Proceedings of Workshop on Vision for Robots in IROS'95 Conference, Pittsburgh, PA, 1995, pp.87-96, 1995.

Purdue Experiments in Model-Based Vision for Hallway Navigation

Akio Kosaka and Juiyao Pan
Robot Vision Laboratory
1285 EE Building, Purdue University
West Lafayette, IN 47907-1285, U.S.A.
e-mail: kosaka@ecn.purdue.edu, juiyao@ecn.purdue.edu

Abstract

In the Robot Vision Laboratory of Purdue University, we have developed two robust architectures for vision-guided mobile robot navigation in indoor environments. The first approach, FINALE, enables the robot to autonomously navigate in hallways at speeds around 8m/min through vision-based position-updating, utilizing a geometrical map, model-based reasoning and Kalman filtering. The performance of the navigation is not impaired by the presence of stationary or moving obstacles. The second approach, FUZZY-NAV, performs vision-based navigation in a more human-friendly manner, utilizing simply a topological map and the combination of neural networks and fuzzy logic. This approach also allows the robot to simultaneously navigate and avoid obstacles, both static and dynamic. In this paper, we describe the above two approaches and show experimental results to demonstrate the effectiveness and the robustness of our approaches.*

1 Introduction

In recent years, there has been increasing interest in using robots to perform mundane tasks that require mobility in such environments as hospitals, factory plants, nuclear reactor sites, underwater sites, and battlefields.

Two divergent research areas have emerged in the mobile robotics community. In autonomous land vehicle research, the goal is to develop a vehicle that can navigate at relatively high speeds using sensory feedback in outdoor environments such as fields or roads. These vehicles require massive computational power and powerful sensors in order to adapt their perception and control capabilities to the high speed of motion in complicated outdoor environments.

*This work was supported by the Office of Naval Research under Grant ONR N00014-93-1-0142.

The second area of the research deals with mobile robots which are designed to work in indoor environments or in relatively controlled outdoor environments. As we will show in this paper, such robots provide testbeds for solving research problems in perception, planning and control.

When we attempt to use a mobile robot in such real indoor environments, the robot can never follow pre-planned paths exactly because of motion uncertainties and unexpected obstacles. Therefore, vision-based position updating is an important issue in mobile robotics research. In this paper, we will discuss such issues, focusing mainly on vision-based updating of robot positions and motion trajectories. First we will briefly review research in vision-based navigation. We will then present two approaches to vision-based navigation which have been developed in the Robot Vision Laboratory at Purdue University, considering the following two situations: (1) when a geometric map of the environment is provided, and (2) when only a topological map is provided to the robot. We will also show experimental results to demonstrate the effectiveness of these approaches.

2 Related Work in Vision-Guided Mobile Robot Navigation

In vision-guided mobile robotics, three major research areas have been paid much attention: map-building, map-based navigation, and tracking-based navigation.

When a robot is located in an unknown environment, the robot needs to explore the environment in order to build its map. Such maps are then used for subsequent navigation tasks. Map-building by mobile robots typically utilizes a sequence of images taken by single or stereo cameras [3, 4, 10, 13].

For map-based navigation, Sugihara [19] and Atiya and Hager [2] developed methods for the robot pose estimation when a 2D landmark map is given to the robot without any information about the robot position and orientation.

Another approach to map-based navigation is the use of a 3D CAD model of environments to estimate the robot position and orientation. In this approach, an approximate location of the robot is given to the robot in a partially known environment. In order to perform an automatic landmark selection and an efficient landmark recognition, Fennema, Hansen, et al. [6], Tsubouchi and Yuta [20], Kak and Andress [7], Kosaka and Kak [8], Christensen, et al. [5] use a simple 3D CAD model of the environment to generate an expectation view and to compare it with an actual image taken by the robot for the robot pose estimation.

Tracking-based navigation keeps track of static landmarks (targets) in the environment to generate an adequate motion of the robot [1]. Other interesting approaches on vision-guided mobile robot navigation are found in [12, 17].

