Integration of Reactive Behaviors and Enhanced Topological Map for Robust Mobile Robot Navigation

Byeong-Soon Ryu and Hyun Seung Yang

***Abstract*— A navigation system for an autonomous mobile robot working in indoor environments is presented. The system takes advantage of *deliberative* (plan-based) approach and *reactive* (behavior-based) approach. In the reactive part of the system are local *behaviors* that are independent, action-generating entities. We also provide higher-level deliberative modules that make interactions manageable so the system can accomplish more meaningful tasks. The deliberative modules control the activation and deactivation of individual local behaviors based on current *situations*, representing the position of the robot and the path to the goal.**

The situation is determined by a mapping subsystem consisting of an *enhanced topological map*, a localization module and a planning module. The use of the explicit world model, especially in a topological manner, makes it possible to reliably localize the robot and plan an efficient path. This paper also provides a detailed description of a localization module based on dead reckoning, and a planning module that selects an efficient and reliable path.

***Index Terms*—Enhanced topological map, mobile robot navigation, reactive behavior.**

I. Introduction

IN ORDER for an autonomous mobile robot to accomplish given tasks in the real world environment, it must make decisions and execute actions in real-time as well as cope with various uncertainties. These uncertainties arise from various reasons: knowledge about the environment is partial and approximate; environmental changes are dynamical and can be only partially predicted; the robot's sensors are imperfect and noisy; and the robot's control is imprecise [1].

A. Control Architectures

Traditional *deliberative approaches* to planning and controlling mobile robots have been criticized for not being able to adequately cope with uncertain and complex environments. As an alternative, a number of *reactive approaches* have been proposed. In the reactive approach, systems consist of collections of local *behaviors* that are independent and action-generating entities [2], [3]. This approach handles uncertainty and unpredictable changes well by giving up the idea of modeling and reasoning about the environment and the future consequences of actions. The global, goal-directed behaviors of such systems are not explicitly planned; instead, they typically emerge from interactions among local behaviors.

The idea of achieving global goals by cooperation among local behaviors is very attractive because it can reduce the complexity of the problem and improve the robustness of the system. To be successful, however, such systems should provide a mechanism that can make cooperation manageable [4]. The problem is that as complexity increases, interaction between behaviors also increases to the point where it becomes difficult to predict overall system behavior. Furthermore, it is quite difficult to make systems where such meaningful behavior as office delivery can emerge from simple behaviors. For those reasons, purely reactive systems that provide only very simple combination mechanisms have been limited to such simple applications as obstacle avoidance and open-space exploration.

One way to solve this problem is to limit interactions by adding a top-down constraint that takes advantage of regularity and knowledge in the operational domain in order to coordinate actions. This strategy is embodied in the deliberative approach, which hierarchically partitions problems into manageable subtasks and explicitly controls interactions between them [4]. As the deliberative modules utilize environmental models, such meaningful tasks as office delivery can also be accomplished.

A number of systems that realize the idea of integration have been proposed since the end of the 1980's [1], [5]–[8]. Although they share the idea of integration, each differs from the others in the way it represents world models, selects relevant behaviors, and combines outputs of selected behaviors. Payton represents the world in grid-type models, and then plans paths on the grids [5]. His behaviors are reflexive and combined with a "winner-take-all" mechanism. Arkin utilizes a meadow map but it is unclear how this map is related to the motor and perception schemas [6]. Behaviors, called motor schemas, are combined by the so-called "potential field." Mataric represents the world with active nodes linked topologically [7]. She provides localizing and planning methodologies, but they are too simple to cope with sensory uncertainties. Behaviors are controlled under the subsumption architecture. Saffiotti has not kept any global model but utilizes the *local perceptual space* that provides the

Manuscript received August 28, 1997; revised June 2, 1999.

B.-S. Ryu is with the Center for Artificial Intelligence Research, Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea.

H. S. Yang is with Center for Artificial Intelligence Research, Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea.

Publisher Item Identifier S 1083-4427(99)06984-2.

context [1]. The activation of his fuzzy behaviors is determined by the context and the output of relevant behaviors are then combined into a resulting command by a fuzzy "or" operation and defuzzification. His work has influenced us to use similar selection and combination methods. His work, however, does not provide the localization and planning mechanisms which our work does. Chung *et al.* integrate a topological map with reactive behaviors [8]. Their behaviors produce outputs of two types: vector and permission ring. The integration is achieved by two modules: a *coordinator* selects relevant behaviors and then a *blender* combines the outputs generated by the selected behaviors. Although they provide the means of using the topological map and localizing the robot, their localization scheme has the weaknesses of ignoring the metric information.

B. World Models

In this paper, the robot's working environment is represented in an *enhanced topological map* (ETMap). The ETMap has basically a topological structure that consists of node and links. However, it is more enhanced than traditional topological maps in that it utilizes rough metrical information and provides localization and planning methodologies that are efficient and reliable even with imprecise sensors. In fact, a variety of maps and corresponding algorithms to localize the robot and plan the optimal path have been proposed for mobile robot navigation. These approaches are of three types: grid-based approach, geometric approach, and topological approach.

The grid-based approaches [9], [10] are adequate for local navigation like obstacle avoidance. However, they are rarely used in localization since they have a high computational cost in matching the sensed features against the internal environment model. In the geometric approaches [11]–[13], the environment is represented with geometric beacons on a three-dimensional (3-D) coordinate system; then the position of the robot is estimated by matching the sensed features against the world models. Various techniques (e.g., Kalman filtering) have been proposed to combine over time the estimates provided by a number of sensory matching techniques. However, those algorithms are vulnerable to sensing errors and environmental uncertainties because they rely on precise metrical information.

In contrast, topological approaches are known to be able to overcome the fragility of purely geometrical methods since they do not require precise metrical information [14]–[18]. Furthermore, the elements of the topological map are strongly related to the semantics of the environments while the other two maps put more emphasis on geometrical information. Therefore, the topological map is more capable than the others in managing reactive behaviors.

Previous work on the topological map, however, has not provided reliable localization methods. Most researchers have concentrated only on adjacency relationships to distinguish between nodes with landmarks of identical type and similar sensory information. However, when the adjacent nodes have similar sensory information, their systems may not distinguish between the nodes because the robot's sensors are imperfect and may miss several landmarks. To solve this problem, we propose a localization algorithm that utilizes metrical information and dead reckoning. We also propose a planning algorithm that considers sensory uncertainties. The localization module first finds a possible location and then fine-tunes its location by detecting the landmarks. The planner selects a path so that there are no ambiguous landmarks within the selected interval. Thus, the localization modules don't have to disambiguate among similar landmarks.

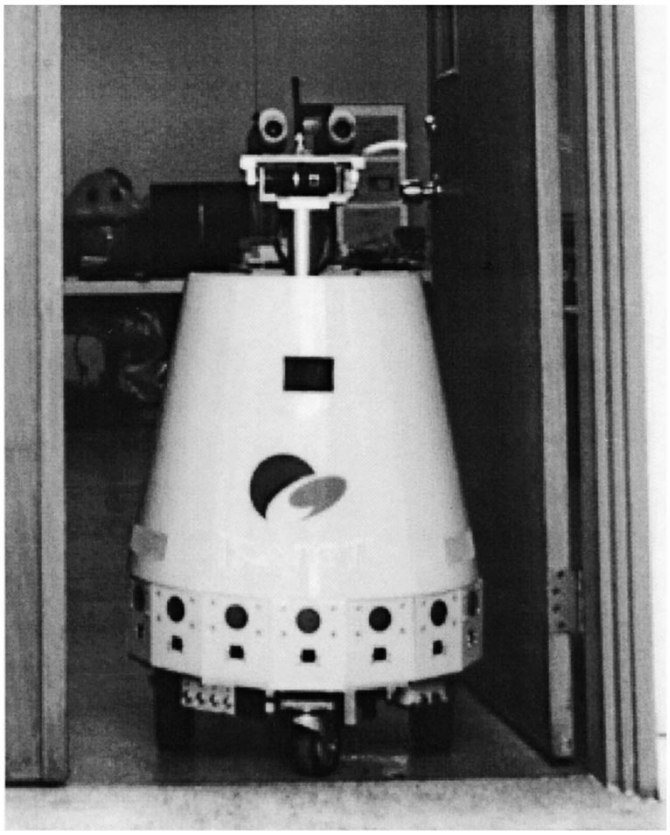

Fig. 1. CAIR-2.

