AIAA-94-1246-CP

MOBILE ROBOT EXPLORATION AND NAVIGATION OF INDOOR SPACES USING SONAR AND VISION

David Kortenkamp*, Marcus Huber, Frank Koss, William Belding,
Jaeho Lee, Annie Wu, Clint Bidlack and Seth Rodgers

Artificial Intelligence Laboratory
The University of Michigan
Ann Arbor, MI 48109

Abstract

Autonomous mobile robots need to integrate many different skills in order to perform complex tasks. In particular, they need to explore, sense, map and navigate in unknown or partially known environments. This paper describes a robot system that is designed to perform a find-and-deliver task in an office-building-like environment. The robot's initial orientation and location within the environment are not known, but the robot does have an *a-priori* map of the environment. We describe a sensor-based map representation that the robot uses while exploring its environment. We also describe how the robot determines its initial position and orientation within the environment, how it explores the environment for a visually-tagged object, how it recognizes the object and how it delivers the object. The robot also updates its map to reflect changes in the environment. While the entire robot system has not yet been integrated, each subsystem described in this paper has been implemented and tested.

Introduction

Autonomous mobile robots need to explore, sense, map, navigate and perform tasks in the environments in which they find themselves. Often these five functions are studied separately, with little or no attention given to how they are all integrated to produce a completely autonomous mobile robot. In this paper we concentrate not on completely describing any single aspect of robot exploration, sensing, mapping or navigation, but instead on how many different skills can be integrated into an autonomous robot that performs a sophisticated task. Unfortunately, time constraints prevented a complete integration of all of the described skills on the mobile robot. All of them, however, were tested individually and their integration is planned.

*Now at The MITRE Corporation, 1120 NASA Road 1, Houston TX 77058, e-mail: korten@aio.jsc.nasa.gov

Task description

The task our robot is designed to perform is to find a single, visually tagged object somewhere in a large, office-like environment and to "deliver" the object to a designated room. The robot is given a crude map shortly before being asked to perform the task. However, the map does not show obstacles that may block hallways or doors, nor does the map show all of the doors in the environment. The robot does not know its starting position or orientation with respect to the map. The delivery object is in one of the rooms and is a coffee pot marked with a black-and-white 'X'. The robot need not actually pick-up the coffee pot, only approach it. Some, but not all, of the doors are tagged with a visually distinct bar-code; bar-coded doors are noted on the map and the delivery room will be one of them. The robot has 30 minutes to complete the task, which was one of three tasks that comprised the AAAI '93 Robot Competition and Exhibition held in Washington DC on July 11-16, 1993.

The task is challenging to mobile robots because it requires the integration of many mobile robot skills. The robot must initially explore the environment and determine its position and orientation with respect to the *a-priori* map. The robot must then plan an exploration strategy that will allow it to examine each room for the coffee pot. This strategy must be flexible in the face of unexpected obstacles. Finally, the robot must use visual sensing to detect the coffee pot, plan a path from the object to the delivery room and then follow that path.

Robot description

Our robot is a Cybermotion K2A called CARMEL (Computer-Aided Robotics for Maintenance, Emergency and Life Support) (see Figure 1). It has a ring of 24 sonar sensors and a rotating B&W camera. Three computers are on-board CARMEL, one computer each for the motors and sonar sensors and a 486-PC for high-level processing. The 486-PC has a framegrabber and performs all image processing. CARMEL has a basic obstacle avoidance competence

Figure 1: The mobile robot CARMEL.

provided by an algorithm called VFH [2, 3, 4]. VFH constructs a certainty grid of sonar hits and uses it to continually compute a new direction that will take the robot towards its target while avoiding obstacles.

Overview

We first present the robot's representation of its environment. This is the representation that is entered into the robot from an *a priori* map. The robot must then *register* itself (i.e., determine its orientation) with respect to the environment; we give a basic registration algorithm. Next the robot must *localize* itself with respect to the *a priori* map; two different localization algorithms are presented. Once the robot is registered and localized, it can begin exploring the environment and looking for the coffee pot. We describe our vision algorithm to detect the coffee pot and also describe how we update the *a priori* map to reflect changes in the environment. Finally, the robot must navigate from the room that contains the coffee pot to the delivery room. This sequence is shown in the flow chart in Figure 2.

Representing the environment

The map in our representation is a graph of nodes. Each node represents a region of the environment that has a common sonar signature. Each link between nodes represents a bidirectional connection between the two regions. The first issue when creating such a representation is to decide on an appropriate sonar signature that will distinguish between different regions.

Detecting region boundaries

There have been many approaches to using sonar sensors to define distinctive places in an environment, including [10, 1, 6, 7, 9, 8]. In our approach, a region in the environment is characterized by having a common sonar signature throughout its extent, where the

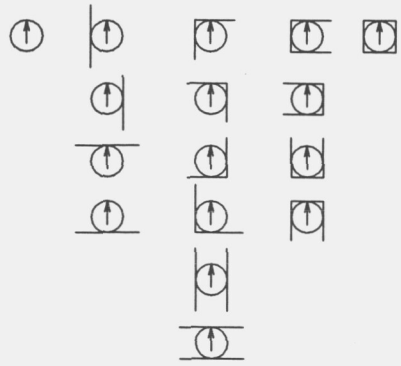

Figure 3: The sonar signature feature set detectable by CARMEL.

