Integration of Representation Into Goal-Driven Behavior-Based Robots

Maja J. Mataric

Abstract—**We describe an implemented architecture that integrates a map representation into a reactive, subsumption-based mobile robot. As an alternative to *hybrid* methods, which separate the reactive and traditional planning parts of the control system, we present a fully integrated reactive system that removes the distinction between the control program and the map. Our method was implemented and tested on a mobile robot equipped with a ring of sonars and a compass, and programmed with a collection of simple, incrementally designed behaviors. The robot performs collision-free navigation, dynamic landmark detection, map construction and maintenance, and path planning. Given any known landmark as a goal, the robot plans and executes the shortest known path to it. If the goal is not reachable, the robot detects failure, updates the map, and finds an alternate route. The topological representation primitives are designed to suit the robot's sensors and its navigation behavior, thus minimizing the amount of stored information. Distributed over a collection of behaviors, the map itself performs constant-time localization and linear-time path planning. The approach we present is qualitative and tolerant of sensor inaccuracies, unexpected obstacles, and course changes. It extends the repertoire of integrated reactive systems to tasks requiring spatial modeling and user interaction.**

I. Introduction

INACCURATE sensors, world unpredictability, and imperfect control often cause the failure of traditional planning and navigation methods for real-time mobile robots. More reactive approaches to navigation have been explored in [1], [5], and [33]. In particular, [5] proposed the *subsumption architecture* as an incremental method for building layers of robot competencies, consisting of simple rules that tightly couple sensing and action.

The subsumption architecture has been used successfully in fully reactive systems such as [4], [7], [9], and [15]. So far, these systems have been limited to applications requiring no explicit internal representation, which imposed a fundamental limitation on the domain of applications for the architecture. The classical problem of path planning, for example, requires some representation of space. Any solution superior to random walk necessitates an internal model of the robot's current location, the desired goal location, and the relationship between the two.

Path planning is discussed extensively in the literature [21], [23], [34]. Most solutions rely on centralized world models whose compatibility with completely reactive systems is debatable [3]. *Hybrid* systems offer a compromise by employing a reactive system for low-level control and a planner for higher level decision making. They separate the control system into two or more communicating but basically independent parts.

In contrast, we address the problem of integrating representation into a fully reactive, nonhybrid system with the goal of maintaining a map of the environment and using it for path planning. We introduce a distributed map representation that merges directly into a homogeneous subsumption-based system, thus eliminating the need to separate the planning and execution parts of the system. Our approach extends the repertoire of integrated, fully reactive systems to domains requiring internal spatial representation.

Our system's task is to explore an office environment, and to construct and maintain a map based on landmarks it discovers. The user can select a particular landmark (e.g., a specific corridor) or landmark type (e.g., the nearest corridor) as the goal. The robot then employs the map to plan and execute the shortest known path to that landmark. After reaching the destination, the robot is either given another goal or continues to explore and update its map. If the robot fails to reach the goal, it detects its failure and changes the map appropriately.

All algorithms we describe were implemented on a mobile robot. The data were gathered by running the robot in unaltered office environments with static and dynamic obstacles including furniture, other robots, and people.

II. The Robot, Toto

Toto, the testbed robot we constructed, consists of a circular omnidirectional three-wheeled base capable of following an arbitrary continuous path. On the base is mounted a ring of 12 ultrasonic ranging sensors, ranging from 0.9 to 32 ft, and a flux-gate compass, providing 4 bits of bearing (Fig. 1). The robot is programmed in the Behavior Language, a rule-based parallel programming language that compiles into the subsumption architecture [5].

The robot was tested over a period of two months in over 40 trials in a cluttered office environment. The data were gathered by attaching a marker to the base of the robot and recording its path on the floor covered with 1-ft^2 tiles.

III. Sensor Characterization

Real sensors are noisy and inaccurate. Maximizing their reliability often involves data interpretation using complicated physical models. In contrast to explicit error modeling,

Manuscript received May 15, 1991; revised December 16, 1991. This work was supported in part by the Artificial Intelligence Center of Hughes Research Laboratories, and in part by the University Research Initiative under Office of Naval Research Contract N0014-86-K-0685. Portions of this paper were presented at the IEEE Conference on Robotics and Automation, Cincinnati, OH, May 1990.

The author is with the MIT Artificial Intelligence Laboratory, Cambridge, MA 02139.

1042-296X/92$03.00 © 1992 IEEE

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

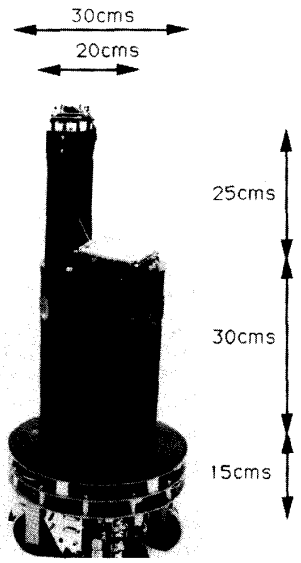

Fig. 1. The robot testbed: an omnidirectional three-wheeled base equipped with a ring of 12 ultrasonic ranging sensors and a flux-gate compass.

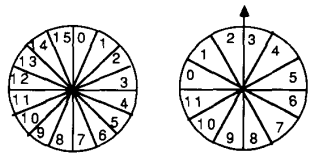

Fig. 2. The organization of the compass and sonar regions. The sonar cones remain constant relative to the shown forward-pointing vector. The compass reflects the local magnetic field. The two sensors are used independently.