C. Mobile Robot CAIR-2 and the Simulator

We have implemented and tested the proposed navigation system both on a simulator and a real mobile robot, CAIR-2, shown in Fig. 1. It is approximately 100 cm tall and 60 cm (diameter) wide, and has two driving wheels. It is equipped with 16 ultrasonic rangefinders for measuring ranges between 30 and 300 cm and eight infrared sensors for proximity sensing within 30 cm. Two video cameras with a pan-and-tilt mechanism attached on the head are used for object recognition, scene understanding, target tracking, and stereo vision.

The simulator shown in Fig. 2, has the same wheel configuration as CAIR-2. It also has a positional error whose degree can be set manually. In the simulator, the robot's footprints are marked in dotted lines and the estimated positions of the robot are marked in small circles. In the current implementation, it has only ultrasonic rangefinders, so the landmark detectors utilize only these ultrasonic rangefinders. Unfortunately, the ultrasonic rangefinders don't provide enough information to

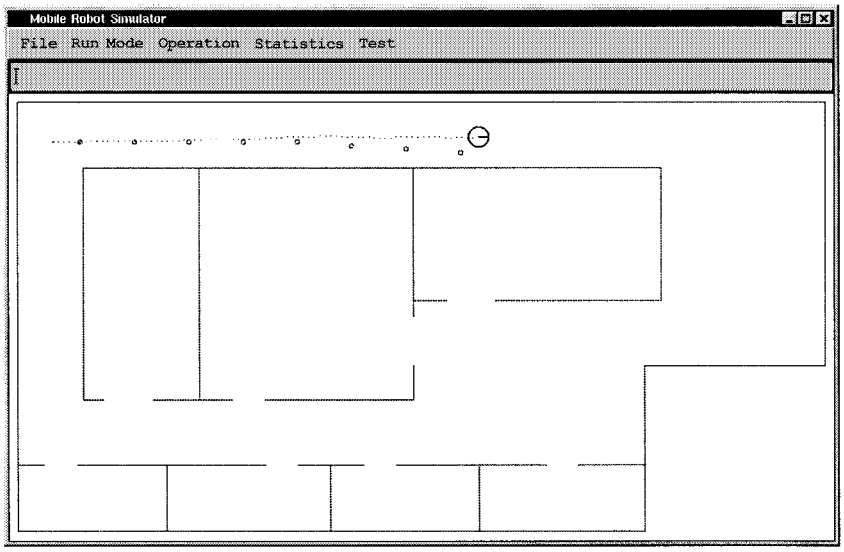

Fig. 2. Mobile robot simulator which has almost the same wheel configuration as CAIR-2. The robot's footprints are marked in dots and the estimated positions of the robot are marked in small circles.

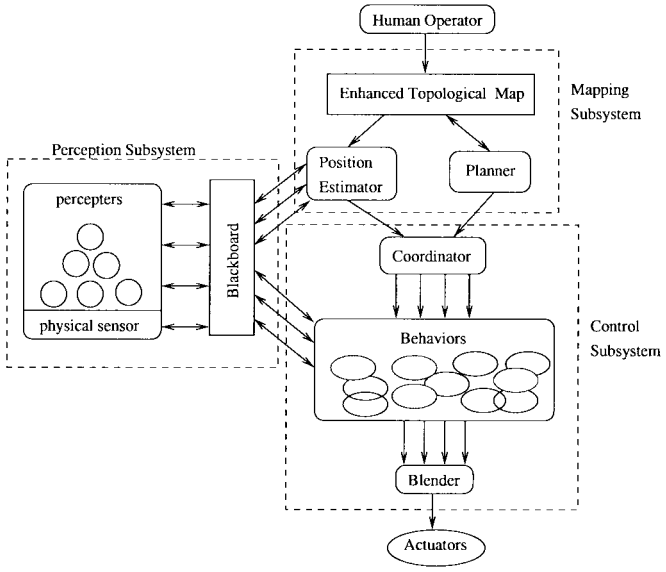

Fig. 3. Block diagram of the proposed navigation system architecture.

distinguish locations. However, several experiments show that the control and mapping systems proposed here are capable of reliably guiding the robot using only the ultrasonic rangefinders.

II. Navigation System Architecture

Fig. 3 depicts a block diagram of the proposed navigation system architecture. The system consists of three functional subsystems: a mapping subsystem, a control subsystem, and a perception subsystem. We have designed each subsystem to be independent from the others, so that we can modify one of them without changing the rest.

Most tasks to be accomplished by mobile robots are navigational ones; i.e., the robots must navigate autonomously from one place to another. Thus, a human operator gives the robot his instructions with a world model. In this paper, we propose an *enhanced topological map* as a world model. The map represents the world in a topological manner with rough metrical information and provides information to two modules: a *position estimator* and a *planner*. Those modules determine the current situation of the robot, and are capable of coping with environmental and sensory uncertainties.

The control subsystem is composed of various modules called *behaviors*, each of which has its own goal and runs independently from the others. As mentioned earlier, behaviors are good at coping with uncertain and dynamic environments, however, some additional mechanisms to manage the behaviors are required to be really useful in the real world applications. For this reason, the proposed system provides two modules. One is a *coordinator* that controls the activation and deactivation of certain behaviors according to the current situation determined by the mapping subsystem, and the other is a *blender* that combines the multiple action commands produced by the activated behaviors into one composed for the actuators.

The perception subsystem also consists of independent computational modules called *perceptors*. They are structured hierarchically from raw sensory data to high-level abstract information. Since each of these has its own thread of computation, delay in a perceptor does not cause delay in the other perceptors. Therefore, the *Avoid Obstacle* behaviors can obtain the raw ultrasonic data whenever they want, even if some time-consuming perceptors like a *scene analyzer* is still processing visual data. Use of a global storage called *blackboard* for communications among the perceptors and other two subsystems also makes the system more manageable and extensible.

III. Mapping Subsystem

A. Enhanced Topological Map

The basic structure of the *enhanced topological map* (ETMap) is a topological model consisting of nodes and links.