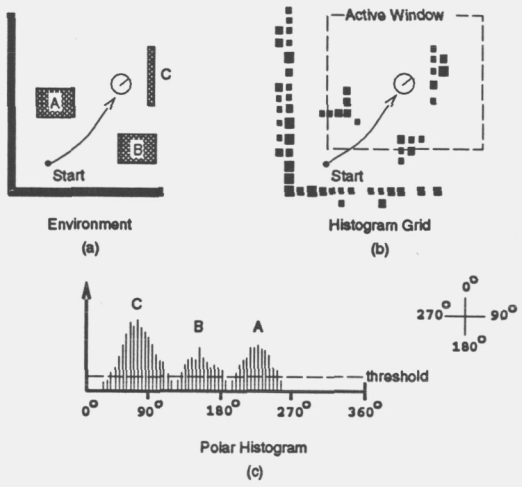

Figure 4: The VFH obstacle avoidance algorithm.

signature is the pattern of free or blocked space to the front, back and sides of the robot. Thus, there are 16 unique sonar signatures in a rectilinear environment (see Figure 3 for a complete listing of the 16 sonar signatures). Our approach is unique in that it is directly tied to an obstacle avoidance algorithm—the Vector Field Histogram (VFH) [4].

The VFH algorithm first creates a histogram grid, which is a certainty grid representation of the objects surrounding the robot as detected using the robot's sonar sensors. VFH then takes a local window of the certainty grid and converts it into a polar representation called the polar histogram. A certainty grid and its corresponding polar histogram are shown in Figure 4. The polar histogram shows the obstacles in each direction around the robot. To avoid obstacles, VFH simply chooses the free direction of travel that is nearest to the desired direction of travel. This same polar representation is used to produce the sonar signature.

A simple example will best show how the polar histogram is used to detect a region boundary. The robot is started down the hallway (the direction of the hallway is determined by an algorithm described in Section 3) and VFH automatically aligns the robot and

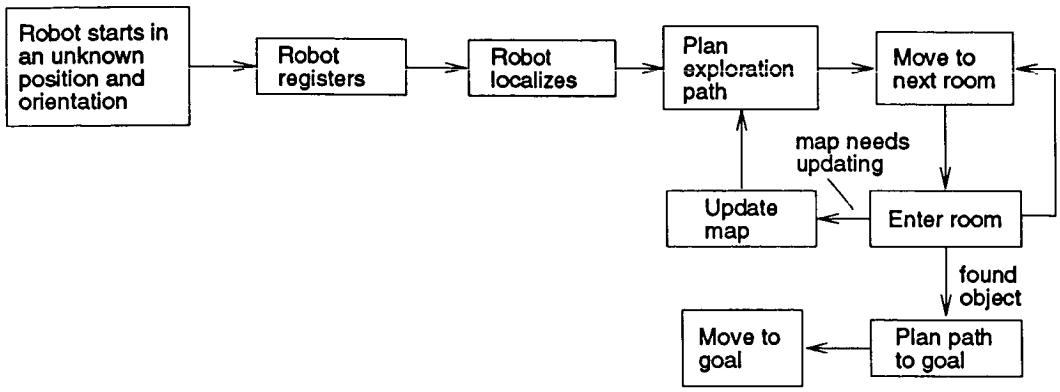

Figure 2: Flowchart for accomplishing the find and deliver task.

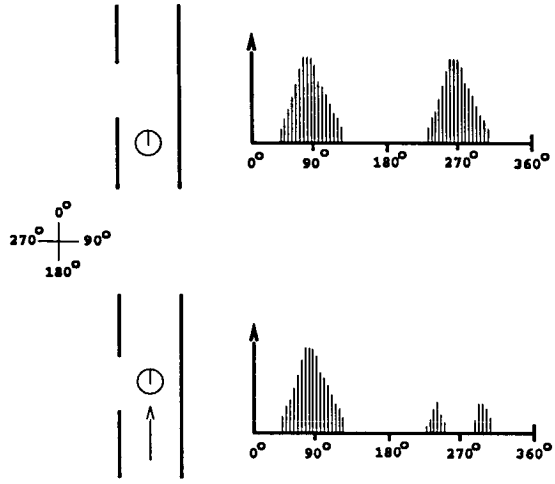

Figure 5: Detecting region boundaries using VFH.

positions it in the middle of the hallway. When the robot is positioned in the middle of a hallway the polar histogram has two "mountains" for the two walls of the hallway (Figure 5(top)). The presence of a "mountain" means that the robot is blocked to that side. In this example, the sonar signature is: (front = open, back = open, right = closed, left = closed). As the robot moves down the hallway and approaches the doorway, the "mountain" on that side of the robot will disappear (Figure 5(bottom)). So the sonar signature is now: (front = open, back = open, right = open, left = close). By "camping out" at the polar histogram segments corresponding to the front, back, left and right of the robot, changes in the sonar signature can be immediately detected.

In tests on the repeatability of this algorithm, CARMEL was asked to repeatedly stop at the same region boundary in the basement of our laboratory. Over ten consecutive runs, the largest difference in position along the hallway's axis between any two runs was 520mm and the largest difference in position perpendicular to the hallway axis along any two runs was 290mm. During these runs, obstacle avoidance was performed and the robot was running at a speed approaching 400 mm/sec.

While our boundary detection algorithm works fine in hallway environments, it has not been extensively tested in rooms. We rely instead on the dead reckoning capabilities of our robot to move into and out of rooms. Extending our approach to rooms as well as hallways is a topic of future research.