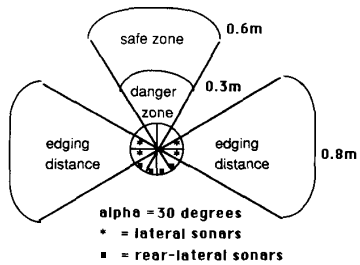

Fig. 3. The perceptual zones around the robot corresponding to the relevant obstacle and boundary conditions in the environment. The regions are used to implement basic collision-free navigation rules that combine into a robust boundary tracing behavior.

we minimized overhead computation by using qualitative, functional descriptions of the sensors describing merely the properties relevant to the robot's task.

Much work has been done on formalizing the limitations of ultrasonic ranging sensors and suggesting various analytical approaches to their application [19], [20]. Our method relies on a single sufficient characteristic of the sensor: its high accuracy (near 95%) for incident angles less than 15° from the surface normal [32]. Long returns may result from specular reflection, and are thus less reliable. We utilize the returns in the short range, as well as the qualitative properties of the data, such as relative differences between readings rather than their exact values.

The error characteristics of the compass are quite different. Its absolute heading reading is grossly inaccurate in the presence of interfering magnetic fields and metal structures in the environment (up to 2 of the available 4 bits of resolution, or 50%), while it is locally consistent to up to 90% accuracy. To maximize its utility, we structured our algorithms to rely on the repeatability rather than on the absolute accuracy of the sensed compass direction.

Fig. 2 illustrates the organization of the sonar and compass regions on the robot. The sonar cones remain constant with respect to the forward-pointing vector, whereas the compass reflects the local magnetic field. The two sensors are used independently.

IV. The Basic Navigation Algorithm

The robot's control system consists of three competencies integrated into a homogeneous behavior-based representation: 1) basic navigation (obstacle avoidance and boundary tracing), 2) landmark detection, and 3) map-related computation (map construction, map update, and path planning). The competencies were designed and added incrementally, each relying on those below.

Lower level competencies do not depend on the higher level ones, but we designed them by keeping in mind the entire integrated system. The basic navigation algorithm was designed to facilitate landmark recognition as well as map construction [27]. Its primary task was to keep the robot moving safely through an unaltered office environment, avoiding collisions with objects and people. The rules for reactive wandering were augmented with a rule that made the robot maintain a small distance from objects, i.e., avoid open areas. This behavior is useful as the accuracy of the sonars is maximized in the proximity of detectable objects, which is where the robot can obtain the most information about the environment.

The robot's velocity was limited by the sonar refresh rate: 200 ms per pair of sensors; a new data set for the entire sonar ring was obtained at 0.83 Hz. This limited the robot's velocity from the maximum of 2 to 0.2 m/s. Based on the circular arrangement of the sonars, obstacle avoidance and boundary tracing were implemented by segmenting the space around the robot into relevant sensory regions (see Fig. 3). The area in front of the robot was divided into two regions: the danger zone and the safe zone. The threshold between the zones (0.3 m) was derived from the robot's velocity and the minimum range of the sonar sensors (0.25 m). It guarantees that at least two sets of sonar data are available before reaching an obstacle, thus decreasing the probability of collision.

An object in the danger zone is an obstacle. An object in the safe zone causes the robot to turn appropriately to avoid it. The edging distance is a threshold dividing the area on each side of the robot. The robot stays within the edging distance of the boundary it is following. The following basic behaviors utilize the zones to produce boundary tracing:

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

```
(defbehavior stroll
   (cond
       ((and (< = (min (sonars 1 2 3 4)
        danger-zone))
             (not stopped))
          (stop))
       ((and (< = (min (sonars 1 2 3 4)
        danger-zone))
             (stopped))
          (move backward))
       (t
          (move forward))))
```

Stroll: If an obstacle is detected within the danger zone, the robot stops. If stopped within a danger zone, it backs up. This allows the robot to escape tight situations while minimizing unnecessary motion in avoiding transient obstacles. If no obstacles are detected, the robot is repeatedly given a target distance, resulting in smooth continuous motion. *Stroll* alone provides safe forward motion.

```
(defbehavior avoid
   (cond
       ((and (< = (sonar 1 or 2) safe-
        zone)
             (< = (sonar 3 or 4) safe-
              zone))
          (turn left))
       ((< = (sonar 3 or 4) safe-zone)
          (turn right))))
```

Avoid: The robot turns (by a small, fixed angle $\alpha = 30° =$ width of sonar cone) in the opposite direction from an obstacle within the safe distance of its front sonars. If obstacles are detected on both sides, oscillation is prevented by consistently choosing the same default direction (left). In conjunction with *stroll*, this rule generates collision-free wandering behavior.

```
(defbehavior align
   (cond
       ((and (< (sonar 7 or 8) edging-
        distance)
             (> (sonar 5 or 6) edging-
              distance))
          (turn right))
       ((and (< (sonar 9 or 10) edging-
        distance)
             (> (sonar 11 or 0) edging-
              distance))
          (turn left))))
```

Align: If an object is detected within the edging distance range of one of the rear-lateral sonars (10 and 9 or 8 and 7) and not by the lateral sonars (0 and 11 or 5 and 6) on the same side, the robot makes an $\alpha°$ turn in that direction. The combination of *avoid*, *stroll*, and *align* allows the robot to follow straight and convex curved boundaries.

```
(defbehavior correct

(cond
       ((and (< (sonar 11) edging-
        distance)
             (> (sonar 0) edging-
              distance))
          (turn left))
       ((and (< (sonar 6) edging-distance)
             (> (sonar 5) edging-
              distance))
```

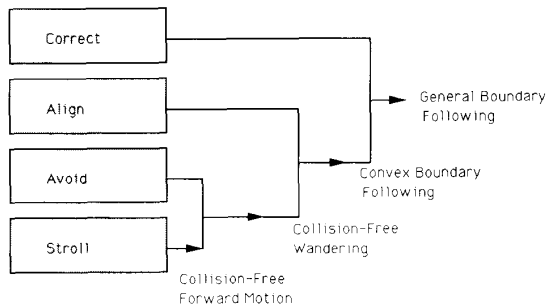

Fig. 4. A schematic showing the incremental interaction of the low-level navigation behaviors resulting in boundary tracing. The addition of each new behavior adds to the overall competence of the robot.

