space corresponding to the hallway line g_{i+1}. In order to deal with such mistakes in the selection of the image lines, and also to bring into consideration the consequences of image digitization and other sources of noise, we will assume that if the vector $\mathbf{z}_i^*$ denotes the Hough parameters of the *correct* correspondent of g_i, and $\hat{\mathbf{z}}_i$ the measured parameters of the selected line, then[21]

$$\hat{\mathbf{z}}_i = \mathbf{z}_i^* + \xi_i, \tag{7.19}$$

where ξ_i includes two types of errors; the selection error, which is the error caused by having selected the wrong line from the uncertainty region in the Hough space, and the errors introduced by phenomena such as sampling and quantization of the image. As was mentioned in the footnote in the preceding subsection, the perspective geometry of camera imaging can cause the latter error to seriously affect the calculation of the updated position vector; hence it must be taken into account.

For the random error vector ξ_i, we may assume that its mean is zero and its covariance matrix is known from experimentation. [If for some reason the mean value is not zero, a bias term can be subtracted from the ξ's.] Then

$$E[\xi_i] = \mathbf{0} \tag{7.20}$$

$$E[\xi_i \, \xi_i^{\mathrm{T}}] = V_i \tag{7.21}$$

$$E[\xi_i \, \xi_j]^{\mathrm{T}} = 0 \quad \text{for } i \neq j, \tag{7.22}$$

where we have also assumed that for different i and j, the observation noise terms are uncorrelated.

In keeping with our earlier notation, $\mathbf{p}$ denotes the initial position vector for the robot, initial in the sense that it designates the random variable before any matching via the Kalman filter. The statistics of $\mathbf{p}$ are a function of the position encoder reports from the robot, as explained in Section 4. Also, after the completion of the sequential computation dictated by Eq. (7.18), let $\mathbf{p}^*$ denote the final position vector of the robot. The statistics of $\mathbf{p}^*$ give us the best localization of the robot after all the matches between the hallway lines that can be seen in the expectation map and the lines extracted from the image. Evidently, prior to the completion of the computations, after each match between a hallway line and an image line, the estimate of the robot position vector will differ from both $\mathbf{p}$ and $\mathbf{p}^*$. In the notation used in Eq. (7.18), $\mathbf{p}^{(i)}$ is the position vector after the match for the ith hallway line g_i. We will now assume that

$$\mathbf{p}^{(i)} \equiv (\bar{\mathbf{p}}^{(i)}, \Sigma_p^{(i)}), \tag{7.23}$$

where $\bar{\mathbf{p}}^{(i)}$ is the mean value of the position random variable $\mathbf{p}^{(i)}$ and where $\Sigma_p^{(i)}$ represents the revised estimate of the variance of the position vector after the match for g_i.

As eloquently stated by Ayache and Faugeras [AyaFau89], the derivation of a Kalman filter requires that any constraints on the parameter to be estimated, if they are nonlinear, be linearized in the neighborhood of their operating ranges. The constraint equation applicable in our case, Eq. (7.10), is nonlinear. To discuss the linearization of this equation, let us assume that we have already finished the computations implied by the first $(i-1)$ terms on the right of Eq. (7.18). This computation will yield the position vector $\mathbf{p}^{(i-1)}$ and, taking into account the hallway line g_i, the Hough parameter vector $\mathbf{z}_i$. Now let $\hat{\mathbf{z}}_i$ denote the actual Hough space parameters of the image line *selected* from the image and the Hough space regions as delineated by the statistics of $\mathbf{z}_i$. Since we have no way of telling whether or not the image line selected is the correct one, all we can do at this time is examine the constraint equation in the vicinity of the measured point $\hat{\mathbf{z}}_i$ and, $\mathbf{p}^{(i-1)}$, the position vector already estimated. Linearizing the constraint equation in the vicinity of these values, we get

[21] We have agonized much about subjecting the reader to the confusion that may be caused by our use of $\mathbf{z}_i$, $\hat{\mathbf{z}}_j$, and $\mathbf{z}_i^*$, but there does not seem any easy way to avoid this multiplicity of symbols, all dealing essentially with the same thing, the parameters of the image lines. All formalisms that deal with predictions and measurements in a sequential framework appear to suffer from this sort of difficulty. To help the reader, we will summarize here the interpretations that these symbols carry. The symbol $\mathbf{z}_i$ is the random vector obtained by projecting the model line g_i into the camera frame through the latest uncertainty in the robot position. The uncertainty region defined by $\mathbf{z}_i$ tells us where to look for the image line which would correspond to g_i. The symbol $\hat{\mathbf{z}}_j$ is also a random vector and represents the measured Hough parameters of a particular line selected from the image. It is most important to note that the source of randomness in $\hat{\mathbf{z}}_j$ is entirely different from the source of randomness in $\mathbf{z}_i$; whereas the randomness in the former is entirely due to the measurement phenomena, such as image quantization noise and selection ambiguity, the randomness in the latter is due to the uncertainties in prediction caused by the uncertainties in the position of the robot. For a given model line g_i, from the uncertainty region defined by $\mathbf{z}_i$ we select for matching a particular image line h_j, the parameters of the image line being $\hat{\mathbf{z}}_j$. The symbol $\mathbf{z}_i^*$ is not a random vector, but a deterministic vector that, although not known, designates the Hough parameters of the true image line corresponding to the model line g_i.

$$\mathbf{f}(\mathbf{x}_i, \mathbf{z}_i^*, \mathbf{p}) = \mathbf{0}$$

$$\approx \mathbf{f}(\mathbf{x}_i, \hat{\mathbf{z}}_i, \bar{\mathbf{p}}^{(i-1)}) + \frac{\partial \mathbf{f}}{\partial \mathbf{z}_i}(\mathbf{z}_i^* - \hat{\mathbf{z}}_i) + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}(\mathbf{p} - \bar{\mathbf{p}}^{(i-1)}), \quad (7.24)$$

where the partial derivatives are evaluated at the points $(\mathbf{x}_i, \hat{\mathbf{z}}_i, \bar{\mathbf{p}}^{(i-1)})$. The partial derivatives used in Eq. (7.24) are shown in the Appendix. This equation may now be rewritten in the following fashion that is standard to Kalman filter development:

$$\mathbf{y}_i = M_i\,\mathbf{p} + \mathbf{u}_i \quad (7.25)$$

where $\mathbf{y}_i$, as a function of the actual measurement $\hat{\mathbf{z}}_i$ and the position estimate already known, $\bar{\mathbf{p}}^{(i-1)}$, will be called the ith Kalman measurement vector. The matrix $\mathbf{M}$ tells us how this measurement vector is related to $\mathbf{p}$, the subject of our sequential estimation, and $\mathbf{u}_i$ gives us a linearized version of the random error associated with the Kalman measurement vector. These quantities are given by

$$\mathbf{y}_i = -\mathbf{f}(\mathbf{x}_i, \hat{\mathbf{z}}_i, \bar{\mathbf{p}}^{(i-1)}) + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}\bar{\mathbf{p}}^{(i-1)} \quad (7.26)$$

$$M_i = \frac{\partial \mathbf{f}}{\partial \mathbf{p}} \quad (7.27)$$

$$\mathbf{u}_i = \frac{\partial f}{\partial \mathbf{z}_i}(\mathbf{z}_i^* - \hat{\mathbf{z}}_i). \quad (7.28)$$

The statistics of the vector $\mathbf{u}_i$ are easily obtained from Eqs. (7.19) through (7.22) and are given by

$$E[\mathbf{u}_i] = \mathbf{0} \quad (7.29)$$

$$U_i = E[\mathbf{u}_i\,\mathbf{u}_i^{\mathrm{T}}] = \frac{\partial \mathbf{f}}{\partial \mathbf{z}_i} V_i \frac{\partial \mathbf{f}^{\mathrm{T}}}{\partial \mathbf{z}_i}, \quad (7.30)$$

where the partial derivatives are calculated at the points $(\mathbf{x}_i, \hat{\mathbf{z}}_i, \bar{\mathbf{p}}^{(i-1)})$. The covariance matrix V_i was defined in Eq. (7.22).

Kalman filter theory tells us that the robot position vector is updated by the equations

$$\bar{\mathbf{p}}^{(i)} = \bar{\mathbf{p}}^{(i-1)} + K_i(\mathbf{y}_i - M_i\,\bar{\mathbf{p}}^{(i-1)}) \quad (7.31)$$

$$K_i = \Sigma_p^{(i-1)} M_i^{\mathrm{T}} (U_i + M_i\,\Sigma_p^{(i-1)} M_i^{\mathrm{T}})^{-1} \quad (7.32)$$

$$\Sigma_p^{(i)} = (I - K_i\,M_i)\,\Sigma_p^{(i-1)}, \quad (7.33)$$

where K_i is called the Kalman gain of the filter. The derivation of the Kalman filter is shown, for example, in [Jaz70]. The sequential operation of the Kalman filter is depicted pictorially in Fig. 35.

7.4. Assessing Match Probabilities

As mentioned before, for the implementation of the Kalman filter we must first project the robot position uncertainty into the image plane and the Hough space, and, from the uncertainty regions thus constructed, we must select one image line for each model line. While the Kalman filter gives us revisions of the robot position vector after a match between a model line and a selected image line, the filter theory itself has nothing to say about how an image line should be selected from the uncertainty regions. In other words, for each mapping in the search space displayed in Fig. 34, the Kalman filter gives us revised estimates of the robot position uncertainty after every match between a landmark and a candidate

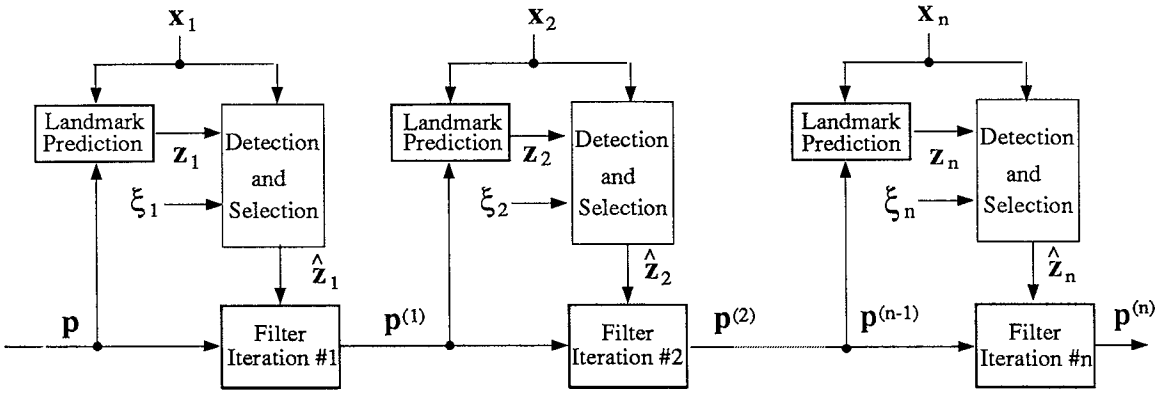


FIG. 35. This figure depicts the sequential operation of the Kalman filter. The goal is to sequentially update the statistics of the robot position vector $\mathbf{p}$ after each match between a landmark and an image feature. After n such matches, the final estimate of the robot position vector is denoted $\mathbf{p}^{(n)}$. The image feature corresponding to the landmark $\mathbf{x}_i$ is predicted to be within the uncertainty region corresponding to the random variable $\mathbf{z}_i$ given the robot position vector $\mathbf{p}^{(i-1)}$. The random variable ξ_i accounts for the detection and selection errors associated with the image feature chosen to correspond to the landmark $\mathbf{x}_i$.

image feature, but how to select a candidate in each row of Fig. 34 is beyond the capabilities of the Kalman filter. Therefore, as was mentioned before, the Kalman filter must be combined with backtracking. To bring about this backtracking, we must evaluate the quality of each overall mapping by computing the objective function in Eq. (7.18), which in turn requires that we be able to estimate the probability that $g_i \rightarrow h_j$ given the current estimate $\mathbf{p}^{(i-1)}$. We will now derive a result for the generic form prob$[g_i \rightarrow h_j \mid \mathbf{z}_i, \mathbf{p}]$ and then use it with the appropriate substitution of variables in Eq. (7.18). Recall that $\hat{\mathbf{z}}_j$ is the measured Hough space vector for the image line h_j that is being considered for a possible match with the model line g_i.

In order to derive a formula for the generic form prob$[g_i \rightarrow h_j \mid \mathbf{z}_i, \mathbf{p}]$, recall that $\mathbf{z}_i$, the prediction vector associated with the model line g_i, is obtained by projecting g_i into the camera frame via the uncertainties in position vector $\mathbf{p}$; the statistics associated with $\mathbf{z}_i$ tell us where in the image and the Hough space we should look for a match for g_i. We then select a line h_j from the region defined by $\mathbf{z}_i$, the Hough parameters of this line designated by $\hat{\mathbf{z}}_j$. Now the question to answer is what is the probability that $\hat{\mathbf{z}}_j$ represents the parameters of the correct match. We will write

$$\text{prob}[g_i \rightarrow h_j \mid \mathbf{z}_i, \mathbf{p}] = \text{prob}[\text{detectable}(g_i) \mid \mathbf{z}_i, \mathbf{p}] \times \text{prob}[\mathbf{z}_{\text{measured}} = \hat{\mathbf{z}}_j \mid \mathbf{z}_i, \mathbf{p}], \tag{7.34}$$

where we have made explicit the fact that, due to various reasons listed below, a given model line g_i may or may not be detectable in the image.

(i) The image correspondent of a model line may fall outside the uncertainty region. For example, if for the purpose of feature extraction we define the uncertainty region on the basis of two units of Mahalanobis distance, there is roughly a 14% probability that the image line would fall outside the uncertainty region and would thus be undetectable by our processing algorithms. Let $P_{\text{outside\_ur}}$ denote the probability that the image projection of a model line will fall outside the uncertainty region.

(ii) Occlusion resulting from the uncertainties in the position of the robot. When an expectation map, such as the one shown in Fig. 24, is made, it is from the vantage point corresponding to the mean position vector at the end of the commanded motion. Since the actual position of the robot will, in general, be different from the mean position, some of the features visible in the expectation map will not be visible in the camera image. Let $P_{\text{occlusion}}$ denote the probability that due to the robot not being at the mean location of its uncertainty region, a model line, visible in the expectation map, is not visible in the camera image.

(iii) Some of the landmarks may become occluded by the obstacles. Also, due to illumination effects, some of the landmarks may simply not have sufficient contrast in order to be detectable. Let P_{obs} designate the probability that a model line will be sufficiently occluded by an obstacle and therefore will not be detectable.