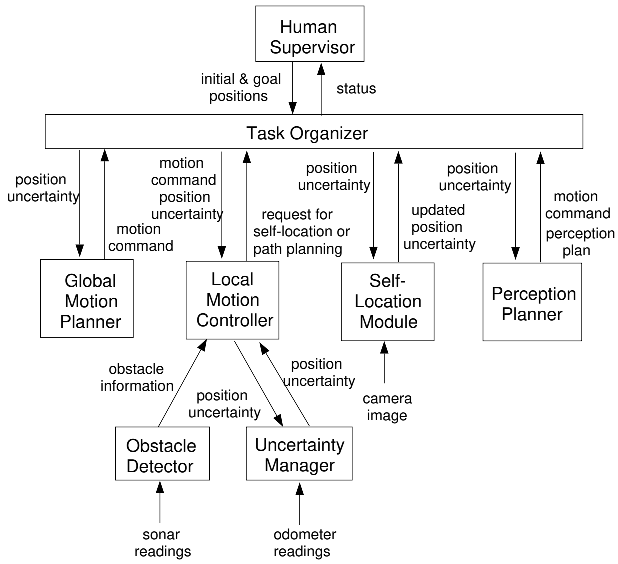


Figure 1: *Overall architecture of FINALE.*

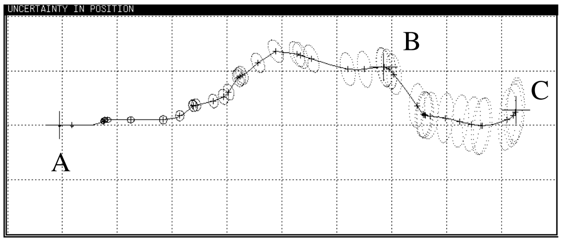
3 FINALE: Geometrical Approach

This section presents our first approach to vision-guided mobile robot navigation using a geometric map of the environment [8, 9]. This approach, the combination of model-based reasoning and Kalman filtering, enables us to design robust reasoning and control architectures for vision-guided mobile robot navigation in indoor environments. The robot is capable of autonomously navigating in hallways at speeds around 8 m/min, using vision for self-location and sonars for collision avoidance. The performance of this robot is not impaired by the presence of stationary or moving objects in the hallways. The robot simply treats everything that is not in its geometrical model base as visual clutters. By maintaining models of uncertainties and their growth during navigation, the robot is able to place bounds on where it should look for important visual landmarks; this gives the robot immunity against visual clutters.

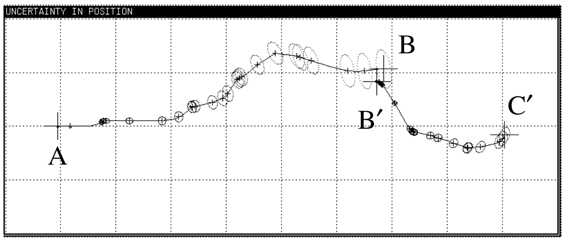
3.1 Architecture of FINALE

Fig. 1 depicts the model-based reasoning and control architecture called *FINALE.* A human supervisor provides a rough estimate of the initial robot position and the goal position to the robot. Given these initial and goal positions, the Task Organizer organizes the overall control of navigation on a real-time basis by monitoring status reports from other modules in the system. The Task Organizer first requests the Global Motion Planner to create a global path from the current position to the goal position. When the Global Motion Planner returns the path consisting of a sequence of *linear translational and rotational moves*, the Task Organizer requests the Local Motion Controller to navigate the robot on the planned path, avoid obstacles, and monitor the growth of uncertainty in the robot position. The Obstacle Detector continuously uses an ultrasonic sonar ring mounted in front of the robot to acquire the range information about obstacles. When an obstacle is detected, the Local Motion Controller reduces the speed of the motion and finds a free space for motion so that the robot can successfully avoid the obstacle.

The Uncertainty Manager continuously monitors the positional uncertainty of the robot every 500 $msec$, in terms of the mean vector and the covariance matrix of the robot position vector as shown in Fig. 2(a). If the positional uncertainty is too large for subsequent motions, the Local Motion Controller issues a self-location request to Task Organizer and stops the motion. Upon receiving the request for self-location, the Task Organizer first requests the Perception Planner to determine the best viewing angle to perform self-location based on the number of visible landmarks in a scene expectation map. If the current orientation is not suitable for the self-location exercise, the Perception Planner will produce a motion command that will lead the robot to a better view with a larger number of visible landmarks. Once the robot rotates to the direction determined by the Perception Planner, the Self-Location Module starts a self-location exercise by first obtaining a camera image from a single TV Camera. On detecting that the self-location process has started, the Task Organizer issues a restart motion command to the Local Motion Controller.



(a)



(b)

Figure 2: *Shown here is the growth in the xy-position uncertainty of the robot. Ellipses shown in the figure represent one unit of Mahalanobis distance from the mean positions. (a) The robot starts moving with a small uncertainty at point A. While avoiding obstacles, the robot maneuvering toward point C. The robot temporally stops at point B where a camera image is taken for a self-location exercise. (b) A self-location exercise is being executed while the robot is moving from point B to point C. At point C, the Self-Location Module accomplishes the task, and the new position estimate B' for the point B is obtained. Then the retroactive updating of the robot positions along the path from B to C is carried out using the motion history stored in the Uncertainty Manager. The new position estimates are displayed from B' to C'.*

If the Self-Location Module reports success, the Task Organizer will request the Local Motion Controller to stop motion, and will then send to the Local Motion Controller the updated position uncertainty. Since the robot is no longer at the location where the camera image for self-location was taken, the Local Motion Controller will request the Uncertainty Manager to re-estimate the current position using the motion history stored in the Uncertainty Manager. This process is called the *retroactive updating* of the robot position. (See Fig. 2(b).)

After the self-location exercise and the retroactive updating of position uncertainty are accomplished, the Task Organizer will issue to the Global Motion Planner the request for replanning a path from the currently updated position to the goal position. The robot will then start moving again toward the goal.

The above process will be iterated until the robot reaches the final goal position. The decision of termination is made by the Uncertainty Manager on a real-time basis by monitoring the difference of the coordinates between the current and goal positions.