Correct: This behavior prevents the robot from losing track of a lateral boundary containing a sharp turn by monitoring the pair of sonars on each side of the robot (0 and 11 or 5 and 6). If the rear of the two sonars (11 or 6) detects an object within the edging distance and the front (0 or 5) does not, the robot makes an $\alpha°$ turn in the same direction. By turning, it gets the boundary in the range of both of the sensors in the pair, effectively returning to a position aligned with the boundary. In conjunction with the rest of the wandering behaviors, *correct* allows the robot to track arbitrarily sharp turns. Fig. 4 illustrates the incremental addition of navigational competencies resulting in boundary tracing behavior.

The boundary-following behavior does not distinguish between, or differentially treat, various kinds of boundaries in the environment. The behavior is general, independent of what types of objects and structures form the boundaries, as long as they are detectable by the sonars.

Fig. 5 shows a cumulative plot of four real-time runs in a large, unaltered office area. The room is cluttered with chairs, tables, doors, a water fountain, moving people, and other robots. The data show reliable boundary following in all trials, independent of the robot's starting position. Fig. 6 plots five independent real-time runs showing the robot's convergence to the middle of a cluttered corridor. The robot remains in the middle of the empty corridor because the corridor width is less than the sum of the edging distance on both sides of the robot. In a wider corridor, the robot follows the wall it initially approaches. Even without the use of position control to direct the robot to specific Cartesian positions, the navigation rules result in stable paths that show repeatable convergence around the detected object boundaries.

V. Landmark Detection

The robust, repeatable navigation behavior enabled the robot to safely wander about and explore its environment. Next we added the ability to detect landmarks used in spatial mapping. Sonar-based systems usually perform landmark detection

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

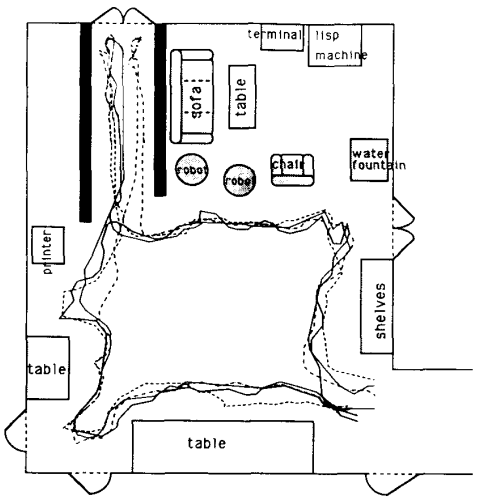

Fig. 5. A plot of four independent real robot runs manifesting consistent boundary following in an unaltered room. The data consist of inflection points in the robot's actual traversed paths, connected by straight line segments.

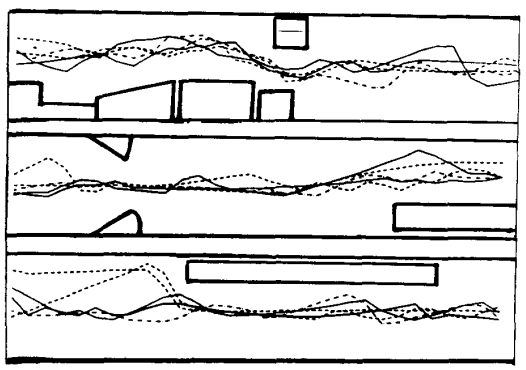

Fig. 6. A plot of five independent real robot runs showing convergence to the middle of a somewhat cluttered corridor. The corridor is shown in three parts, but the data demonstrate continuous runs left to right, bottom to top.

by matching sensory patterns to stored landmark models or signatures (e.g., [12]). These approaches are "static" in their use of a discrete snapshot of the world based on a single set of sensor data. However, the expected accuracy of any one data point is low, and different sonar signatures are generated in different trials due to sensor error and noise. Static matchers compensate by maintaining error estimates and utilizing positional precision that is often difficult to maintain due to wheel slipping and other factors producing further cumulative errors.

We explored an alternative, "dynamic," approach to landmark detection based on continuously monitoring the robot's sensors and taking advantage of the robot's underlying boundary tracing behavior. The landmark detector looks for features in the world that have physical extensions detectable over time, i.e., it monitors for consistencies in the sensory data as the robot is moving next to objects in the environment. Spurious sensor errors are filtered out through dynamic averaging. Three specific sensory conditions are monitored: the compass bearing, to determine whether the robot is moving straight, and the sonar data on both sides of the robot, to check for a persisting boundary on either or both sides. Whenever the robot repeatedly detects short readings on its right side and its averaged compass bearing is stable, the confidence counter for a right wall is incremented. An analogous rule applies to left walls. Simultaneous sufficient confidence in both walls indicates a corridor. When the appropriate combination of confidences reaches a preset threshold τ, the corresponding landmark is detected, and the confidence counter is reset. τ is the minimum length of a continuous landmark boundary, represented in counter units. To adapt the algorithm to other types of environments, τ is set to the shortest landmark we want the robot to detect. In our case, based on the constant velocity assumption, τ is equivalent to the maximum length of nonlandmark obstacles in the office environment (chairs, table legs, trash cans, filing cabinets).

