Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models

Min Meng and A.C. Kak

Large strides recently have been made in the model-based approach to vision-guided mobile robot navigation in indoor environments. Although the model-based method does indeed result in very robust reasoning and control architectures, the fact remains that this approach requires precise geometrical modeling of those elements of the environment that are considered visually significant — a requirement that can be difficult to fulfill in some cases. The need for geometrical modeling of the environment also makes such systems "non-human-like." We have, therefore, recently developed a new kind of reasoning and control architecture for vision-guided navigation that makes a robot more "human-like." This system, called NEURO-NAV, discards the more traditional geometrical representation of the environment, and instead uses a semantically richer nonmetrical representation in which a hallway is modeled by the order of appearance of various landmarks and by adjacency relationships. With such a representation, it becomes possible for the robot to respond to human-supplied commands such as, "Follow the corridor and turn right at the second T junction." This capability is achieved by an ensemble of neural networks whose activation and deactivation are controlled by a supervisory controller that is rule-based. The individual neural networks in the ensemble are trained to interpret visual information and perform primitive navigational tasks such as hallway following and landmark detection.

Imitating Human Navigation

It has now been convincingly demonstrated that by combining model-based reasoning with Kalman filtering, it is possible to design very robust reasoning and control architectures for vision-guided mobile robot navigation in indoor environments. For example, the robot described in [1], [2] is capable of autonomously navigating in hallways, at speeds around 8 m/min, using vision for self-location and ultrasound for collision avoidance. The performance of this robot is not impaired by the presence of furniture and other stationary or moving objects in the hallways. The robot simply treats everything that is not in its geometrical model base as visual clutter. By maintaining models of uncertainties and their growth during navigation, the robot is able to place bounds on where it should look for important landmarks; this gives the robot immunity against visual clutter. Other contributions along similar lines that preceded the work reported in [1], [2] are the work of Ayache and Faugeras [3], Kriegman *et al.* [4], and Fennema *et al.* [5].

Model-based approaches of the kind reported in [1], [2] require that geometrically precise models of the 3D environment be available. While it is true that not every feature of the environment need be represented in the model, for obvious reasons a sufficient number of visually significant features, called landmarks, must be represented. We believe that the need for geometrical models of the environment sets these model-based approaches apart from the manner in which humans navigate.

Inspired by how easily humans navigate, we have developed a new system; we call it NEURO-NAV. The key ideas in NEURO-NAV were formulated through observations of human navigational behavior. First, human navigators do not need to calculate the exact coordinates of their position while navigating over roads or through hallways. The road-following or the hallway-following behavior exhibited by humans is a *reactive behavior* that is learned through experience. Second, given a goal, human navigators can focus attention on particular stimuli in their visual input and extract meaningful information very quickly. Third, extra information may be extracted from the scene during reactive behavior; this information (e.g., approaching an intersection) will usually be stored away and may be retrieved subsequently for higher level reasoning.

These observations have dictated the design of NEURO-NAV. Besides containing a rule-based supervisory controller, NEURO-NAV consists mainly of a collection of neural networks that mimic the human reactive navigational behaviors. NEURO-NAV does not require geometrically precise 3-D models of the environment. Instead, a geometrically much simpler yet semantically richer hallway model, similar to human navigational maps and consisting of corridors, junctions, dead ends, and landmarks (e.g., doors, bulletin boards), aids the high level reasoning and path planning process during navigation. Using this simple model, the supervisory controller can issue commands to the robot, such as, "go down the corridor A and turn right at next junction." Given these commands, the robot travels on a planned path using its ability to perform primitive navigational tasks such as hallway following, i.e., navigating down the hallway without running into walls, and landmark detection.

Extended version of a paper presented at the 1992 IEEE International Conference on Systems, Man, and Cybernetics. *The authors are with the Robot Vision Lab, 1285 EE Building, Purdue University, W. Lafayette, IN 47907-1285.* *This work was supported by the Office of Naval Research under Grant ONR N00014-93-1-0142.*

0272-1708/93/$03.00©1993IEEE

Before closing this section, we would like to mention an earlier contribution in the area of mobile robot navigation using neural networks. We are referring to the work of Pomerleau [6] in which reduced-resolution raw images and range data for road scenes are fed into a two-layer feed-forward neural network to produce steering commands. While clearly a pioneering effort, Pomerleau's work suffers from the shortcoming that a monolithic neural network is used to generate the final decisions for the vehicle. Monolithic neural networks are not amenable to interaction with higher-level supervisory control in an autonomous system. In contrast, as in our work, when a control architecture uses an ensemble of neural networks, each dedicated to some primitive task, control becomes much more flexible and more receptive to the incorporation of heuristic knowledge supplied by a human expert.

Here, we will very briefly present the overall architecture of NEURO-NAV, leaving some of the details to [7]. We will focus on one specific module of this architecture, the corridor follower. This article is an extended version of [8].

Architecture of NEURO-NAV

As depicted in Fig. 1, the reasoning and control architecture of our vision-guided mobile robot navigation system includes a non-metrical model of the hallway, a path planner, a human supervisor, a hallway follower, a landmark detector, an obstacle-avoidance module, and a rule-based supervisory controller. Here we will not say much about the ultrasonic obstacle-avoidance module, save that it utilizes a semi-ring of Polaroid transducers and calculates the directions and the distances to obstacles on the basis of time taken by the echoes to return to the transducers. The computed distance to an obstacle is used for local and immediate modifications to the path traversed by the robot.

Hallway Modeler

Clearly, given the missions of NEURO-NAV, the geometric modeling of the environment used in, say, [1], [2] is not what is needed here. As mentioned before, NEURO-NAV views a task, such as hallway following, as a reactive behavior, which means the motions of the robot will be triggered by visual cues such as the orientation of the hallway floor edges with respect to the direction of the robot. A task such as hallway following, therefore, has no need for the usual metrical details of a geometry-based modeler. For these reasons, in the modeling needed for NEURO-NAV, we have eschewed geometry in favor of a more non-metrical approach. The components of a hallway model in NEURO-NAV are determined more by their semantic and functional significance than by geometry. What we mean by this statement is made clear by the following.