3.2 Motion Uncertainty

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). We assume that the robot is located in the position (p_x, p_y) in the world coordinate frame with orientation angle ϕ. As the robot travels, its position is defined by the 3D vector $\mathbf{p} = (p_x, p_y, \phi)$. *The problem of the position updating is, therefore, to estimate vector* $\mathbf{p}$ *by visual observations.*

The motion commands issued forth by the Local Motion Controller are either *pure translations* or *pure rotations*. Obviously, due to differential slippage in the wheels and other effects, the actual location and the orientation of the robot will always differ from the commanded location or orientation. We have carried out exhaustive experimental studies on the discrepancies between the commanded motions and the actual executed motions and thus constructed models of dependence of positional uncertainty in response to motion commands. Our experiments corroborate the assumption made earlier by Smith and Cheeseman [18], Kriegman et al. [10], and others that positional errors for a mobile robot can be modeled by Gaussian distributions and can be characterized by second order distributions. Therefore, if $\mathbf{p}$ and $\mathbf{p}'$ denote the position vectors of the robot before and after a commanded motion where $\mathbf{p}$ is a Gaussian random vector with the mean $\bar{\mathbf{p}}$ and the covariance Σ_p, then we may claim that $\mathbf{p}'$ is also a Gaussian random vector with the mean vector $\bar{\mathbf{p}}'$ and the covariance matrix Σ'_p. The relationship between Σ_p and Σ'_p is evidently a function of the length of a commanded translation and the degree of a commanded rotation. From the empirically constructed models of uncertainty growth, it is straightforward to compute Σ'_p from Σ_p. Of course, in reality, the overall motion of the robot will be a sequence of commanded translations and rotations. By invoking the uncertainty propagation formula separately for each commanded translation and each commanded rotation, we can easily construct the uncertainties that would apply at the end of any robot travel.

Fig. 2(a) is a record of growth of uncertainty during an actual run of the robot in which the robot had to maneuver around an obstacle. In this figure, the robot starts moving from point A with a small initial uncertainty. Each uncertainty ellipse indicates the size of uncertainty in the xy position in the world coordinate frame. More precisely, one unit of Mahalanobis distance from the mean position is shown. In the architecture shown in Fig. 1, the Uncertainty Manager keeps track of the uncertainty as the robot responds to the commanded motions from the Local Motion Controller.

3.3 Vision-Based Self-Location

We will briefly explain how the Self-Location Module performs a vision-based self-location with the help of a 3D CAD model of the environment and the prediction of motion uncertainties. Fig. 3 shows the process of our self-location algorithm.

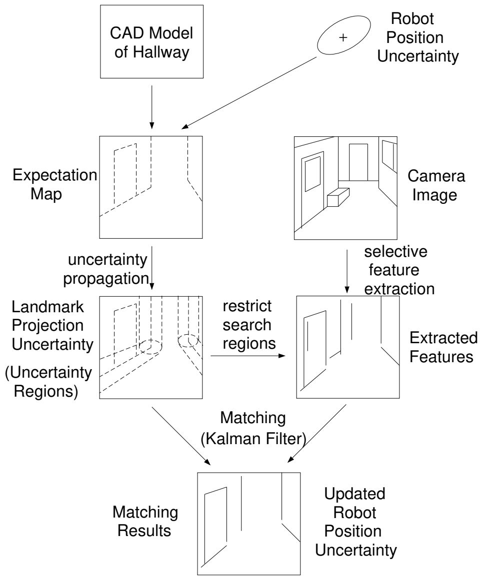


Figure 3: *Process of self-location algorithm.*

(1) Prediction of visible landmarks

Using a CAD model of the environment and an initial estimate of the predicted mean values of the robot position vector, FINALE first generates a scene expectation map (expectation view) from the camera viewpoint. Fig. 4 shows the expectation map with white lines superimposed on an actual image taken by the camera mounted on the robot. From this expectation map, line segments whose lengths in the view are larger than a prespecified threshold are selected as visible model landmarks (features) so as to be easily extracted from an image.

Figure 4: *A camera image taken for self-location. The white lines superimposed here indicate a scene expectation map rendered from a 3D model of the hallway.*

A key element of this vision-based navigation is the propagation of the robot position uncertainty into both the camera image and the feature space in which many of image features used for model matching are easily extracted. In fact, the locational uncertainty of model landmark features in the camera image (expectation map) can also be predicted by the mean vector and the covariance matrix. Such a propagation can be realized by propagating the prediction error covariance Σ_p of the robot position vector $\mathbf{p}$ into both the camera image and the feature space, using the Jacobian matrices of the transformations involved [8].

The uncertainty propagated into the camera image bounds the regions where the model landmarks should exist in the camera image. For example, we are able to define such a region by applying an appropriate threshold to the covariance. We will call this region the *uncertainty region* in the camera image.

Similarly, we can propagate the robot position uncertainty to a feature space such as the Hough space [8]. The Hough space, formed by the Hough transform, is specified by two parameters (ρ, γ) to represent a line in the camera image – the perpendicular distance ρ from the image origin to the line, and the orientation angle γ of the normal vector to the line.