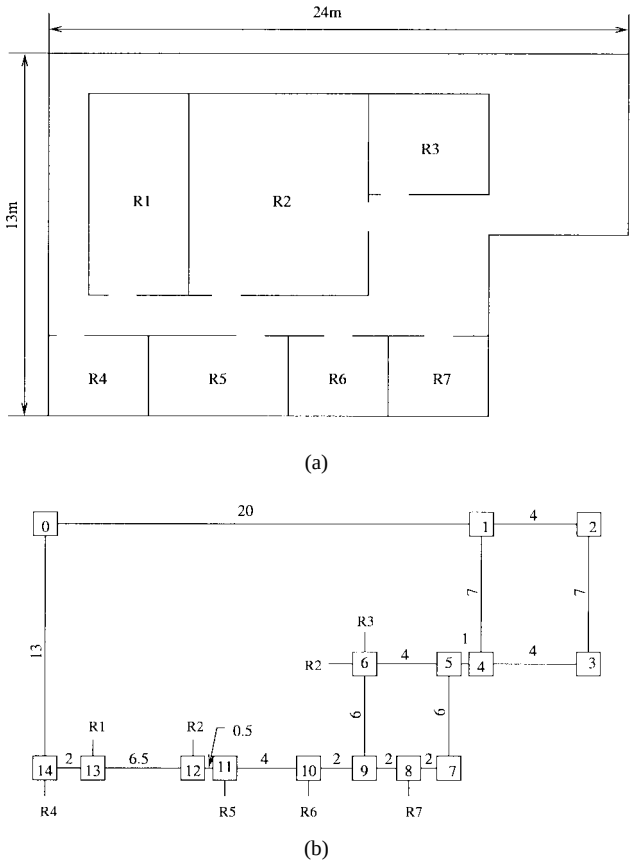

Fig. 4. (a) Typical indoor environment and (b) its corresponding ETMap. In both (a) and (b), R1–7 represent the rooms. In (b) boxes represent nodes while lines represent links and numbers on the lines are approximate distances in meters between the nodes.

However, it is different from traditional topological maps in that it is designed with special emphasis on the efficient and reliable localization and navigation of mobile robots even with imprecise sensors and simple perception modules. As a result, it has the following three characteristics:

- topological structure with rough metrical information;
- reliable localization algorithm based on dead reckoning;
- planning algorithm for reliable localization and navigation which considers sensory uncertainties.

Fig. 4 shows a typical indoor environment and the corresponding ETMap. As shown in the figure, the ETMap has a topological structure where nodes are *landmark places* (LP's) and arcs are *adjacency links* (ALinks). The LP's are drawn in rectangles while the ALinks are drawn in lines. In addition to a pure topological structure, the ETMap has such rough metrical information as the length and orientation of the ALinks, although need not be precise. Indeed, in our implementation, the length precision is about 10 cm and four primary directions (north, south, east, and west) are exploited.

1) Landmark Places: The Landmark Places (LP's) that correspond to the nodes of the ETMap are places where several landmarks can be detected and the robot can localize itself. A variety of sensors are available to detect landmarks. Examples of such landmarks detectable by ultrasonic range finders are "DOOR," "OPEN (JUNCTION)," and so on.

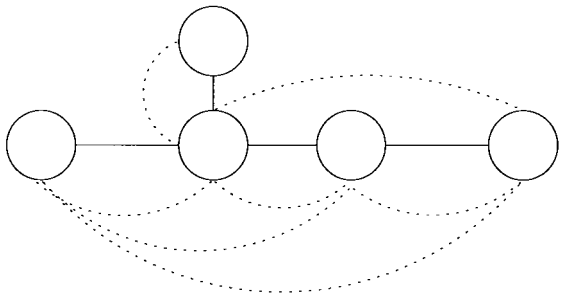

Fig. 5. Adjacency links and neighborhood links.

Each of the LP's has not only landmark types but also such landmark information as landmark direction and the extent to which landmarks are observable.

An LP is somewhat similar to the *distinctive place* (DP) proposed in [15]. However, the LP has explicit information about landmarks while the DP is defined implicitly with a distinctiveness measure. On the other hand, *Gateway* proposed in [19] is a subset of the LP. While Gateway only marks the transition from one space to another, the LP also provides the means of localizing the robot even within adjacent spaces.

2) Adjacency Links: LP's are linked together by Adjacency Links (ALinks). ALinks represent spatial adjacency, i.e., there is no other LP between two LP's linked with an ALinks. As in Fig. 4, ALinks have rough metrical information such as length and orientation of links. In addition, ALinks also have the same information about landmarks as LP's.

The reason why the ALinks have information about landmarks is that there are some transient landmarks that can be detected only by detecting transitions in perceptual states. For example, when we consider node 1 in Fig. 4, the robot can detect "SOUTH_OPEN" and recognize node 1 when the robot moves from node 0 to node 1. However, it can't recognize node 1 with "SOUTH_OPEN" when it moves from node 2 because the link between node 2 and node 1 also has "SOUTH_OPEN."

3) Neighborhood Links: Besides ALinks, the ETMap has auxiliary links called *neighborhood links* (NLinks). While ALinks denote physical adjacency, NLinks denote logical adjacency, which we call *neighborhood*. We define the neighborhood of an LP (starting LP) as a set of LP's that can be reached by the robot from the starting LP without an abrupt change of orientation. Fig. 5 illustrates the difference between ALinks and NLinks. Although we consider only linear paths, the same concept of NLinks can be adapted to curved paths such as outdoor roads. NLinks are set automatically using the ALinks when the robot initializes the ETMap.

The major reason why we use NLinks is to find more efficient paths. Generally, the robot pays a high price to detect a landmark. The more information the landmark gives, the higher the cost. This cost influences not only computation time but also robot speed. For example, if the robot detects a landmark with its vision sensor that takes about 1 s to process and has to take an image at least every 50 cm not to miss the landmark, then the robot's speed can't exceed 50 cm/s.

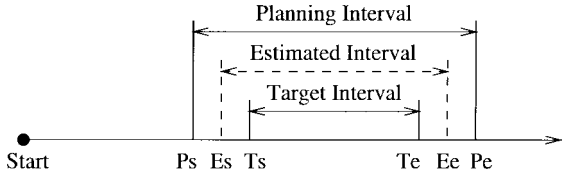

Fig. 6. Distance intervals.

However, if the robot can detect a landmark and localize itself using only cheap sensors like ultrasonic range finders, it can move faster. Moreover, if the robot can localize itself even while skipping some landmarks, it can move at maximum speed. Therefore, by navigating along NLinks, we can obtain a more efficient path.

Another important difference between ALinks and NLinks is that the former represent the structure of the environment while the latter represent the information related to navigation. NLinks have a selected set of landmarks for efficient and reliable navigation while the ALinks have all landmarks related to the links. In addition, each NLinks keeps a list of behavior modules selected by the coordinator to be activated when the robot navigates along the NLink.

4) Distance Intervals: To plan the path and localize the robot, the system introduces some notions of *distance intervals* (DI's) as shown in Fig. 6. The *Target Interval* (TI) denotes the interval where the robot can detect landmarks. The other two intervals are wider than the TI due to dead-reckoning error. The *Estimated Interval* (EI) is the interval within which the estimation lies when the robot is actually in the TI. Since the robot knows only the estimation of its position, it uses the EI to localize itself. The *Planning Interval* is the interval within which the robot actually lies when the estimation of its position lies within the EI. The PI refers to the actual position of the robot when it estimates that its position is within the EI. So the PI is used by the path planner.