We chose three large, stable, and reliably detectable landmark types: left walls (LW), right walls (RW), and corridors (C). A default landmark type was added, corresponding to long irregular boundaries (I). It enables the system to represent the environment as a collection of contiguous strings of features describing the path the robot traversed. Wherever in the environment the robot happens to be, it finds and follows a boundary enabling it to classify the area into one of its four landmark types.

The landmarks used by the system are designed to be reliably detectable, but due to their size produce a rather sparse representation of space. By introducing additional sensors or position control, the granularity of landmarks can be refined. While additional behaviors would be required for detecting different landmark types, the connectivity and path planning properties of the representation would remain linear.

The qualitative, procedural nature of the landmark detection algorithm adds robustness to the system by not relying on sensor precision or position control. Fig. 7 illustrates the locations of landmark detection over three trials in a room cluttered with furniture and people. Even without position control, the data show some clustering. The same landmarks, with slight deviations in the compass bearing due to the averaging process, are detected repeatedly independent of the robot's starting position or the exact path followed.

Besides following object boundaries, the robot must also be able to find landmarks located in the middle of open areas, separated from the continuous boundary it begins to explore. This ability is provided by a behavior that occasionally moves the robot out into an open area. When stimulated to go into an open area, the robot continues to move straight until it encounters an object whose boundary is either new or recognized as one it has followed before. The following sections describe how the mapping algorithm distinguishes between the two possibilities and how the open-space behavior is triggered.

VI. The Mapping Algorithm

The robot's top-level task is to map the structure of the environment based on the spatial relationships of the land-

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

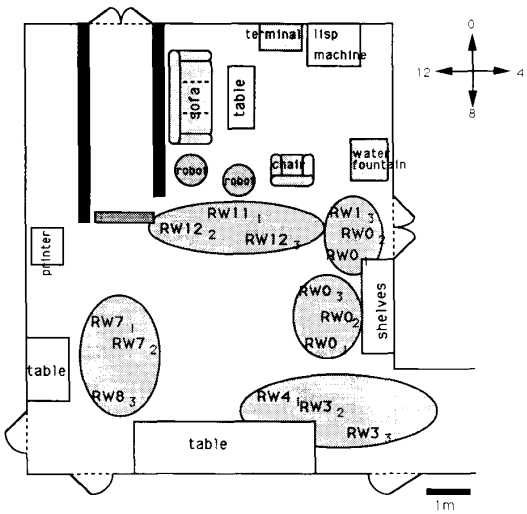

Fig. 7. The robot's landmark detection performance over three trials in the same room. Transient obstacles and people are not shown. Each landmark consists of the type and the associated compass bearing (e.g., RW0 = right wall north). The shown landmark locations correspond to the exact position of detection. The subscript indicates the trial. The locations corresponding to the same landmark are indicated by a common shaded area.

marks and to use this map to find paths to any previously visited landmark that the user chooses as the goal. This differs fundamentally from building a detailed map of the world that includes features of smaller size and higher probability of impermanence. The aim is to produce and maintain a coarse-scale map that allows the robot to get within the sensing range of the goal. Reaching an exact location can be accomplished by augmenting the system with special-purpose motion planning based on the specific task and sensors used. The system is suitable for various applications that require the use of a map, such as sentry and surveillance tasks, as well as prioritized tasks, such as hazardous area maintenance, plant watering, or supply delivery in a large office complex.

Reaching places that cannot be sensed from the robot's current position necessitates some form of path planning, which, in turn, requires a world model. The traditional approach to path planning involves some type of a reasoning engine that generates a plan by manipulating a Cartesian map usually stored in a centralized data structure (e.g., [8], [14], [16], [21], etc.). The success of the plan depends on the accuracy of the geometric information in the map. In contrast to Cartesian metric representations, graphs are convenient for encoding topological, qualitative information (e.g., [8], [10], [17], [18], etc.). Similarly, our approach directly constructs and utilizes a graph in which each node represents a unique landmark, and neighbor links indicate physical adjacency, thus producing a structure isomorphic to the topology of the environment.

A. The Distributed Nature of the Representation

In contrast to implementing the graph as a manipulable data structure, we represent it as a collection of concurrently active behaviors in which each landmark is a behavior. The links in the graph serve a dual purpose: they encode topological

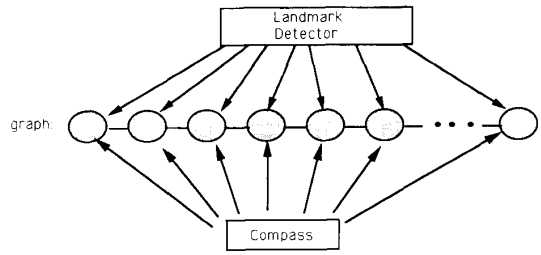

Fig. 8. The graph representation consists of a number of concurrently active nodes, each of which corresponds to a unique landmark in the world. Each node receives inputs from the compass and the landmark detector, and communicates with its neighbors in the graph.

relationships between landmarks and serve as message wires allowing information exchange between neighbors. Each landmark behavior receives inputs from the landmark detector and the compass, and outputs to other behaviors in the network (Fig. 8).

Unlike a centralized data structure, the graph as a whole is not manipulable since each of its nodes is an independently acting behavior. Its distributed representation is a natural extension of the subsumption architecture. Like all other behaviors in the system, graph behaviors are collections of simple rules. The rules in the landmark behaviors allow the nodes to match the detected landmarks to the particular landmark they are encoding (i.e., to localize the system) and to send various activation messages used for path planning.

B. Map-Building Through Graph Construction

As the robot explores the environment, the landmarks it detects are broadcast to all the nodes in the graph. Initially, the nodes are empty, and the first landmark is assigned to the first node by recording its descriptor. The landmark descriptor is a tuple $< T, C, L, P >$:

$T \in \{LW, RW, C, I\}$ is the qualitative landmark type.