For the purpose of illustrating our modeler, consider the hallway segment shown in Fig. 2(a) and its corresponding model, represented as an attributed graph, in Fig. 2(b). As shown there, the nodes of this graph consist of corridors, junctions, and dead ends. The links of the graph, also attributed, contain information regarding the physical distance between the landmarks represented by the nodes at the two ends of the link. A node is represented by a list of attribute names and pointers. For example, a corridor node contains the attributes *name*, *primary direction*, *left landmarks*, *right landmarks*, *behind node*, and *beyond node*. The pointers that are the values of attributes like *left landmarks*, etc., point to a list of landmarks such as doors, junctions, alcoves, etc. An example of a corridor node, corridor C2 in Fig. 2(b) in the hallway model, is shown in Fig. 3. In a sense, each extended node, such as corridor C2, is assumed to be centered at its middle point, making the distance between C2 and J1 equal to half the length of the corridor. This is done merely to facilitate path planning. NEURO-NAV itself is fully aware of the fact that C2 is an extended physical entity. As the reader has surely surmised from our description, a hallway model is defined with respect to a particular direction of travel for each of its corridors; this points to an interesting difference between the traditional geometrical models and the non-metrical models such as in NEURO-NAV. Of course, as further elaborated in [7], the robot is free to travel in any direction in a hallway. In other words, if so dictated by the initial position of the robot and the desired destination location, the path planner is free to send the robot through a segment of a hallway in a direction opposite to the one encoded for the segment in the representation.

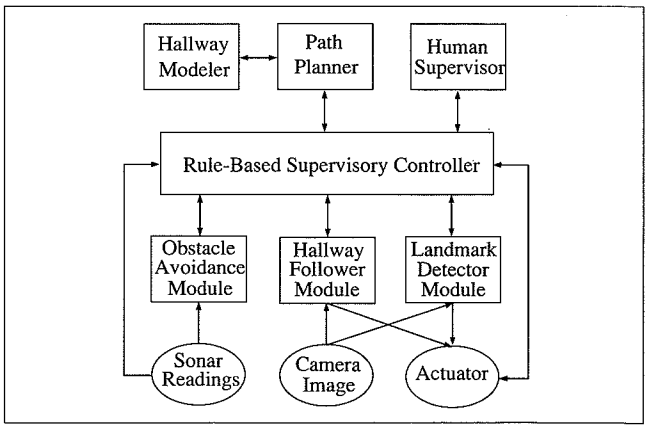


Fig. 1. Architecture of NEURO-NAV.

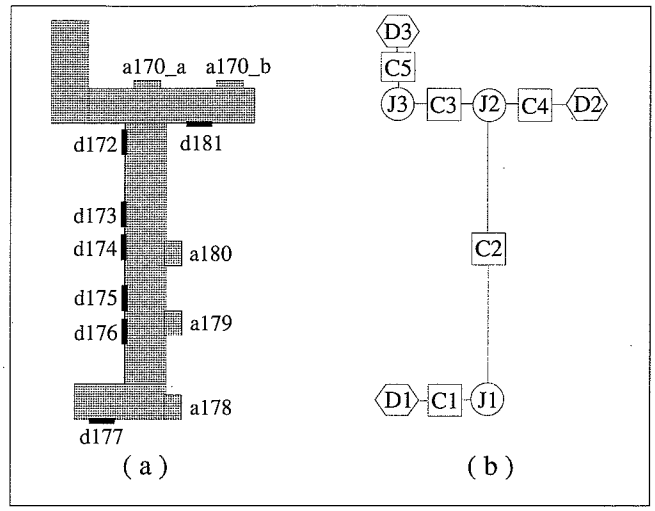


Fig. 2. (a) A hallway section where d_i's are doors, a_i's are alcoves. (b) Representation of the hallway section in (a). C_i's are corridors, J_i's junctions, and D_i's dead ends in a non-metrical attributed-graph representation.

name:	C2
primary direction:	north
left landmarks:	door, d176
	door, d175
	door, d174
	door, d173
	door, d172
right landmarks:	power_panel, p3
	alcove, a179
	power_panel, p2
	alcove, a180
	bulletin_board, b2
behind node:	junction, J1
beyond node:	junction, J2

Fig. 3. The data structure used for the node that represents the corridor C2 in Fig. 2(b).

Path Planner

There are basically two approaches to path planning. In the mobile robotics context, the first approach, called the configuration space based approach, may be implemented by first constructing a binary array whose nonzero elements correspond to the floor of the hallways and then modifying the binary array in such a manner that, from the standpoint of collision avoidance, the motion of the mobile robot on the floor is equivalent to the motion of a point object in the hallways. This approach was pioneered by the work of Lozano-Perez and Wesley [9] in the context of path planning for arm robots and, most recently, used in the FINALE architecture [1]. A major disadvantage of the configuration space based approach is the need for path replanning whenever the robot deviates from its originally computed path, as is wont to happen during collision avoidance exercises.

The need for replanning is eliminated in a second approach to path planning; in this approach each object is considered to give rise to a potential field and a path to the goal is calculated by searching through the valleys of the overall potential field. Although in recent years this approach, pioneered originally by Khatib [10], has proved popular in a number of different domains, especially the domain of assembly motion planning [11], nonetheless it is not suitable for our purposes since, like the configuration space based approaches, it is highly demanding of the detailed geometrical knowledge of the navigational environment.

Our path planner differs fundamentally from both the configuration space based approach and the potential field based approaches. The new path planner, reported here only very briefly, simulates human cognition in the sense that the paths produced are less geometrical and more descriptive and yet useful enough for a mobile robot to use to get to its destination. For example, for the hallway shown in Fig. 2, if the robot is placed initially facing north in corridor C2 and its destination is in front of the dead-end D2, the path planner will output the following string of symbols: "*follow corridor, turn right at next T-junction, stop at next dead-end.*" It is important to realize that this path to the destination is semantically rich and is essentially devoid of explicit geometry, at least in the sense geometry is used in, say, the FINALE system in [1]. It would not be far fetched to say that the output of the path planner in NEURO-NAV shares many similarities with how a human navigator might communicate with the driver of a vehicle.

In our approach to path planning, the initial and the destination positions of the robot, given by a human supervisor, are specified relative to landmarks in the hallway model. If the robot is going to navigate in the hallway shown in Fig. 2, the position of the robot can be described using symbol streams like: (landmark: *X*; node: *Y*; direction: *Z*) or just (node: *Y*; direction: *Z*). The first choice is evidently more specific than the second for describing the initial position or the desired destination position of the robot. As an example, when the robot is in the vicinity of door d176 in Fig. 2(a) facing north, the position would be described by the list (*landmark: d176; node: C2; direction: north*). On the other hand, if the human did not wish to be specific about the precise location of the robot, the path planner in NEURO-NAV would not complain if the position of the robot is described by the list (*node: C2; direction: north*). The reader should note that, as with some other aspects of NEURO-NAV, this manner of supplying the robot position to the path planner makes for a more human-like interface with the robot.

