Autonomous Robots 6, 293–308 (1999)
© 1999 Kluwer Academic Publishers. Manufactured in The Netherlands.

Integrating Exploration, Localization, Navigation and Planning with a Common Representation

ALAN C. SCHULTZ, WILLIAM ADAMS AND BRIAN YAMAUCHI*
Navy Center for Applied Research in Artificial Intelligence (NCARAI), Naval Research Laboratory, Washington, DC 20375-5337, USA
schultz@aic.nrl.navy.mil

Abstract. Two major themes of our research include the creation of mobile robot systems that are robust and adaptive in rapidly changing environments, and the view of integration as a basic research issue. Where reasonable, we try to use the same representations to allow different components to work more readily together and to allow better and more natural integration of and communication between these components. In this paper, we describe our most recent work in integrating mobile robot exploration, localization, navigation, and planning through the use of a common representation, evidence grids.

Keywords: mobile robots, localization, planning, navigation, exploration, evidence grids, integration

1. Introduction

A central theme of our research is the view of integration as a basic research issue, studying the combination of different, complementary capabilities. One principle that allows integration is the use of unifying representations. Where reasonable, we try to use the same representations to allow different components to work more readily together and to allow better and more natural integration of and communication between these components. In the work reported here, the unifying representation is the evidence grid, a probabilistic metric map. In this paper, we describe how using evidence grids as a unifying representation not only allows for better integration across techniques, but also allows reuse of data in learning and adaptation.

We have developed and integrated techniques for autonomous exploration, map building, and continuous self-localization. Further, we have integrated these techniques with methods for navigation and planning obtained from other research groups, modifying their systems to use our common representation. In addition, this integrated system includes methods for adapting maps to allow for robust navigation in dynamic environments. As a result of this integration, the system allows a robot to enter an unknown environment, map it while remaining confident of its position, and robustly plan and navigate within the environment in real time.

In the next section, we describe the common representation we use for integrating the various techniques. In Sections 3 and 4, we present our results in localization and exploration, along with our integration of these techniques. The integration of a mechanism to make the the map adaptive to changes in the environment is presented in Section 5. In Sections 6 and 7 we introduce the components for planning and reactive navigation, and show how they integrate into the system using our representation. In Section 8, we describe the overall integrated architecture and describe experiments to verify that the resulting system works robustly and repeatably, and present the results of these experiments.

2. Unifying Representation

We use evidence grids (Moravec and Elfes, 1985) as our spatial representation. An evidence grid is a probabilistic representation which uses Cartesian grid

*This work was sponsored by the Office of Naval Research.

cells to store evidence that the corresponding region in space is occupied.

Each cell contains a real value in the range $(-1, 1)$ that represents the amount of evidence that a cell is occupied (1) or unoccupied (-1), or indicates that there is not enough information to determine the occupancy of the cell.[1]

Evidence grids have the advantage of being able to fuse information from different types of sensors. To update an evidence grid with new sensor readings, the sensor readings are interpreted with respect to a sensor model that maps the sensor datum at a given pose to its effect on each cell within the evidence grid.[2] The interpretation is then used to update the evidence in the grid cells in real time using a probabilistic update rule. Evidence grids have been created that use different updating methods, most notably, Bayesian (Moravec and Elfes, 1985), and Dempster-Shafer (Hughes and Murphy, 1992). In the results reported here, Bayesian updating is used.

In this study, we use sonar sensors in combination with a planar structured light range finder. Sonar sensors can provide only coarse evidence of occupied space due to their wide field, but they are very effective at determining empty space, as an object anywhere within that space would likely have resulted in a shorter sensed range. The structured light range finder has the opposite properties. It can sense occupied space at a high resolution, but its horizontal, 2-D nature prevents it from sensing objects above or below the structured light plane. It therefore cannot be used with any confidence to rule the intervening space as empty.