Assume that the dead-reckoning error is ϵ and proportional to the distance traveled, then each interval is defined as follows:

$$\begin{aligned}
\mathrm{TI} &= (T_s, T_e) \\
\mathrm{EI} &= (E_s, E_e) \\
&= ((1-\epsilon)T_s, (1+\epsilon)T_e) \\
\mathrm{PI} &= (P_s, P_e) \\
&= \left(\frac{E_s}{1+\epsilon}, \frac{E_e}{1-\epsilon}\right) \\
&= \left(\frac{1-\epsilon}{1+\epsilon}\,T_s, \frac{1+\epsilon}{1-\epsilon}\,T_e\right). \qquad (1)
\end{aligned}$$

B. Localization

Localization is a critical issue in mobile robot navigation. Although various localization schemes for topological maps have been proposed, most previous works using topological approaches have inherent limitation in that it concentrates only on finding distinctive places to localize the robots but seldom uses metrical information. Even if they provide probabilistic or heuristic tools to select the best possible one among similar places, they have inherent limitation since they ignore metrical information. Suppose a robot is navigating down a corridor that has two adjacent and similar doors as shown in Fig. 7. Since the robot's sensors are imperfect, one of the doors can be missed. For example, if the robot should miss door 1 and detect a door in front of door 2, it might be unable to decide whether it is door 1 or door 2 unless it refers to such metrical information as the traveled distance.

On the contrary, the algorithm proposed in this paper can solve this problem by utilizing dead reckoning. It first calculates the EI of the target location and then navigates until the estimation of position using dead reckoning falls into the EI. Finally, it fine-tunes the robot's position using sensor-based landmark detection algorithms. This idea was motivated by the human behavior under the similar situation. When we enter an unfamiliar area, we look up the map and look for landmarks marked in the map. We may not evaluate the probabilities but rather try to find distinctive landmarks. In addition, we often estimate the possible distance of the landmark so that we can realize in case we proceed too far without detecting any landmark.

Since the EI's in Fig. 7 are not overlapped, the algorithm can decide which door it has detected. Then, how does the algorithm estimate the EI? Is it possible to estimate the EI reliably with only dead reckoning? Generally speaking, it is hard to use dead reckoning alone because it has a large cumulative error. We can see in Fig. 2 that there is a large difference between the actual and the estimated positions.

However, we have found that the *projected traveled distance* (PTD), which is the traveled distance projected to the major axis of the robot's motion, can be estimated quite reliably within the small error bound when the motion of the robot is linearly constrained. The motion of the robot is linearly constrained when the robot navigates along a linear corridor, along a linear wall, along a linear road, or continuously aiming at a static target. Such situations are common in the robot's working environments. In a sense, an indoor environment can be regarded as a combination of linear corridors.

Below is a brief description of the proposed algorithm.

Step 1: We assume that the initial configuration of the robot is known although it need not be very precise.

Step 2: It selects the next place from the path planned by the planner. The planner plans the path so that any path between adjacent places is linear.

Step 3: It extracts information on the EI of that place. The EI is precalculated with a constant error ratio for every NLink when the robot plans the path.

Step 4: It resets the orientation of the robot so that the direction of the path to be followed becomes zero. Since we know the initial configuration, there is no problem in resetting the orientation at the initial place. However, at other places, the orientation of the robot should be estimated before reset because the configuration of the robot at that time is only estimated by dead reckoning and thus may have a lot of error in its orientation. We have developed an algorithm to estimate the orientation using ultrasonic range finders and the Hough transform [20].

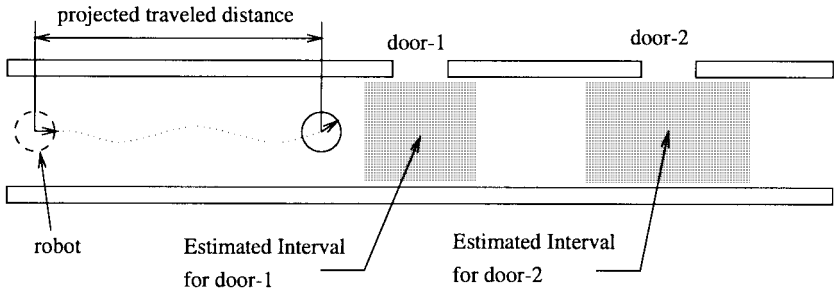

Fig. 7. An illustration of the proposed localization scheme. The width of the Estimated Interval is proportional to the distance from starting position.

Step 5: While the robot navigates along the path, it estimates the PTD and moves to Step 6 when the PTD gets into the EI.

Step 6: It invokes the landmark detection modules related to the place.

Step 7: If any landmark detection module detects a landmark, this means that the target place is recognized. If the target place is the goal place, quit navigation. Otherwise, go to Step 2.

Step 8: If the PTD exceeds the EI without detecting any landmark, the localization module signals the robot to invoke the failure recovery module.

1) Projected Traveled Distance: We are now going to show that the Projected Traveled Distance (PTD) can be estimated quite reliably. We prove it analytically and then show some experimental results.

The kinematics of a robot with two driving wheels is given as follows:

$$\begin{aligned}\dot{x} &= ((v_r + v_l)/2)\ \cos\theta = v\ \cos\theta \\ \dot{y} &= ((v_r + v_l)/2)\ \sin\theta = v\ \sin\theta \\ \dot{\theta} &= (v_r - v_l)/b = w\end{aligned} \tag{2}$$

where the state of the system (2), $q = [x, y, \theta]^T$, is the position of the wheel axis center, (x, y), and the orientation of the robot, θ, with respect to the x-axis. The velocities v_l and v_r are the tangent velocities of each wheel at its center of rotation and b is the distance between the wheels. Suppose the state of the robot at time t_0 is $q_0 = [x_0, y_0, \theta_0]^T$ and the robot moves with certain linear and angular velocities, v_1 and w_1, respectively, for the time interval Δt. Then the new state of the robot, $q_1 = [x_1, y_1, \theta_1]^T$, is obtained as follows:

$$\begin{aligned}x_1 &= x_0 + \int_{t_0}^{t_1} v_1\ \cos\theta\ dt \\ &= x_0 + v_1(\sin\theta_1 - \sin\theta_0)/w_1 \\ &= x_0 + v_1\frac{\sin\theta_1 - \sin\theta_0}{\theta_1 - \theta_0}\ \Delta t \\ y_1 &= y_0 + \int_{t_0}^{t_1} v_1\ \sin\theta\ dt \\ &= y_0 - v_1\ (\cos\theta_1 - \cos\theta_0)/w_1 \\ &= y_0 - v_1\frac{\cos\theta_1 - \cos\theta_0}{\theta_1 - \theta_0}\ \Delta t \\ \theta_1 &= \theta_0 + w_1\Delta t.\end{aligned} \tag{3}$$

When the robot moves along a wall, it rarely changes its direction except when it avoids obstacles. Generally, the number of obstacles is relatively small and the influence of the obstacles can be ignored. Therefore, $\theta_1 \approx \theta_0$ and the equation becomes

$$\begin{aligned}x_1 &= x_0 + v_1\ \cos\theta_1\Delta t \\ y_1 &= y_0 + v_1\ \sin\theta_1\Delta t \\ \theta_1 &= \theta_0.\end{aligned} \tag{4}$$

Then, the position of the robot after k time steps, (x_k, y_k), is

$$\begin{aligned}x_k &= x_0 + \sum_{i=1}^{k} v_i\Delta t\ \cos\theta_i \\ y_k &= y_0 + \sum_{i=1}^{k} v_i\Delta t\ \sin\theta_i.\end{aligned} \tag{5}$$

Let ε_i be a proportional error in velocity and $\hat{q}_i = [\hat{x}_i, \hat{y}_i, \hat{\theta}_i]^T$ be an estimation of the state of the robot at time i. Then the estimation of the position of the robot at time k is

$$\begin{aligned}\hat{x}_k &= \hat{x}_0 + \sum_{i=1}^{k} v_i(1+\varepsilon_i)\Delta t\ \cos\hat{\theta}_i \\ \hat{y}_k &= \hat{y}_0 + \sum_{i=1}^{k} v_i(1+\varepsilon_i)\Delta t\ \sin\hat{\theta}_i.\end{aligned} \tag{6}$$