Given the initial and the destination locations specified as above, the path planner then examines the adjacency matrix data structure for the hallways. Dijkstra's algorithm [12] is used to traverse the adjacency matrix using the distances between nodes as cost functions to produce a sequence of nodes as a solution. This sequence of nodes where the first node is the initial node and the last node the destination node represents the path between the initial and the destination positions. The path planner subsequently translates this sequence of path nodes into a sequence of descriptive commands. For example, when the initial and the destination positions are given by (*landmark: d176; node: C2; direction: north*) and (*node: D2; direction: east*), respectively, the command sequence output by the path planner is (*follow corridor, turn right at next T junction*, and *stop in front of the dead end*). Associated with each command in this sequence are lists of landmarks on the left and right faces, and on the behind

and the beyond junctions. These landmarks are listed in the order of appearance to be seen by the robot.

Before closing the subject of path planning, we want to mention that when the calculated paths are described using semantically rich descriptors, path replanning becomes unnecessary. Since the description of the path along which the robot is moving is not burdened by geometrical precision, the robot now has latitude in locating itself with respect to its environment. To elaborate, as long as the robot can see a particular set of landmarks, each possibly recognized by a special neural network, the robot knows where it is with respect to the final goal. *In the manner of a human, the robot would not compute its exact location, but would still know how to get to its destination.*

Human Supervisor

The human supervisor in NEURO-NAV has two roles. When the robot is in autonomous mode, the human supervisor specifies the initial and destination positions of the robot and this information is passed on to the path planner. On the other hand, when the robot is in human-supervised mode, the human supervisor directly gives out the commands, often in the form of a sequence that is similar to what is produced by the path planner when the robot is in the autonomous mode. The menu of commands that is available to a human supervisor is shown in Table I. The first four commands are explicit directions for the robot to follow. The remaining eighteen commands are implicit and the robot executes them with the help of the hallway follower module, the landmark detector module, and the rule-based supervisory controller.

Hallway Follower and Landmark Detector

The hallway follower module and the landmark detector module in Fig. 1 are composed of collections of neural networks, each network trained to perform a specific task. The hallway follower module consists of three submodules, the corridor follower, the junction_left follower, and the junction_right follower, each consisting of two neural networks. (The reason two neural networks are necessary in each submodule will be explained shortly.) The function of the corridor follower is to keep the robot going straight down a hallway and to do so even when the robot is forced by obstacles to deviate from a straight path. The function of the junction_left follower is to ensure that the robot is able to turn left through a junction. The function of the junction_right follower is the same, except that now the robot has to turn right through the junction.

The landmark detector module consists of a neural network that is capable of detecting both junctions and dead ends, and, at the same time, of making a qualitative assessment of the distance between the robot and the junction or dead end. The distance estimate is qualitative in the sense that the output nodes of this neural network correspond to "far," "at," and "near," and the estimation is in terms of these semantic labels.

Supervisory Controller

During navigation, appropriate networks are activated and deactivated by the rule-based supervisory controller. Further discussion on this orchestration of all the neural networks in the system by the supervisory controller is presented in [7]. We only wish to mention here that when obstacles are detected by ultrasonic sensors, the supervisory controller temporarily suspends vision-based servoing. All the maneuvering required for collision avoidance is governed by the collision avoidance module in Fig. 1. The supervisory controller resumes vision-based navigation after the possibility of collision is eliminated.

Corridor Follower

We will now explain in greater detail the structure of the corridor follower, which, as was mentioned previously, is contained in the hallway follower module in Fig. 1. The design of the corridor follower is based on the following rationale: i) in order for a robot to go straight down a hallway, the perspective projection in the camera image of either the left or the right hallway edge must be within a certain angular range; and ii) if the robot is not headed straight down the hallway — a condition

Table I
Command Menu Available to a Human Supervisor

Commands					
	name	**argument**		**name**	**argument**
1	turn left	(degrees)	12	stop	T junction
2	turn right	(degrees)	13	turn left	left_T junction
3	go straight	(meters)	14	go straight	left_T junction
4	stop		15	stop	left_T junction
5	follow	corridor	16	turn right	right_T junction
6	stop	dead-end	17	go straight	right_T junction
7	turn left	junction	18	stop	right_T junction
8	turn right	junction	19	turn left	intersection
9	stop	junction	20	turn right	intersection
10	turn left	T junction	21	go straight	intersection
11	turn right	T junction	22	stop	intersection

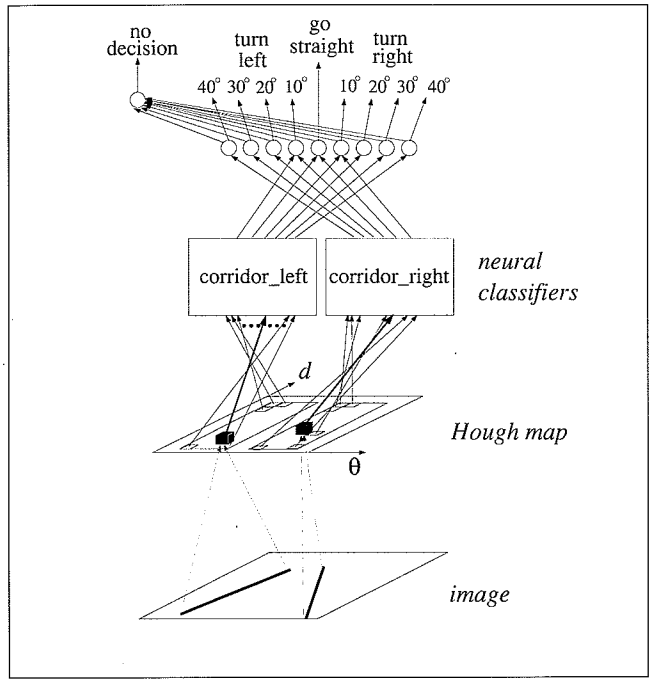


Fig. 4. The structure of the corridor follower is shown here. As shown, the disjoint regions of a Hough map are fed into two different neural networks for corridor following.