We will now assume that these three phenomena are independent. We will therefore use the following expression for prob[detectable$(g_i) \mid \mathbf{z}_i, \mathbf{p}$]:

$$\text{prob}[\text{detectable}(g_i) \mid \mathbf{z}_i, \mathbf{p}] = (1 - P_{\text{outside\_ur}})\,(1 - P_{\text{occlusion}})(1 - P_{\text{obs}}), \tag{7.35}$$

where the product rule is dictated by the fact that in order for a model line to be detectable, it must be detectable in a conjunctive sense with regard to all three phenomena. For obvious reasons, the probabilities $P_{\text{occlusion}}$ and P_{obs} would be hard to derive analytically. Fortunately, these two probabilities do not lend any differential weight to the overall objective function. What that means is that all the paths in the search space of Fig. 34 will be equally affected by a given specification of $P_{\text{occlusion}}$ and P_{obs}. This is particularly true of P_{obs} when all obstacles are executing truly random motions around the robot. Therefore, we will simply ignore $P_{\text{occlusion}}$ and P_{obs} from the rest of the discussion, which leads to the following expression for the probability of detection:

$$\text{prob}[\text{detectable}(g_i) \mid \mathbf{z}_i, \mathbf{p}] = (1 - P_{\text{outside\_ur}}). \tag{7.36}$$

Before discussing how the probability $P_{\text{outside\_ur}}$ is specified, we will now address the issue of estimating prob$[\mathbf{z}_{\text{measured}} = \hat{\mathbf{z}}_j \mid \mathbf{z}_i, \mathbf{p}]$, the second factor in Eq. (7.34). The question here is that given the prediction random vector $\mathbf{z}_i$, what is the probability that the measured Hough parameters of the selected image line would be $\hat{\mathbf{z}}_j$? Invoking from Section 4 the assumption that $\mathbf{z}_i$ is a Gaussian random vector of mean $\bar{\mathbf{z}}_i$ and covariance $\Sigma_{\rho\gamma,i}$, we can write

$$\text{prob}[\mathbf{z}_{\text{measured}} = \hat{\mathbf{z}}_j \mid \mathbf{p}] = \Delta\mathbf{z}\left(\frac{1}{2\pi\,|\Sigma_{\rho\gamma,i}|}\exp\left[-\frac{1}{2}(\hat{\mathbf{z}}_j - \bar{\mathbf{z}}_i)^{\mathrm{T}}\,\Sigma_{\rho\gamma,i}^{-1}\,(\hat{\mathbf{z}}_j - \bar{\mathbf{z}}_i)\right]\right), \tag{7.37}$$

where $\Delta\mathbf{z} = \Delta\rho\,\Delta\gamma$ is the cell size in the Hough space. We will assume the Hough space is sampled uniformly and that therefore the cell size is constant. It is important to note that the exponent on the right hand side of Eq. (7.37) equals the square of the Mahalanobis distance from the mean prediction vector $\bar{\mathbf{z}}_i$ to the measurement vector $\hat{\mathbf{z}}_j$. If the measurement $\hat{\mathbf{z}}_j$ equals the mean prediction vector, the probability of match becomes maximum. As the Ma-

halanobis distance between the actual measurement vector and the mean prediction vector gets larger, the probability of the image vector being a good match becomes smaller. For a given measurement vector $\hat{\mathbf{z}}_j$, the calculation of the probability in Eq. (7.37) is straightforward since, as we explained in Section 4, we already know how to transform the uncertainty in robot position into the uncertainty in the Hough space. In other words, for a given model line g_i and for a given position random vector $\mathbf{p}$, we already have the formulas that give us $\bar{\mathbf{z}}_i$ and $\Sigma_{\rho\gamma,i}$.

The distribution shown in Eq. (7.37) also helps us calculate the probability $P_{\text{outside\_ur}}$ since this probability is given by

$$P_{\text{outside\_ur}} = \int_{\mathbf{z} \notin \text{Hough\_uncert\_region}} \frac{1}{2\pi \, |\Sigma_{\rho\gamma,i}|} \exp\left[-\frac{1}{2} (\mathbf{z} - \bar{\mathbf{z}}_i)^{\mathrm{T}} \, \Sigma_{\rho\gamma,i}^{-1} \, (\mathbf{z} - \bar{\mathbf{z}}_i) \right] d\mathbf{z}, \tag{7.38}$$

where, as indicated, the integral is computed outside the Mahalanobis distance that defines the uncertainty region for $\mathbf{z}_i$ in the Hough space. In other words, the integral is computed wherever

$$(\mathbf{z} - \bar{\mathbf{z}}_i)^{\mathrm{T}} \, \Sigma_{\rho\gamma,i}^{-1} \, (\mathbf{z} - \bar{\mathbf{z}}_i) \geq d, \tag{7.39}$$

where the threshold d means that the uncertainty region is defined by d units of Mahalanobis distance. This then takes care of specifying $P_{\text{outside\_ur}}$ in Eq. (7.36).

Since it is quite likely that some of the model lines would be mapped to nil, we also need to assess the probability prob$[g_i \rightarrow \text{nil} \mid \mathbf{z}_i, \mathbf{p}]$. Since the factors that cause a model line to be nondetectable are exactly the same as the factors that would cause a model line to be matched to nil, we could use

$$\text{prob}[g_i \rightarrow \text{nil} \mid \mathbf{z}_i, \mathbf{p}] = 1 - \text{prob}[\text{detectable}(g_i) \mid \mathbf{z}_i, \mathbf{p}]. \tag{7.40}$$

However, in practice, it is not feasible to use this probability. The reason for why this obvious expression for nilmapping probability cannot be used can be explained simply if for a moment we ignore the nondetectability of lines caused by the second and the third factors mentioned at the beginning of the section. In other words, we will assume, for the sake of explanation here, that the only reason a model line is nondetectable is because its image counterpart falls outside the uncertainty region defined by the prescribed Mahalanobis distance. In this case, prob$[g_i \rightarrow \text{nil} \mid \mathbf{z}_i, \mathbf{p}]$ will equal the integral shown in Eq. (7.38). If we use, say, two units of Mahalanobis distance to prescribe the uncertainty region, the integrals under the tails of the Gaussian distribution will equal 0.14, meaning that the probability of any model line g_i being matched to nil is 14%. On the other hand, even if we find the correct match for a model line at the center of the Gaussian distribution, the probability that the match is correct would only equal the height of the Gaussian expression times the size of the Hough cell—this product will in practically all cases be much smaller than 14%. As given by Eq. (7.36), the probability associated with any correct match, assuming it has been detected, is limited by the size of a Hough cell. On the other hand, the probability of a nilmap is proportional to the integral of the Gaussian function over a large chunk of the Hough space. Consequently, with, say, a 14% probability of a nilmap, the objective function in Eq. (7.18) would be maximized by simply mapping every model line to nil, which would be an absurd thing to do.

To cope with this difficulty, and as was done in [BoyKak88], we do not include the contributions from nilmaps in the calculation of the objective function by Eq. (7.18). The nilmaps are kept track of separately, with the goal that we must select that mapping which contains a minimum number of nilmaps. We therefore use the following optimality criteria for the construction of a mapping from the model lines to the image lines:

> An optimal mapping is that path in the search space of Fig. 34 which has the minimum number of nilmaps and a maximum value for the objective function of Eq. (7.18). The objective function takes into account only the non-nilmapped factors from the right side of Eq. (7.18).

To implement this optimality criterion, the growth of any path in the search space whose nilmap count exceeds the minimum encountered for any of the previous paths is stopped and the node used as a backtracking point. And, of all the paths retained—they would have the same minimum number of nilmaps—the one selected has the highest value of the objective function. To expedite computations, in addition to comparing with the minimum number of nilmaps encountered before, we use a threshold on the maximum number of nilmaps along any path. Clearly, a path whose nilmaps are a significant fraction of the number of model lines could not possibly be very useful for self-location.

Our experience gained through actual experimentation has also shown that it pays to accept a mapping function, regardless of the value of the objective function, if the number of non-nilmaps along a path in Fig. 34 exceeds a threshold.

7.5. Correspondence Finding Procedure

We now summarize, using pseudocode, the complete procedure for finding the correspondences between the lines in an expectation map and the lines that can be

```
procedure correspondence_finding( )
    1. S ← ∅;
    2. U_S ← initial robot position uncertainty;
    3. M ← set of model lines;
    4. C ← set of candidate pools;
    5. D_S ← 0;
    6. status ← not_yet;
    7. G ← ∅;
    8. U_G ← ∅;
    9. D_G ← 0;
   10. match(M, C; S, U_S, D_S; G, U_G, D_G; status);
   11. report(status, G, U_G);
   12. end.
```

Variables:

M: a set of model lines such that $\mathbf{M} = \{g_1, g_2, \cdots, g_n\}$.

R: a set of model lines which have not been used in the search and $\mathbf{R} \subseteq \mathbf{M}$.

C: a set of candidate pools such that $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_n\}$ where $\mathbf{C}_i = \{h_{j_i} \mid j = 1, 2, ..., n\}$ and h_{j_i} is an image line candidate for the model line g_i.

S: a partial map such that $\mathbf{S} = \{(g_i, h_{j_i}) \mid i = 1, 2, \cdots\}$ where h_{j_i} may be nil.

$\mathbf{U_S}$: the robot position uncertainty obtained from a a partial map **S**. $\mathbf{U_S}$ consists of the mean vector $\mathbf{p_S}$ and its covariance matrix Σ_S.

G: a set of matches such that $\mathbf{G} = \{(g_i, h_{j_i}) \mid i = 1, 2, \cdots n\}$ and $|\mathbf{G}| = n$ where h_{j_i} may be nil. In other words, the set **G** indicates a full map.

$\mathbf{U_G}$: the robot position uncertainty obtained from a full map **G**. $\mathbf{U_G}$ consists of the mean vector $\mathbf{p_G}$ and its covariance matrix Σ_G.

status :
a status indicates the result of the procedure *match*. The status may be one of the following: *not_yet, success, found, fail. Not_yet* indicates the status that the algorithm has not yet found any full map, while *found* indicates the status that at least one full map is discovered. *Success* indicates the status that the algorithm has found an optimal map, while *fail* indicates that the current path does not lead to any full map.

FIG. 36. The top level procedure for establishing correspondences between the hallway landmarks and the features extracted from an image.

extracted from an image. The main procedure, *correspondence_finding*( ) is shown in Fig. 36, whose bottom half contains the definitions of the variables used. Lines 1 through 5 of the main program define the global variables used. The set **M** contains the model lines; only those lines whose lengths exceed a threshold, set currently at 50 pixels, are selected for inclusion in **M**. Based on the estimated uncertainty in the position of the robot, candidate sets of image lines for each model line are extracted using the projected uncertainty in the image space and the Hough space. The candidate pool for model line g_i is denoted by the set $\mathbf{C}_i$. As a match is established between the model line g_i and some image line h_{j_i}, all the candidate pool $\mathbf{C}_i$ are pruned to reflect the reduced uncertainty in the position of the robot. This pruning is accomplished by line 23 of the procedure *match* that is called by *correspondence_finding* in line 10.

When first called in line 10 of *correspondence_finding*, the procedure *match*, displayed in Fig. 37, has **M**, **C**, **S**, $\mathbf{U_S}$ and $\mathbf{D_S}$ for input variables and **G**, $\mathbf{U_G}$, $\mathbf{D_G}$, and *status* as output variables. Note that **S**, the set of (g_i, h_{j_i}) matches established so far, is empty when *match* is first called. We will refer to **S** as a partial map. Also note that $\mathbf{U_S}$ is the robot position uncertainty computed on the basis of the partial map **S**. Of course, initially, that means that $\mathbf{U_S}$ is the robot position uncertainty established on the basis of the commanded motion. The value of the input variable $\mathbf{D_S}$ is the value of the objective function $\mathbf{D}_0$, defined in Eq. (7.18), calculated on the basis of nonnilmaps contained in **S**. Since initially **S** is empty, $\mathbf{D_S}$ is zero. That takes care of all the input variables when *match* is first called.

The procedure *match* returns instantiations for the output variables **G**, $\mathbf{U_G}$, $\mathbf{D_G}$, and *status*. The variable **G** is the final set of matches (g_i, h_{j_i}) that optimize the criterion mentioned in the previous section. The returned value of $\mathbf{U_G}$ is the final uncertainty in the position of the robot, established on the basis of the matches contained in **G**, and $\mathbf{D_G}$ is the final value of the objective function.

To briefly describe the procedure *match*, in line 1 it compares the number of nilmaps in the partial map **S** with the minimum number of nilmaps in any previously de-

```
procedure match(R, C; S, U_S, D_S; G, U_G, D_G; status):
 1.  if (G ≠ ∅ and #nils(S) > #nils(G)) then status ← fail; return;
 2.  if #nils(S) > threshold(nil) then status ← fail; return;
 3.  if #success(S) > threshold(success)
 4.    then begin G ← S; U_G ← U_S; D_G ← D_S; status ← success; return; end;
 5.  if |S| = |M|
 6.    then if G = ∅
 7.           then begin G ← S; U_G ← U_S; D_G ← D_S; status ← found; return; end;
 8.         else if ( #nils(S) < #nils(G) ) or ( #nils(S) = #nils(G) and D_S > D_G )
 9.                then begin G ← S; U_G ← U_S; D_G ← D_S; status ← found; return; end;
10.              else status ← fail; return;
11.  for each g_i ∈ R
12.    begin
13.      #c_i = 0;
14.      for each h_j ∈ C_i
15.        begin
16.          compute d_{i,j} = mahalanobis_distance((g_i, h_j), U_S);
17.          if d_{i,j} < threshold(mahalanobis) then #c_i ← #c_i + 1;
18.        end;
19.    end;
20.  find the model line g_k ∈ R such that #c_k ≤ #c_i for all g_i ∈ R;
21.  R ← R − {g_k};
22.  for each h_j ∈ C_k
23.    if the Mahalanobis distance d_{k,j} > threshold(mahalanobis) then C_k ← C_k − {h_j};
24.    else p_{k,j} = log prob[g_k → h_j | U_S];
25.  sort C_k in decreasing order with respect to p_{k,j};
26.  if C_k = ∅ then
27.    begin
28.      S ← S ∪ {(g_k, nil)};
29       match(R, C; S, U_S, D_S; G, U_G, D_G; status);
30.    end;
31.  else
32.    while C_k ≠ ∅ do
33.      begin;
34.        h_{kq} ← the first element in C_k;
35.        C_k ← C_k − {h_{kq}};
36.        S_new ← S ∪ {(g_k, h_{kq})};
37.        D_new ← D_S + p_{k,kq}
38.        update_uncertainty(U_S, U_new, (g_k, h_{kq}));
39.        match(R, C; S_new, U_new, D_new; G, U_G, D_G; status_new);
40.        if status_new = success then
41.          status ← success; return;
42.        else if status_new = found then status ← found;
43.      end;
44.  return;
```

FIG. 37. This procedure is the workhorse of the top level shown in Fig. 36.

rived full map **G**. The node in the search space of Fig. 34 where this condition is violated then serves as a backtracking point. Line 2 makes sure that the total number of nilmaps in the partial map **S** do not exceed a threshold; if they do then backtracking would again be initiated by line 2. In line 3, *#success*(**S**) returns the number of non-nilmaps in the partial map **S**. If the condition in line 3 is satisfied, then, in accordance with our comments in the previous subsection, the recursion is terminated in line 4 and the map contained in **S** accepted as the final map, hence the assignment of **S** to **G**.

Lines 5 through 10 are invoked when a path in the search space of Fig. 34 reaches the bottom of the space, in other words when the cardinality of **S** equals the cardinality of **M**. In this case, if there is no prior full map **G**, the partial map **S** becomes the first full map (line 7). If a prior full map was obtained, then in line 8 a comparison is made of nilmaps contained in **S** with the nilmaps in the prior **G**. If the number of nilmaps in **S** is strictly less than that in **G**, in line 9 **S** becomes the new **G**. The same happens if the number of nilmaps in **S** equals that in **G**, provided the value of the objective function for **S** is larger the value for **G**.

In line 11, the variable **R**, initially the same as **M** and subsequently reset in line 21, is the set of model lines not yet used for mapping. Lines 11 through 21 are invoked for finding that landmark g_k which has the smallest candidate pool C_k. Note the symbol $\#c_i$ designates the cardinality of the candidate pool C_i; this cardinality is calculated in lines 12 through 19 by counting only those pool members that are inside of the user-specified Mahalanobis threshold using the latest estimate U_S of the robot position uncertainty.[22]

The landmark with the smallest candidate pool is then considered in lines 22 through 43 for matching. First considering those landmarks that have the smallest candidates pools leads to more efficient backtracking. Consider the fact that at the very least a landmark will have a pool of size one containing a nil. Therefore, by considering landmarks with small sized candidate pools first, we more quickly assign the nils than would otherwise be the case. As a result, we derive greater power from the optimal criterion stated in Section 7.4 that says that a mapping function should be rejected if the number of nilmaps used exceeds some threshold.

To find the correspondent for the landmark g_k, in lines 22 though 25 we first prune C_k by deleting those candidates that are outside the Mahalanobis threshold, and then order the remaining C_k on the basis of the probability measures computed in line 24. If the resulting candidate set C_k is null, we add the pair (g_k, nil) to the partial map S in line 28 and make a recursive call to match in line 29. If C_k is not empty, we choose the first candidate in C_k as a possible match for the landmark g_k. One cycle of the Kalman filter is now invoked in line 38 to update the robot position uncertainty U_S producing a new estimate U_{new} of this uncertainty. After that, in line 39, we reinvoke match recursively. The status statements in lines 40 through 42 should become obvious from the definition of status in Fig. 36.