For the sonar sensor model, grid cells in an arc at the sensed range receive a higher evidence of being occupied, while cells between the sensor and the sensed distance receive reduced evidence of being occupied. Since a sonar sensor is more likely to detect an object near its axis, cells closer to the sensor's axis receive larger adjustments than cells far from the axis. More information on the sonar sensor model is available in (Moravec, 1988). The sensor model for the structured light range finder provides strong evidence at the cell where the range datum lies, but makes no adjustment to any intermediate cells.

In order to reduce the effect of specular reflections, we have developed a technique we call laser-limited sonar. If the laser returns a range reading less than the sonar reading, we update the evidence grid as if the sonar had returned the range indicated by the laser, in addition to increasing the occupancy probability of the cells actually returned by the laser.

Although evidence grids may represent a three-dimensional space, our initial results examine a single horizontal layer of the evidence grid that is located at the height of the sensors.

We create two types of representations with the evidence grids: short-term perception maps, and long-term metric maps. The short-term maps store very recent sensor data that does not contain significant odometry error, and these maps are used for obstacle avoidance and for localization. The long-term maps represent the environment over time, and are used for navigation and path-planning.

2.1. Long-Term Maps

A long-term map is an evidence grid representation of the environment that is built from many sensor readings, over a long time period in that region of space. Typically, each long-term map will represent approximately one "room" in the environment. All sensor data contributes to this map, and this map can be used by other robotic processes, such as navigation and path planning. Figure 1 shows an evidence grid of the robotic laboratory at NCARAI. The white space represents cells that have evidence of the cell *not* being occupied (free space), the darker areas represent evidence of the cell being occupied, and the gray areas (like in the outer parts of Fig. 1) indicate cells where neutral evidence exists.

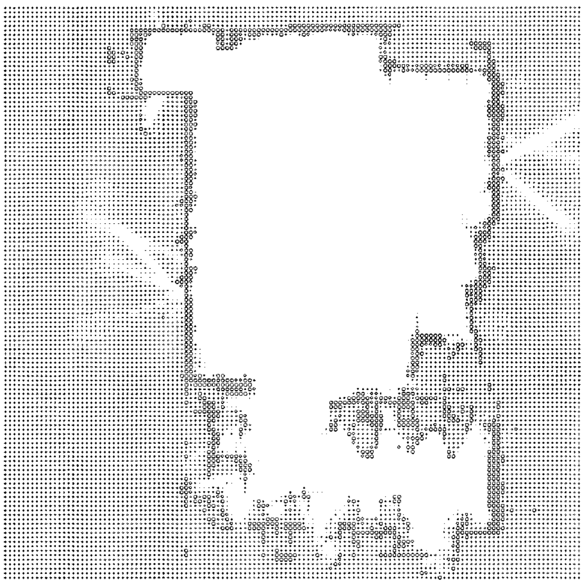
Figure 1. Long-term map of laboratory.

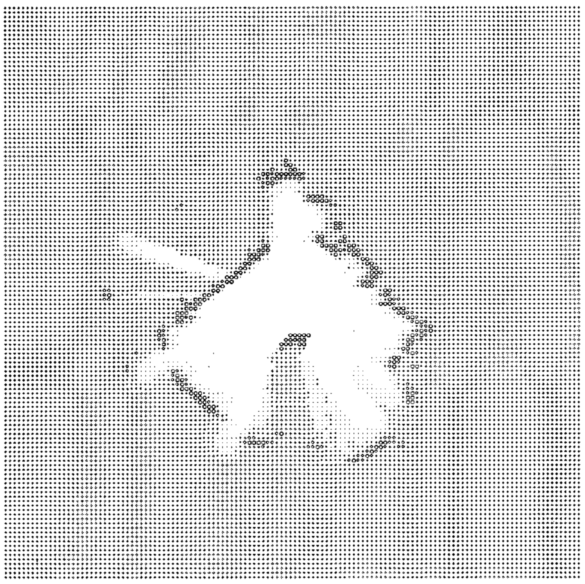

Figure 2. A short-term perception map.