(2) Extraction of image features

For each model landmark feature, image features are extracted *only* from its uncertainty region in the camera image. Since we use the Hough transform to extract image features, this extraction process is facilitated by applying the Hough transform to only uncertainty regions in the Hough space.

To give the readers a sense of the magnitude of the propagated uncertainty, we have shown in Fig. 5 the uncertainty ellipses associated with the endpoints of the edges shown in the superimposed expectation map in Fig. 4. These ellipses correspond to one unit of Mahalanobis distance. We only need to extract line features from the uncertainty regions projected into both the image and the Hough space, which gives FINALE great immunity against visual clutters. Fig. 6 shows extracted image features corresponding to the visible model landmark features in Fig. 5.

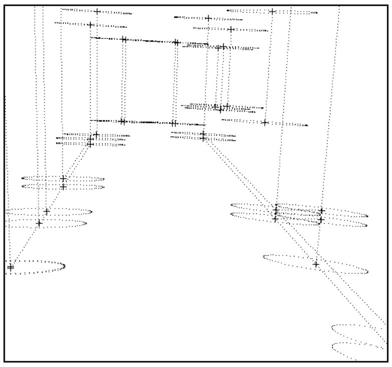


Figure 5: *Ellipses shown represent the uncertainties associated with the endpoints of the linear model features in Fig. 4.*

Figure 6: *Linear features extracted from the camera image are imposed on the image. Note that the image features are only extracted within appropriate uncertainty regions of the image.*

(3) Updating of robot position vector through optimal correspondence search

Due to visual occlusions and background conditions, the line extraction by the Hough transform may extract more than one image feature in the uncertainty region. In order to precisely estimate the robot position vector $\mathbf{p}$, we need to find a correct correspondence between model landmarks and image features. Our algorithm performs an optimal correspondence search using parametric reasoning for model and image feature relations. This algorithm, performing a constrained search, is verified to be robust even in the presence of occlusion and a large displacement of expectations and actual views [8].

A key idea of this constrained search is that a single match between a model landmark and an image feature constrains the robot position uncertainty, updating the statistics of the uncertainty. In practice, we use the Kalman filter to realize this updating. Given a single match of model landmark and image feature, robot position uncertainty represented by the pair of the mean vector and the error covariance matrix is updated to the pair of the new mean vector and the new covariance matrix. *Since this update reduces the robot position uncertainty, it also reduces the uncertainty regions in both image and feature spaces for other model landmarks, and therefore allows subsequent model landmarks to be easily matched to image features in the subsequent search process.* In addition, the new estimate of the robot position vector is indeed obtained as the by-product of the Kalman filter-based updating.

In the actual steps of the search process, the algorithm looks for an optimal correspondence in a branch-and-bound depth-first search mode with appropriate optimality criteria. See more details in [8].

Figure 7: *Matched model features are projected back into the camera image. For this projection process, we use the estimated mean position of the robot to demonstrate the accuracy of the position estimation.*

To show how precisely FINALE can calculate the exact position of the robot, Fig.7 depicts a reprojection into the camera image of those model landmarks that were used by the Kalman filter for finding correspondences with the image features.

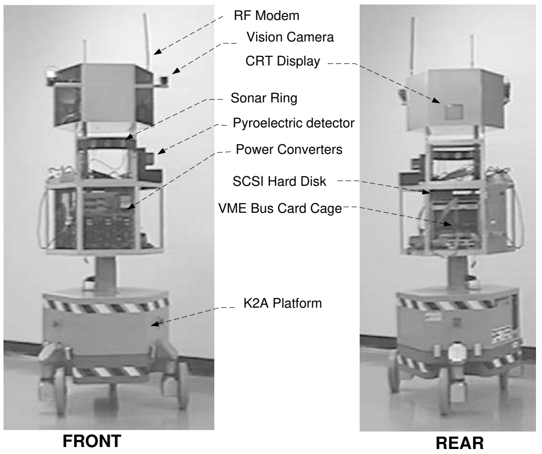


Figure 8: *Shown here are pictures of the front and the rear of our mobile robot.*

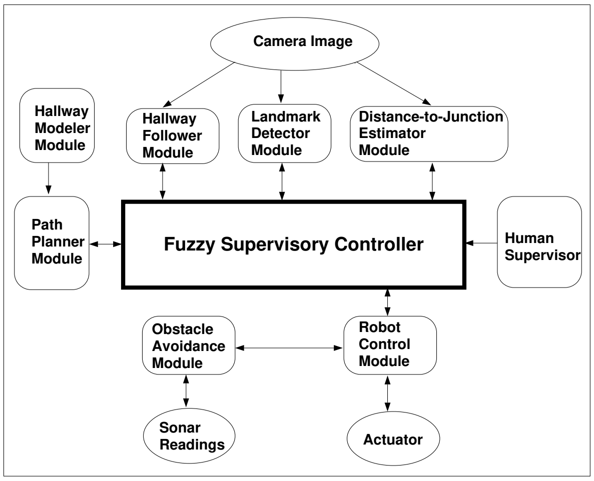


Figure 9: *Architecture of FUZZY-NAV.*

3.4 Experimental Results

Fig. 8 shows our mobile robot which contains a Cybermation K2A platform and a VME-based computing hardware. The VME card cage contains a MC68040 microprocessor based MVME167 single-card computer, an Imaging Technology frame grabber, and a sonar transducer board. A semi-ring of five ultrasonic sonars provides the range information for obstacle avoidance. The MVME167 board runs the VxWorks real-time operating system.