In Fig. 38 we have pictorially illustrated the sequential reduction of the uncertainty in the position of the robot as landmarks and image features are paired up by the procedures discussed. Each row of this illustration corresponds to one step of the sequential matching process, or, more precisely, to one call to the procedure *match* of Fig. 37. Initially, we seek a match for the vertical line g_1 in the expectation map shown on the left in the top row. Shown in the middle figure of the top row is the image frame containing the lines corresponding to the detected Hough space parameters. The lines in the shaded region of this frame are within the Hough space uncertainty ellipse for landmark g_1. The size of the Hough space uncertainty ellipse is determined by the uncertainty in the position of the robot, this initial uncertainty being shown on the right in the top row. If we match g_1 with h_1, shown in the middle of the second row, the robot position uncertainty becomes as shown at right in the second row. Also, the uncertainties associated with landmarks g_2 and g_3 change; these uncertainties are displayed by the shaded regions in the middle of the second row. Now suppose landmark g_2 is matched with the image feature h_2 shown in the middle of the third row. This causes the robot position uncertainty to change into the ellipse shown on the right in the third row. Also, the uncertainties associated with landmarks g_3 and g_4 become what is depicted by the two shaded regions in middle frame of the third row. Matching g_3 with h_3 reduces the robot position uncertainty to the small circle shown on the right in the last row. Of course, this sequence of matches would only be accepted if the value of the associated objective function and the number of nilmaps used (we did not use any in our example) satisfy the optimality criterion.

8. PATH PLANNING, PATH REPLANNING, AND PERCEPTION PLANNING

It is obvious that the method of self-location discussed in the preceding section needs to be embedded in a larger

[22] The reader is probably wondering why in line 17 we don't delete those pool members that are outside the Mahalanobis threshold, instead of just counting those that are within the threshold. The reason has to do with the fact that a deletion of a candidate in line 17 would make it permanently unavailable for the rest of the search path (without backtracking). By deleting candidates in line 23 we allow such candidates to be considered again later on.

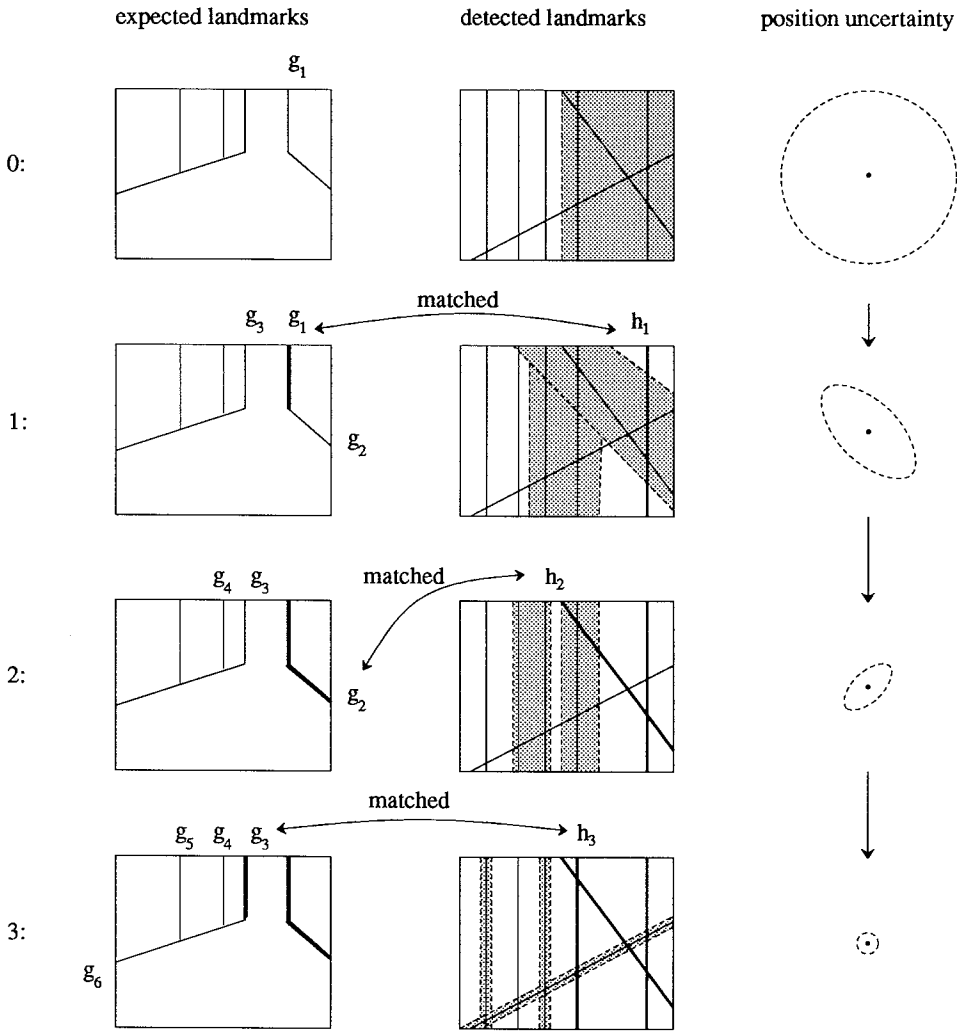


FIG. 38. The sequential approach to the reduction of uncertainty in the position of the robot, illustrated pictorially.

processing scheme that would be capable of the following:

(1) accepting a destination from the human supervisor (currently the destination in our system is specified by the goal (x, y) coordinates and the final orientation desired for the robot);

(2) planning a path from the initial position and orientation to the destination position and orientation;

(3) keeping track of the uncertainties during the various translational and rotational motions as the robot starts its travel toward the destination;

(4) stopping for the self-location exercise when the accumulated uncertainties hit a maximal human-specified bound;

(5) if the orientation of the robot for the self-location exercise is such that the number of visible landmarks is too low, being able to examine the environment using a different viewpoint (this is an example of animate vision [Bal89]);

(6) replanning the path to the destination from the point where self-location is carried out;

(7) being able to carry out collision avoidance using the ultrasonic sensors on the robot during motions from one self-location point to another;

(8) being able to keep track of the growth of uncertainties during the reactive behavior exhibited during collision avoidance and being able to stop for self-location when the uncertainties hit a user-specified limit.

All of these capabilities are supported by the computational framework presented in Fig. 2. In this section, we will discuss further the precise nature of the data struc-

ture and the computations that support the path and perception planning components.

8.1. Path Planning

There exists a voluminous amount of literature on the subject of path planning. Instead of citing all the pertinent references, we will refer the reader to the book by Latombe [Lat91]. Suffice it to say that there now exist two major approaches to path planning, one based on configuration space ideas first pioneered by Lozano-Perez and Wesley [LozWes79] and the other using potential functions, their use first proposed by Khatib [Kha86] in the context of realtime robot control. In the configuration space approach, the environment is typically modeled by a set of polygons and a solution path which consists of linear motions with the shortest distance considered [LozWes79, Lat91, Koi89]. Also, the robot is shrunk to a point, while the objects in the environment are enlarged by the size of the robot. The desired trajectory is then calculated for the point robot so that the enlarged fictitious objects in the environment can be avoided. Several algorithms have been proposed for the extraction of minimum distance paths using this approach.

In the potential field approach to path planning, a globally computable scalar function with potential-field-like properties is associated with each object. To find collision-free paths, one must search through the valleys of the composite potential field corresponding to all the objects. This search process has two components, one that deals with keeping track of different branches when a valley bifurcates and making sure that at each point along the chosen path there exists some orientation of the robot that would result in collision-free motion. As demonstrated in [HwaAhu88, Lat91], the determination of collision-free orientations along the potential-field valleys requires gradient-descent calculations. While such calculations are necessary in the context of assembly motion planning [GotKak91a, GotKak91b], they would be excessively burdensome for our purposes especially because the clearances involved in our case are not that tight and because computational time is of the essence.

We will now describe a rather simple approach, particularly appropriate to a robot traversing a flat plane, to path planning that suffices for our case. As we will demonstrate, this approach supports all the path planning and replanning capabilities mentioned at the beginning of this section.

In order to make a path from the initial position to the goal position, we first construct a bitmap of the floor of the hallway from the data structure of Fig. 15; next, we extract a radial-valued skeleton of this bitmap, the limbs of this skeleton corresponding roughly to the midpoints of the hallway; finally, for each pair of start and destination positions we smooth the relevant sections of the skeleton to reduce the number of turns required. The skeletal segment thus obtained is the desired path from the start position to the destination position provided the start point lies on one of the limbs of the skeleton. If the start point does not lie on the hallway skeleton—a possibility that occurs invariably during path replanning after collision avoidance—a procedure is used to connect the start point with the farthest possible point on the skeleton. We will now discuss each of these steps in some detail:

Step 1: Construct a Bitmap Representation of the Floor

As discussed in Section 5.1, a hallway is represented by a threaded tree, whose face-level nodes contain pointers that allow a fast traversal of all the basic faces of the hallway in clockwise scan from any point in the hallway. Therefore, it is easy to extract from the hallway data structure of Fig. 15 an ordered list of vertices that define the hallway, each vertex being a pair of coordinates in the xy-plane of the world frame. The polygon corresponding to this ordered list of vertices will be referred to as the *floor polygon*.

Next, the floor polygon is triangularized by an algorithm of complexity $O(N^2 \log N)$, where N is the number of sides of the floor polygon [PreSha85]. Triangularization is a necessary precursor to the construction of the floor bitmap since the triangular facets help us determine in time $O(1)$ whether or not a point on the floor is inside or outside the triangle. So, in order to construct a bitmap for the floor we simply lay out a sampling grid on the xy-plane of the world coordinate frame. This sampling grid is then raster scanned and the inside–outside predicate tested for each point with regard to, in the worst case, all the triangles. In our current implementation, the complexity of this operation for each point on the sampling grid is $O(M)$, where M is the number of triangles generated by the triangularization algorithm.[23] In our current implementation, we use a 5-cm resolution for the sampling grid. Figure 39 shows the bitmap representation of the floor polygon for the hallway of Fig. 4. After obtaining the bitmap of the floor polygon, we construct the configuration space for the robot, taking into account the radius of the robot, by essentially eroding the boundary of the floor bitmap by the radius of the robot. The actual implementation consists of first constructing a binary mask that has a circular region of ones representing the robot and zeros elsewhere. By raster scanning the robot mask over the floor bitmap and taking a logical AND of

[23] It would of course be easy to considerably reduce this complexity by using rectangular enclosures for the triangles and then using some sort of a hashing scheme (a two-dimensional hash function utilizing the coordinates of one of the vertices of the enclosure should do) to examine a minimum number of triangles.

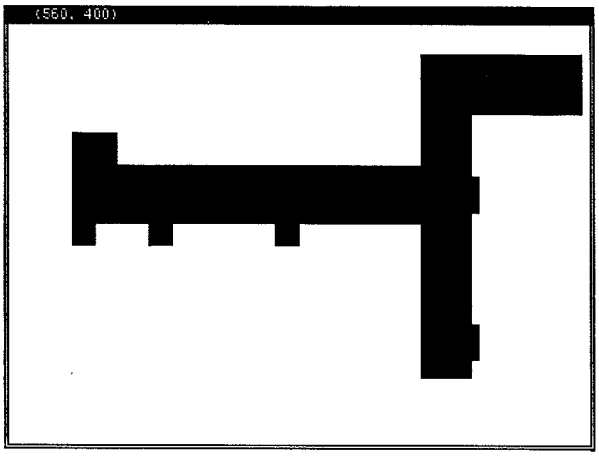


FIG. 39. The bitmap representation of the floor polygon.

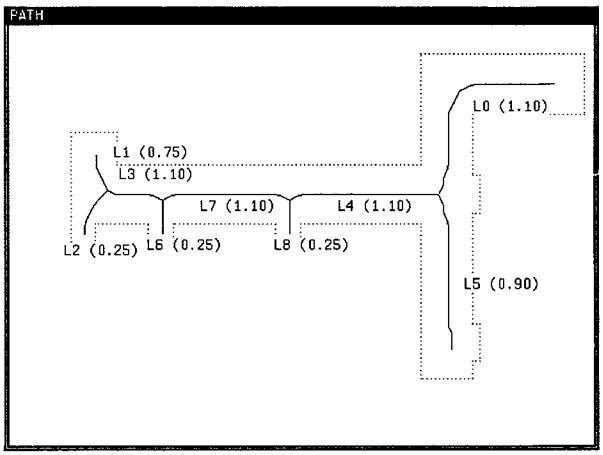


FIG. 41. The radial-valued skeleton obtained from the bitmap of the configuration space of Fig. 40. The limbs of the skeleton are identified separately. Also shown for each limb is the most frequently occurring value of the radius in meters.

the superimposed pixels, we can output a one or a zero for the configuration space. This configuration space represents the set of all points that can be occupied by the center of the robot base without the robot colliding with the walls of the hallway. Figure 40 illustrates the configuration space for the floor polygon shown in Fig. 39.

Step 2: Extraction of Floor Skeleton from the Bitmap

The bitmap of the configuration space obtained in Step 1 is now processed to yield a radial-valued skeleton of the floor. A radial-valued skeleton is a regular skeleton augmented by what is referred to as the "peel-off" number, which is the number of deletions of boundary "layers" that resulted in the creation of the skeletal pixel. This peel-off number is approximately equal to the minimum chessboard distance from the skeletal pixel to the boundary of the hallway [RosKak82]. The algorithm for computing the radial-valued skeleton of a binary pattern, given in [BoyKak86], consists of raster scanning the bitmap with a 3×3 binary mask and using a predicate that simultaneously tests whether or not a pixel is a border pixel and whether or not the deletion of a pixel would alter the topological connectivity of the binary pattern. The scanning with the mask and predicate testing is done sequentially from each of the four directions (west, east, south, and north). Figure 41 shows the radial-valued skeleton for the bitmap of the configuration space in Fig. 40. Although we have gray-scale-modulated the skeleton to display the radial values at each point, the variations in the gray scale are admittedly hard to discern. So, along each limb we have also shown, using a parenthesized number, the most frequently occurring value of the radius in meters. However, note that in general the value of the radius will not be constant along a limb.

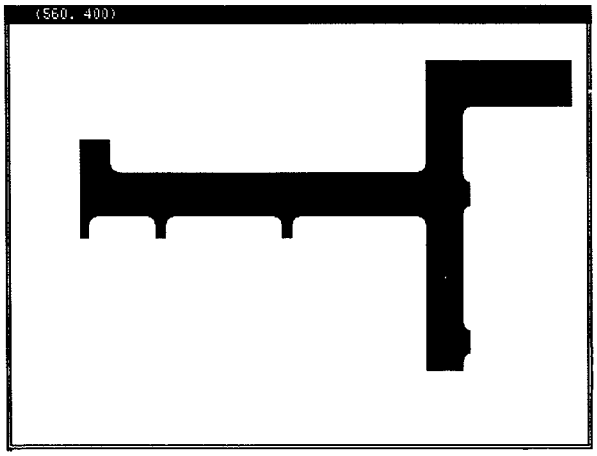


FIG. 40. The configuration space for the floor polygon shown in Fig. 39 is obtained by eroding the boundary of the floor bitmap by the radius of the robot.