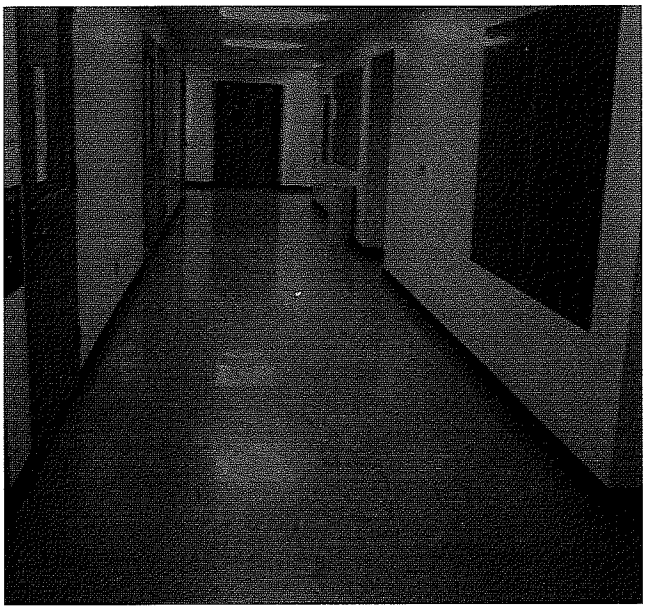


Fig. 5. A typical image seen by the mobile robot during hallway navigation.

that could be caused by an attempt at collision avoidance — there exists a correlation between the turn the robot must make back towards the straight-down-the-hallway direction and the extent to which the perspective projection of the hallway edge is outside the previously stated angular range. This rationale evidently dictates that it be possible to extract line features from images and, since the robot has to servo with respect to the line features, it be possible to carry out a fast extraction of these line features. These considerations have led to the corridor follower shown in

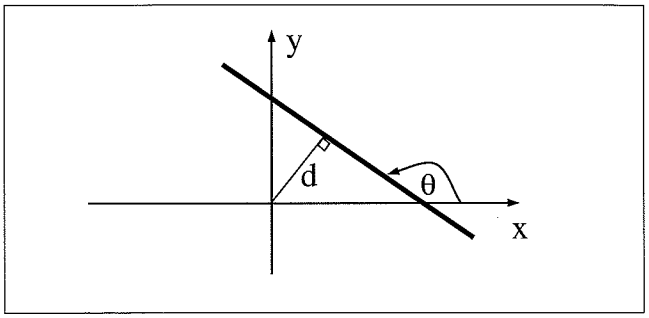


Fig. 6. The Hough space representation spans the parameters θ and d that characterize a line feature in an image.

Fig. 4. The input to the two neural networks shown there consists of the left and the right regions of the Hough map derived from a single camera image, such as the one shown in Fig. 5. For readers conversant with vision algorithms, a robust method for detecting edges that may be continuous or broken consists of first applying an edge detector to an image and then mapping all the edge points into what is called a Hough space, whose each cell is indexed by the slope of an edge and by the perpendicular distance of a line through that edge from the image center (Fig. 6). To speed up this process, in NEURO-NAV the camera image is downsampled from a 512×480 matrix to a 64×60 matrix with no noticeable effect on the abilities of the corridor follower. A more modularized schematic of the corridor follower module is shown in Fig. 7. Actually, that processing chain serves all the functions of the hallway follower, which include corridor and junction following, and of the landmark detector. The distance estimates mentioned in that figure are for the landmark detector.

To understand why different regions of the Hough map are fed into different neural networks in Fig. 4, we must first explain the purpose of the *corridor_left* and the *corridor_right* neural networks shown there. Note that the goal of corridor following

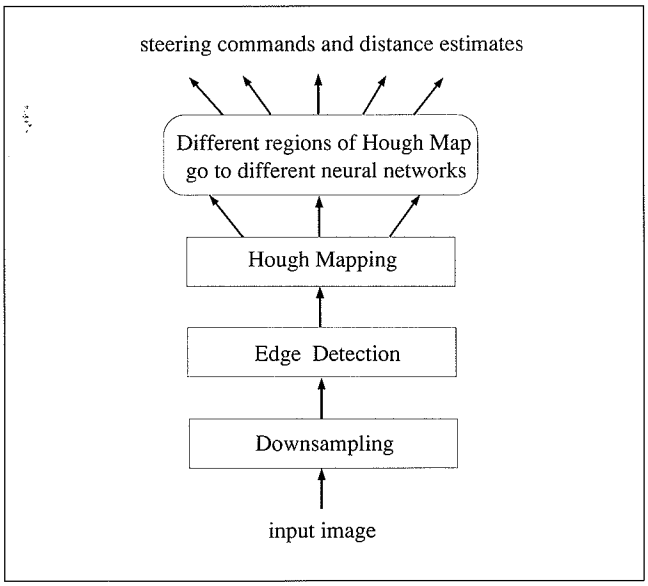


Fig. 7. The flow of processing that takes a camera image for input and produces steering commands for the robot at the output. Also produced at the output are qualitative estimates of the distance between the robot and any junctions/dead ends in front of the robot.

is to navigate down the corridor without running into the walls on the left or the right of the robot. In Fig. 4, one neural network, *corridor_left*, is trained to be sensitive to the left hallway floor edge; while the other neural network, *corridor_right*, is sensitive to the right hallway floor edge. Each of these two networks produces six output commands representing steering angles at 10° increments. The output of the two neural networks compete to contribute to the final decision using the principle of "maximum takes all." If neither network can produce a decisive output (i.e., values of all output nodes are below some threshold), a "no decision" output is rendered.

Now back to the explanation regarding the segmentation of the Hough map, it can be shown that the Hough map can be structured in such a manner that even for significant deviations in the orientation of the robot from its ideal direction, all the edge features corresponding to the left wall will be in the left half of the Hough space and all the edge features corresponding to the right wall in the right half of the Hough space. For illustration, we have shown in Fig. 8 the different regions of a Hough map for a typical camera image; this segmentation of the Hough map depends on how the camera is mounted on the robot and the figure shown applies to our robot. Therefore, it is sufficient to feed into the *corridor_left* only those Hough cells that are in the left half plane but not the extreme left columns, and into the *corridor_right* only those that are in the right half plane, but again not too close to the extreme right edge of the Hough space. Fig. 9(a) shows a typical downsampled hallway image. The detected edges are shown in Fig. 9(b), the extracted floor edges in Fig. 9(c), and finally the Hough space representation of the floor edges used as input to the neural networks is shown in Fig. 9(d). Note that the discrimination between the floor edges and the nonfloor edges is made with the help of the Hough map.

The neural networks that perform corridor following, like all the other neural networks employed in NEURO-NAV for primitive navigational tasks, are three-layer feed-forward neural networks trained by the backpropagation learning algorithm. The network structures are determined empirically by initially assigning an arbitrarily large number of nodes to the hidden layers, studying the convergence of the system during the learning process, and reducing the number of the nodes in the hidden layers to see whether or not the network still converged with the reduced number of nodes. In this manner, one ends up with a network with a small number of nodes for the hidden layers while the network is guaranteed to converge. This approach is particularly effective if a small learning rate is used in the backpropagation algorithm. The network structure of the neural network *corridor_left* is shown in Fig. 10. It is composed of 117 input units which cover a designated region of the Hough map, two hidden layers consisting of 12 and 6 hidden units, respectively, and 6 output units. In addition, three bias units are included in the network to represent the internal thresholds of the units in the hidden and the output layers.