We remark here that the navigation experiments succeeded roughly 90% of the time. This number was obtained by conducting a large number of experiments over a run of 50 m of hallways with two-right angle turn at both ends in the presence of unknown obstacles such as humans and furniture. The self-location experiments showed that the average positional error was 2 cm and the orientational error 0.2 *degree*, while the average time needed for self-location was 30 *seconds* which allowed the robot to navigate roughly at 8 m/min speed.

4 FUZZY-NAV: Topological Approach

We report here another robust vision-based control architecture for indoor mobile-robot navigation named *FUZZY-NAV* [12, 16].† On one hand, this architecture takes advantage of the high-throughput of neural networks for the processing of camera images. On the other, it employs fuzzy logic to deal with the uncertainty in the inferences drawn from the vision data. Note that the architecture we present in this paper allows our robot to simultaneously navigate and avoid obstacles, both static and dynamic.

As indicated in the previous section, the FINALE system is heavily geometrical, in the sense that it requires that a 3D model of the hallways be known.

Our next architecture, FUZZY-NAV, presented in [12, 16], is more human-friendly, in the sense that it used topological models of hallways. From the topological models of space the path planner in FUZZY-NAV outputs a sequence of navigation commands like "*[straight to the second T-junction, turn right, straight to the third door on the left]*." In executing these commands, the robot invokes an ensemble of neural networks for steering control. With the same hardware as for FINALE, the FUZZY-NAV system is able to generate a steering command every three seconds.

4.1 Architecture of FUZZY-NAV

As depicted in Fig.9, the heart of the FUZZY-NAV system is the Fuzzy Supervisory Controller connected to a human supervisor and several peripheral modules.

(1) Hallway Follower

The Hallway Follower consists of two competing neural networks that help the robot stay in the middle of the hallway when no obstacles are present. Fig. 10 shows the structure of the Hallway Follower. Different regions of the Hough transform of the camera image are fed into the input nodes of these neural networks. The two neural networks together have ten output nodes corresponding to the following steering commands: *left-40 left-30 left-20 left-10 go-straight right-10 right-20 right-30 right-40* and *no-decision*.

†FUZZY-NAV is the extension of our previous work called NEURO-NAV in [12].

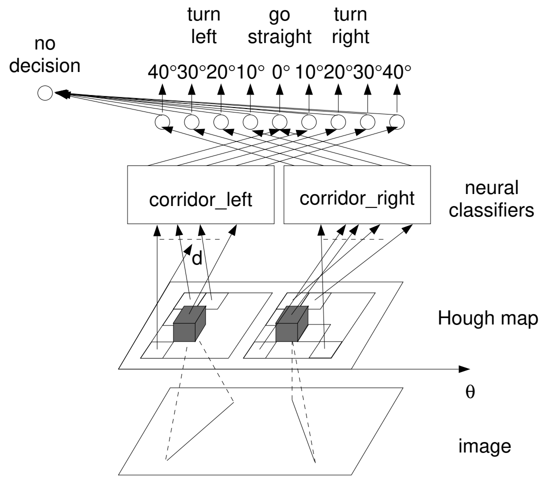


Figure 10: *The structure of the Hallway Follower is shown here. The distinct regions of a Hough map are fed into two different networks for hallway following.*

(2) Distance-to-Junction Estimation Module

The Distance-to-Junction Estimation Module gives the robot a sense of how far it is to a hallway junction. Certain regions of the Hough space representation of the camera image correspond to those straight lines in a hallway that are horizontal but perpendicular to the hallway in which the robot is traveling. By feeding these regions of the Hough space into an appropriately trained neural network, an estimate of the distance to the end of the hallway can be constructed. The output nodes of this network are *far, near, close* and *no-decision*.

(3) Obstacle Avoidance Module

The Obstacle Avoidance Module in Fig.9 consists of a semi-ring of ultrasonic sensors. Through these sensors, the module can estimate both the direction of and the distance to the obstacle. When the distance to the obstacle falls below a certain preset threshold, the Collision Avoidance Module interrupts the CPU and takes over the Robot Control Module. As long as an obstacle can be detected by any of the sonar sensors, the Robot Control Module remains completely under the control of the Obstacle Avoidance Module. When no obstacles are detected, control reverts back to the Fuzzy Supervisory Controller.

(4) Landmark Detector

The Landmark Detector contains neural networks, each specialized for a specific recognition task. One of the main neural networks is for recognizing door frames of closed doors from a certain range of perspectives. This module, obviously of critical importance if our robot is to succeed in simultaneous navigation and object finding, is still undergoing development and refinement in our laboratory.