For the purpose of path planning, we next decompose the skeleton into its limbs and represent each limb in the computer memory by the following triple:

$$\begin{aligned}&(\text{limb\_ID} \quad \text{length} \quad \{(x_0, y_0, r_0), (x_1, y_1, r_1),\\ &\quad \ldots, (x_{m-1}, y_{m-1}, r_{m-1})\}),\end{aligned} \tag{8.1}$$

where *limb_ID* is a symbolic name for the limb, *length* the length of the limb in pixels, and for each index $i = 0, 1, \ldots, m-1$, the tuple (x_i, y_i, r_i) the ith element of the limb, where r_i is the radius at the skeletal pixel (x_i, y_i). The skeleton of Fig. 41 will be represented in the computer memory by nine limbs. Simultaneously with the decomposition of the skeleton into its limbs, we also create an adjacency matrix that tells us which limbs are directly connected by virtue of being adjacent. The adja-

cency matrix for the skeleton of Fig. 41 is shown below:

	L_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8
L_0	1	0	0	0	1	1	0	0	0
L_1	0	1	1	1	0	0	0	0	0
L_2	0	1	1	1	0	0	0	0	0
L_3	0	1	1	1	0	0	1	1	0
L_4	1	0	0	0	1	1	0	1	1
L_5	1	0	0	0	1	1	0	0	0
L_6	0	0	0	1	0	0	1	1	0
L_7	0	0	0	1	1	0	1	1	1
L_8	0	0	0	0	1	0	0	1	1

Step 3: Extraction of Skeletons for a Global Path

As we mentioned earlier, the human supervisor provides a pair of points for the construction of the robot's path, namely the initial point P_{start} and the goal point P_{goal} of the path, both points being specified by (x, y) coordinates in the xy-plane of the world coordinate frame. Of all the limbs generated in the previous step, we now find the two limbs that are closest to P_{start} and P_{goal}. For any point P in the xy-plane, the closest limb is found by calculating the Euclidean distance from P to every point on a given limb and then repeating this calculation for all the limbs in the hallway skeleton. [It would be possible to make this computation more efficient by using, say, rectangular enclosures for the limbs, but we have not done so yet.] Let the symbolic identifiers of these two limbs be L_{start} and L_{goal}.

Using Dijkstra's algorithm [AhoHop74], the path planner now traverses the adjacency matrix, treating the length of the limbs as a cost function, and finds a sequence of limbs

$$\{L_{i_1}, L_{i_2}, \ldots, L_{i_m}\},$$

where the indices i_1 and i_m are those that correspond respectively to the limbs L_{start} and L_{goal}. The sequence of limbs returned by Dijkstra's algorithm is that for which the total cost function is minimum and that includes the lengths associated with the end limbs L_{start} and L_{goal}. (As we will argue later, such a sequence of limbs does not necessarily correspond to the shortest path between P_{start} and P_{goal}.) For example, with P_{start} and P_{goal} as shown in Fig. 42, the sequence of limbs extracted in this manner is

$$L_1, L_3, L_7, L_4, L_5.$$

Before the construction of the actual path, the sequence of limbs returned by Dijkstra's algorithm is now smoothed to remove unnecessary turns, because, as we will show in the section on experimental results, a rotational motion is capable of introducing more uncertainty than a pure translational motion. Smoothing of the limbs

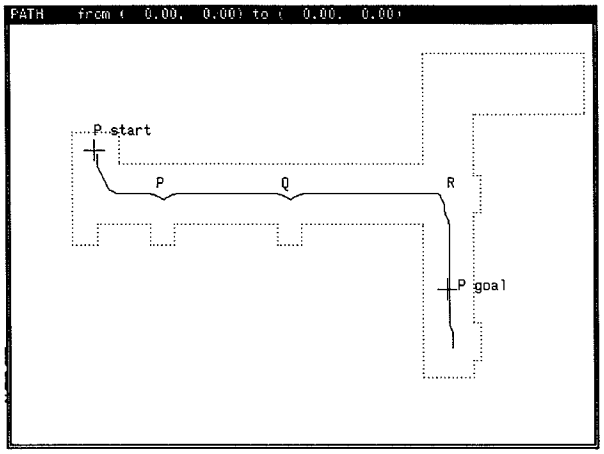


FIG. 42. For designated start and goal positions, the sequence of limbs which constitutes the global path.

is accomplished by first treating the limbs returned by Dijkstra's algorithm as a continuous space curve and then applying a five-point window operator to this curve. The five-point window that runs along the space curve computes the local direction of the space curve in the xy-plane by treating the five-element segment as a fragment of a straight line. Then by imposing tolerance intervals on these local angles, straight segments of the space curve can be easily extracted. Figure 43 shows the output of this smoothing process for the P_{start} and P_{goal} of Fig. 42.

As is clear from Fig. 43, while the window operator removes some of the finer turns in the skeletal path, it is incapable of eliminating the larger turns, such as those introduced at P, Q, and R in Fig. 42 by the skeletonizing program. To get rid of these turns, the path planner first extracts the longest straight line segments of the space curve, these segments being stored in a sequence to pre-

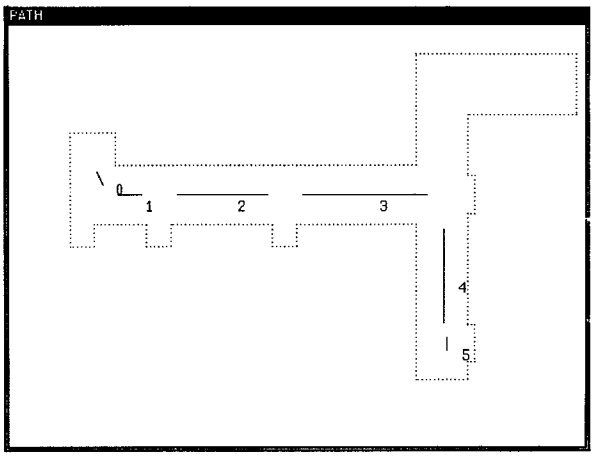


FIG. 43. Straight segments extracted from the limbs on the global path shown in Fig. 42.

serve their adjacency relations. For the space curve of Fig. 42, the segments thus obtained are shown in Fig. 43. The planner now attempts to join these straight segments by extending each toward the next in the sequence. The extension is made by testing each pair of straight segments for collinearity within a user-specified threshold; if found collinear, the endpoints of the lines are simply joined, provided such a join does not violate any feasibility restrictions. To explain what we mean by feasibility of restriction, assume that the list of straight segments is $\{\overline{Q_0R_0}, \overline{Q_1R_1}, \ldots, \overline{Q_{n-1}R_{n-1}}\}$ and let us focus on the join between R_i and Q_{i+1} assuming that the straight segments $\overline{Q_i,R_i}$ and $\overline{Q_{i+1}R_{i+1}}$ have been found to be collinear. We temporarily connect R_i and Q_{i+1} by a line segment and check the feasibility of this line segment for robot travel. In order to perform the test, we discretize the new line segment $\overline{R_iQ_{i+1}}$ with the same resolution as the floor bitmap. Let $(\bar{x}_k, \bar{y}_k)$ be the kth discretized point on the analytically defined line segment $\overline{R_iQ_{i+1}}$. We now go back to the bitmap of the configuration space to check whether or not the discretized point $(\bar{x}_k, \bar{y}_k)$ is an interior point of the bitmap of the configuration space. This test can be done by simply examining the bit of the pixel corresponding to $(\bar{x}_k, \bar{y}_k)$.

When the two consecutive straight segments in the sequence $\{\overline{Q_0R_0}, \overline{Q_1R_1}, \ldots, \overline{Q_{n-1}R_{n-1}}\}$ are not collinear, the two near endpoints are simply joined by extending the segments, that the two segments will join is guaranteed by the noncollinearity condition.

Figure 44 shows the final result obtained by using this approach on the output of the smoother shown earlier in Fig. 43. As the reader can see, all the unnecessary turns have been eliminated.

When a sharply curving section of a hallway lies between straight sections, the approach discussed above will fail to join since a feasible join between two collinear segments, such as the segments P and Q shown in Fig. 45, will not be found. So if the feasibility condition is violated for any pair of consecutive segments in the sequence $\{\overline{Q_0R_0}, \overline{Q_1R_1}, \ldots, \overline{Q_{n-1}R_{n-1}}\}$, the above procedure is implemented recursively by first locating the midpoint M of the skeleton between the endpoints, such as the points P and Q in Fig. 45, and then invoking the procedure on the two segments $\overline{PM}$ and $\overline{MQ}$.

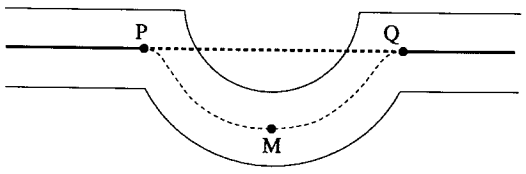


FIG. 45. For turning hallways, as shown here, connecting the straight segments extracted from the limbs of the skeleton requires a recursive invocation of the procedure by identifying the midpoints of the turns.

Step 4: Connecting Initial and Goal Points with Skeletons

The path constructed so far still does not start from P_{start} and does not end at P_{goal}, even though these extremal points were used for the determination of L_{start} and L_{goal}. Therefore, the path planner now joins the start point P_{start} with the furthest endpoint for any of the straight segments produced by Step 3. This is done by testing straight-line connections from P_{start} with all the endpoints for navigational feasibility, using the same procedure as described in Step 3 and accepting that endpoint which is feasible and yet furthest from P_{start}. If such an endpoint does not exist, then we simply draw a perpendicular from P_{start} to L_{start}. Exactly the same is done for the goal point P_{goal}. The final path thus constructed for the P_{start} and P_{goal} of Fig. 42 is shown in Fig. 46.

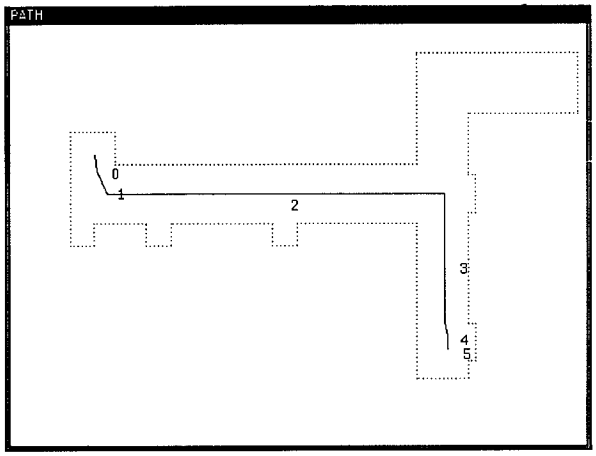


FIG. 44. Straight segments of Fig. 43 after they are connected to form a minimum-turn global path between the designated start and goal positions.

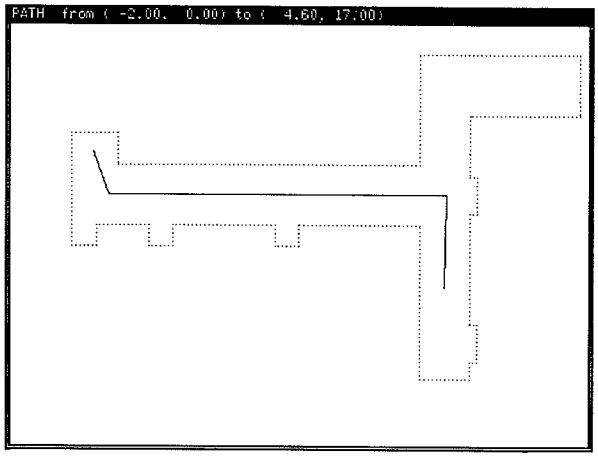


FIG. 46. A minimum-turn global path from the initial point to the goal point.

8.2. Path Replanning

After the robot has gone through a self-location exercise by using the procedure described in Section 7, its actual location will, in general, be found to deviate from the originally planned global path. What that implies is that after every self-location exercise, the robot must replan its path to the goal. As we mentioned in Section 8.1, the original global path consists of a sequence of limbs of the hallway skeleton, as produced by Step 3 of the preceding subsection. Path replanning is accomplished by repeating Step 4 with P_{start} replaced by the robot position found by the self-location procedure.

8.3. Perception Planning

The Perception Planner, which sits in module 3 of the overall architecture of Fig. 2, has a critical role to play in regulating the deliberative behavior of the robot, especially so under the dynamic conditions that might be created by obstacles in motion and the accumulated uncertainty in the position of the robot since its last exercise at self-location. It does so by monitoring the mean $\bar{\mathbf{p}}$ and the covariance matrix Σ_p of the robot position. In our current implementation, these two parameters are queried by Perception Planner every 0.5 s.

The conditions that the Perception Planner responds to and the actions taken by it are as follows:

(1) When the monitored uncertainty in robot position exceeds a threshold, the perception planner stops the robot and invokes the self-location procedure of Section 7. As the reader will recall, Σ_p measures the variance in p_x, p_y, and ϕ. Clearly, at any position of the robot, the variance is not a single number but defines an ellipse in the three dimensions spanned by p_x, p_y, and ϕ. Theoretically, we would like the Perception Planner to stop the robot as soon as the maximum of the values on the variance ellipse exceeds a threshold, but since it is rather impossible to make a direct comparison between translational and rotational variances, we have adopted a two-pronged approach. The Perception Planner separately keeps track of the variances in the xy-plane and for the ϕ variable. For example, shown in the upper part of the Fig. 47 are the variance ellipses corresponding to the position variables p_x and p_y of the robot[24] while the robot is engaged in collision avoidance behavior. At point A of this plot, the robot has zeroed out its uncertainty by using the self-location procedure of Section 7. Subsequently, during the motions dictated by the collision-avoidance behavior, the uncertainty ellipses grow larger, until at point B the Perception Planner decides to bring the robot to a halt due to the maximal variance associated with the ellipse at that point. Given a particular variance ellipse, its maximal variance is measured by finding the larger eigenvalue associated with the upper left 2×2 submatrix of the Σ_p matrix.

The rotational variance is kept track of separately and compared to its own user-specified threshold. Shown in the lower part of Fig. 47 is the growth in rotational variance as the robot is engaged in the collision avoidance behavior. The Perception Planner stops the robot for self-location whenever either the rotational variance exceeds its threshold or the principal eigenvalue of the xy-variances exceeds its threshold.

(2) If the procedure of Section 7 fails at self-location, the failure being recognized by the number of nilmaps in the mapping function exceeding a user-specified threshold, the Perception Planner will cause the robot to execute a rotation so that a different and visually richer viewpoint can be used. The visual richness of a viewpoint is measured by simply counting the number of landmarks in the expectation map. So, when the originally used viewpoint fails, the robot examines a sequence of viewpoints at 20° intervals, measures the richness of each, and accepts the first that meets the visual richness criterion.

Although we will not dwell on the matter here, it is possible to operate the robot in a mode where the Perception Planner interacts with the Path Planner in such a manner that the next expected stop for self-location will be rich in visual cues in terms of the number of landmarks that would be visible from the orientation of the robot at the end of the travel. Such interaction between the two planners makes sense only when it is known a priori that the collision avoidance behavior will cause only minor deviations from the planned paths.

8.4. Collision Avoidance

Since the focus of this paper is on the deliberative behavior of the robot as influenced by visual inputs, we shall not say much about the specifics of the reactive behavior for collision-avoidance. Suffice it to say that the robot is equipped with a semiring of ultrasonic transducers. Ultrasonic echoes are analyzed for the time of return and thus the locations of the obstacles deduced. A local process on the computer on board the robot constantly monitors these echo returns and commands the robot away from the directions in which the closer obstacles are perceived.

9. EXPERIMENTAL RESULTS

The aim of this section is to discuss the experimental aspects of our work.

[24] In terms of the Σ_p matrix, the ellipses shown in the upper part of Fig. 47 are formed by the upper left 2×2 submatrix of the Σ_p matrix at each position of the robot.