$C \in [0..15]$ is the averaged compass bearing.

$L \in [1..127]$ is a rough estimate of the landmark's length.

$P = (x, y)$ s.t. $-128 \leq x, y \leq 127$ is a coarse position estimate.

L is derived from the number of times the same landmark is consecutively detected. Assuming constant velocity, this value serves as an implicit representation of distance and time. P is derived by periodically summing the compass vector, the magnitude of which is based on the constant velocity assumption. Once created, the newly added node is *activated*. The active node corresponds to the position of the robot within the map.

Whenever a landmark is detected, it is matched to all the nodes in the graph. The matching process results with either a unique match or no match (see next section for details). Localization is a simple process of comparing the landmark descriptor $< t, c, l, (x, y) >$ with the robot's current sensory information $< t', c', l', (x', y') >$. For the landmark type, an equality test suffices ($t =? t'$), whereas the compass and position values are compared with an allowable error margin ($| c - c' | \leq 30$ degrees). The relatively large compass error could be used based on the heuristic that physically

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

adjacent landmarks (walls and corridors) in a typical office environment are usually orthogonal or collinear. The rough position estimate was recalibrated whenever the robot returned to a previously visited landmark and was scaled by an error factor e directly proportional to the size l of the landmark. e allows for the possibility that the robot is positioned anywhere along the length of a landmark and takes advantage of the fact that large landmarks permit large error. For an n-unit-long corridor, e covers the area of the rectangle of length n, the height h of an average corridor, and rotated in the direction c. Formally:

$$| (x - x') \cos c + (y - y') \sin c | < n/2$$

$$| -(x - x') \sin c + (y - y') \cos c | < h/2.$$

Finally, each landmark has a dual, depending on the direction of the robot's motion relative to the boundary being followed (e.g., a left wall going north is equivalent to a right wall going south, LW0 = RW8). Based on the landmark descriptor, landmark matching, performed in parallel, runs in $O(1)$ constant time regardless of the size of the graph.

C. Landmark Disambiguation

Whenever a node is activated, it spreads *expectation* to its neighbor in the direction of robot's travel along the path, priming it for upcoming activation. Matching an expecting node to a found landmark verifies the correctness of the graph. The notion of expectation provides a contextual clue by expanding the matching window to two nodes instead of one. This information helps disambiguate between nodes with identical type and similar compass bearing.

When the robot initially returns to a previously visited landmark, the node corresponding to the location will not be expecting activation since the topological link between it and the beginning of the path has not yet been established. The match is recognized by comparing the location of the landmark to the stored position estimate. Although the estimate is very inaccurate, the matching tolerance is bounded by the size of the landmark, and it suffices for disambiguating two otherwise identical landmarks. The use of the position estimate does not constitute position control, however, since it is used exclusively for landmark disambiguation and not for controlling the robot's motion. The combination of expectation and position estimation allows for uniquely disambiguating the landmarks.

Consequently, matching always produces a unique match or no match. If no match is found, the landmark is assumed to be new and is assigned to a free node. The newly added landmark is connected to the currently active landmark by a topological wire. Fig. 9 shows the graph representation the robot constructed for the office environment shown in Fig. 7. Fig. 10 illustrates an environment containing multiple cycles and its corresponding graph.

The unique representation of landmarks eliminated false positive matches in all trials. Due to sensor noise, false negatives occurred in approximately one third of the trials when the robot, while following a boundary, failed to recognize it as a landmark. If this happened during the discovery phase, it resulted in a sparser map that would later get augmented if the robot was allowed repeated runs through the same environment. In the converse case, failing to detect a previously detected landmark was ignored if the subsequent landmark matched in type and position. Otherwise it was recognized as a new location and added to the network as an alternative direction to pursue. This case accounted for situations in which the environment could change, such as a doorway that could be open or closed. Finally, failing to detect a landmark while on the way to a goal did not affect the goal-finding behavior unless the skipped landmark was the goal itself or a junction of two or more paths in the network.

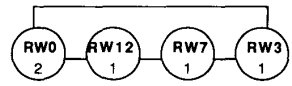

Fig. 9. The graph the robot produced in the environment shown in Fig. 7 in the second trial. The first trial produced (RW0, RW11, RW7, RW4) while the third trial produced (RW0, RW12, RW8, RW3); the three networks have correct, identical topology, and the difference in compass values falls within c's error margin. The nodes are ordered left to right by discovery time. Besides the landmark type and compass bearing, the figure also shows the relative landmark length.

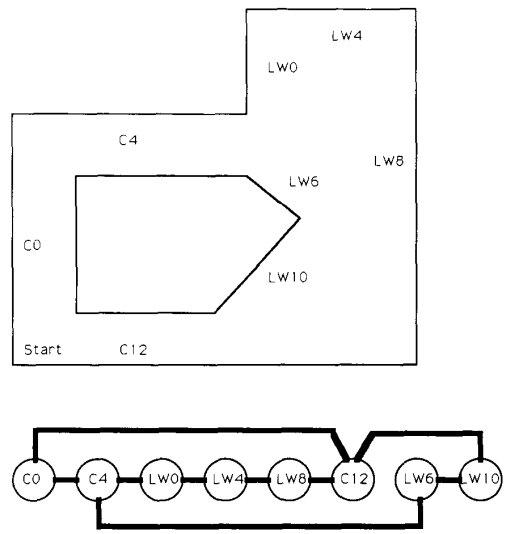

Fig. 10. An example of an environment containing multiple cycles on the robot's traversal path, and the graph the robot produced.