When the initial position is given, $\hat{q}_0$ becomes identical to q_0. The positional error at time k, $E^k = (E_x^k, E_y^k)$, is then given as follows:

$$\begin{aligned}E_x^k &= |\hat{x}_k - x_k| \\ &= \left|\sum_{i=1}^{k} v_i(1+\varepsilon_i)\Delta t\ \cos\hat{\theta}_i - \sum_{i=1}^{k} v_i\Delta t\ \cos\theta_i\right| \\ &= \left|\sum_{i=1}^{k} v_i\Delta t(\cos\hat{\theta}_i - \cos\theta_i) + \sum_{i=1}^{k} \varepsilon_i v_i\Delta t\ \cos\theta_i\right| \\ E_y^k &= |\hat{y}_k - y_k| \\ &= \left|\sum_{i=1}^{k} v_i(1+\varepsilon_i)\Delta t\ \sin\hat{\theta}_i - \sum_{i=1}^{k} v_i\Delta t\ \sin\theta_i\right| \\ &= \left|\sum_{i=1}^{k} v_i\Delta t(\sin\hat{\theta}_i - \sin\theta_i) + \sum_{i=1}^{k} \varepsilon_i v_i\Delta t\ \sin\theta_i\right|.\end{aligned} \tag{7}$$

Let D be the total distance traveled so that it satisfies $D = \Sigma\ v_i\Delta t$. Several experiments show that the ε_i's are small

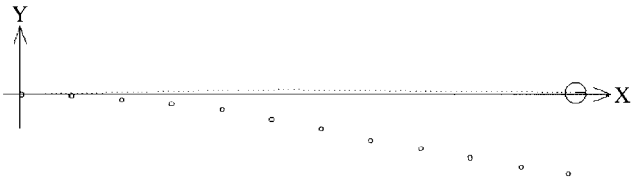

Fig. 8. Simulated robot movement along the x-axis. The dots represent real positions of the robot while the small circles denote the estimated positions.

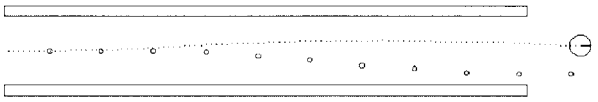

Fig. 9. Simulated robot movement along a corridor.

TABLE I
RESULTS OF THE SIMULATIONS

	Actual	Estimated	$E_x(\%)$	$E_y(\%)$
Fig. 8	(1647,3)	(1634,-235)	0.79	14.45
Fig. 9	(1632,12)	(1643,-48)	0.67	3.68

Fig. 10. CAIR-2 navigating down a corridor. Position and sonar measurements were collected while CAIR-2 navigates and plotted offline. Although it looks curved due to the position estimation errors, corridor is actually straight.

enough to ignore, $\theta_i \approx 0$ and $\hat{\theta}_i \leq \alpha$ for a quite small constant α in most cases. Therefore, (7) can be simplified as follows:

$$E_x^k \leq D(1-\cos\alpha)$$
$$E_y^k \leq D\ \sin\alpha. \tag{8}$$

Considering the characteristics of the sine and cosine functions around the origin, E_x^k is relatively smaller than E_y^k, and E_x^k is tolerable for qualitative indoor navigation. For instance, suppose the directional error α is 10°, then E_x^k and E_y^k become $0.015D$ and $0.174D$, respectively, when the robot moves 10 m, E_x^k is only 15 cm while E_y^k is 174 cm. Therefore, the PTD can be quite reliably estimated as long as the robot moves along a wall.

Figs. 8 and 9 show the experimental results in simulation. The robot moves along the x-axis in Fig. 8 and along a corridor in Fig. 9. In those figures a dotted line denotes the robot's trail while the small circles denote the estimated positions. Table I shows the actual and estimated positions and positional errors of those experiments.

In Fig. 10, CAIR-2 navigates along a corridor and the sonar measurements are plotted with the estimated positions of the robot. In this experiment, the actual PTD is 1590 cm and the estimated PTD is 1582 cm. This means a 0.50% error.

TABLE II
MEASURED POSITIONS AND CORRESPONDING ERRORS WHEN CAIR-2 RAN TEN TIMES BETWEEN (0, 0) AND (1000, 0) ALONG A CORRIDOR

No.	Measured position (cm)	Error (%)
1	1000.3	0.03
2	-8.5	0.88
3	1003.6	1.21
4	-6	0.96
5	995.1	0.11
6	-10	0.51
7	998.3	0.83
8	-19.5	1.78
9	1001.7	2.12
10	1	0.07

In the final experiment, CAIR-2 ran ten times between two locations, (0,0) and (1000,0), along a corridor adjusting its direction using the estimated direction of the wall. Table II shows the list of the actual positions measured by the authors and the corresponding positional errors. In those experiments, the errors in estimating the x value, or the PTD, do not exceed 3%; when the robot moves 10 m, the error does not exceed 30 cm. Thus, the PTD can be estimated quite accurately and used to localize the robot with the proposed localization scheme.

2) Failure Detection and Recovery: The planning algorithm described in the next subsection selects a path and corresponding landmarks so that only one location in the path has the selected landmarks. So, the localization algorithm doesn't have to distinguish among the locations. Instead, it must concern itself with the situation where it doesn't detect any landmarks in the EI. When such a situation occurs, the localization module signals "Failed to Detect" and invokes the recovery module.

There may be many possible approaches to recovery. In our case, the recovery module let the robot turn around and search for the landmarks again more cautiously within the EI.

C. Planning with Sensory Uncertainty

As explained before, we plan the optimal path not on the ALinks but on the NLinks for more efficient navigation. Efficiency is achieved by skipping several LP's. However, at the same time, the planner should be concerned with reliable sensing. The proposed localization algorithm first finds the EI using dead reckoning and then detects the landmarks within the EI. In order for the algorithm to be successful, the landmarks must be unique in the EI. In other words, there should be only one door in the EI when we try to find a door. However, because of cumulative dead reckoning error, the EI becomes wider as the distance of the NLink becomes longer. Therefore, we should select the longest NLink where EI has at least one landmark unique in the EI and reliably detectable.

Fig. 6 defines three distance intervals. In order to check how reliable the NLinks are, the planner compares the landmarks included in the PI. The PI is divided into three subintervals:

TABLE III
A CERTAINTY MATRIX FOR THE LANDMARKS

	Door	Open	Front Wall
Nothing Detected	0.1	0.1	0.1
Door Detected	0.7	0.25	0
Open Detected	0.2	0.65	0
Front Detected	0	0	0.9

pre-TI, TI, and post-TI. Let the subintervals be denoted simply by A, B, and C, respectively. Then they are defined as follows:

$$\begin{aligned} A &= \{x | P_s \le x < T_s\} \\ B &= \{x | T_s \le x < T_e\} \\ C &= \{x | T_e \le x < P_e\}. \end{aligned} \tag{9}$$

A NLink is reliable if at least one landmark included in the sub-interval B is not included in A and C. Since the sensors are not perfect, we represent the sensory uncertainties with probabilities.

The probabilities that the robot detects or does not detect the target LP, or detects a wrong LP are

$$\begin{aligned} P(Success) &= P_A(\overline{D}) \cdot P_B(D) \\ P(Fail) &= P_A(\overline{D}) \cdot P_B(\overline{D}) \cdot P_C(\overline{D}) \\ P(False) &= 1 - P(Success) - P(Fail) \end{aligned} \tag{10}$$

where the $P_*(D)$ and $P_*(\overline{D})$ denote the probabilities that the robot detects or doesn't detect the landmarks included in the TI when the robot is actually in sub-interval "*". *Fail* occurs when the robot passes over the EI without detecting any landmarks and *False* occurs when the robot detects the landmarks in the wrong sub-interval. The latter is more critical and should be avoided because if the robot moves in the wrong way, it is hard to recover. Therefore, we check the false situation at first and discard it if $P(False)$ exceeds a certain amount.