Map representation

Each region of the environment, which corresponds to a sonar signature, is represented by a node. A node contains the extent of the region (i.e., its length and width), a global (x,y) position of the center of the region and connections to neighboring regions. Figure 6 shows an example configuration of the arena where the robot will be working. As shown in Figure 7, the whole area is divided into regions based on the sonar signature. The regions are further distinguished by either being a hallway region or a room region. Each hall section has one node at the center of the hall (nodes with "H" prefix). Every exit of the room also has one node close enough to the entrance (nodes with "R" prefix). Each room section has some extra virtual nodes (nodes with "V" prefix) for each side of the walls of the room. These virtual nodes serve two purpose. First, they are used to figure out the boundary of the room. Since each node has its (x, y) coordinate, we need at least two room-nodes to calculate the boundary of a rectangular shape room. Second, they are used for map modifications which will be explained later in this paper.

Rooms can have up to four exits, one each to the north, south, east and west. If a room has more than one exit on each side it will be split into several virtual rooms. Large open space, such as lobbies, are also classified as rooms and may have to be split into several virtual rooms. For example, rooms 7,8 and 9 in Figure 7 are all virtual rooms contained within a single open area.

Registration

In order for our representation scheme to work, CARMEL must be able to determine the main axes of the corridors, so that it can start searching for region boundaries to its left, right, front and back. We call this *registration*. Currently, CARMEL can only

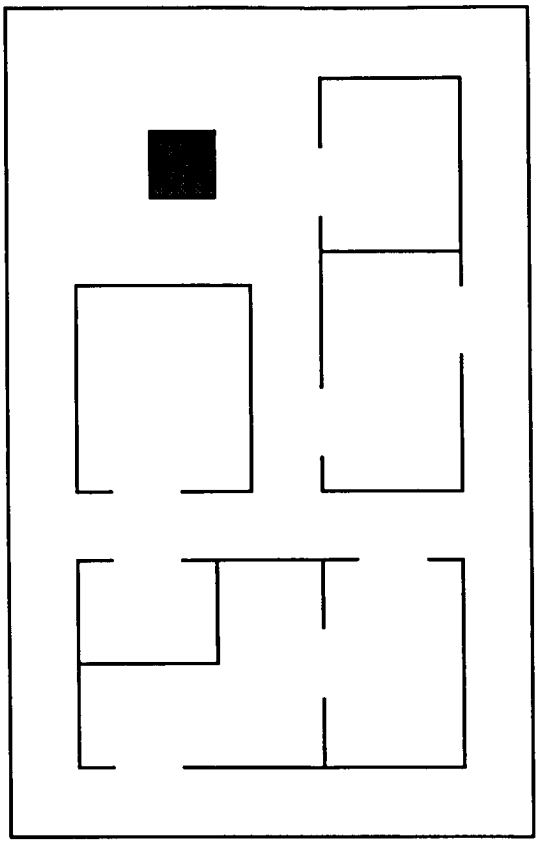

Figure 6: Arena Configuration

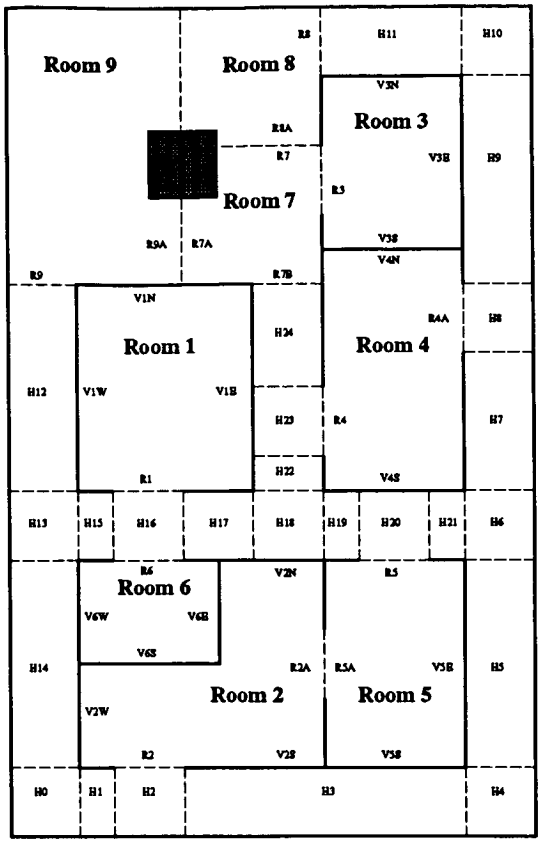

Figure 7: Arena Map

register itself in a hallway; if CARMEL starts in a room it must wall-follow until it enters a hallway. To register in a hallway, CARMEL starts to travel in any free direction. As it travels, the VFH obstacle avoidance algorithm will automatically align CARMEL between the two walls of the hallway. While moving, CARMEL saves its (x,y) positions along the way and fits a line to them. The orientation of this line is used to determine the axis of the hallway. CARMEL can reregister during the task to correct dead reckoning errors.

During initial registration, when the robot has no information as to its orientation in the environment, CARMEL also stores and averages the direction of free space. This should always fall along the axis of the hall. However, obstacles in the hallway, doorways, and intersecting corridors cause CARMEL to drift from the middle of the hall. Therefore line fitting, which is a simple chi-square fit, is used. This occurs after CARMEL has traveled a minimum distance and the rate of change in the average free-space direction falls below a threshold. If CARMEL becomes trapped before this time, it turns around and starts the process again assuming that it has reached a blockade in the passage or the end of a hall. If the orientation of the fit line is too far from the average free-space direction, it is assumed that CARMEL has not been traversing a hall or has "fallen" into a room or an intersecting hallway. In either case, all data are disregarded and the entire process, including wall-following if necessary, is repeated.