(5) Path Planner

The Path Planner uses a topological model of the hallways and the information supplied by the human about the desired destination location to spit out a sequence of navigational commands. For example, for the hallway shown in Fig.14, if the destination point is close to Rm170b, the path planner will issue the following commands to the Fuzzy Supervisory Controller:

follow-corridor,
stop-at-junction,
turn-right-at-junction,
follow-corridor,
stop-at-deadend.

4.2 Vision-Based Fuzzy Supervisory Control

It is clear that the Fuzzy Supervisory Controller uses the information received from the Hallway Follower, the Distance-to-Junction Estimator, and the Landmark Detector neural networks to decide how to control the robot. This control must of course be in accordance with the sequence of top-level commands received initially from the Path Planner. The Fuzzy Supervisory Controller actually uses three linguistic variables, *distance-to-junction, turn-angle, distance-to-travel.* Associated with these linguistic variables are a total of sixteen fuzzy terms. The linguistic variable *turn-angle* represents the rotational command for the robot to face straight to the hallway while navigating. The variable *distance-to-junction* represents the estimate of the distance from the robot position to the hallway junction. The linguistic variable *distance-to-travel* stands for the current best estimate of how far the robot should plan on traveling straight barring any encounters with obstacles. While the values of the linguistic variables *turn-angle* and *distance-to-junction* are derived from the vision data by one of the neural networks, the linguistic variable *distance-to-travel* is given a value by the firing of one or more rules in the Supervisory Controller. As with all systems using fuzzy logic, the membership functions for the terms were obtained empirically [16].

The Fuzzy Supervisory Controller, in reality, is based on our fuzzy expert system shell [15]. There are two important aspects for the workings of the Fuzzy Supervisory Controller. First, unlike the more traditional rule-based systems, a rule is possibly fired even when the data and the rule-antecedent are not identical patterns. For example, a fact like

(turn-angle right-20)

can match a rule antecedent like

(turn-angle right-30)

if there is an overlap between the membership functions for right-20 and right-30. The second aspect is when the Fuzzy Supervisory Controller asserts multiple fuzzy terms for the same linguistic variable, their membership functions must be combined after they are weighted in accordance with the dictates of fuzzy logic [11]. And, then the composite membership functions must be defuzzified to yield a single numerical value for, say, the turn angle for the robot. The reader is referred to [14, 15] for the implementation details regarding how to fulfill those aspects and other important details.

We now explain how the vision-based fuzzy control will be performed with the neural networks by the Fuzzy Supervisory Controller. Fig.11 shows the flow of control for processing individual navigational commands issued either by the Path Planner or by a human. As mentioned before, the robot is supplied with an initial location and a final destination, both described using minimal metrical information. Using the topological model of the hallways, the path planner then generates a sequence of commands. Assume that the sequence is the same as before and that the first of the commands is "follow-corridor". The Fuzzy Supervisory Controller invokes the neural networks in the Hallway Follower to fulfill this command. To feed the appropriate information into the input nodes of these neural networks, the system grabs an image, initially of size 512 $\times$ 480, downsamples the image to size 64 $\times$ 60, and applies an edge detector to this image, as shown in Fig. 11. The extracted edges are mapped into the Hough space and different regions of the Hough space fed into different networks [12].

The important point to note about supervisory control is that all three modules that use vision data – the Hallway Follower, the Distance-to-Junction Estimator, and the Landmark Detector – take their input from the Hough representation of the camera image. For each camera image, all the three modules are fired sequentially. Therefore, immediately after each steering command generated by the Hallway Follower, the Supervisory Controller has available to it an estimate of the distance to the next junction in the hallway. See [16] for more discussions on fuzzy supervisory control.

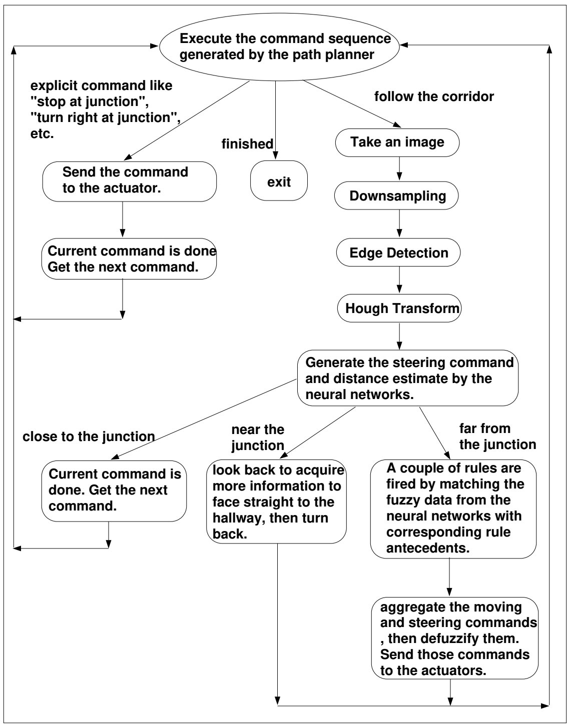


Figure 11: *Flow of control for processing individual navigational commands issued by the Path Planner.*