In order to evaluate the $P_*(D)$ and $P_*(\overline{D})$, we first evaluate the probabilities related to each of the landmarks. We use a similar certainty matrix which Nourbakhsh *et al.* used for their mobile robot [17]. The certainty matrix for ultrasonic range finders is shown in Table III. The value of a certainty matrix at (i, j) represents the probability that the robot detects landmark l_i when the landmark is actually l_j. The probability can be written as $P(d_i|l_j)$.

Suppose a sub-interval, *, having a set of landmarks, L_*. The probability, $P_*(d_i)$, that the robot detects a landmark, l_i, within the subinterval is $P_*(d_i) = \vee_{l_j \in L_*} P(d_i|l_j)$, where $\vee$ denotes a probabilistic "or" operation. When the planner selects a set of landmarks, L_B, to be detected for localization, the probability for each landmark included in the L_B should be combined. Therefore, $P_*(D)$, the probability that the robot detects any landmark included in the L_B within the subinterval * is obtained as follows:

$$\begin{aligned} P_*(D) &= \bigvee_{l_i \in L_B} P_*(d_i) \\ &= \bigvee_{l_i \in L_B} \bigvee_{l_j \in L_*} P(d_i|l_j). \end{aligned} \tag{11}$$

The probability, $P_*(\overline{D})$, that the robot doesn't detect any landmarks within the sub-interval is then $P_*(\overline{D}) = 1 - P_*(D)$.

The probabilities given in (10) provide the reliability of the NLinks. For the sake of efficient navigation, they should be combined with other costs such as the length of the NLinks and the sensory cost. We define the cost in terms of time. Each landmark restricts the speed of the robot to a certain degree. Therefore, the maximum speeds of the robot within PI's are determined by the selected landmarks. In other places outside the PI's, the robot can move at maximum speed. Therefore, the cost C is defined as follows:

$$C = \begin{cases} C(Success) + C(Fail) & \text{if } P(False) \le \beta \\ \infty & \text{otherwise} \end{cases} \tag{12}$$

where $C(Success)$ and $C(Fail)$ are defined as follows:

$$\begin{aligned} C(Success) &= P(Success)\left(\frac{E_s}{V_m} + \frac{E_e - E_s}{V_s}\right) \\ C(Fail) &= P(Fail)\left(\frac{E_s}{V_m} + 3\frac{E_e - E_s}{V_s}\right) \end{aligned} \tag{13}$$

where the E_s and E_e are the starting and end point of the EI as depicted in Fig. 6, and V_m is the maximum velocity and V_s is the minimum of constrained maximum velocities related to the selected landmarks; i.e., the robot can not exceed V_s in detecting the selected landmarks. The definition of $C(Fail)$ implies that the robot scans the EI once more. The false situation is pre-checked with small constant β.

With the definition of the cost, the planner selects the optimal path and corresponding sets of landmarks. First, it selects a set of landmarks for each NLink so that the $P(Success)$ and V_s are high and thus eventually the cost of the NLink becomes lower. It then applies Dijkstra's shortest path algorithm to find the lowest cost path.

IV. CONTROL SUBSYSTEM

A. Reactive Behaviors

In order to cope with a complex, uncertain and dynamically changing environment, we designed the control subsystem to be reactive. Many independent and reactive behavior modules cooperate to make the overall behaviors of the robot feasible. The most important factors considered by us in designing the behaviors are *extensibility* and *information loss*. The advocates of the behavior-based controller argue that complex behavior emerges from cooperation among multiple local behaviors. However, the real world is very complex, so a lot of behaviors are needed to cope with such an environment. Therefore, the controller should be easily extensible. Among previous work, Arkin's *motor schemas* [6] and Saffiotti's *fuzzy behaviors* [1] are more adequate than the others in this sense. Their systems have no interconnections among behaviors and allow easy add/removal of behaviors into/from the controller.

Those systems having no inter-connections among behaviors should provide an additional mechanism to blend the action commands produced by multiple independent behaviors. Such a mechanism helps a system to be extensible, but the system may suffer from information loss. Since the behaviors

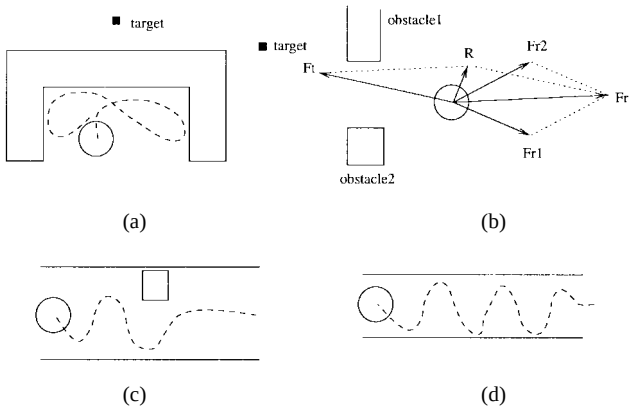

Fig. 11. Inherent limitations of the potential field approach: (a) a trap situation due to local minima (b) no passage between closely spaced obstacles (c) oscillations in the presence of obstacles (d) oscillations in narrow passage.

use virtually direct information, they exhibit little information loss. However, the blending module gets the information only indirectly via behavior output. Thus, the blending module may not notice exactly why certain behaviors produce such and such outputs, and it may produce unwanted result.

This kind of problem can be found in the *potential field* approach, which is very popular and attractive because of its elegance and simplicity. Koren and Borenstein [21] have pointed out the inherent limitation of this approach as illustrated in Fig. 11. The first problem, the local minima, is not the problem of the potential field approach but the problem of purely reactive systems. Such a problem can be resolved integrating the deliberative approach. The remaining three problems are due to the information loss between the behaviors and the blending module. For example, Fig. 11(b), both obstacle1 and obstacle2 produce repulsive force F_r1 and F_r2, respectively, which are combined into F_r. This repulsive force is then combined with attractive force F_t into a resultant force R, which makes the robot turn right. As a result, the robot can not pass between the obstacles in spite of the sufficient opening of the passage. Such a mistake is due to the repulsive forces. The F_r1 can be translated into this statement: "Go in the direction opposite to the obstacle 1," while the expected statement is: "Don't go in the direction of obstacle 1." Therefore, the information that the robot can go any direction except toward obstacle 1 is distorted and lost.

In order to take "extensibility" and "no information loss," we have designed the behaviors as computational modules that run independently without any inter-connections with other behaviors and produce action commands in desirability functions. More precisely, we say that each behavior B is implemented as Fig. 12 and produces an action command in a desirability function as follows:

$$Des_B: Control \rightarrow [0,1]. \tag{14}$$

Saffiotti [1] also used the notion of desirability. However, our work is different from his in that our behaviors are computational modules while his behaviors are fuzzy rules. We decided to build the behaviors computationally rather than logically (with fuzzy rules) since the former provides more efficient and flexible implementation.

```
BH_BEGIN
    initialization;

    loop forever
        gather sensory information;
        produce desirability function;
    end loop
BH_END
```

Fig. 12. Frame of a behavior.