For reregistration during the task, the previous orientation can be used to judge the accuracy of the calculated orientation. When there is a large difference between the previous and new orientations, either the old one can be maintained or the process can be repeated. Maintaining the old orientation repeatedly is dangerous because CARMEL's orientation can become very inaccurate over a period of time.

We evaluated the registration algorithm in two situations. The first situation was in a corridor with no obstacles or openings into rooms or intersecting hallways. These experiments set out to confirm that the algorithm will correctly identify the hall axis regardless of CARMEL's initial orientation. In the second set of experiments, CARMEL was placed in a more complex area which included an obstacle and an opening into a room. The intention of these runs was to determine the robustness the algorithm, as it currently stands, in a more realistic situation.

Twenty-seven runs were carried out in the obstacle-free hallway. CARMEL's initial orientation with respect to the hall axis varied from 10 to 170 degrees in 20 degree steps. CARMEL's initial orientation was determined by eye and so is inaccurate by up to a degree or two. At each initial orientation, three runs were made. CARMEL determined the actual hallway axis to within six degrees in all but one run. In this case, the calculated hall axis was off by nine degrees. No run required more than approximately four

meters. The accuracy of the registration and the distance covered during a run were acceptable and within the limits of the environment that was expected to be encountered.

The second set of runs had two parts. The first part was carried out with CARMEL given an initial orientation of 30 degrees. The second part used an initial orientation of 70 degrees. Eight runs were carried out with each orientation. In the first part, CARMEL determined the hall axis to within five degrees each time, except one in which it wandered into the room through the opening. The average distance required was approximately five meters. In the second part, CARMEL entered the room twice, but determined the hallway axis to within four degrees in the other six runs. The average length of the successful runs was about 2.6 meters.

While these runs were far from exhaustive, they do show that this method of registration is useful. The most difficult problem is that of wandering into a room. This can either be avoided or detected, with the first preferable. The difficulty with preventing CARMEL from drifting into a room is that there is no simple way to distinguish between an opening into a room and a narrowing of the corridor due to obstacles. The former should not be entered while the latter should be. Detecting the entry into a room should be simpler. The chi-square fit provides a goodness of fit measure, namely χ^2. When CARMEL enters a room, its path is generally straight but roughly perpendicular to the hall axis. This should yield a very poor value for χ^2. The use of this value and the return of CARMEL to the hallway it left are still being investigated.

Localization

Once registered, the next critical issue is the determination of the correct location and orientation of the robot. We call this process *localization*. CARMEL accomplishes localization through the accumulation of information, in the form of local sonar signature features, during its initial movement through the halls of the "office" environment, and through observation of visual tags identifying doors. We would like CARMEL to localize itself as quickly as possible, however, so that in the absence of door markers we try to use the sonar signature features. In both approaches CARMEL is given a map representing the environment in which it will be placed. However, the map can be in error in that doorways may exist where they are not so indicated on the map, and doorways may be blocked where they are indicated on the map. The localization schemes must therefore deal with these problems.

We have implemented two approaches, one based upon heuristics and confidence factors, the other upon probabilistic reasoning using a belief network. Our localization methods only work in hallways, so that if CARMEL's initial location was within a room we would first have to find an exit using a wall following behavior. In this section we describe each of the approaches and show them in operation. Although both were implemented, neither have actually been fully integrated into the office exploration system.

Rule Based Localization

One method of localization that has shown to be successful is a rule based system. Before CARMEL makes a move during rule based localization, it computes scores over all of its possible starting locations in the *a-priori* map.

Creating a Score Distribution

Since direction is ambiguous to CARMEL at first, we run our scoring algorithm four times, rotating CARMEL's map to a new cardinal orientation each time. So for n possible starting locations, there are $4n$ total scores computed.

The basic scoring algorithm is a modified depth-first recursion, which runs as follows:

The first feature node seen by CARMEL is compared with the start node in this orientation. The comparison scores points depending on how many sonar signature features (i.e., walls or openings on the four sides of the robot) match and on the measured extent of a region. However, points scored for extent matching are fewer because we have determined that distance data tend to be more erroneous than the detected sonar signature features.

Each node adjacent to the current one on CARMEL's constructed map is checked to see if it has been examined yet in this orientation. If the neighboring node has not yet been examined, then it is compared with the node corresponding to it on the *a-priori* map. The algorithm continues this recursively for all of the paths the robot can follow from the start node. The score for each recursion is added to the total score for that start node in that particular orientation.

While the algorithm recurses, it also tries to answer this question:

> *If CARMEL started in this start node with this orientation, where would CARMEL be now?*

If the algorithm can determine this, it makes note of the fact. We refer to a current location inference of this type as a *location resolution*. There is no guarantee that a start node–orientation configuration will produce a location resolution.

In the event that a path to an adjacent node exists in CARMEL's map but a wall exists on the *a-priori* map, the routine attempts to figure out possible locations across the wall that might correspond to the node CARMEL saw. If a possible match is found, then the routine continues from there; otherwise, no points are scored for that area on CARMEL's map.

After all of the location–direction combinations are examined, the algorithm normalizes the raw scores by computing the mean and standard deviation of the score set. Then each original score is replaced by the number of standard deviations the score was above the mean. The resulting scores are now less dependent on the number of nodes seen by CARMEL.

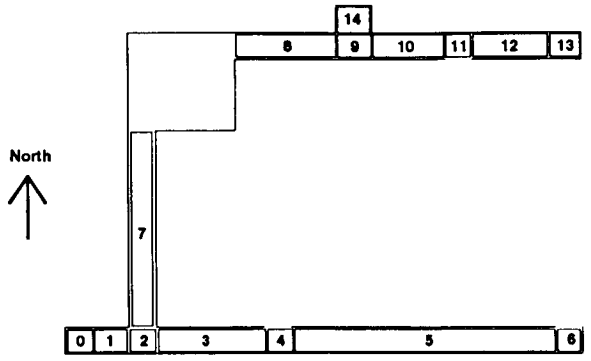

Figure 8: Map of the experimental space showing a priori regions.