The backpropagation learning algorithm is a gradient descent error-correction algorithm that minimizes the errors between the desired outputs and the actual computed outputs by modifying the connection strengths, or weights, between the units in the network [13]-[16]. The learning algorithm consists of two phases. In the feed forward phase, the input pattern is presented to the input units, fed forward through the hidden layers, and the

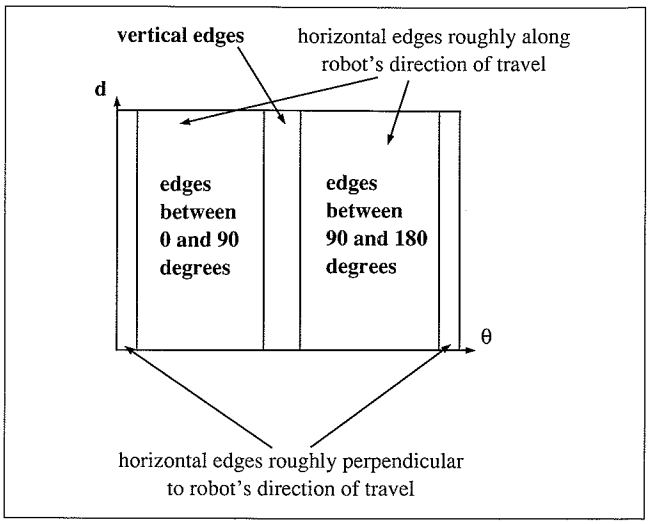


Fig. 8. Segmentation of the Hough map.

actual output is calculated at the output layer. At each layer, the output of a unit j is calculated as

$$O_j = f\left(\sum_{i=1}^{k} O_i w_{ij} + \theta_j\right)$$

where O_i and O_j are the outputs of nodes i and j, w_{ij} the connection strength from node i to node j, and θ the internal threshold of node j. For the network of Fig. 10, for a given node j the summation shown will span the nodes that are in the layer below. The function $f()$ is the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}$$

that transforms the input values into continuous values between 0 and 1.

In the backpropagation phase, the weights and the internal thresholds are modified using the errors between the desired and the actual outputs. More specifically, they are adjusted by the following equations:

$$\Delta w_{ij}(t+1) = \eta O_i \delta_j + \alpha \Delta w_{ij}(t)$$
$$\Delta \theta_j(t+1) = \eta \delta_j + \alpha \Delta \theta_j(t)$$

where η is a small positive constant called the learning rate, δ_j is the error signal at node j, and α is the momentum coefficient that determines the effect of past learning on the current weight changes. If node j is in the output layer, the error signal δ_j is calculated as

$$\delta_j = O_j(1 - O_j)(O_j^d - O_j)$$

where O_j^d is the desired output for node j. If node j is in any of the hidden layers, the error signal is calculated as

$$\delta_j = O_j(1 - O_j)\sum_i \delta_i w_{ji} .$$

For a given node j, the summation here spans all the nodes in the layer above. In this manner, the error signals are propagated

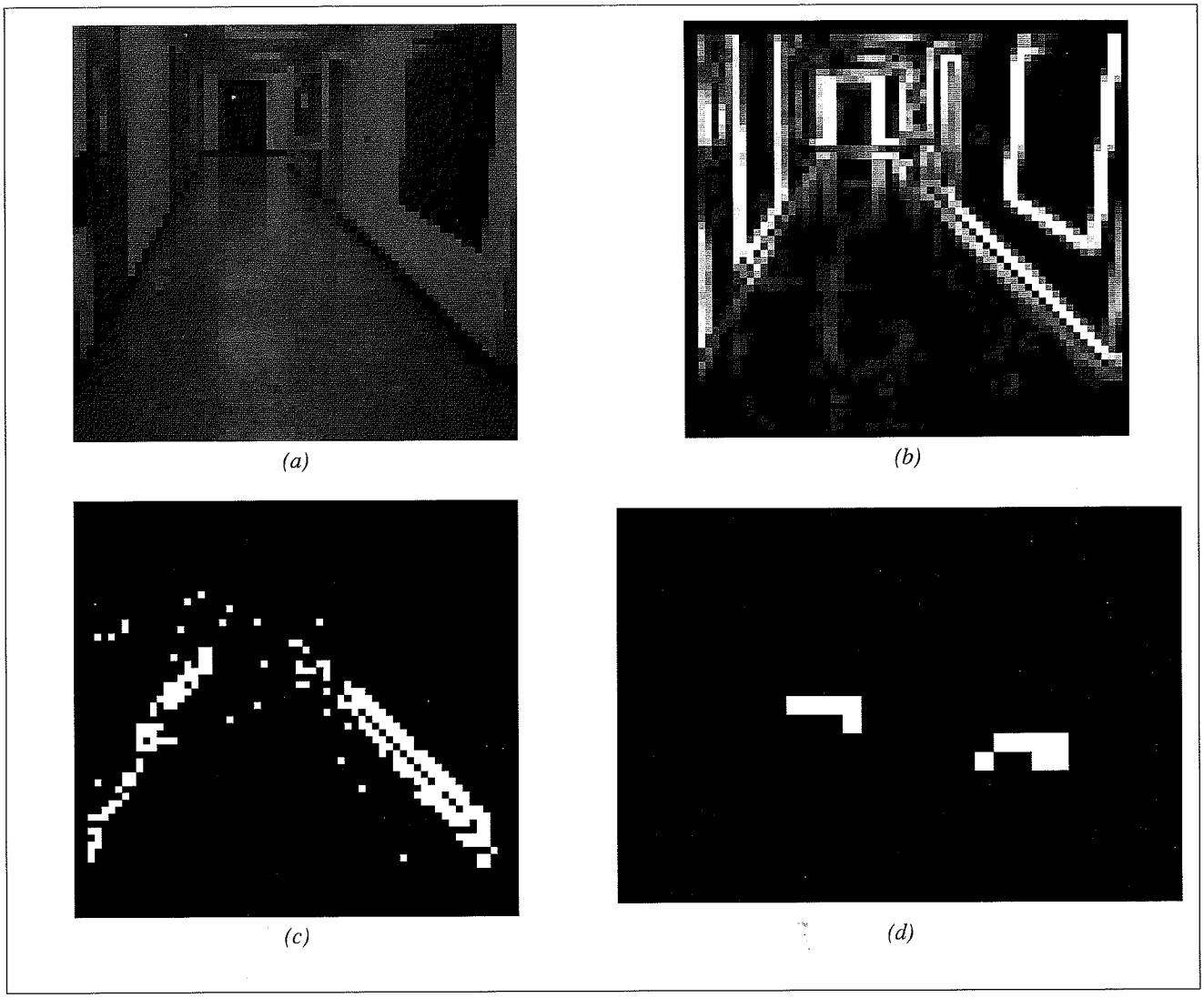


Fig. 9. Shown here are the edges, the extracted floor edges, and the Hough map representation for a reduced-resolution camera image shown in (a). (a) Resampled image. (b) Detected edges. (c) Extracted floor edges. (d) Corresponding Hough space.