2.2. *Short-Term Perception Maps*

A short-term (or local) perception map represents the immediate temporal and spatial environment of the robot as an evidence grid. Only very recent sensor readings of the robot contribute to the local perception map. Several local perception maps of the robot's environment may exist at the same time, each with a different amount of sensor data contributing to the "maturity" of that map. A local perception map is considered mature when it has accumulated the maximum desirable amount of positional error. After a map has matured, it is used and then discarded.

Figure 2 shows a local perception map recorded by the robot while it was in the upper, left hand corner of the room in Fig. 1. (The short-term map is rotated 45° with respect to the long-term map.) The white and dark areas have the same meaning as in the previous figure. Note that objects are present in the local perception map that were not present when the long-term map was created.

3. Learning Where You Are: Continuous Localization

Evidence grids provide a uniform representation for fusing temporally and spatially distinct sensor readings. However, the use of evidence grids requires that the robot be localized within its environment. Due to odometric drift and non-systematic errors such as slippage and uneven floors, odometry errors typically accumulate over time making localization estimates degrade. This can introduce significant errors into evidence grids as they are built. We have addressed this problem by developing a method for *continuous localization*, in which the robot corrects its position estimates incrementally and on the fly (Schultz and Adams, 1998).

Continuous localization exploits the fact that the robot's odometric error usually increases gradually over time, except in extreme cases such as when the robot hits an obstacle. By re-localizing often, less effort is required to correct the error in odometry.

Continuous localization builds local perception maps of the robot's local environment. These maps typically contain very small amounts of error, and are used to locate the robot within a global, long-term map via a registration process. (In Section 4 we will describe how these long-term maps are created.) The results from this process are used to correct the robot's odometry.

Figure 3 shows the process of continuous localization. The robot builds a continuous series of local perception maps of its immediate environment. At the beginning of each interval, a new local perception map is created. During the time interval, new sensor data are fed to the new map and the previous maps still in memory. Each local perception map is of short duration and contains only a small amount of dead reckoning error. After several time intervals, the oldest (most "mature") local perception map is used to position the robot within the long-term map by registering the two

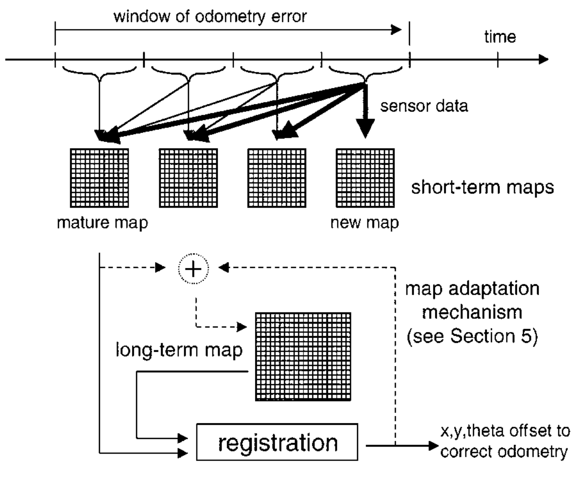

Figure 3. Continuous localization.

maps and is then discarded. The number of local perception maps that exist simultaneously and the amount of data that is entered into each map are runtime parameters of the system.

The registration process involves a search in the space of offsets in translation and rotation that minimizes the error in the match between the short-term and long-term maps. Since we expect the odometry error to be small, we restrict the registration search to be between ± 6 in. in translation and $\pm 2^\circ$ in rotation. (These values can also be changed as runtime parameters.) This restricted search space allows the search to be completed quickly, specifically before the interval expires and the next registration is attempted.