Move Planning

CARMEL scores the possible moves it can make based on the location resolutions. CARMEL looks at each location resolution that it has and computes a shortest path to the nearest door marker for that resolution. The first move of this path is considered. This first move is either one of the four directions, or no movement at all (the special case where CARMEL is hypothetically in the vicinity of a door marker).

For each first move of a particular type, weight is added to that corresponding move possibility. The weight added depends on the score found for the location–orientation pair that the first move was derived from. If CARMEL saw a door marker at this stop, or if any location–orientation score for one of the first moves is above a certain threshold, then the "don't move" move choice is given a score of infinity, and CARMEL assumes localization is complete. In the former case, CARMEL assumes it is now at the location on the *a-priori* map where the door marker is. In the latter case, CARMEL assumes it is at the location resolution found for the above–threshold score.

As a final factor in move choice, we programmed CARMEL to select from these possible movement choices the highest scoring direction that has the shortest path to an unexplored region on CARMEL's map. This ensures that CARMEL covers unexplored space as efficiently as possible while searching out door tags. It also guarantees that CARMEL will not 'paint itself into a corner' or oscillate between adjacent nodes while exploring (two problems that occurred without this adjustment).

Experimental Results

Here are some results of a typical run with the rule based algorithm. The map given CARMEL is shown in Figure 8, and a door tags are located at nodes 0, 6, and the north end of 7.

We placed CARMEL in feature region 2 and aligned the robot so that it faced south on the map. CARMEL was told that it was starting somewhere along the south hall (nodes 0–6). CARMEL localized after the third move without using door markers.

In Table 1, the Move # column shows the current move. The Best Resolved column reports the best location–direction pair, i.e., CARMEL's top choice(s) for where it may be. The number is the feature node label corresponding to the map, and the direction is the direction CARMEL thinks it's facing. For example, 2–south means CARMEL thinks it may be at node 2 facing south.

The 'Best' Move(s) column indicates the best moves computed by the possible move scoring routine. Directions are displayed here as the true direction on the map for ease in interpretation. Multiple directions indicate a 'tie', in which case the final move is chosen from them based on exploration preference.

The Move Choice column is the actual move made by CARMEL. It may differ from the previous column if CARMEL chose an unexplored area over a high score direction.

It may seem strange at first that the 'best' move and the move choice are totally uncorrelated after the first move. One must remember that the 'best' move is a result of weighting all possible moves for all resolvable pairs, so it doesn't necessarily represent the true best move that can be made, especially in a symmetric environment. The moves chosen were carried out because CARMEL picked a direction, and preferred to explore new area on a 'next best' score rather than backtrack on a best one (which only may be best by a margin). Also, it is important to remember that the Best Resolved nodes are not the only nodes used in determining direction. All of the possible location resolved nodes are considered, weighted only by the location–orientation score associated with them. The Best Resolved values are therefore only displayed to show how quickly the algorithm can localize.

Belief Network Approach

In the second localization approach, the dependence of the sensed features on the world map, the robot's initial orientation, and the direction of travel of the robot as it attempts to localize itself, is modeled using a belief network [5, 11]. As the robot moves about and sees new features, the belief network accumulates a history of the features observed and the movements that the robot has made. These observations can then be propagated through the network, resulting in a probabilistic distribution over the possible locations the robot may be in currently. The robot considers itself localized when one of the locations achieves a level of confidence about a certain threshold. If CARMEL is not yet localized, it can use this distribution to determine the most likely direction in which to travel to facilitate better localization. Currently, this amounts to moving in the direction most likely to take it to a room tag, the most unambiguous localization feature detectable by CARMEL.

Belief network operation

The belief network that we used is shown in Figure 9. This network models the dependencies between the robots initial location, its initial orientation, and the sonar feature that it "sees". The modeling is accomplished both through the topology of the network as well as the probability tables (both conditional and priors). The conditional probability that a certain

Move #	Best Resolved	'Best' Move(s)	Move Choice
start	2-south,4-south	east,west	east
1	3-south,5-south	west	east
2	4-south	west	east
3	5-south	no move	no move

Table 1: Results from an experiment using rule-based localization.

feature is detected, given a particular location and orientation, is based on heuristic calculations of the correlation between what should be observed and that which is actually observed. The conditional probability that the robot is in a particular location, given a previous location and orientation, is a simple boolean function based on the map, where the probability is 1.0 if the locations are adjacent and joined by a path, and 0.0 if they are not.

Upon initialization, an observation of the robot's initial surroundings is placed in the FEATURE1 node of the network and then propagated throughout the network. The resulting posterior probability distribution in the LOCATION1 and ORIENTATION nodes reflect the evidence's impact upon the likely starting location and orientation of CARMEL. The robot can use this revised information in its planning to either facilitate improved localization or to switch to exploration of the office environment if the probabilities are suitably high enough to justify this.