4.3 Experimental Results

Fig. 12 shows a downsampled camera image in the left upper corner; the extracted edges are shown in the right upper image. The edge image is then mapped into the Hough space. Different regions of the Hough space are used as input for the different neural networks in the system. The portion $[d_1, d_2] \times [\theta_1, \theta_2]$ is used as input region for the Hallway Follower module, whereas the portion of the Hough space bounded by $[d_3, d_4] \times [\theta_3, \theta_4]$ is used as input for the Distance-to-Junction Estimator Module. The lower left in Fig. 12 shows that part of the Hough space derived from the edge image of upper right which is used as input to Hallway Follower Module. The lower right shows that part which is used as input to the Distance-to-Junction Estimator Module.

On the CRT display of the robot as shown in Fig.8, composite images such as the one in Fig. 12 pop up as the robot is navigating down the hallway. As the vision data is processed by each of the three vision-based modules, one of the three iconic figures displayed in Fig. 13 also pops up. For example, after the Hallway Follower has processed part of the Hough space of the camera image shown in Fig. 12, the iconic figure at the upper left in Fig. 13 pops up on the display. In this

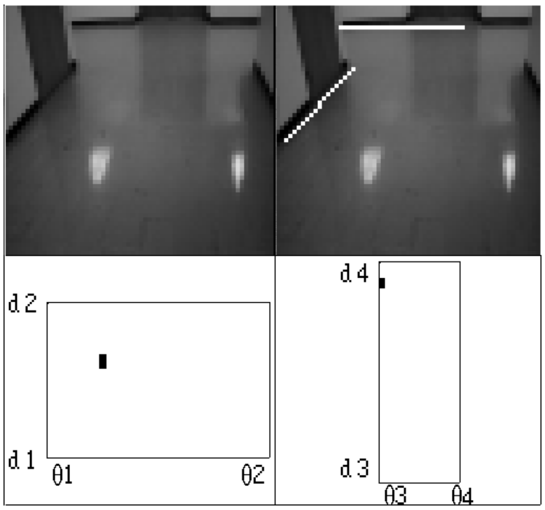


Figure 12: *Shown here are the downsampled image on the left upper corner, the extracted edges on the left lower corner, the Hough space for generating the steer command on the right upper corner and the Hough space of the horizontal lines for generating the distance estimate on the right lower corner.*

example, the output node *right_10* lights up, meaning that the Hallway Follower is recommending that the robot turn to the right by 10 degrees in order to stay in the middle of the hallway. Subsequently, the iconic figure shown at the upper right shows up; this corresponds to the output node *far* lights up. Therefore, the Distance-to-Junction Estimator is indicating that the distance to the junction is *far*. All of this information is fed into the Fuzzy Supervisor Controller. When this happens, the iconic figure at the bottom in Fig. 13 pops up on the screen. This figure shows that the Fuzzy Supervisory Controller takes in the recommendations *far* for distance to junction and *right-10* for the turn angle and issues forth its decisions regarding what the robot actions should be. As the iconic diagram shows, in this case a total of three rules were fired that made assertions about how far the robot should continue to travel in a straight line, two of these asserted that the value of the linguistic variable *distance-to-travel* should be *long* while one rule asserted that it should be *medium*. The same three rules also asserted that the the value of the linguistic variable *turn-angle* should be *zero*, *right-20* and *right-10*. The defuzzification of these assertions made by the Fuzzy Supervisory Controller result in the values of 5.9 *feet* for the latest estimate of how far the robot should travel before expecting to see the junction and that the turn angle at this time should be 9 *degrees* in order to bring the robot to approximately the middle of the hallway.

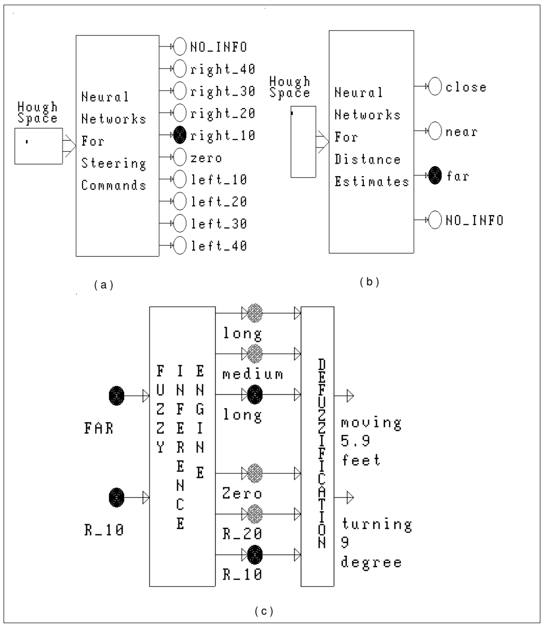


Figure 13: *Shown here are the outputs of the neural networks given the Hough spaces in Fig. 12 and the outputs of fuzzy inference engine.*

Fig. 14 shows a typical run of the robot controlled by FUZZY-NAV with human obstacles. The robot must work around the obstacle as it tries to reach its destination. During this maneuvering around the obstacle, the robot is completely under the control of the Obstacle Avoidance Module. After the robot has cleared the obstacle, it resumes its normal vision-based navigation to the destination point. The Obstacle Avoidance Module is also invoked if the robot gets too close to any of the walls.