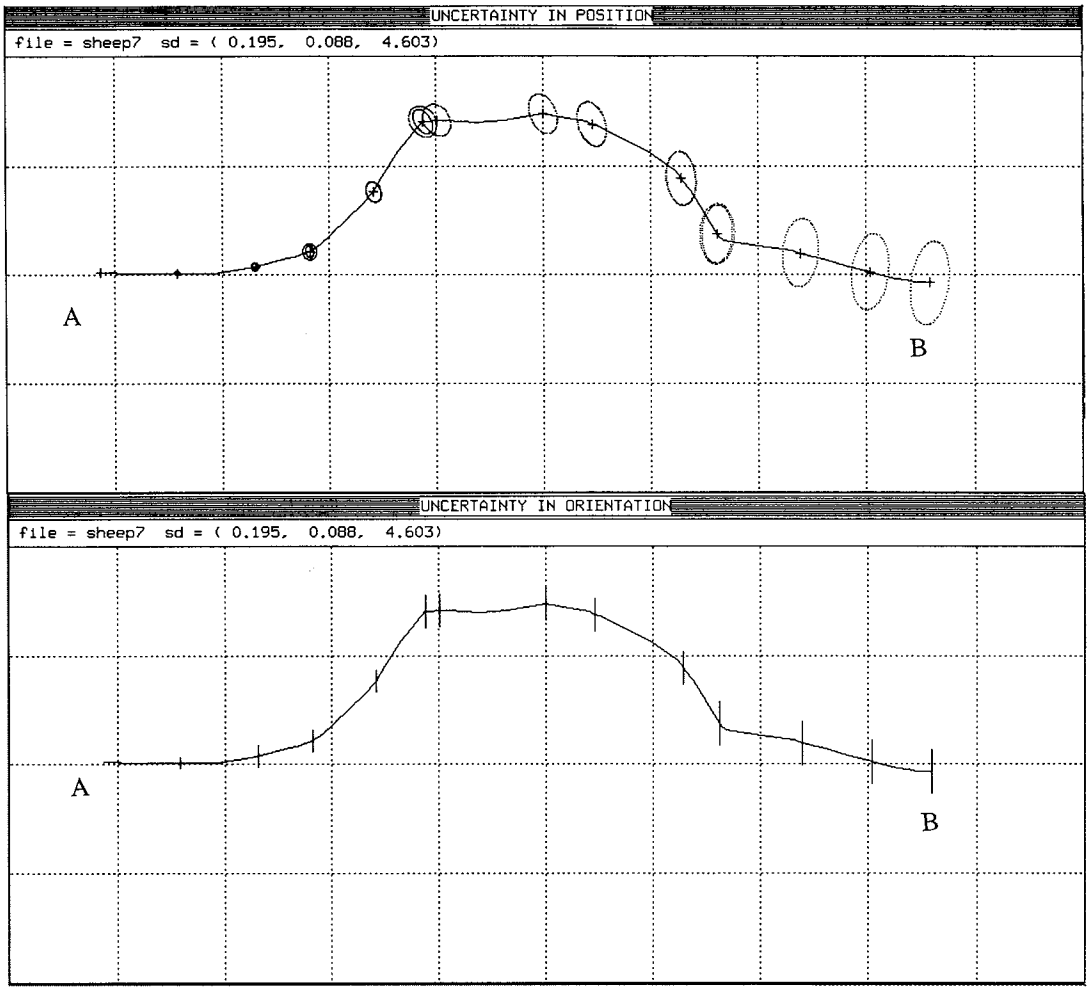


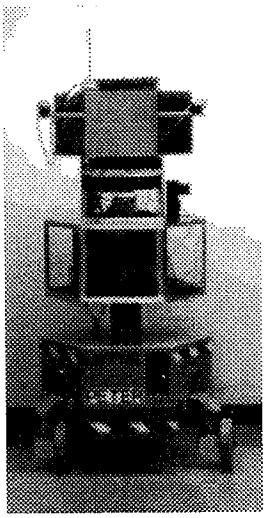
FIG. 47. The Perception Planner keeps track of the growth of the uncertainty in the position parameters of the robot. Shown here is data when the robot is engaged in collision-avoidance behavior. Starting at position A in the hallway, the robot turns left to avoid a moving obstacle and then tries to return to its originally planned path by turning right. During this process, the uncertainty in the translational parameters p_x and p_y grows as shown by the ellipses in the upper part of the figure. Shown separately in the lower part of the figure are bars whose lengths are proportional to the standard deviation in the rotational angle ϕ.

9.1. The Mobile Robot Used in This Research

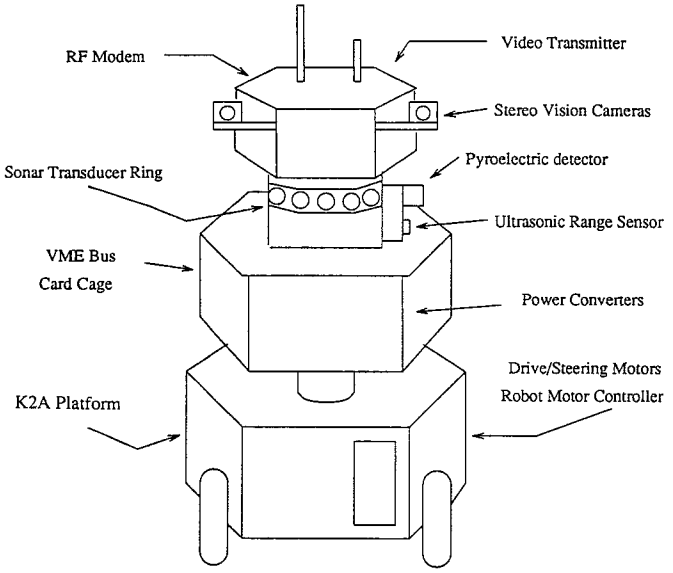
We start with a brief presentation of the various features of the mobile robot, deferring the details to a longer description in [LopKak89]. The mobile robot, PETER, consists of a Cybermation K2A platform, on top of which we have built our own "electronic torso" (Fig. 48). This torso contains a VME-bus based computing hardware which acts both as a supervisor for the factory-supplied motion controller in the platform and as a processor for autonomous reactive behavior for collision avoidance using a semiring of ultrasonic sensors. Since vision processing is done off-board, the VME-bus based computing hardware also maintains a communication link with a SUN4 processor.[25] A video transmitter continually broadcasts the camera image to an off-board frame grabber, which in turn sends the digitized image to the SUN4 processor. Although the robot is equipped with two cameras for research in navigation using binocular stereo, only monocular vision was used for the research reported here.

From the standpoint of what follows in the rest of this section, the important thing to note is that the robot exe-

[25] It is important to note that all the vision computations currently being executed off-board on a SUN4 processor could easily be implemented on an on-board single-board computer that would plug into the VME chassis. Actually, we are in the process of making this change at this time. These additional on-board computations will of course increase the drain on the power supply which will be beefed up with the installation of additional batteries on the robot.



(a)



(b)

FIG. 48. The mobile robot PETER. (a) The picture of the mobile robot. (b) The hardware components.

cutes two distinct motions: a translational motion and a rotational motion. While the former is nominally free of rotations, the latter is nominally free of translations; in other words, given no differential slippage between the wheels, the robot is capable of rotating about a point without undergoing any translations. Therefore, our robot is a holonomic device.

9.2. Motion Uncertainty Experiments

In Section 4.2 we discussed the characterization of the two basic motions of the robot from the standpoint of uncertainty. We said that for the case of purely translational motions, when the robot is commanded to go straight through a distance d_0, its resulting motion can be represented by the parameters d, α, and β, where d is the straight line distance between the start point and the point where the robot stops, α the angle which shows the direction of travel of the robot in terms of where the robot stops, and β the angle between the old y_r axis and the new y_r axis (Figs. 5). As stated earlier, for a given commanded d_0, the entities d, α, and β are random variables, characterized by their mean values, their standard deviations, and correlation coefficients with one-another; all of these characterizations were captured in the covariance matrix of Eq. (4.8). Note that all of these characterizations are functions of the command variable d_0. For seven different values of the command parameter d_0 (0.5, 1.0, 2.0, 3.0, 4.0, 5.0, and 6.0 m), the robot was ordered to execute the translational motion 84 times. For each value of d_0 and for each set of repeated experiments, the response of the robot was measured and statistically analyzed for the determination of the mean values $\bar{d}$, $\bar{\alpha}$, $\bar{\beta}$, the standard deviations σ_d, σ_α, σ_β, and, the correlation coefficients $\rho_{d\alpha}$, $\rho_{d\beta}$, $\rho_{\alpha\beta}$. In Figs. 49a–49e we have shown these statistical parameters of the translational motion of the robot. Although difficult to discern from the figure, in (a) the value of the mean $\bar{d}$ is always less than the value of d_0, which we believe is a consequence of the slippage of the wheels. The mean values $\bar{\alpha}$ and $\bar{\beta}$ of the angles α and β are, as shown in (b), quite close to zero; however, the standard deviations of the same two parameters shown in (d) are significant. To appreciate the significance of these standard deviations, Fig. 49d tells us that, if we assume d, α, β to be Gaussian, with more than 60% probability the orientation of the robot will be off by close to 2° if the robot is commanded to execute a translational motion of 6 m. Extrapolating the σ_β curve in Fig. 49d, this means that if the robot were commanded to move straight by 48 m, the orientation of the robot would be off by 16° with a high probability. With such a large error in orientation, not to mention the errors in the angle α (note that, as shown in Fig. 49e, α and β become highly correlated the longer the commanded distance) and the actual distance traveled d (σ_d, shown in Fig. 49c, being a measure of this error), the robot would certainly not be capable of navigating by just dead reckoning if any appreciable distances are involved. What is particularly noteworthy about the plots in Fig. 49 is the smoothness of the dependencies of the experimentally measured statistical parameters on commanded motion parameter d_0. That implies that when the robot is commanded to translate by d_0 whose value is other than those used in the statistical studies, we can either interpolate or extrapolate, as required.

An identical statistical study was made of rotational motions. As described in Section 4.2, a commanded rotation is designated by θ_0 and resulting position of the robot

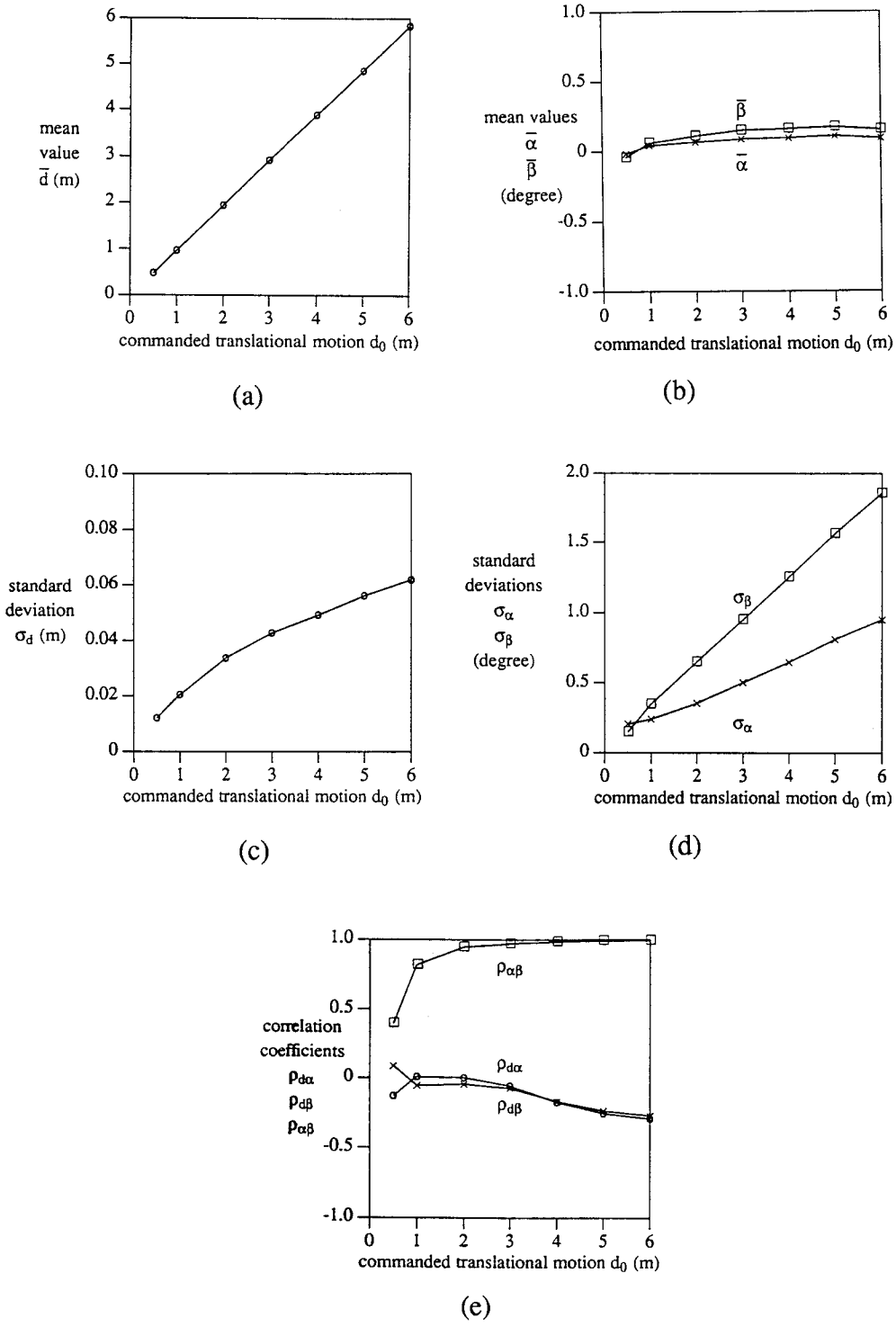


FIG. 49. When the robot is commanded to translate through a distance d_0, the resulting motion is characterized by the parameters d, α, and β. Shown here are (a) the mean value d, (b) the mean values $\bar{\alpha}$ and $\bar{\beta}$, (c) the standard deviation σ_d, (d) the standard deviations σ_α and σ_β, and (e) the correlation coefficients $\rho_{d\alpha}$, $\rho_{d\beta}$, and $\rho_{\alpha\beta}$.

by u, v, and θ; u and v are differential-slippage-caused translational components of the resulting motion and θ the rotation executed by the robot. For each commanded θ_0, the statistical characterizations of u, v, and θ are captured by the covariance matrix of Eq. (4.14). For each value of θ_0 equal to -90.0, -60.0, -45.0, -30.0, -20.0, -15.0, -10.0, -7.0, -5.0, 5.0, 7.0, 10.0, 15.0, 20.0, 30.0, 45.0, 60.0, 90.0°, the commanded rotations were repeated

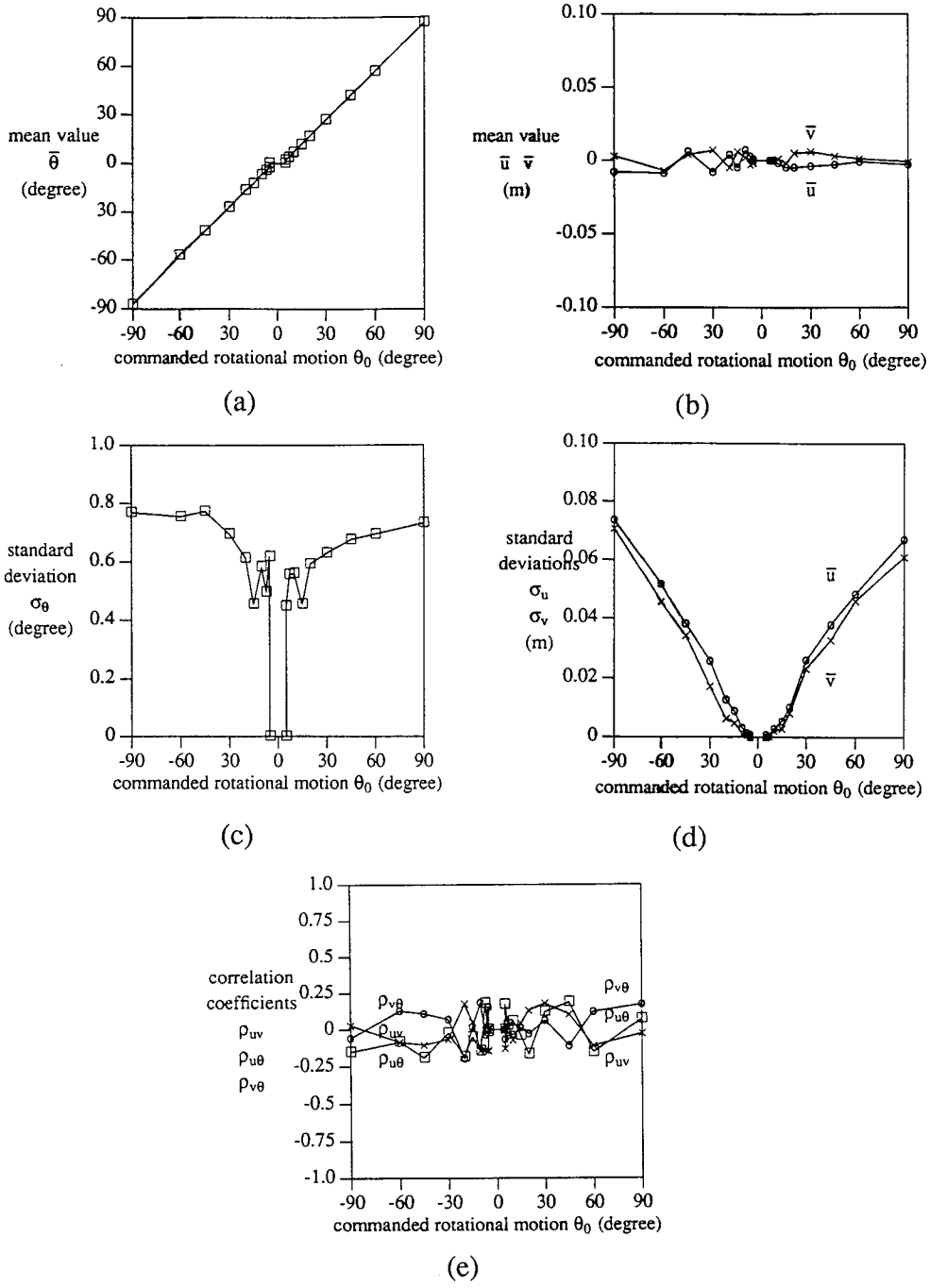


FIG. 50. When the robot is commanded to execute a purely rotational motion through an angle θ_0, the resulting motion is characterized by the parameters θ, u, and v. Shown here are: (a) the mean value $\bar{\theta}$, (b) the mean values $\bar{u}$ and $\bar{v}$, (c) the standard deviation σ_θ, (d) the standard deviations σ_u and σ_v, and (e) the correlation coefficients ρ_{uv}, $\rho_{u\theta}$, and $\rho_{v\theta}$.