D. Path Planning and Optimization

The map provides the structure for relating the robot's current position and the goal. Its distributed nature allows for the path to be computed by the individual map components using local operations only. We use a variation of *activation spreading* from the goal in all directions throughout the graph. Activation is propagated through the nearest-neighbor links. The goal node repeatedly sends out a *call* that eventually reaches the currently active node. Equivalent to parallel search, this process is guaranteed to terminate in worst case linear time $O(n)$ in the size of the graph [26].

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

Path optimization by topological distance is a natural consequence of this process. Whenever the robot follows the landmarks in the direction of the spreading call, it is guaranteed to proceed on a shortest topological path to the goal. The call originating from the node closest to the robot's current position will reach it first, given uniform activation dissipation. Weighting each landmark by its physical length allows for computing the physically shortest path within the graph. As a call propagates from the goal to the current node, it sums the lengths of all the landmarks it passes. Upon reaching the active landmark, the value of the call approximates the physical length of the traversed path. The shortest incoming call is chosen at each landmark. Making a local greedy choice at each node results in the global, physically shortest known path within the graph, in $O(n)$ [28].

Graph cycles do not cause problems because of the greedy nature of the algorithm. Since the length of a cycling call increases monotonically, it is never selected as the optimal path. Additionally, the maximum length of any path is bounded by the size of the graph so indefinite activation propagation cannot happen.

The activation from the goal node is received by all of the nodes in the graph. As the robot traverses a path, it chooses the optimal direction to pursue from any landmark. Consequently, if the robot veers away from the optimal path or is intentionally placed elsewhere in the environment by the user, once localized it will pursue the optimal path from the current location [25].

The goal is reached when the currently recognized landmark matches the goal landmark. This condition terminates the activation spreading, and the robot can pursue another goal or continue exploring the environment and augmenting and verifying the map. If the path to the goal is blocked, the robot will persistently fail to make a transition from the current landmark to its neighbor. After a fixed time period, it gives up pursuing the blocked path, terminates activation spreading, and removes the topological link between its current position and the one it failed to reach. When the change in the graph is complete, activation spreading from the goal is started up again, in order to find another path, if one exists. The ability to detect failure and update the network structure allows the system to adapt the map representation to a dynamically changing environment.

In evaluating the performance of the system, the robot was presented with various obstacles including furniture and people walking in its way. If the desired path was temporarily blocked, the low-level navigation behavior ensured that no collision occurred by turning the robot away from the obstacle. Simultaneously, activation from the goal forced the robot to turn in the direction of the desired path. The conflict of the two motivations resulted in taking the first free turn toward the direction of the goal. Only if the path to the goal was completely blocked was it eventually abandoned.

The user can interact with the system in a number ways depending on the type of the specified goal. The goal types vary from very specific (e.g., a particular corridor, the first discovered landmark in the graph, etc.) to less general (e.g., nearest north-going corridor) to very general (e.g., nearest corridor). All landmarks that match the goal descriptor become goals and spread activation. Based on the greedy algorithm, the robot always pursues the path to the nearest one.

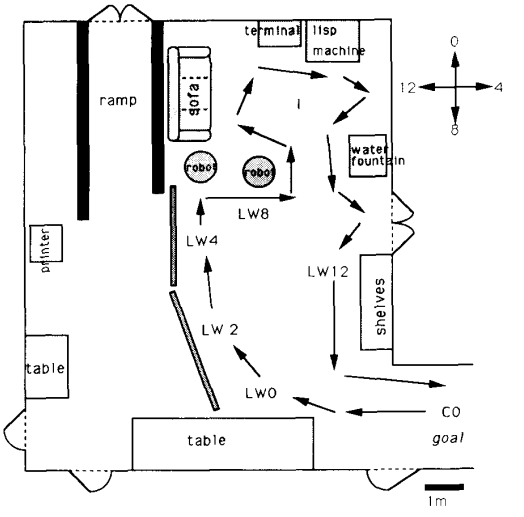

Fig. 11. In the shown environment, the robot records a cluttered area as a long irregular boundary (indicated by landmark type I).

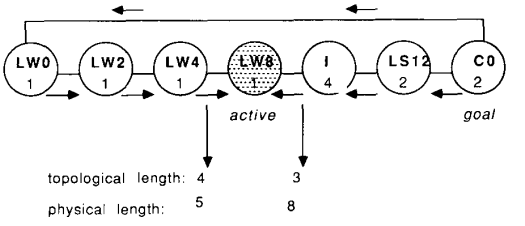

Fig. 12. The graph the robot constructed of the shown environment. Shaded nodes indicate the robot's current location and the goal, and the arrows indicate the direction of activation propagation. To test the robot's ability to use the correct measure of optimality, the corridor was chosen as the goal when the robot was located at LW8. The topologically shortest path involves going through a cluttered area with a long irregular boundary. Consequently, the robot consistently chose the topologically longer but physically shortest path around the room.

Fig. 11 shows an environment used for testing path finding and optimization. After exploring the environment and learning its structure, the robot is given the corridor as the goal. Given a choice of paths from its current position (LW8), the system takes into account that the shortest topological path does not correspond to the shortest physical one (Fig. 12). In seven consecutive trials, the system consistently correctly chose the topologically longer but physically shorter path to the goal.

VII. Hardware Implications

The system we present is best suited for coarse-grained parallel hardware. In general, graph structures with arbitrary, dynamically assignable connections between nodes can escalate to full connectivity and do not scale well. In contrast, our approach employs only a few global broadcast connections

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

in addition to nearest-neighbor connections between adjacent graph nodes.

To further limit the graph connectivity, we utilize some domain knowledge about the robot's environment. Based on the boundary following behavior, the robot has no more than a small, fixed number f of directions to pursue from any given location in an office environment. Consequently, we can bound the out degree of the graph to f. In our implementation, $f = 8$, bounded by the resolution of the compass in the half-plane (from any convex boundary the robot can pursue at most as many directions as it can distinguish with the compass). This results in the total connectivity linear $O(n)$ in the size of the graph.