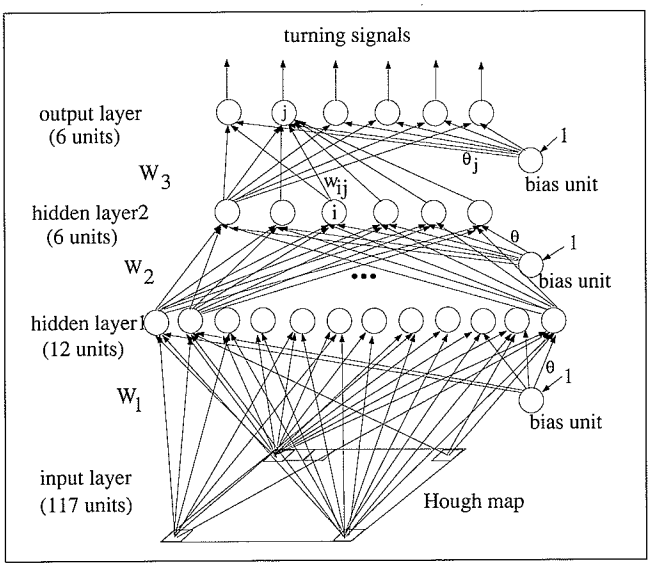


Fig. 10. The structure for the corridor_left network (not all links are shown).

backwards from the output nodes to the input nodes. During the training phase, the feedforward and the backpropagation phases are repeated, cycling through many inputs and desired outputs, until the errors are below some threshold.

We now describe briefly how junction/dead-end detection, contained in the Landmark Detection Module in Fig. 1, works in NEURO-NAV. To present the main point of the junction/dead-end detection, as the robot is approaching a hallway junction, say junction J2 in Fig. 2(a), the numerous horizontal edges that define what the robot sees straight ahead will exhibit a downward motion in the camera image. This fact is exploited for the qualitative estimation of the distance between the robot and a junction/dead-end by first applying a difference operator to Hough space representation of successive images; the difference operator is applied to only those regions of the Hough space that are populated by the horizontal lines in the image. (For the configuration of the camera on our robot, the first few columns and the last few columns in the Hough map correspond to the horizontal edges.) The output nodes of this network, as was

mentioned before, stand for the labels "far," "at," and "near," these labels being qualitative estimates of the distance of the robot from the far wall of the junction.

The neural networks for junction following, contained in the hallway follower module, share similarities with the corridor follower, also contained in the same module. To elaborate, a junction_left follower, used for turning left through a junction, consists of two neural networks, one tracking the right edge of the floor and the other the left edge. In actual execution, the latter network is initially ineffective as it either sees nothing or whatever it sees is in the wrong cells of the Hough map. However, as the robot comes close to finishing up a left turn, this network keeps the robot from turning too far. In a similar fashion, junction_right follower consists of two neural networks also.

Experimental Results

The neural networks described above were trained to assist our robot in navigating the hallways around our laboratory. Training data can be selected from snapshots of the hallway, but for convenience, synthesized images rendered by an existing three-dimensional modeler of the hallway were used. As described earlier, the structures of the neural networks were determined empirically. The networks *corridor_left* and *corridor_right* are each composed of 117 input units, two hidden layers consisting of 12 and 6 hidden units, respectively, and 6 output units. We used 72 patterns, with twelve patterns corresponding to each of the six steering directions, for training the neural networks of the corridor follower. For each of the two networks, each training cycle consisted of feeding a pattern three times into the feedforward network, backpropagating the errors, and adjusting the weights. One training loop consisted of processing all 72 patterns in this manner. We used 1000 such loops. Therefore, each neural network in the corridor follower of Fig. 4 was subject to 216 000 forward propagations and backpropagations before the connection weights were accepted. We chose to use 1000 training loops because the average output error magnitudes after that many training loops were below 0.01. The average errors during the 1000 training loops for networks *corridor_left* and *corridor_right* are shown in Fig. 11(a) and (b). The learning rate and the momentum coefficient were both set to 0.05.

The networks junction_left and junction_right mentioned in the previous section are each composed of 117 input units, 12 and 4 hidden units in the hidden layers, and 2 output units indicating two different degrees of steering angles. For example, the two output units of the junction_left network issue left steering commands of 30° and 60°. (Note that for turning, say, left, the two neural networks that are used in a competitive mode, in the same manner as in corridor following, consist of the the corridor_left part of Fig. 4 and the just-mentioned junction_left network. While the latter network causes the robot to turn left by either 30° or 60°, the former keeps the robot from turning left too far.) A total of 60 training patterns was used to train these networks, with the learning rate and the momentum coefficient both set to 0.1. The network junction/dead-end detector, again mentioned in the previous section, consists of 21 input units that cover a designated region of the Hough map, 6 and 4 units in the two hidden layers, and 3 output units standing for "far," "near," and "at." This network was trained with 47 training patterns while the learning rate and the momentum coefficient were both set to 0.05. The average error rates during the training processes for these networks are shown in Fig. 11(c) and (d).