20 times and the means $\bar{u}$, $\bar{v}$, $\bar{\theta}$, the standard deviations σ_u, σ_v, σ_θ, and the correlation coefficients ρ_{uv}, $\rho_{u\theta}$, and $\rho_{v\theta}$ computed; these results are plotted in Figs. 50a–50e.[26] The reader will notice in Fig. 50a a small horizontal jog in the middle of the plot for the mean value $\bar{\theta}$. What that jog shows is that the robot is incapable of rotations when the commanded rotation θ_0 is of value 5° or less on the linoleum floor of our hallways.[27]

[26] Compared to the set of commanded translations for the study in Fig. 49, the set of commanded rotations for the study in Fig. 50 is larger. The larger set was made necessary by the nonsmoothness of the displayed curves in the vicinity of $\theta_0 = 0$.

[27] Therefore, when very small rotations are called for during actual navigation, it becomes necessary to execute two opposite rotations through sufficiently large angles such that the difference of the two rotations is the desired small-angle rotation.

As expected, the mean translations, $\bar{u}$ and $\bar{v}$, for any commanded rotation are approximately zero (Fig. 50b). However, as shown in Fig. 50d, the standard deviations σ_u and σ_v can be significant, especially for larger values of commanded rotations. To impress on the reader the largeness of this standard deviation, with a high probability the robot center will get displaced by as much as 7 cm when the commanded rotation is close to 90°. *Therefore, large rotations are capable of introducing significant uncertainties in the position of the robot.*

9.3. A Study of the Accuracy of Self-Location

We now report the results of a study we undertook to determine the accuracy of the Kalman filter-based self-location procedure of Section 7. We placed the robot at a number of different but known positions. The robot was also provided with a covariance matrix for the position uncertainty that was typical of what it would encounter during navigation. The robot was then asked to self-locate by taking a camera image and comparing the image with the expectation map that was rendered by assuming the robot was located at its mean position. More specifically, the robot was told that its mean position was

$$\bar{\mathbf{p}} = [0 \text{ m}, \quad 0 \text{ m}, \quad 0°]^T \tag{9.1}$$

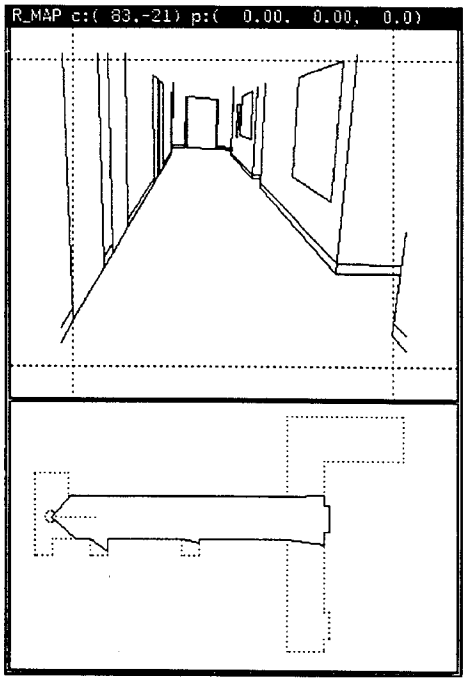


FIG. 51. An expectation map from the mean position within the uncertainty circle shown in the lower figure. Also shown in the lower figure is the visible_space as defined in Section 5.

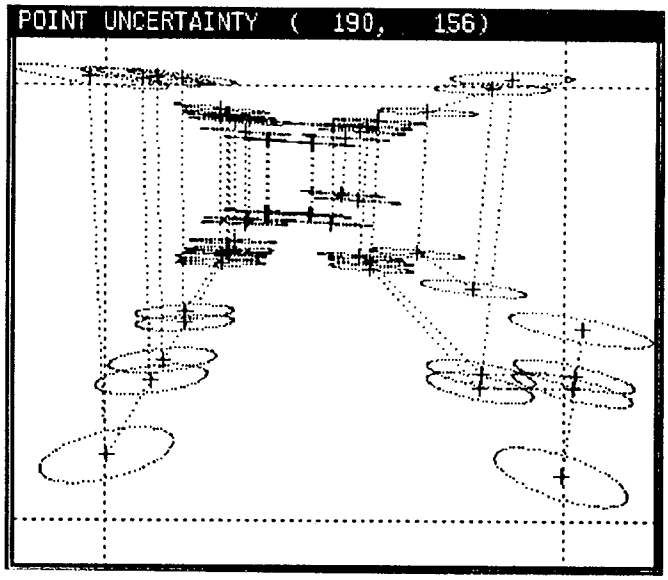


FIG. 52. The uncertainty ellipses for the vertices of the edges in the expectation map of Fig. 51. These ellipses correspond to one unit Mahalanobis distance.

and that the covariance matrix associated with the position vector was

$$\Sigma = \begin{bmatrix} (0.25 \text{ m})^2 & 0 & 0 \\ 0 & (0.25 \text{ m})^2 & 0 \\ 0 & 0 & (5°)^2 \end{bmatrix}. \tag{9.2}$$

The robot was now moved to the positions created by sampling p_x and p_y from the sets $\{-0.25 \text{ m}, 0, +0.25 \text{ m}\}$ and ϕ from the set $\{-5°, 0, +5°\}$, which gave us a total of 27 positions. It follows from the covariance matrix of Eq. (9.2) that in this study the maximum displacements in the robot location and orientation are right at one unit of Mahanalobis distance. The rendered expectation map from the mean position, given by Eq. (9.1), is shown in Fig. 51. The uncertainty regions for the various landmarks in the expectation map are shown in Fig. 52 for the case of image space and in Fig. 53 for the case of Hough space. These uncertainty regions correspond to a unit Mahanalobis distance. (Note however that, as explained in Section 6, two units of Mahanalobis distance are used for the detection of image lines corresponding to any particular model line.) Shown in Fig. 54 is the image taken from one of the 27 positions. To see the difference between the expectation map, rendered by assuming that the robot is at the mean position within its uncertainty ellipse, and the camera image taken from where the robot actually happens to be, in Fig. 55 we have superimposed the two. When the uncertainty regions are used to guide the edge detector, the output of the edge detector and also the camera image itself from one of the 27 positions

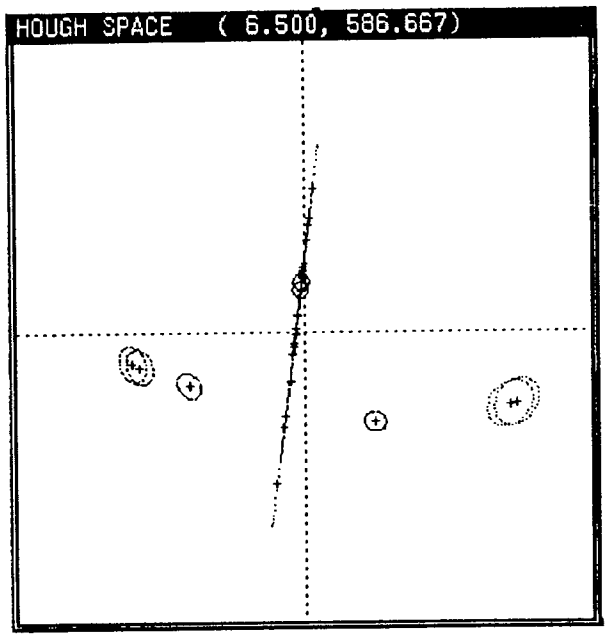


FIG. 53. The Hough space uncertainty for the lines in the expectation map of Fig. 51. These ellipses correspond to one unit of Mahalanobis distance.

of the robot are shown in Fig. 56. The actual position vector for the position of the robot from where the camera image of Fig. 56 was taken was

$$\mathbf{p} = [0.245\ m,\ -0.276\ \text{m},\ -6.0°]^T. \qquad (9.3)$$

Figure 57 shows the projection of the landmarks from the expectation map into the camera image assuming the robot is located at the vector **p** estimated by the Kalman filter-based formalism of Section 7. So, any misregistration errors in Fig. 57 are an immediate reflection of the errors in the estimation of the robot position vector **p**. As the reader will notice there are some discrepancies between the edges of the image and the overlaid edges from the rendering process. For the specific case corresponding to the position where the underlying image of Fig. 54 was taken, the estimated value of position vector **p** was

$$\mathbf{p}^* = [0.237\ \text{m},\ -0.292\ \text{m},\ -6.11°]^T. \qquad (9.4)$$

The convergence to this value of **p** is graphically illustrated by the *xy*-uncertainty ellipses of Fig. 58. Initially, the uncertainty in the *xy*-plane is given by the upper-left 2×2 submatrix of the covariance matrix of Eq. (9.2) and corresponds to the large circle in the figure. Subsequently, each match between a model line and an image

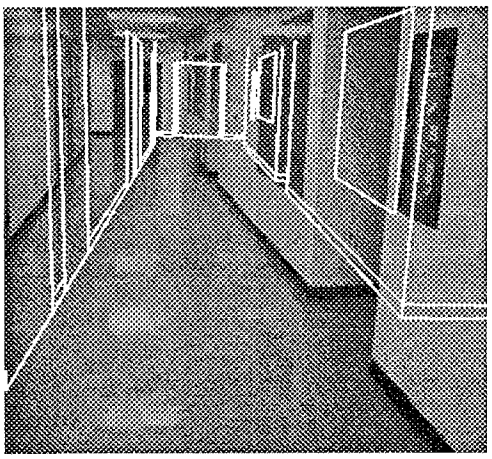


FIG. 55. The expectation map of Fig. 51 is superimposed here on the camera image of Fig. 54 to highlight the misregistration between the two.

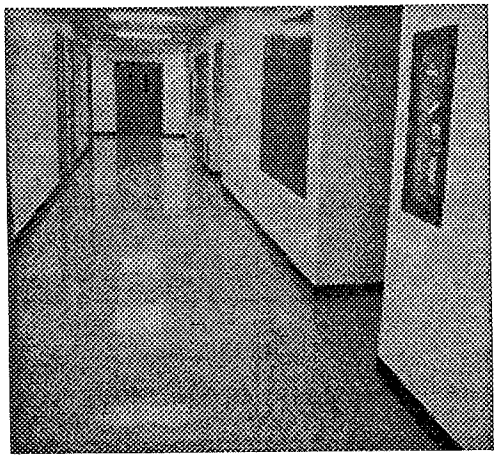


FIG. 54. Camera image taken by the robot from one of the 27 positions around the mean position.

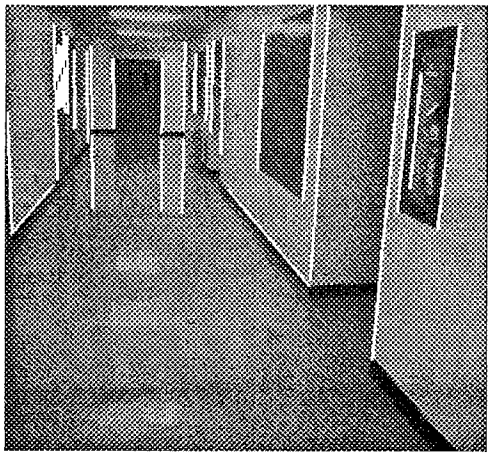


FIG. 56. Detected image lines are superimposed in the original image.

FIG. 57. Matched model lines are projected back into the camera image. For this projection process we use the estimated mean position of the robot. How accurately the projected lines coincide with the corresponding edges in the image is one measure of the accuracy of the method for calculating robot position.

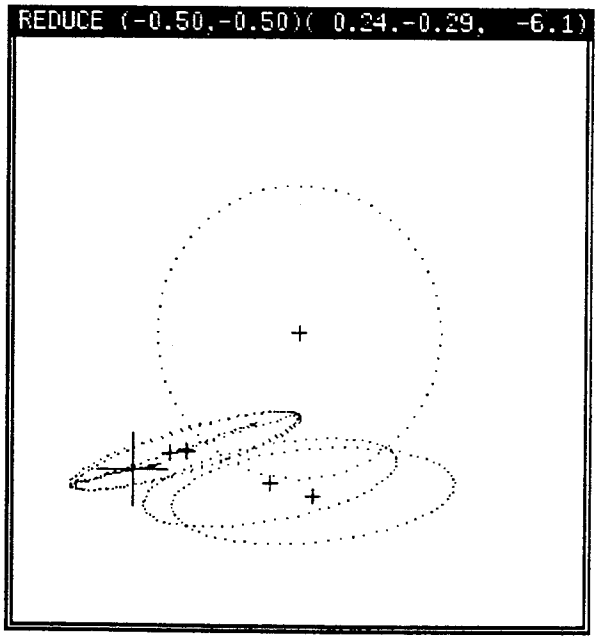


FIG. 58. The uncertainty reduction process by the Kalman filter. The initial uncertainty in the position of the robot is represented by the large circle. Each smaller ellipse represents the updated robot position uncertainty after each match between a model line and an image line.

line reduced the robot position uncertainty. Fig. 58 shows this reduction sequentially along the path finally selected in the search space of Fig. 34. The final uncertainty in the robot position in this case given by

$$\Sigma_p = \begin{bmatrix} 0.000788 & -0.000369 & 0.004560 \\ -0.000369 & 0.002567 & -0.001665 \\ 0.004560 & -0.001665 & 0.029997 \end{bmatrix}. \quad (9.5)$$

We summarize the results of the 27 experiments. The following statistics show the performance of the self-location procedure executed for this set of experiments.

Average number of landmarks:	29.7
Average error in position:	2.0 cm
Average error in orientation:	0.16°
Maximum error in position:	7.8 cm
Maximum error in orientation:	1.50°
Minimum execution time:	24.4 s
Average execution time:	33.2 s
Maximum execution time:	48.5 s

The time spans are measured for the whole processing including image acquisition, expectation map making, landmark detection in the image, and optimal correspondence-finding. The error associated with the position estimation is sufficiently small for navigation. The time required for the self-location procedure is approximately 30 s. This performance provides the speed of 8 m per minute for the indoor navigation.