B. Blender

As many behaviors can be simultaneously active in the controller, each aiming at one particular goal, many desirabilities are produced. All these desirability functions are merged into a composite. Then the defuzzification module converts the resulting tradeoff desirabilities into one crisp control decision.

Fig. 13 illustrates how multiple action commands are merged into a composite one. In this figure, the x-axis represents the directions. "Goal" denotes the action command produced by the *Move to Goal* behavior—its current task is to orient the robot toward the goal which is about 10° left, while "avoid" is the action command produced by the *Avoid Obstacle* behavior. The *Avoid Obstacle* behavior has found a small obstacle at the front. The desirability produced by the *Avoid Obstacle* behavior denotes that the robot should not move forward, but this doesn't prevent it from going in other directions.

In the current implementation, the desirabilities are combined by a simple summation. Although this summation makes the action command exceed the maximum of the probability value, it is not serious in this case because we need only a direction which has a maximum desirability. The composite action command is represented as "SUM" in Fig. 13.

The composite action command is then defuzzified to get an actual action command used to control the actuators. There are also many kinds of defuzzifing methods. In this paper, we take an action command having "maximum desirability" as an input to the actuators. Another possible method is "center of mass." In this method, the center of mass of the desirability distribution is selected. However, this method may produce undesirable result. Consider the case illustrated in Fig. 13. The composite action command has two peaks and the center of mass is the center of these two peaks; this can make the robot run into the obstacle. In contrast, the "maximum desirability" method does not suffer from such a problem, however, it may cause oscillation between the two peaks.

C. Coordinator

We hypothesize that many behavior modules are needed to make the system more intelligent and reliable. They need not, however, be active all the time. We need only a subset of behaviors active at any one time. For example, a behavior "Pass the doorway" is not needed when the robot explores an

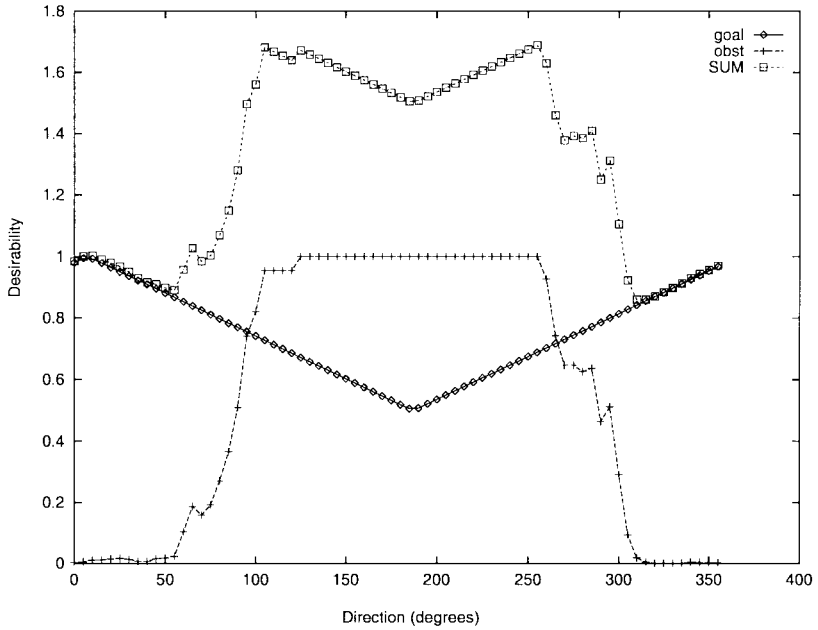

Fig. 13. Action commands of *Move to Goal* (goal), *Avoid Obstacle* (obst) and their composition (SUM).

TABLE IV
CORRESPONDENCES BETWEEN THE NLINK TYPES AND THE BEHAVIORS

From	To	Behaviors
Corridor	Room	Pass doorway
		Avoid Obstacle in doorway
Corridor	Corridor	Follow a corridor
		Avoid Obstacle

open space. A module called "coordinator" is exploited for this purpose. It manages the activation of the behaviors according to the *situation*.

As the robot navigates, the situation continuously changes. In order to cope with such changes, the robot should be able to recognize the situation. The proposed system recognizes the situation using the localization module and the planning module, which are described in the previous section. Since we use a topological map, the situation can be determined only roughly, i.e., one example of a possible description of the situation is "the last identified location is Room #1024 and the robot is navigating toward Room #1025." However, it is sufficient for the coordinator because a more precise variation of the situation can be handled by the behaviors.

The coordinator selects a set of behaviors according to the current and next locations. In fact, the corresponding sets of the behaviors for each NLinks are determined in the initialization time when the map is prepared. As mentioned earlier, the robot navigates along the NLinks so that the robot's situations correspond to the NLinks. Table IV shows some correspondence between the NLink types and the behaviors. The NLink type is defined by the types of the current and next locations.

When the robot recognizes that it has reached a sub-goal node and is going to navigate to the next subgoal along the NLink, the coordinator deactivates the previously activated behaviors and activates the behaviors that correspond to the NLink.

We may find robot systems that function similar as our Coordinator. For instance, Brooks' subsumption architecture has no separate module to select behaviors but the behaviors are layered so that higher-level behavior can inhibit or subsume lower-level behavior [2]. Saffioti's system, on the other hand, has control structures that are a set of landmarks [1]. The possibility of certain landmarks being detected effects the activation of the behaviors. The advantage of our coordinator over the above mentioned work might be that it utilizes an enhanced topological map that provides efficient and reliable localization and planning mechanisms.

V. EXPERIMENTAL RESULTS

Figs. 14 and 15 show the results of our experiments. In these experiments, the robot navigated in the environment illustrated in Fig. 4. The robot is commanded to navigate from place 0 to place 10 in the first experiment and from places 0 to 14 in the other. In the second experiment, we cut the link between places 0 to 14 intentionally to test the localization algorithm over the long run. The robot successfully reached its goal position in both experiments. It is remarkable that in the second experiment, the robot overcame a large positional error that occurred in the corridor linking places 0–1. The results prove that the proposed localization scheme is reliable even with a somewhat large positional error and imprecise sensors.

In each experiment, the planner selected the paths "$0 \rightarrow 14 \rightarrow 10$" and "$0 \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 12 \rightarrow 14$," respectively. The robot went to place 10, directly skipping all places from place 14 in experiment 1, but it visited place 12 when it went from places 9–14. This was because the

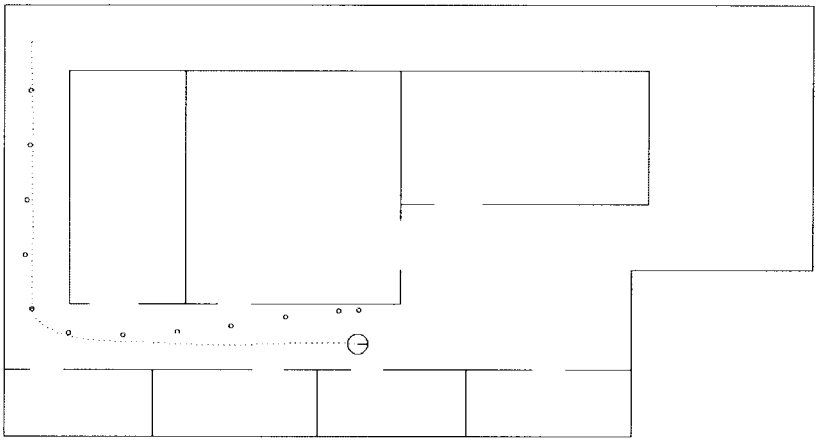

Fig. 14. Experiment 1: navigate from place 0 to place 10.