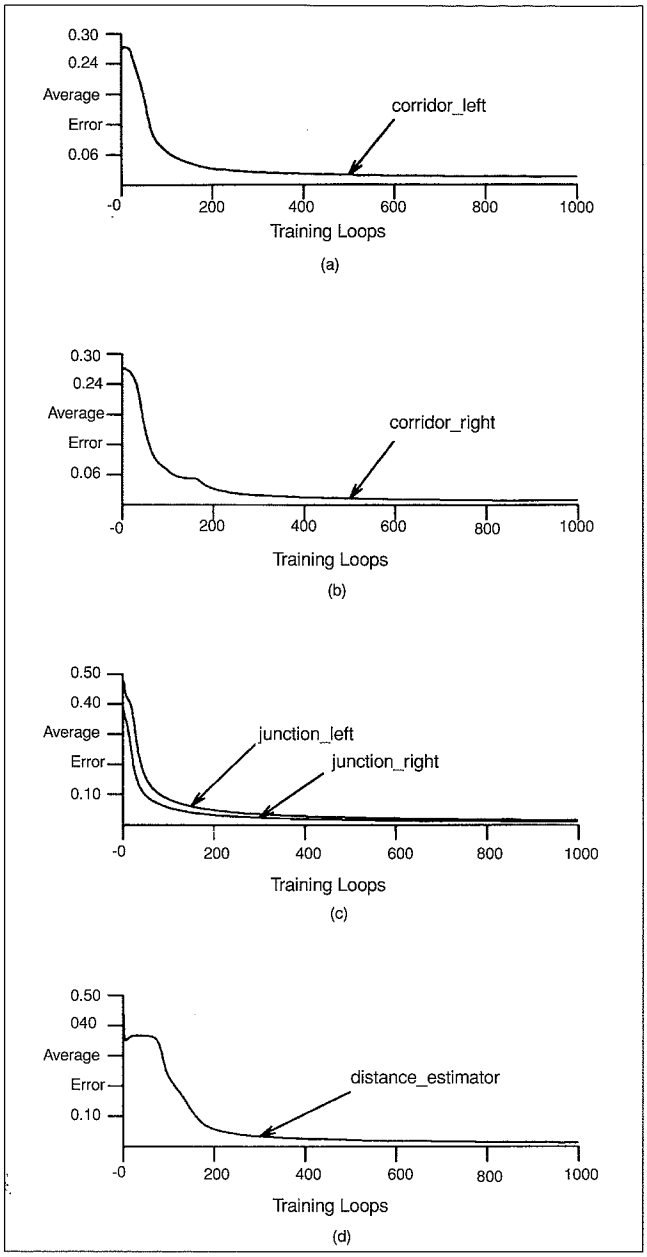


Fig. 11. Average error rates for training the various neural networks in NEURO-NAV by the backpropagation algorithm.

The trained neural networks were then incorporated into NEURO-NAV. Using 448 test patterns not used for training, NEURO-NAV generated 387, or 86%, correct steering angles and 43, or 10%, incorrect angles. The remaining outputs (4%) indicated "no decision" which includes the situations where no hallway floor edges were seen by the robot. In the 10% of the cases when an incorrect steering angle was output, the steering command was off by one "notch", meaning 10°. When a "no decision" was rendered, the robot was instructed to turn slightly so that a new viewing angle may produce a more decisive output. Shown in Fig. 12 are the trajectories of our robot in two experi-

mental runs. The run shown in Fig. 12(a) is for the case of no obstacles in the hallway; therefore, vision is the only sensing used and the ultrasonic obstacle avoidance module plays no role here. For the run in Fig. 12(b), a human obstacle is in the middle of the hallway; now the ultrasonic collision avoidance module and vision-based processing must work in concert. For both runs in Fig. 12, although at times incorrect outputs were generated, the robot was able to detect the error in its subsequent moves and correct its course.

At this point, one might ask why not train one large neural network that would cover all regions of the Hough map and produce a final command directly. Our decision to train individual small networks to perform primitive tasks is based mainly on three reasons. First, we would like to create generic behaviors. In the case of corridor following, by training the two networks individually, we are not forcing the two networks to agree with each other. Consequently, the corridor follower will not be limited to hallways of the same width. Second, a large neural network is often viewed as a large black box which produces output that cannot be influenced by other intelligent agents. Using smaller networks that produce intermediate results allows a higher level knowledge based system, in our case the rule-based supervisory controller, to participate in the decision making process. Third, smaller neural networks with less complicated decision space are easier to train.

NEURO-NAV is implemented on a Cybermotion platform with a turret of our own design containing a VME bus-based MC68030 processor. A picture of the robot is shown in Fig. 13(a), with Fig. 13(b) a schematic of the various features of the robot. Using ordinary laboratory computing hardware of 16 MIPS power, NEURO-NAV can process a camera image and produce a navigational output within approximately 2 s.

More Human-Like Behavior

We believe we have succeeded in developing a vision-guided navigation system for indoor robots that endows the robot with primitive behaviors not too different from those used by humans. The behaviors, such as corridor following, junction following, and junction/dead-end detection, are "programmed" into neural networks that are fed from different and pre-designated regions of the Hough space representation of camera images.

The NEURO-NAV approach is radically different from the deliberative approach to vision-guided mobile robot navigation presented in [1] that uses model-based reasoning and Kalman filtering. An advantage of the deliberative approach, especially of the type presented in [1], is that the performance of the robot carries certain statistical guarantees, at least as long as the geometrical models of the environment and the models of uncertainty remain valid. On the other hand, the advantage of the more behavior-based approach in NEURO-NAV is that the robot is more human-like in the manner in which it solves navigation problems; in its use of environment representations that are akin to human cognitive maps; in the fact that the robot does not care about the geometrical precision of either the representation of the environment or of its own location; and finally, in the nature of the commands that NEURO-NAV generates for robot's motion, these commands being similar in structure to the commands a human navigator would issue to a driver.

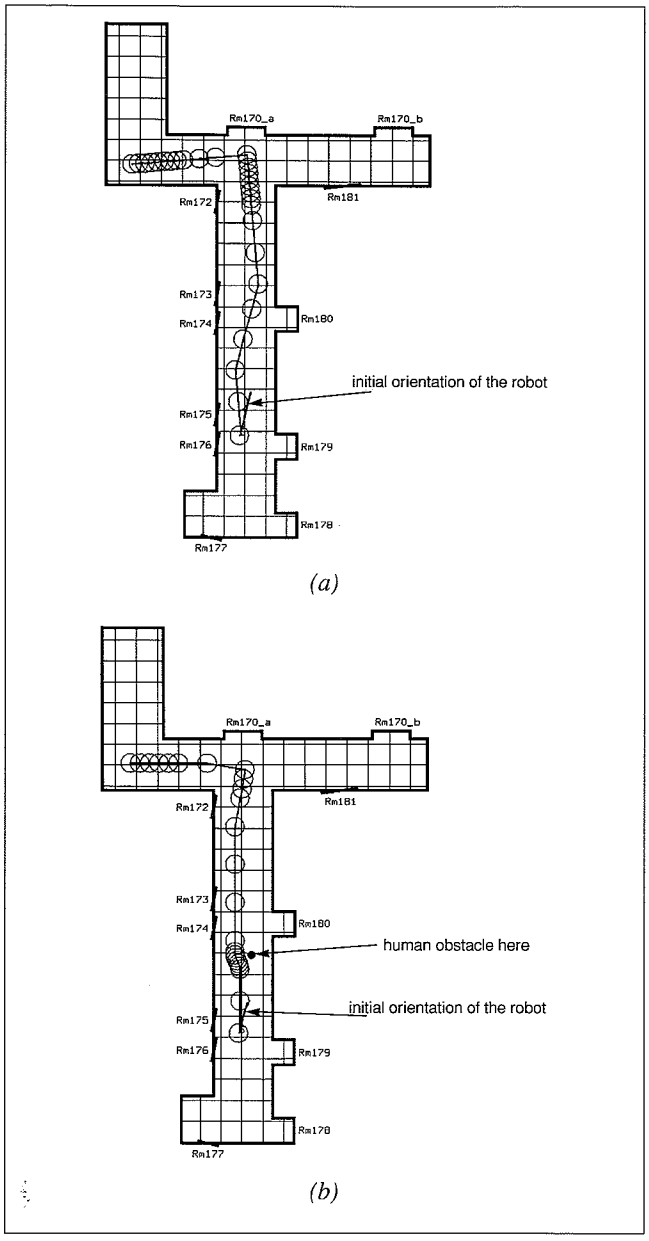
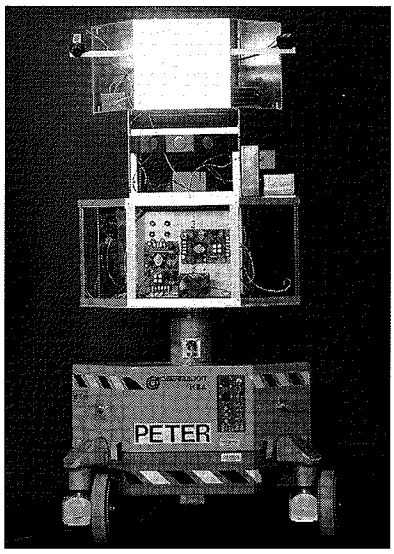