9.4. Navigation in a Complex Environment

One night when our janitor was cleaning one of the classrooms and had stacked up the chairs in the hallway, we decided to test the navigation system to get some idea of its robustness (or fragility). The robot was asked to navigate from start position P_{start} to the goal position P_{goal} shown in Fig. 42. The robot was initially taken with joystick control to P_{start} and was also provided with a mean vector and a covariance matrix to characterize the positional uncertainty at P_{start}. The global path planned by the robot is the same as before and is shown in Fig. 46.

To get a fix on its precise location at P_{start}, the robot first rendered an image that is shown in Fig. 59a; this rendering was made assuming that the robot was at the initially supplied mean position. The camera image from what is nominally P_{start} is shown in Fig. 59b. The robot calculated the uncertainties associated with each of the expectation map landmarks and used them to guide the landmark detector, whose output is shown in Fig. 59c. The process of matching the landmarks in the expectation map with the extracted landmarks in Fig. 59c resulted in an improved estimate for the position of the robot. The projection of the landmarks into the camera plane using the new estimated position of the robot resulted in the overlay shown in Fig. 59d. The robot then replanned its path to the goal position and started to execute translational and rotational motions along this path

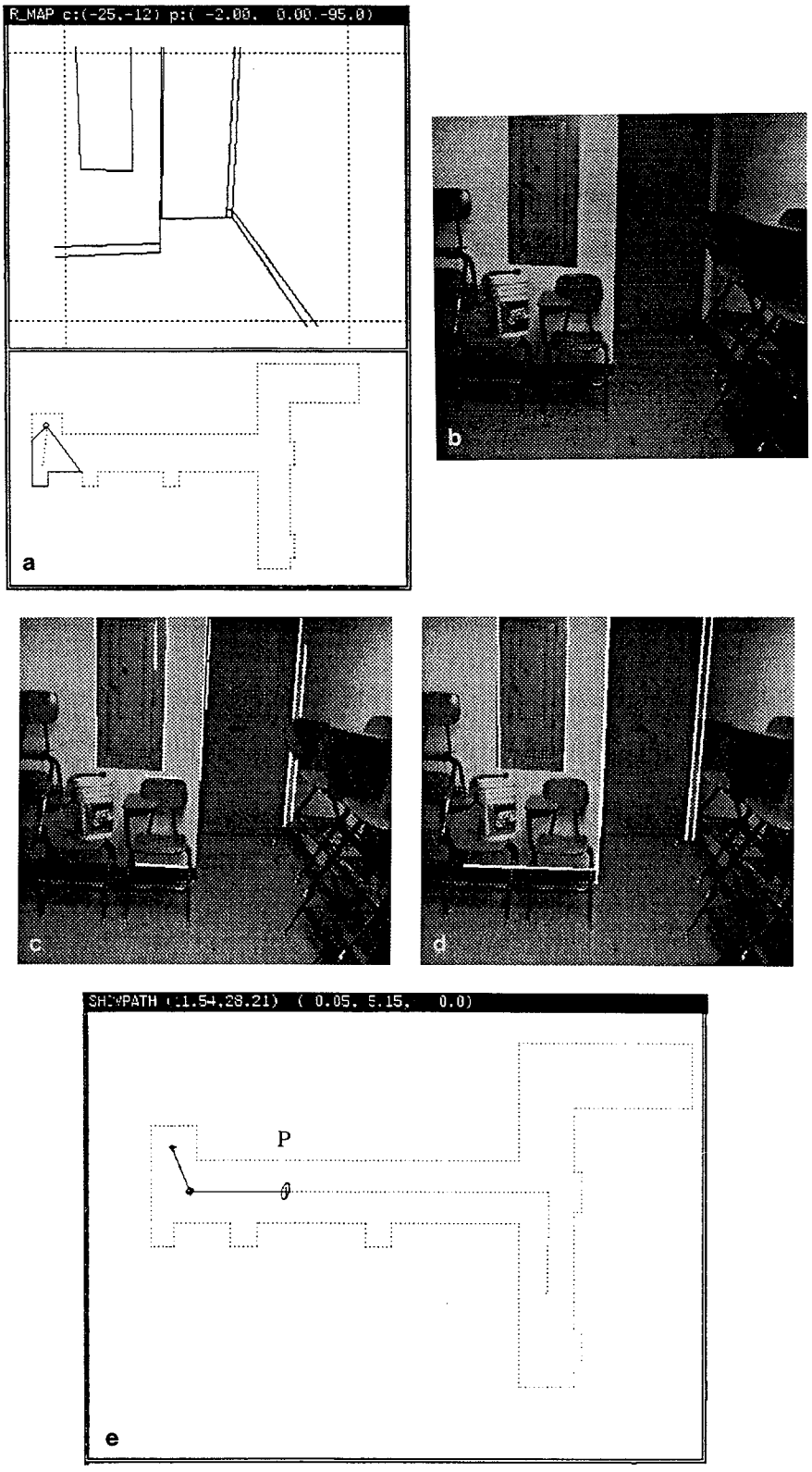


FIG. 59. This sequence of pictures, (a) through (e), demonstrates the result of the self-location experiment at the initial position for the navigation experiment under discussion. (a) An expectation map rendered by assuming that the robot is located at the mean of its uncertainty ellipse. The thick lines in the lower part of (a) indicate the visible space. (b) Camera image taken by the robot. (c) Lines extracted from the camera image are shown superimposed on the image. Note that the lines are only extracted within appropriate uncertainty regions of the image. (d) Using the newly estimated position of the robot, the hallway lines are projected back into the camera image to demonstrate the accuracy of position estimation. The robot then executes the motions shown in (e). At the terminus of this motion, Perception Planner determines that the accumulated uncertainty in the position of the robot, as depicted by the ellipse shown at the terminal point, exceeds the acceptable threshold. The robot is therefore brought to a stop for the next exercise in self-location.

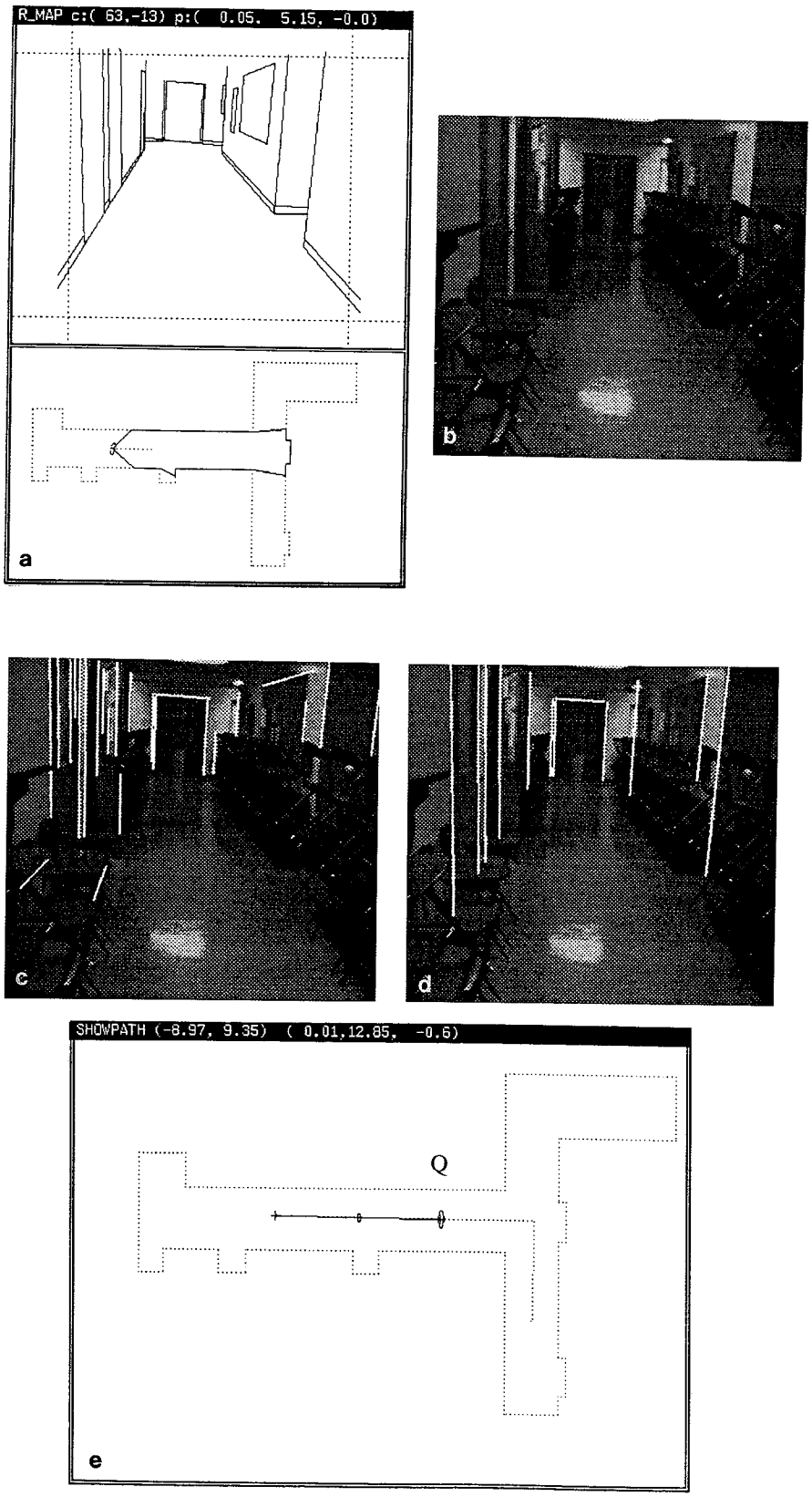


FIG. 60. The sequence of figures, (a) through (e), demonstrates the self-location experiment that is carried out at point P shown in Fig. 59e. Plates (a) through (d) correspond to similarly labeled plates in Fig. 59. Subsequently, the robot executes the motions shown in (e). Perception Planner keeps track of the growth in position uncertainty during these motions and brings the robot to a halt at point Q shown in (e).

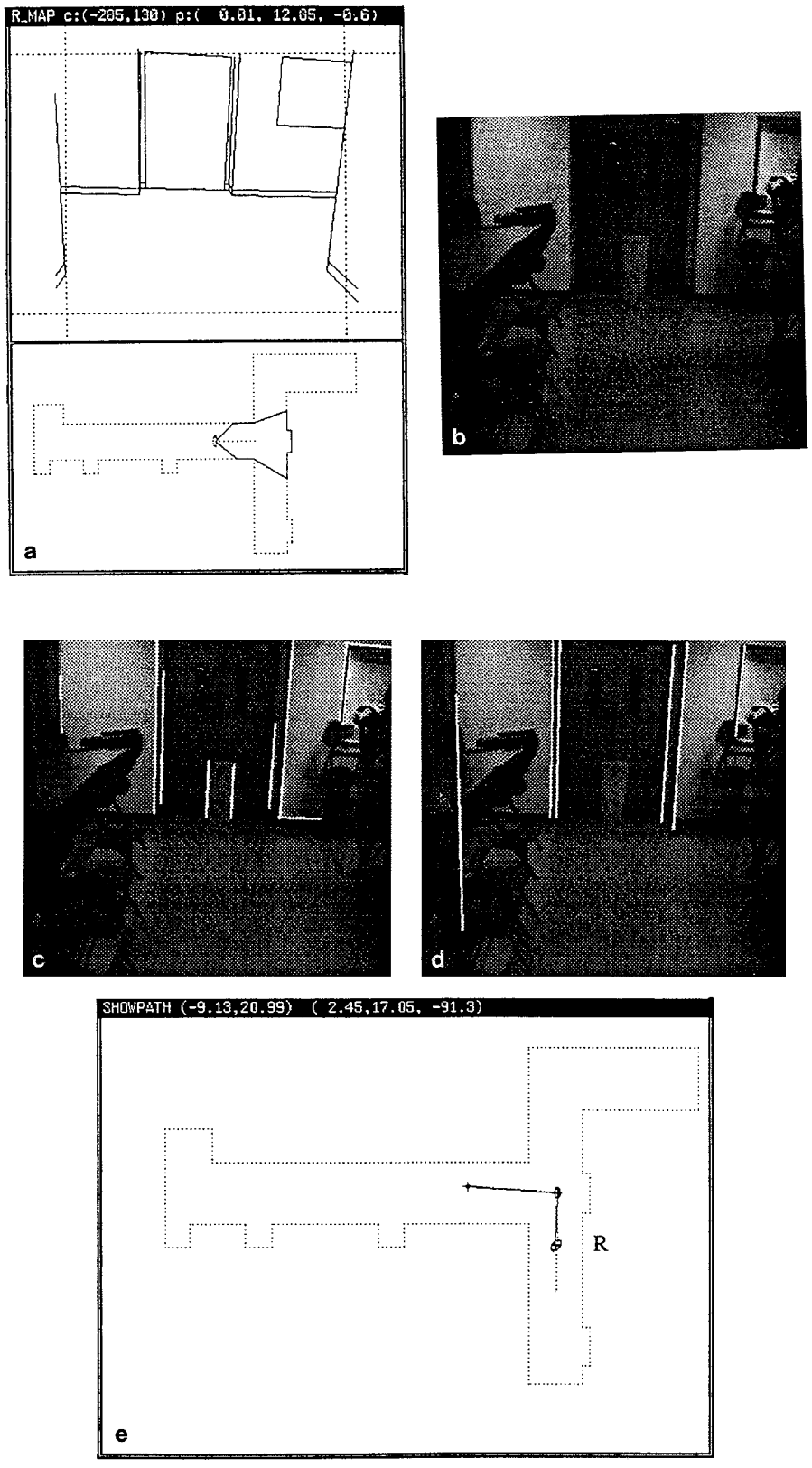


FIG. 61. The sequence of figures, (a) through (e), demonstrates the self-location experiment that is carried out at point Q shown in Fig. 60e. Plates (a) through (d) correspond to similarly labeled plates in Fig. 59. Subsequently, the robot executes the motions shown in (e) here. Perception Planner keeps track of the growth in position uncertainty during these motions and brings the robot to a halt at point R shown in (e).