The choice of a parsimonious representation that encodes only the necessary information simplified the control system and resulted in notably small object code. For example, a network of 10 landmarks takes up 51 kilobytes. The division between code and data is blurred since the graph representation, which comprises most of the code, is actually data. This means that scaling to larger maps requires only a linear increase in hardware. This contrasts with approaches relying on a reasoner that may suffer from a combinatorial explosion with an increase in the problem size.

The approach we present has been shown to be biologically feasible and to resemble some properties of the spatial mapping mechanism of a rat [26]. In particular, the distributed nature of the representation and its direct integration with action has biological analogs [29].

VIII. Related Work

Path planning systems initially relied on purely deliberative, nonreactive solutions (e.g., [6], [8], [14], [16], [21], [30], etc.). More recently, most path planning systems implemented on physical robots have introduced a reactive layer and have been implemented in the hybrid style (e.g., [2], [31], etc.). Exceptions include [10], which suggests a completely reactive subsumption-based scheme for navigation by path remembering, but this scheme was never fully developed or implemented. Lumelski and Stepanov [22] describe an entirely local navigation strategy independent of a map. Using Cartesian coordinates of the goal, the system relies on position control to either reach the goal or recognize failure. While general, the system does not build or maintain a representation of the environment in order to optimize its paths to the goal. Kuipers and Byun [18] describe a qualitative spatial learning method based on a landmark strategy similar to ours. The key differences lie in the nature of its landmarks (they are signature based), the lack of metric information (no metric path optimization is done), the static world assumption (no moving obstacles are allowed), and the fact that the system was tested in simulation. Examples of relevant graph discovery and exploration work are given in [11] and [13]. However, these are theoretical results that, in order to be applied to robots, require a perfect ability to determine the local graph structure. Such ability has not yet been demonstrated in physical robots.

IX. Limitations and Extensions

In order to implement a nonhybrid architecture, we attempted to minimize the translation overhead between the robot's sensor space and the map representation. We chose a topological scheme, and instead of place-and-path graphs, used extended landmarks and their adjacency relationships. These primitives were chosen because they are directly available from the robot's sensors and its low-level, reactive control system.

Consequently, the method we describe is based almost entirely on topological information. This simplifies computation but also limits it to optimizing paths only within the previously traversed path set. In an extension of the existing approach, the rough position information already available in the system could be used for making geometric inferences. For instance, the information could be used to generate shortcuts by producing novel, previously unexplored paths [24]. A related extension to the system would enable the robot to explore by following directions into as yet undiscovered areas.

The current system allows the user to select a specific landmark or a landmark type as a goal by pressing buttons on the top of the robot. While the landmark primitives are well suited to the robot's sensors and its task, they are not intuitive for the user. A mechanism for translating between the robot's and the user's map representation would make the interaction with the robot easier. Additionally, such a mechanism would allow for translating the robot's internal representation into a form that could be used by other robots equipped with different types of sensors. Using the combination of the topological structure (the graph) and the metric data (the landmark lengths, the rough position estimates, the robot's velocity, and the landmark threshold), it is possible to transform the map into other forms more accessible to people or other robots. Currently, the robot's representation is well suited for its sensors and its task. However, if a different representation is needed in the future, the described transformation tool would be an interesting extension of this work.

X. Conclusion

We have described a strategy for integrating a distributed spatial representation into a fully reactive, subsumption-based mobile robot. The robot performs navigation, spatial mapping, and path planning, in real time based on real sensory data. Additionally, the robot can interact with a human operator and receive a variety of goal directives. The strategy is implemented as a collection of concurrently executing behaviors performing both representation and action tasks.

The approach we presented derives its strengths from three main properties of the representation: it is qualitative, procedural, and distributed. The qualitative nature keeps the granularity of the representation low but minimizes the computational overhead. Qualitative sensor characteristics are used to construct a fault-tolerant navigation behavior that facilitates a simple procedural landmark detection algorithm. The distributed representation utilizes the benefits of parallel computation in allowing for constant-time localization and linear-time path planning. The decentralized nature of the map permits

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.

the planning computation to be performed by the map itself rather than by a separate planner. The system detects failures and dynamically adapts the map.

The presented architecture is an alternative to the hybrid approach of separating the reactive and the planning parts of the control system. By removing the distinction between the control program and the map, the described distributed representation introduces added power to fully reactive subsumption-based architectures by extending their domain to applications requiring internal spatial models and interaction with the user.

ACKNOWLEDGMENT

R. Brooks originally proposed the idea of a distributed representation. J. Robles provided many helpful comments on earlier drafts of this paper, as did R. Brooks, N. Pollard, S. Narasimhan, and three constructive anonymous reviewers.

REFERENCES

[1] P. Agre and D. Chapman, "Pengi: An implementation of a theory of activity," in *Proc. 6th Nat. Conf. Artificial Intell., AAAI-87* (Seattle), 1987, pp. 268–272.

[2] R. Arkin, "Toward the unification of navigational planning and reactive control," in *Proc. AAAI Spring Symp. Robot Navigation Working Notes*, Mar. 1989, pp. 1–5.

[3] R. Brooks, "Intelligence without representation," *Aritificial Intell. J.*, vol. 47, pp. 139–159, Jan. 1991.

[4] ——, "A robot that walks; Emergent behavior from a carefully evolved network," *Neural Comput.*, vol. 1, no. 2, pp. 253–262, 1989.