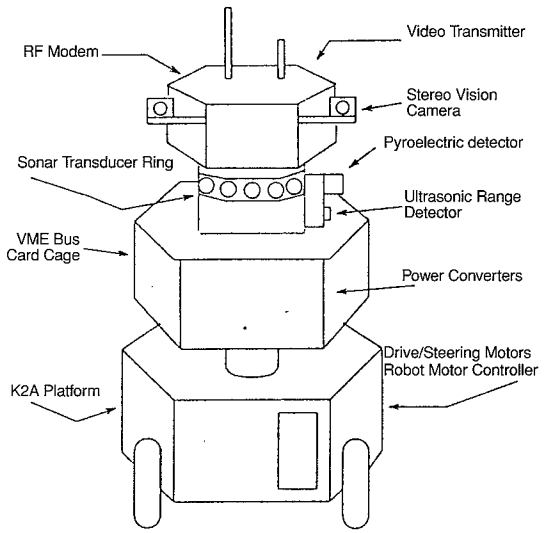
Fig. 12. (a) An actual experimental run of the robot under the control of NEURO-NAV. There were no obstacles in the hallway for this run. (b) An actual experimental run of the robot under the control of NEURO-NAV. In this case, a human obstacle is at the location shown. The obstacle is not constrained to be stationary.

Acknowledgment

The authors wish to thank Steve Blask for his help with the development of the graphical interface for NEURO-NAV. Juiyao Pan's help was indispensable for the running of the navigation experiments on the robot; he also implemented the multi-processing capability under VxWorks which increased the computational efficiency of the system. Ken Groom eliminated some of the most persistent bugs in the communication hardware onboard the robot; his help is much appreciated, as is the help rendered by Andy Jones who, together with Ken Groom, "cleaned up" the various power supplies on the robot. Matt Carroll was the original



(a)



(b)

Fig. 13. (a) Picture of the robot. (b) Schematic of the various features of the robot.

designer of the turret and the VME bus-based supervisory controller. Finally, one of us (M.M) is grateful to Thach Le for being a source of much-needed moral and intellectual support.

References

[1] A. Kosaka and A.C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *Comp. Vis., Graphics, Image Proc.—Image Understand.*, vol. 5, pp. 271-329, Nov. 1992.

[2] A. Kosaka, M. Meng, and A.C. Kak, "Vision-guided mobile robot navigation using retroactive updating of position uncertainty," in *Proc. 1993 IEEE Int. Conf. Robot. Auto.*, vol. 2, pp. 1-7.

[3] N. Ayache and O.D. Faugeras, "Maintaining representations of the environment of a mobile robot," *IEEE Trans. Robot. Auto.*, vol. 5, no. 6, pp. 804-819, 1989.

[4] D.J. Kriegman, E. Triendl, and T.O. Binford, "Stereo vision and navigation in buildings for mobile robots," *IEEE Trans. Robot. Auto.*, vol. 5, no. 6, pp. 792-803, 1989.

[5] C. Fennema, A. Hansen, E. Riseman, J.R. Beveridge, and R. Kumar, "Model-directed mobile robot navigation," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 6, pp. 1352-1369, Nov./Dec. 1990.

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

[7] M. Meng and A.C. Kak, "NEURO-NAV: A neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the enrironment," in *Proc. 1993 IEEE Int. Conf. Robot. Auto.*, vol. 2, pp. 750-757.

[8] M. Meng and A.C. Kak, "Fast vision-guided mobile robot navigation using neural networks," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 1992, pp. 111-116.

[9] T. Lozano-Perez and M.A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *CACM*, vol.22, no.10, pp. 560-570, Oct. 1979.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90-98, 1986.

[11] S.N. Gottschlich and A.C. Kak, "AMP-CAD: Automatic assembly motion planning using CAD models of parts," School of Elec. Eng., Tech. Rep. TR-EE-91-12, June 1991.

[12] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.

[13] D.E. Rumelhart, J.L. McClelland, and PDP Research Group, *Parallel Distributed Processing*, vol. I. Cambridge, MA: M.I.T. Press, 1987.

[14] R. Lippman, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol.4, pp. 4-22, 1987.

[15] P.K. Simpson, *Artificial Neural Systems*. New York: Pergamon, 1990.

[16] C. Lau, *An Introduction to Neural and Electronic Networks*. San Diego, CA: Academic, 1990.

Min Meng is a Ph.D. candidate in electrical engineering at Purdue University. She is a Research Assistant at the Purdue Robot Vision Laboratory. Her research interests are in the areas of computer vision, mobile robotics, artificial intelligence, neural networks, and fuzzy logic.

A.C. Kak is a Professor in the School of Electrical Engineering, Purdue University, Lafayette, IN. His current research interests are in machine intelligence, robotics, and computer vision. He has coauthored the books *Digital Picture Processing* (New York: Academic Press) and *Principles of Computerized Tomographic Imaging* (New York: IEEE Press). He is also the coauthor of a forthcoming book on robot planning and learning.