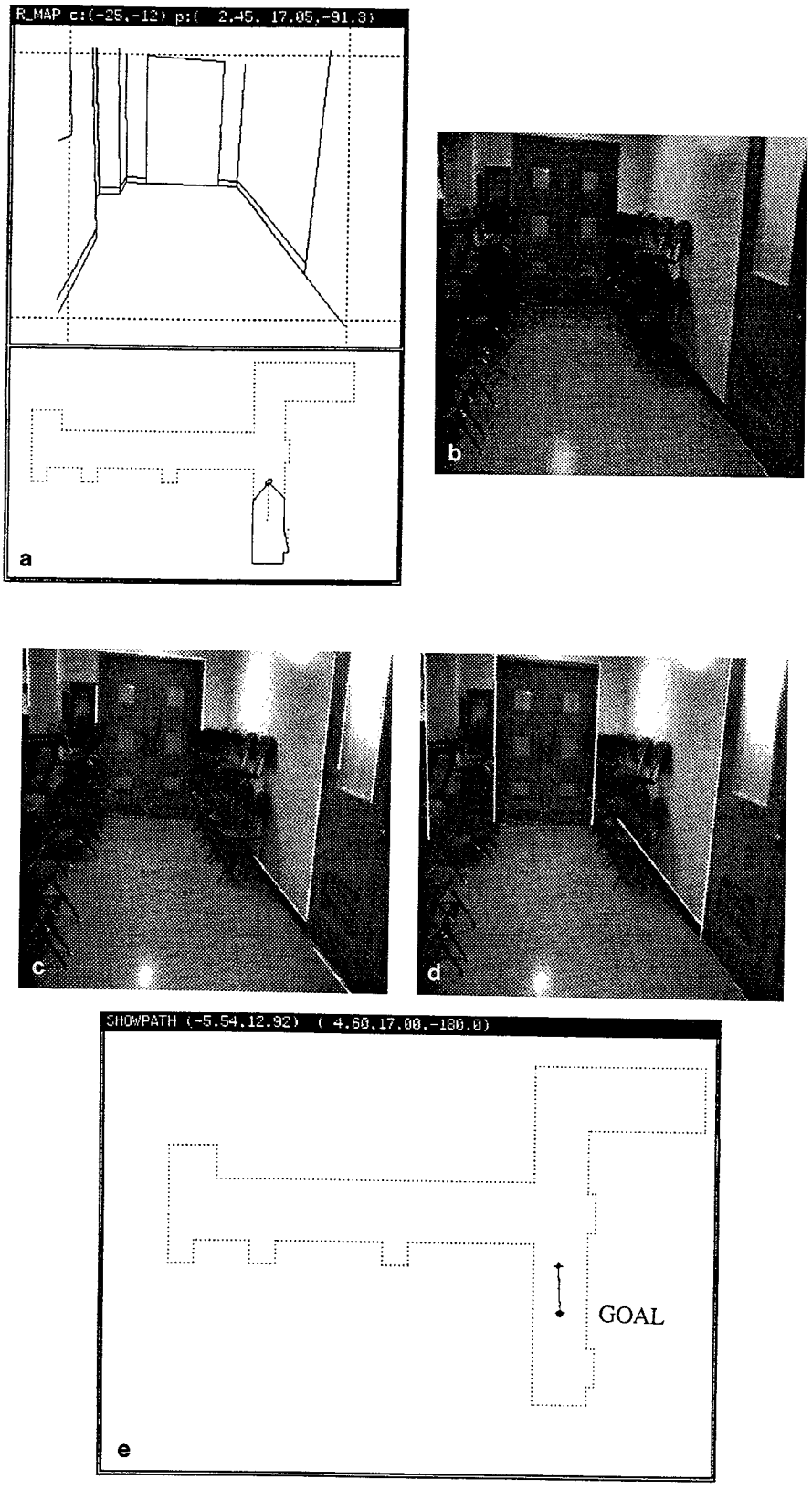


FIG. 62. The sequence of figures, (a) through (e), demonstrates the self-location experiment that is carried out at point *R* shown in Fig. 61e. Plates (a) through (d) correspond to similarly labeled plates in Fig. 59. Subsequently, the robot executes the motions shown in (e) here. Perception Planner keeps track of the growth in position uncertainty during these motions and brings the robot to a halt at point GOAL shown in (e).

while keeping track of the growing uncertainties in its position, as discussed in Section 8. Before arriving at the goal position, the robot stopped at the locations marked P, Q, and, R shown in Figs. 59e, 60e, and 61e, respectively. The expectation map, the camera image together with the extracted features, and the overlay of the landmarks projected onto the camera image using Kalman filter-supplied estimates of robot position are shown in Figs. 60, 61, and 62, respectively, for the self-location done at positions P, Q, and R.

The upshot is that the robot had no difficulty whatsoever with the clutter and occlusion generated by the stacked up chairs in the hallway. We believe that this demonstration makes our formalism the strongest formalism presented so far for indoor mobile-robot navigation using model-based vision.

Because it is difficult to do so, we have not shown any experimental results in navigation that take collision-avoidance into account. Suffice it to say that unless the obstacles create large occlusions in the camera images, the performance is not seriously degraded.

10. CONCLUSIONS

In this paper, we presented a new method for fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. We first discussed a framework for the representation and transformation of uncertainties. We modeled the position uncertainty of the robot and then derived the landmark projection uncertainty and the Hough space uncertainty which was then used for the efficient detection of the landmarks in the image. We then developed a simplified model representation for the hallways that lends itself to fast extraction of the expectation maps as the robot navigates down the hallways. Despite its simplicity, especially so in relation to commercial CAD systems, the data structures used by us for the hallway models retained all the geometric information that was needed for vision processing. We next discussed the subject of landmark detection and presented a new technique for a quick extraction of vertical lines in an image. This technique was based on our derivation that in a perspective image all the vertical world lines would have the same vanishing point regardless of the position of the robot, assuming that the robot was navigating on a flat floor. Subsequently, we presented a formalism that allows landmarks to be matched to image features in a Kalman filter-based sequential scheme in which each match between a landmark and an image feature is used both to update the position of the robot and to estimate the new bounds on the various uncertainties. The paper then went into global path planning and perception planning, both being necessary prerequisites to navigation in the presence of stationary and moving obstacles. Finally, we presented experimental results which demonstrated the power of our fast navigation techniques.

APPENDIX

Formulas for the Derivatives Used in Kalman Filtering

In this appendix, we show the derivatives used for the Kalman filtering formulas of Section 7. Equations (7.26) through (7.33) use two types of derivatives: $\partial\mathbf{f}/\partial\mathbf{z}$ and $\partial\mathbf{f}/\partial\mathbf{p}$. The former consists of a 2×2 matrix, and the latter of a 2×3 matrix.

The derivative $\partial\mathbf{f}/\partial\mathbf{z}$ involves a partial differentiation with respect to the observation vector $\mathbf{z}$. As the reader will recall, the observation vector consists of two random variables, ρ and γ. The derivative is given by

$$\frac{\partial \mathbf{f}}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial f_1}{\partial \rho} & \frac{\partial f_1}{\partial \gamma} \\ \frac{\partial f_2}{\partial \rho} & \frac{\partial f_2}{\partial \gamma} \end{bmatrix}, \tag{A.1}$$

where

$$\frac{\partial f_1}{\partial \gamma} = [-\sin\gamma \quad \cos\gamma \quad 0]\, T\, H \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \tag{A.2}$$

$$\frac{\partial f_2}{\partial \gamma} = [-\sin\gamma \quad \cos\gamma \quad 0]\, T\, H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{A.3}$$

$$\frac{\partial f_1}{\partial \rho} = [0 \quad 0 \quad -1]\, T\, H \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \tag{A.4}$$

$$\frac{\partial f_2}{\partial \rho} = [0 \quad 0 \quad -1]\, T\, H \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}. \tag{A.5}$$

The derivative $\partial\mathbf{f}/\partial\mathbf{p}$ involves partial differentiation with respect to position vector $\mathbf{p}$, which consists of three

random variables p_x, p_y, and ϕ. The matrix expansion of the derivative is

$$\frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial f_1}{\partial p_x} & \frac{\partial f_1}{\partial p_y} & \frac{\partial f_1}{\partial \phi} \\ \frac{\partial f_2}{\partial p_x} & \frac{\partial f_2}{\partial p_y} & \frac{\partial f_2}{\partial \phi} \end{bmatrix}, \tag{A.6}$$

where

$$\frac{\partial f_1}{\partial p_x} = [\cos\gamma \ \sin\gamma \ -\rho]\, T \begin{bmatrix} 0 & 0 & 0 & -\cos\phi \\ 0 & 0 & 0 & \sin\phi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = 0 \tag{A.7}$$

$$\frac{\partial f_2}{\partial p_x} = [\cos\gamma \ \sin\gamma \ -\rho]\, T \begin{bmatrix} 0 & 0 & 0 & -\cos\phi \\ 0 & 0 & 0 & \sin\phi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{A.8}$$

$$\frac{\partial f_1}{\partial p_y} = [\cos\gamma \ \sin\gamma \ -\rho]\, T \begin{bmatrix} 0 & 0 & 0 & -\sin\phi \\ 0 & 0 & 0 & -\cos\phi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = 0 \tag{A.9}$$

$$\frac{\partial f_2}{\partial p_y} = [\cos\gamma \ \sin\gamma \ -\rho]\, T \begin{bmatrix} 0 & 0 & 0 & -\sin\phi \\ 0 & 0 & 0 & -\cos\phi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{A.10}$$

$$\frac{\partial f_1}{\partial \phi} = [\cos\gamma \quad \sin\gamma \ -\rho]\, T \begin{bmatrix} -\sin\phi & \cos\phi & 0 & p_x\sin\phi - p_y\cos\phi \\ -\cos\phi & -\sin\phi & 0 & p_x\cos\phi + p_y\sin\phi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \tag{A.11}$$

$$\frac{\partial f_2}{\partial \phi} = [\cos\gamma \quad \sin\gamma \ -\rho]\, T \begin{bmatrix} -\sin\phi & \cos\phi & 0 & p_x\sin\phi - p_y\cos\phi \\ -\cos\phi & -\sin\phi & 0 & p_x\cos\phi + p_y\sin\phi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}. \tag{A.12}$$

REFERENCES

[AhoHop74] A. V. Aho, J. E. Hopcraft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, MA, 1974.

[AndKak88] K. M. Andress and A. C. Kak, Evidence accumulation and flow of control in a hierarchical spatial reasoning system, *AI Magazine* **9**(2), 1988.

[AyaFau89] N. Ayache and O. D. Faugeras, Maintaining representations of the environment of a mobile robot, *IEEE Trans. Robotics Automation* **5**(6), 1989, 804–819.

[Bal81] D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recog.* **13**(2), 1981, 111–122.

[Bal89] D. H. Ballard, Reference frames for animate vision, in *Proceedings, Eleventh International Joint Conference on Artificial Intelligence*, pp. 1635–1641.

[BoyKak86] K. L. Boyer and A. C. Kak, *Symbolic Stereo from Structural Descriptions*, Purdue University Technical Report, TR-EE 86-12, March 1986.

[BoyKak88] K. L. Boyer and A. C. Kak, Structural stereopsis for 3-D vision, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-10**(2), March, 1988, 144–166.

[Bro85] R. A. Brooks, Visual map making for a mobile robot, in *Proceedings, 1985 IEEE International Conference on Robotics and Automation,* Vol. 1, No. 4, pp. 29–68.

[Bro86] R. A. Brooks, A Robust Layered Control System for a Mobile Robot, *IEEE J. Robotics Automation* **RA-2**(1), 1986, 14–23.

[ChaLau85] R. Chatila and J-P. Laumond, Position referencing and consistent world modeling for mobile robots, in *Proceedings, 1985 IEEE International Conference on Robotics and Automation,* pp. 138–145.

[Con87] J. Connell, Creature design with the Supsumption Architecture, in *Proceedings, International Joint Conference on Artificial Intelligence*, 1987, pp. 1124–1126.

[Cro85] J. L. Crowley, Navigation for an intelligent mobile robot, *IEEE J. Robotics Automation*, **RA-1**(1), March 1985, 31–41.

[DavKun87] L. Davis and T. R. Kunshner, Vision-based navigation: A status report, in *Proceedings, Image Understanding Workshop, February 1987,* pp. 153–169.

[Dur87] H. F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems*, Kluwer, Boston, 1987.

[FenHan90] C. Fennema, A. Hansen, E. Riseman, J. R. Beveridge, and R. Kumar, Model-directed mobile robot navigation, *IEEE Trans. Systems Man Cybernet.* **20**(6), November/December 1990, 1352–1369.

[Gat90] E. Gat, M. G. Slack, R. J. Firby, and D. P. Miller, Planning for execution monitoring on a planetary rover, in *Proceedings, 1990 IEEE International Conference on Robotics and Automation,* pp. 20–25.

[GirSob79] G. Giralt, R. Sobek, and R. Chatila, A multi-level planning and navigation system for a mobile robot; A first approach to Hilare, in *Proceedings, Sixth International Join Conference on Artificial Intelligence*, 1979, pp. 335–337.

[GotKak91a] S. N. Gottschlich and A. C. Kak, Dealing with uncertainties in CAD-based assembly motion planning, in *Proceedings, Ninth National Conference on Artificial Intelligence*, pp. 646–652.

[GotKak91b] S. N. Gottschlich and A. C. Kak, Motion planning for assembly mating operations, in *Proceedings, 1991 IEEE International Conference on Robotics and Automation,* pp. 1956–1963.

[Gri90] W. E. L. Grimson, T. Lozano-Perez, and D. P. Huttenlocher, *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, MA, 1990.

[HwaAhu88] Y. K. Hwang and N. Ahuja, Path planning using a potential field representation, UILU-ENG-88-2251, Coordinated Science Laboratory, College of Engineering, University of Illinois at Urbana–Champaign, 1988.

[Jaz70] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

[KakAnd89] A. C. Kak, K. M. Andress, C. Lopez-Abadia, M. S. Carroll, and J. R. Lewis, Hierarchical evidence accumulation in the PSEIKI system and experiments in model-driven mobile robot navigation, in *Uncertainty in Artificial Intelligence* (M. Henrion, R. Shachter, L. N. Kanal, and J. Lemmer, Eds.), pp. 353–369, Elsevier, North-Holland, 1990.

[KakNof85] A. C. Kak, Depth perception for robots, in *Handbook of Industrial Robotics*, (S. Y. Nof, Ed), pp. 272–319, Wiley, New York, 1985.

[Kha86] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robotics Res.* **5**(1), 1986, 90–98.

[Koi89] A. J. Koivo, *Fundamentals for Control of Robotic Manipulators*, Wiley, New York, 1989.

[KriTri89] D. J. Kriegman, E. Triendl, and T. O. Binford, Stereo vision and navigation in buildings for mobile robots, *IEEE Trans. Robotics Automation* **5**(6), 1989, 792–803.

[KumHan89] R. Kumar and A. Hansen, Robust estimation of camera location and orientation from noisy data with outliers, COINS TR 89-120, Computer and Information Science, University of Massachusetts at Amherst, December 1989.

[Lat91] J. Latombe, *Robot Motion Planning*, Kluwer, Boston, 1991.

[LenTsa88] R. K. Lenz and R. Y. Tsai, Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-10,** 1988, 713–720.

[LiuHua88] Y. Liu, T. S. Huang, and O. D. Faugeras, Determination of camera location from 2D to 3D line and point correspondences, in *IEEE International Conference on Computer Vision and Pattern Recognition*, 1988, pp. 82–88.

[LopKak89] C. Lopez-Abadia and A. C. Kak, *Vision-Guided Mobile Robot Navigation*, Purdue University Technical Report, TR-EE 89-34, Electrical Engineering, Purdue University, West Lafayette, IN, 1989.

[LozWes79] T. Lozano-Perez and M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Commun. ACM* **22**(10), October 1979, 560–570.

[MarBir81] H. A. Martins, J. R. Birk, and R. B. Kelly, Camera models based on data from two calibration planes, *Comput. Graphics Image Process.* **17,** 1981, 173–180.

[Mor81] H. P. Moravec, *Robot Rover Visual Navigation*, UMI Research Press, 1981.

[Mor83] H. P. Moravec, The Stanford Cart and the CMU Rover, *Proc. IEEE* **71**(7), July 1983, 872–884.

[Mor85] M. E. Mortenson, *Geometric Modeling*, Wiley Sons, New York, 1985.

[PreSha85] F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.

[RosKak82] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vol. 2, 2nd ed., Academic Press, New York, 1982.

[ShaHar81] L. G. Shapiro and R. M. Haralick, Structural descriptions and inexact matching, *IEEE Trans. Pattern Anal. Mach. Intelligence* **PAMI-3,** 1981, 504–519.

[SmiChe86] R. C. Smith and P. Cheeseman, On the representation and estimation of spatial uncertainty, *Int. J. Robotics Res.* **5**(4), 1986, 56–68.

[Sug88] K. Sugihara, Some location problems for robot navigation using a single camera, *Comput. Vison Graphics Image Process.* **42,** 1988, 112–129.

[ThoSha87] C. Thorpe, S. Shafer, and T. Kanade, Vision and navigation for the Carnegie Mellon Navlab, in *Proceedings, Image Understand Workshop*, February 1987, pp. 143–152.

[Til80] R. B. Tilove, Set membership classification: A unified approach to geometric intersection problems, *IEEE Trans. Comput.* **C-29**(10), 1980, 874–883.

[Tsa87] R. Y. Tsai, A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE J. Robotics Automation*, **RA-3**(4), 1988, 323–344.

[TsuYut87] T. Tsubouchi and S. Yuta, Map assisted vision system of mobile robots for reckoning in a building environment, in *Proceedings, 1987 IEEE International Conference on Robotics and Automation*, pp. 1978–1984.

[ZheBar91] J. Y. Zheng, M. Barth, and S. Tsuji, Autonomous landmark selection for route recognition by a mobile robot, in *Proceedings, 1991 IEEE International Conference on Robotics and Automation*, pp. 2004–2009.