[5] ——, "A robust layered control system for a mobile robot," *IEEE J. Robotics Automat.*, vol. RA-2, pp. 14–23, Apr. 1986

[6] ——, "Solving the find-path problem by good representation of free space," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-13, no. 3, Apr., 1983, 190–197

[7] R. Brooks and J. Connell, "Asynchronous distributed control system for a mobile robot," in *SPIE's Cambridge Symp. Optical and Opto-Electronic Eng. Proc.*, vol. 727, Oct. 1986, pp. 77–84.

[8] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.*, Mar. 1985, pp. 138–145.

[9] J. Connell, "A colony architecture for an artificial creature," MIT Artificial Intell. Lab., Tech. Rep. 1151, Oct. 1989.

[10] ——, "Navigation by path remembering," *SPIE Mobile Robots III*, vol. 1007, pp. 383–390, 1988.

[11] X. Deng and H. Papadimitriou, "Exploring an unknown graph," in *Proc. 31st Ann. Symp. Foundations Comput. Sci.*, vol. 1, Oct. 1990, pp. 355–361.

[12] M. Drumheller, "Mobile robot localization using sonar," MIT Artificial Intell. Lab., Memo 826, 1986.

[13] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotics exploration as graph construction," Res. Biological and Computat. Vision Tech. Rep. 23, Nov. 1988.

[14] A. Elfes, "A sonar-based mapping and navigation system," in *Proc. IEEE Int. Conf. Robotics Automat.*, Feb. 1986.

[15] A. Flynn, R. Brooks, S. Wells, and D. Barrett, "Squirt: The prototypical mobile robot for autonomous graduate students," MIT Artificial Intell. Lab., Memo 1120, July 1989.

[16] G. Giralt, R. Chatila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," in *First International Symposium on Robotics Research*, M. Brady and R. Paul, Eds. Cambridge, MA: MIT Press, 1984, pp. 191–214.

[17] B. Kuipers, "A qualitative approach to robot exploration and map learning," in *Proc. AAAI Workshop Spatial Reasoning and Multi-Sensor Fusion* (Chicago), Oct. 1987.

[18] B. Kuipers and Y. Byun, "A robust, qualitative approach to a spatial learning mobile robot," in *SPIE Sensor Fusion: Spatial Reasoning and Scene Interpretation*, Nov. 1988, pp. 366–375.

[19] R. Kuc and Y. Di, "Intelligent sensor approach to differentiating sonar reflections from corners and planes," in *Proc. Int. Congress Intell. Autonomous Systems* (Amsterdam), 1986.

[20] R. Kuc and R. Siegel, "Physically based simulation model for acoustic sensor robot navigation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9 no. 6, pp. 766–778, Nov. 1987.

[21] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulation," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 224–238, June 1987.

[22] V. Lumelski and A. Stepanov, "Dynamic path planning for a mobile automation with limited information on the environment," *IEEE Trans. Automat. Control*, vol. CA-31, no. 11, Nov. 1986.

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

[24] M. Mataric, "A distributed model for mobile robot environment-learning and navigation," MIT Artificial Intell. Lab., Tech. Rep. 1228, June 1990.

[25] ——, "Environment learning using a distributed representation," in *Proc. IEEE Int. Conf. Robotics Automat.*, May 1990, pp. 402–406.

[26] ——, "Navigating with a rat brain," in *Proceedings of the 1990 International Conference on Simulation of Adaptive Behavior*, J. Meyer and S. Wilson, Eds. Cambridge, MA: MIT Press, 1990, pp. 169–175.

[27] ——, "Qualitative sonar based environment learning for mobile robots," in *SPIE Mobile Robots IV Proc.*, Nov. 1989.

[28] M. Mataric and R. Brooks, "Learning a distributed map representation based on navigation behaviors," in *Proc. USA–Japan Symp. Flexible Automat.*, June 1990, pp. 499–506.

[29] B. McNaughton, "Neuronal mechanisms for spatial computation and information storage," in *Neural Connections, Mental Computation*, L. Nadel, L, A. Cooper, P. Culicover, and R. M. Harnish, Eds. Cambridge, MA: MIT Press, 1989, pp. 285–350.

[30] H. Moravec and D. Cho, "A Bayesian method for certainty grids," in *Proc. AAAI Spring Symp. Robot Navigation*, Mar. 1989, pp. 57–60.

[31] D. Payton, "Internalized plans: A representation for action resources," in *Designing Autonomous Agents*, P. Maes, Ed. Cambridge, MA: MIT Press, 1991, pp. 89–103.

[32] *Polaroid Ultrasonic Ranging System Handbook, Application Notes and Technical Papers*, Polaroid Corporation, 1987.

[33] S. Rosenschein and L. Kaelbling, "The synthesis of digital machines with provable epistemic properties," in *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, J. Halpern, Ed. Los Altos, CA: Morgan Kaufmann, 1986, pp. 83–98.

[34] C. K. Yap, "Algorithmic motion planning," in *Algorithmic and Geometric Aspects of Robotics*, J. T. Schwartz and C. K. Yap, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 95–143.

Maja J. Mataric was born in Beograd, Yugoslavia, in 1965. She received the B.S. degree in computer science (with honors) from the University of Kansas, Lawrence, in 1987 and the M.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1990. She is currently working toward the Ph.D. degree at the MIT Artificial Intelligence Laboratory.

In addition to her doctoral studies, she has been a Research Assistant at the MIT Artificial Intelligence Laboratory since 1987. She has also worked at the NASA Jet Propulsion Laboratory, the Free University of Brussels AI Laboratory, and the GTE Laboratories Self-Improving Systems Group. She has been involved in the field of autonomous mobile robots, integrated agent architectures, and machine learning. Her current research includes work on the theory of and experiments in multiple robot interaction, imitation and social learning, and behavior synthesis and analysis.

Authorized licensed use limited to: MIT Libraries. Downloaded on November 3, 2008 at 16:41 from IEEE Xplore. Restrictions apply.