In the situation where the resulting probabilities are such that CARMEL is still unsure enough of where it is to warrant further localization, CARMEL plans and executes a move in a direction most likely to take it to a door tag in the shortest amount of time (i.e., shortest distance). When it detects a change in the sonar features around it, it stops and makes another observation. The new sonar feature, as well as the motion the robot made to get to its current location, is fed to the belief network as evidence, propagated, and the resulting probability distributions analyzed. This cycle continues until either CARMEL becomes sure enough of its location based solely upon the sonar features so far detected, at which time it switches to exploration mode, or CARMEL detects a door tag, at which time it knows with certainty where it is, and similarly switches to exploration mode. The belief network in Figure 10 shows the belief network at iteration 3 in the process. As can be seen in this figure, the belief network grows at each iteration, adding new LOCATION, MOTION, and FEATURE nodes. The portion of the network that includes the MOVE node models the dependence of the robot's new location upon the previous location, the original orientation, and the move the robot made to get to the new location.

Experimental Results

We evaluated the belief network's ability to localize CARMEL in the halls of the University of Michigan's Artificial Intelligence Laboratory. The map for the region is shown in Figure 8, with the possible localization locations indicated by the numbered regions. Each of the labeled locations represents a region of

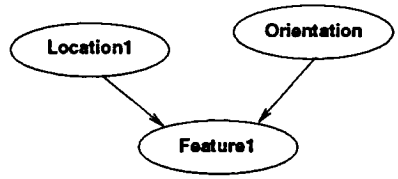

Figure 9: Initial belief network architecture.

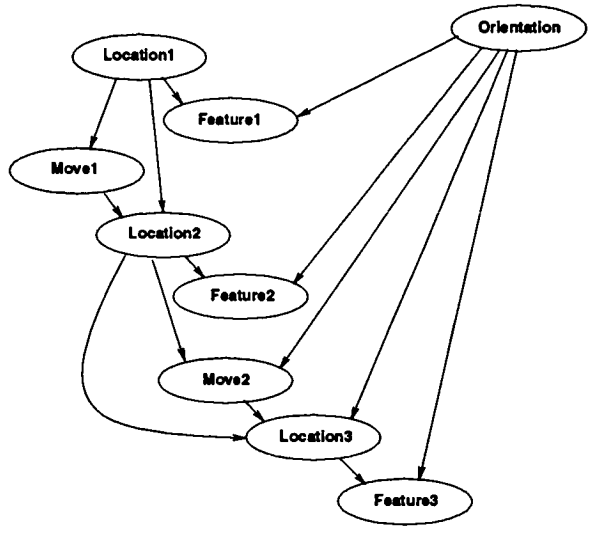

Figure 10: Belief network architecture showing the network at iteration 3 of the process. The Feature and Move nodes are instantiated as evidence, and the Location and Orientation nodes are inferred.

the map that has the same sonar feature type. Travelling between regions (locations), then, implies that the sonar feature must change at the transition point between the regions.

As an example, suppose CARMEL starts in location 2, the T-intersection at the South end of the map, and is initially facing South. The sonar feature observed would be that of a single blocked direction, that directly in front of it. Passing this evidence to the localization network, the resulting probability distribution for the current location is shown in Table 2(top), while the posterior distribution of the ORIENTATION node is shown in Table 2(bottom). These state that CARMEL is either in Location 2, 4, 9 or 11, and is most likely to be facing South.

If CARMEL then moves West (to it, East on the map), it will move until it enters region 3, which has a different sonar feature. The new feature, that of an East-West hall, and the West move that CARMEL made, are both given to the localization network and

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.04	0.04	0.13	0.04	0.13	0.04	0.04	0.04	0.04	0.13	0.04	0.13	0.04	0.04	0.04

North	South	East	West
0.27	0.44	0.15	0.14

Table 2: Probability distribution of location (top) and orientation (bottom) at the start location.

propagated. The new probabilities for the current location and orientation are show in Table 3, which says that CARMEL thinks that it most likely to be at Location 10, and is most likely to be facing South. Again moving West (to it, East on the map), so that CARMEL sees the new feature at Location 4, yields distributions shown in Table 4.

Taking this probability distribution, CARMEL now has greater than 90% confidence that it is at Location 4 and was initially facing South. If there was a door tag at the entrance to the room at Location 4, it could then visually verify that this inference is correct. CARMEL can now reorient itself correctly to the map and position itself at the transition point between the current location (Location region 4) and the previous location (Location region 3). Note that since the belief network has nodes representing each of the previous locations also, it is easy to reason about where the robot has already been simply by looking each of the nodes and determining the highest probability state in each. This facilitates exploration updating in that extra time to perform a backtracking search through the map with the previously performed motions doesn't have to be done in order to see what has already been explored.

Exploration and Navigation

Once CARMEL is localized it can begin to look for the coffee pot. The first step in this process is to plan an exploration path. An exploration path is an exhaustive sequence of rooms to visit from the robot's current location. The sequence is determined based on the travel distance from the current location. CARMEL first selects the closest room (in terms of travel distance, not the Cartesian distance), then adds another room closest to the selected room, and so on. After planning, CARMEL traverses the exploration path stopping in each room to scan with its camera for the coffee pot. Exploration is terminated when the coffee pot is found. The exploration path can be modified to accommodate unexpected blockages or openings.

Planning exploration path using closest-node-first (hill-climbing) method does not necessarily generate the optimal path in terms of total traveled distance*, but in practice this method turned out very fast and the resultant path was quite reasonable. For example, the exploration path from, say, "H18" (in the middle of the map in Figure 7) would be (R4, R7B, R3, R8A, R9A, R1, R6, R5, R2A). This sequence specifies only the room nodes to visit in that order.