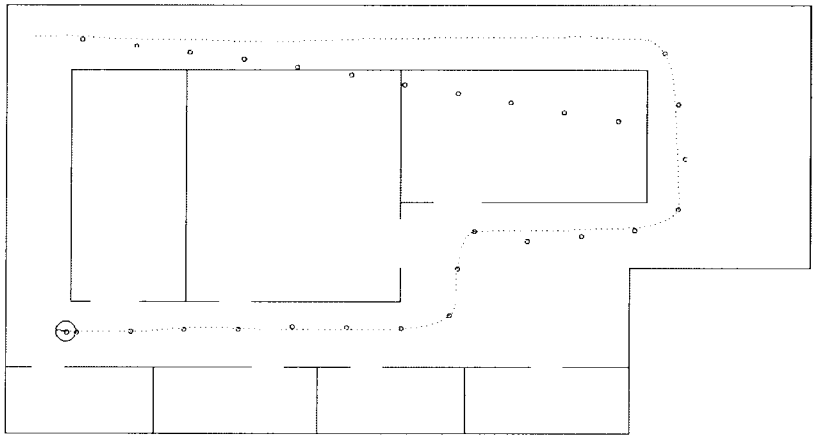

Fig. 15. Experiment 2: navigate from place 0 to place 14.

distance between places 9–14 was long, and place 13 was sensed instead of place 14.

In addition, the proposed control architecture has been proven valid by several real world experiments. Two major experiments/demonstrations were conducted during the '93 Taejon World Exposition and the 4th Annual Mobile Robot Competition sponsored by IJCAI and AAAI held in Montreal, P.Q., Canada, in 1995. In the '93 World EXPO, CAIR-2 reliably demonstrated reactive behaviors such as "Avoid obstacles and spectators" and "Track a moving visual landmark" in an outdoor environment for three months. And in the mobile robot competition, we won the first place award in the "Office Delivery" event that tested autonomous navigation ability in an office-like indoor environment. CAIR-2 used almost the same control architecture as the one proposed in this paper (we have improved it since the competition) and has proven the feasibility of the control architecture. A detailed description of the competition can be found in [8] and [22].

VI. Conclusion and Future Work

We have described a new control architecture for an autonomous mobile robot. The proposed navigation system integrates the advantages of *reactive* approach and *deliberate* approach. In order to integrate them, we provided two modules: one for selecting the relevant behaviors and the other for blending multiple action commands. We also concentrated on describing the mapping system, on which the behavior selection modules highly depend. The mapping system represents the environment in an *enhanced topological map* (ETMap) and provides a localization module and a planning module for efficient and reliable navigation.

The proposed system was implemented and tested on both a real mobile robot, CAIR-2 and a simulator. The experiments show that the proposed system can accomplish the assigned tasks even with imprecise sensors like ultrasonic range finders.

Although the proposed system was tested only for indoor navigation consisting of straight corridors, a similar idea can be applied to other applications such as curved corridors and outdoor navigation with ALV's (autonomous land vehicle) with minor modification. We are planning to extend our work to more complex environment.

References

[1] A. Saffiotti, "Some notes on the integration of planning and reactivity in autonomous mobile robots," in *Proc. AAAI Spring Symp. Foundations Automatic Planning*, 1993, pp. 122–126.

[2] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. 2, pp. 14–23, Mar. 1986.
[3] R. C. Arkin, "Motor schema based navigation for a mobile robot: An approach to programming by behavior," in *Proc. IEEE Int. Conf. Robotics Automation*, 1987, pp. 264–271.
[4] R. G. Simmons, "Structured control for autonomous robots," *IEEE Trans. Robot. Automat.*, vol. 10, no. 1, pp. 34–43, 1994.
[5] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey, "Plan guided reaction," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 1370–1382, Dec. 1990.
[6] R. C. Arkin, "Integrating behavioral, perceptual, and world knowledge in reactive navigation," *Robot. Auton. Syst.*, vol. 6, pp. 105–122, 1990.
[7] M. J. Mataric, "Integration of representation into goal-driven behavior-based robots," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 304–321, June 1992.
[8] J. Chung, B.-S. Ryu, and H. S. Yang, "Integrated control architecture based on behavior and plan for mobile robot navigation," *Robotica*, vol. 16, pp. 387–399, 1998.
[9] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 278–288, June 1991.
[10] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE J. Robot. Automat.*, vol. 3, pp. 249–265, June 1987.
[11] N. Ayache and O. D. Faugeras, "Maintaining representations of the environment of a mobile robot," *IEEE Trans. Robot. Automat.*, vol. 5, pp. 804–819, Dec. 1989.
[12] A. Kosaka and A. C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *CVGIP: Image Understand.*, vol. 56, pp. 271–329, Nov. 1992.
[13] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox, "Dynamic map building for an autonomous mobile robot," *Int. J. Robot. Res.*, vol. 11, pp. 286–298, Aug. 1992.
[14] D. M. Kortenkamp, "Cognitive maps for mobile robots: A representation for mapping and navigation," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1993.
[15] B. J. Kuipers and Y. T. Byun, "A robust, qualitative method for robot spatial learning," in *Proc. 7th Nat. Conf. Artificial Intelligence (AAAI88)*, 1988, pp. 774–779.
[16] T. S. Levitt and D. T. Lawton, "Qualitative navigation for mobile robots," *Artif. Intell.*, vol. 44, no. 3, pp. 305–360, 1990.
[17] I. Nourbakhsh, R. Powers, and S. Birchfield, "Dervish: An office-navigating robot," *AI Mag.*, vol. 16, no. 2, pp. 53–60, 1995.
[18] L. P. Wesley, "Autonomous locative reasoning: An evidential approach," in *Proc. IEEE Int. Conf. Robotics Automation*, 1993, pp. 700–707.
[19] D. Kortenkamp and T. Weymouth, "Topological mapping for mobile robots using a combination of sonar and vision sensing," in *Proc. 12th Nat. Conf. Artificial Intelligence (AAAI94)*, 1994, pp. 979–984.
[20] W. K. Pratt, *Digital Image Processing*. New York: Wiley, 1991.
[21] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE Int. Conf. Robotics Automation*, 1991, pp. 1398–1404.
[22] H. S. Yang, J. Chung, B.-S. Ryu, and J. Lee, "Cair-2: Intelligent mobile robot for guidance and delivery," *AI Mag.*, vol. 17, no. 1, pp. 47–53, 1996.

Byeong-Soon Ryu received the B.S., M.S., and Ph.D degrees from the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, in 1991, 1993, and 1998, respectively.

He was on the research staff at the Center for Artificial Intelligence Research from 1993 to 1998. His current research interests include computer vision, robotics, artificial intelligence, and multimedia.

Hyun Seung Yang received the B.S. degree from the Department of Electronics Engineering, Seoul National University, Seoul, Korea, in 1976, and the M.S.E.E. and Ph.D. degrees from the School of Electrical Engineering, Purdue University, West Lafayette, IN, in 1983 and 1986, respectively.

He was with the Department of Electrical and Computer Engineering, University of Iowa, Iowa City, from 1986 to 1988 as an Assistant Professor. Since 1988, he has been with the Department of Computer Science at Korea Advanced Institute of Science and Technology (KAIST), Taejon, where he is currently a Full Professor. He has been a director of the Computer Vision and Intelligent Robotics Lab, Center for Artificial Intelligence Research sponsored by Korea Science and Engineering Foundation (KOSEF) at KAIST since 1990. His current research interests include computer vision, robotics, artificial intelligence, computer graphics and multimedia.