5 Conclusions

In this paper, we discussed vision-based mobile robot navigation using the vision-based updating of the robot motion. We presented two approaches to vision-guided navigation: geometrical approach (FINALE) and topological approach (FUZZY-NAV). Through actual experiments, we demonstrated the robustness of our approaches of model-based vision to hallway navigation. Currently, we are working on enhancing the level of intelligence for the mobile robot so that the robot can go beyond just navigation and also engage in more complex tasks such as simultaneously finding objects while it is navigating in a dynamic environment.

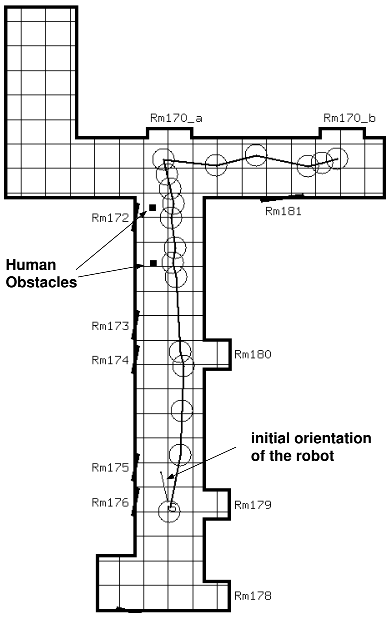


Figure 14: *A typical run under FUZZY-NAV control.*

References

[1] R. C. Arkin and D. MacKenzie, "Temporal coordination of perceptual algorithms for mobile robot navigation," *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 3, pp. 276-286, 1994.

[2] S. Atiya and G. D. Hager, "Real-time vision-based robot localization," *IEEE Trans. on Robotics and Automation*, Vol. 9, No. 6, pp.785-800, 1993.

[3] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot," *IEEE Trans. on Robotics and Automation*, Vol. 5, No. 6, pp. 804-819, 1989.

[4] R. Chatila and J.-P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proceedings of 1985 IEEE Int. Conf. on Robotics and Automation*, pp.138-145, 1985.

[5] H. Christensen, N. Kirkeby, et al., "Model-driven vision for in-door navigation," *Robotics and Autonomous Systems*, Vol. 12, pp.199-207, 1994.

[6] C. Fennema, A. Hansen, E. Riseman, J. R. Beveridge, and R. Kumar, "Model-directed mobile robot navigation," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 20, No. 6, pp.1352-1369, 1990.

[7] A. C. Kak, K. M. Andress, et al., "Hierarchical evidence accumulation in the PSEIKI system and experiments in model-driven mobile robot navigation," in *Uncertainty in Artificial Intelligence* edited by M. Henrion, R. Shachter, et al., Elsevier, pp.353-369, 1990.

[8] A. Kosaka and A. C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *Computer Vision, Graphics and Image Processing – IU*, Vol. 56, No. 3, pp.271-329, 1992.

[9] A. Kosaka, M. Meng and A. C. Kak, "Vision-guided mobile robot navigation using retroactive updating of position uncertainty," in *Proceedings of 1993 IEEE Int. Conf. on Robotics and Automation*, Vol. 2, pp.1-7, 1993.

[10] D. J. Kriegman, E. Triendl, and T. O. Binford, "Stereo vision and navigation in buildings for mobile robots," *IEEE Trans. on Robotics and Automation*, Vol. 5, No. 6, pp. 792-803, 1989.

[11] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller – Part 2," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 20, pp.419-435. 1990.

[12] M. Meng and A. C. Kak, "Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models," *IEEE Control Systems*, pp. 30-39, October 1993.

[13] H. P. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of IEEE*, Vol.71, No.7, July, pp.872-884, 1983.

[14] J. Pan and A. C. Kak, "Design of a large-scale expert system using fuzzy logic for uncertainty-reasoning," to appear in *World Congress on Neural Networks*, July 17-21, 1995, Washington.

[15] J. Pan and A. C. Kak, "FuzzyShell: A large-scale expert system shell using fuzzy logic for uncertainty-reasoning," in preparation for *IEEE Trans. on Fuzzy Systems*.

[16] J. Pan, D. J. Pack, A. Kosaka and A. C. Kak, "FUZZY-NAV: A vision-based robot navigation architecture using Fuzzy inference for uncertainty-reasoning," to appear in *World Congress on Neural Networks*, 1995.

[17] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," Tech. Report, CMU-CS-89-107, 1989.

[18] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. Journal of Robotics Research*, Vol. 5, No. 4, pp. 56-68, 1986.

[19] K. Sugihara, "Some location problems for robot navigation using a single camera," *Computer Vision, Graphics and Image Processing*, Vol. 42, pp. 112-129, 1988.

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