*Finding the optimal exploration path amounts traveling sales man problem

Updating the map

While exploring CARMEL can update its *a-priori* map to reflect blocked hallways and doorways and to note additional doorways that were not in the original map. For blocked hallways or doorways, the connections between the two nodes in the map are cut and the sonar characteristic of the node is modified appropriately. In the case of unexpected openings, new nodes are created, assigned the appropriate signature and connected to adjacent nodes. This information helps CARMEL find the most efficient route to the delivery room once the coffee pot has been found or to replan its exploration path. For example, if the robot sees an unexpected opening to a room from a hall node, it finds the nearest node (either virtual or real room node) of the room and create a new link to that node. Note that each node can have maximum four connections (roughly corresponding to North, West, South, and East) to other nodes. Figure 11 shows a part of the map before the robot sees an unexpected opening at west side of the hall "H5". It first figures out which room is next to west of "H5" from the boundary information of each room. In this case, it is Room 5 and "V5E" is the closest node of that room. Now "H5" should be divided into three sections since the sections are divided based on the sonar-reading changes and new opening will change the sonar signature as shown in Figure 12.

Visual sensing

As CARMEL enters each room it scans for the coffee pot. CARMEL's vision system finds predefined markers (a black 'X' on a white background in the case of the coffee pot) in the environment and determines their pose (3D position and orientation) relative to the robot. We will describe the algorithm for detecting the 'X' here. The algorithm for determining the pose of the 'X' is described in [12].

Marker detection

The marker detection phase is composed of two main routines: the connected components routine and the marker identification routine. The detection phase must be both fast and accurate for the pose estimation algorithm to be useful for real-time tasks.

To maximize speed, we make only one pass through the entire image. During the pass, the image is thresholded and connected components are found and labeled. One pixel components are ignored and not labeled. Size thresholding then filters out most of the non-marker components. Only one pass is made through all possible connected components.

To identify or reject the remaining markers, we use a weighted pattern matching template. An *nxn* template matrix is created for each marker (see Figure 13).

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
< 0.01	0.04	0.01	0.20	0.01	0.20	0.01	0.03	0.05	0.01	0.36	0.01	0.04	< 0.01	< .01

North	South	East	West
0.33	0.62	0.04	0.02

Table 3: Probability distribution of location (top) and orientation (bottom) after first move.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0.01	< 0.01	0.08	< 0.01	0.57	< 0.01	0.08	0.01	0.02	0.07	< 0.01	0.13	< 0.01	0.01	< 0.01

North	South	East	West
0.18	0.800	0.01	0.01

Table 4: Probability distribution of location (top) and orientation (bottom) after two moves.

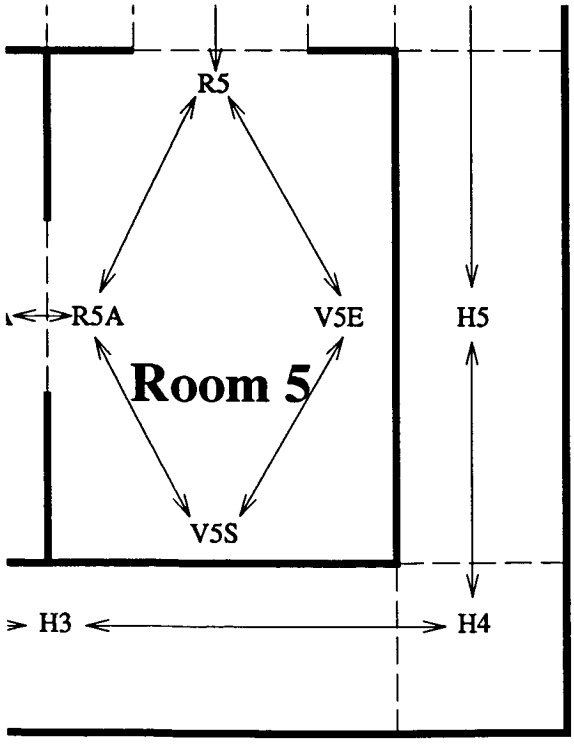

Figure 11: Before Modification

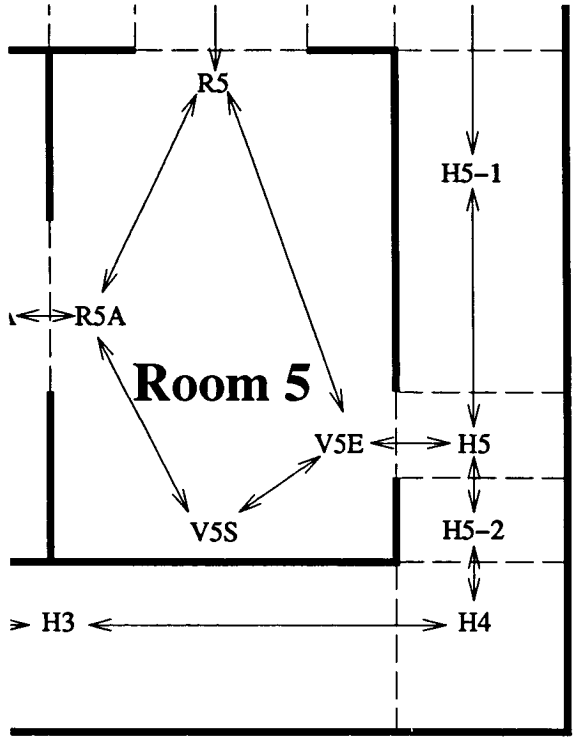

Figure 12: After Modification

1	1	-2	-8	-2	1	1
1	2	0	-1	0	2	1
-2	0	3	1	3	0	-2
-8	-1	1	8	1	-1	-8
-2	0	3	1	3	0	-2
1	2	0	-1	0	2	1
1	1	-2	-8	-2	1	1

Figure 13: Weighted pattern template for the 'X' markers. Positive values indicate expected black areas; negative areas are expected to be white. Certainty increases with magnitude.

b	w	w	w	w	b	b
b	b	w	w	b	b	w
w	b	b	b	b	w	w
w	w	b	b	b	w	w
w	w	b	b	b	b	w
w	b	b	w	w	b	b
b	b	w	w	w	w	b

Sample marker

$$Certainty_x = \frac{\sum_r \sum_c f_x(r,c)}{\sum_r \sum_c |x_{rc}|} = \frac{92}{96} = 0.9583$$

$$f_x(r,c) = \begin{cases} |x_{rc}| & \text{if correct color} \\ 0 & \text{otherwise} \end{cases}$$

Figure 14: Sample marker with calculated 'X' certainty value. "b" indicates a black pixel; "w" indicates a white pixel. r counts rows; c counts columns.

Increasing n increases the resolution of the template, but also increases the process time. We found $n = 7$ to be a good compromise. This weighted template indicates which areas are expected to be black and which ones white. The weights for our matrix are currently determined by trial and error, but we could easily replace these with machine generated weights if a learning program were implemented. The marker template which a component most resembles is selected as the "guess" for that component. The program generates a certainty measure with each guess (see Figure 14) and uses this measure to accept or reject the guess. Each marker can have one or more templates. The additional templates may be used to improve marker recognition from views other than straight on.

We also use additional heuristic information in identifying the markers. Some heuristics were not learned or incorporated until after the program had been tested. For example, diagonal lines often scored high enough certainty values to be considered 'X's. Once we realized this, adding a specific test to verify that each possible 'X' is not a diagonal line solved this problem. To avoid slowing down the program too much, specific heuristic tests were kept to a minimum.

Navigation

Once the coffee pot is found, CARMEL uses the vision algorithm's estimation of the pot's relative location to approach it. Since CARMEL doesn't have a manipulator, it is assumed that once CARMEL has approached the coffee pot it has "grabbed" it and can then deliver it to the delivery room. CARMEL plans the shortest path to the delivery room using a standard shortest-path algorithm. CARMEL then follows the path by moving from region to region and detecting region boundaries with its sonar sensors. There are numerous error recovery routines that can cope with changes in the environment and sensor errors.

Conclusion

Unfortunately, time constraints leading up to the competition prevented the complete integration of all of the described skills. In particular, the robot did not perform registration or localization at the competition. Instead the robot was told its orientation and position. During the actual competition, the robot explored several rooms before becoming hopelessly lost, at which time the run was terminated. The most difficult problem encountered was tuning the sonar-based region-finding algorithm to the particular environment. While the algorithm had worked fine in our testing environment (the basement of our laboratory), different characteristics of the competition environment caused many false detections (i.e., defining the start of a new region when there wasn't one) and a few missed detections (i.e., not detecting a new region when there was one). Since the robot's localization depended on matching the regions it found with the *a priori* map, it became lost very quickly.

Our experience demonstrates an important lesson in mobile robotics—if the low-level sensing of the world is not working correctly, then high-level reasoning or map making will be unsuccessful, no matter how elegant their implementations. Our experience also underscores the fact that routines that are demonstrated to work in one environment will not necessarily work in another environment, even if that environment is quite similar. In addition, our experience was not unique—no robot at the competition (out of a dozen entries) successfully completed the task. Obviously, there remains much work to be done in mobile robot exploration and navigation of indoor environments.

Acknowledgments

The authors wish to thank the other members of the CARMEL team, including Dr. Johann Borenstein, Roy Feague, Rob Giles, Kevin Mangis, Patrick Kenny, and Alex Ramos. Dr. Terry Weymouth was the faculty advisor for the team and contributed much to the effort. Support for the CARMEL team was provided by The University of Michigan College of Engineering; the American Association for Artificial Intelligence; ABB Graco Robotics, New Berlin, WI; the NTN Technical Center, Ann Arbor, MI; and

the Environmental Research Institute of Michigan (ERIM), Ann Arbor, MI. Co-authors Huber and Lee are supported by DARPA grant no. DAAE-07-92-C-R012. Purchase of CARMEL and support for research using it is provided by Department of Energy grant no. DE-FG0286NE37969, which also supported co-author Kortenkamp.

References

[1] Kenneth Basye, Thomas Dean, and Jeffrey Scott Vitter. Coping with uncertainty in map learning. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 1989.

[2] Johann Borenstein and Yoram Koren. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Journal of Robotics and Automation*, 7(4), 1991.

[3] Johann Borenstein and Yoram Koren. The Vector Field Histogram for fast obstacle-avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3), 1991.

[4] Johann Borenstein and Yorem Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), 1989.

[5] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, Winter, 1991.

[6] Thomas Dean, Kenneth Basye, Robert Chekaluk, Seungseok Hyun, Moises Lejter, and Margaret Randazza. Coping with uncertainty in a control system for navigation and exploration. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1990.

[7] David Kortenkamp, L. Douglas Baker, and Terry Weymouth. Using gateways to build a route map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1992.

[8] Benjamin J. Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8, 1991.

[9] Maja K. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3), 1992.

[10] David P. Miller. A Spatial Representation System for Mobile Robots. In *Proceedings IEEE International Conference on Robotics and Automation*, St. Louis, 1985.

[11] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[12] Annie Wu, Clint Bidlack, Arun Katkere, Roy Feague, and Terry Weymouth. Vision-based object pose estimation for mobile robots. In *Proceedings of the AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space (CIRFFSS '94)*, 1994.