For each tested pose in the registration search, the mature local perception map is rotated and translated by the difference in pose (the offset) and a match score is calculated based on agreement between the cell values of the local perception map and the long-term map, summed across all cells. The match scores for all tested poses are then used to determine the offset that is likely to have the highest match score. This offset is applied to the robot's odometry, placing it at the pose which causes its local perceptions to best match the long-term map. After the registration takes place the most mature map is discarded, and a new local perception map is created.

Two experiments were performed to determine the effectiveness of continuous localization at reducing odometric error, and to determine which of several match functions and search functions yield better results.

3.1. Effectiveness of Continuous Localization

The first experiment was conducted in a room measuring roughly 26×30 ft, open in the center with bookcases, desks, and chairs around the edges of the room. The robot was commanded to follow a square path near the center of the room, 8 ft on each side, by traveling to each corner's coordinates in turn. Continuous localization ran independently of the motion process, maintaining 4 short-term perception maps and re-localizing approximately every 8 ft (each mature short-term map contained sensor data gathered during the most recent 32 ft of travel). The registration search method used was *center-of-mass* with the *binary* match function (described in detail in Section 3.2).

Ten runs were made, with each run consisting of 80 laps around the square, a distance of 2560 ft

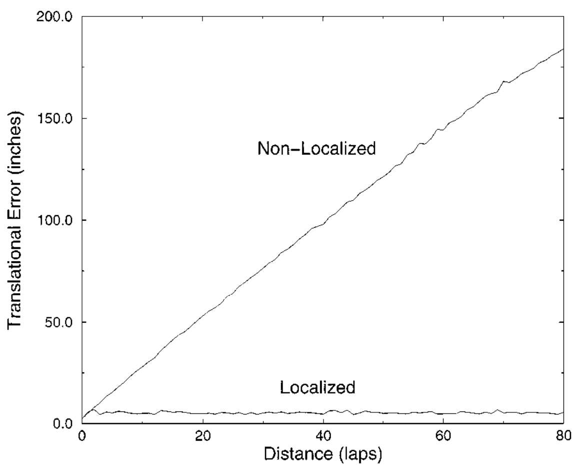

Figure 4. Effect of continuous localization.

(approximately 2 h duration). The distance between the robot's odometic position and its true position was computed at the same corner for each lap. This measure includes rotational error, as motion causes error in orientation to be reflected as an error in position.

The results are displayed in Fig. 4 as an average across all ten runs. The robot's non-localized pose (simple dead-reckoning) steadily drifted, growing without bound. The localized curve shows that continuous localization was able to keep the robot's pose error at a constant level, averaging 5.35 in. (136 mm) with a standard deviation of 2.08 in. (53 mm) across all points of all runs.

3.2. Search and Match Functions

The second set of experiments were run to determine the best of several search routines and matching functions that could be used to register the long-term and short-term perception maps.

In order to describe the search routines, it is useful to first describe the search space in which they work. The search space is all possible poses within ± 6 in. in translation and $\pm 2^\circ$ in rotation of the robot's current pose. This corresponds to a three dimensional space with axes x, y and theta.

The two search routines tested were an *iterated hill climber* and a *center-of-mass calculation*.

The iterated hill climber search (designated in the text and graphs as H) uses an initial resolution to divide the space into pose cells. The match between the short-term perception map and the long-term map is computed for the robot's pose and the center of the 26 immediately neighboring pose cells ($3^3 - 1$). If a neighbor is found with a better match, then the process repeats using that pose cell as the center. If no neighbor is found to be better, then the hill climber re-divides the space at double the resolution and repeats the process. The search stops when a predetermined resolution is reached. For the experiments reported here, an initial step size of 1.5 in. and 1.25° was used, with a final resolution of 0.375 in. and 0.3125°.

The "center-of-mass" search (designated in this paper as C) similarly divides the search space into pose cells, but picks a random pose within each pose cell and uses those random poses to compute a set of match scores that are distributed throughout the search space. The match scores are normalized to the range [0, 1], raised to the fourth power to exaggerate the peak, and then a center-of-mass calculation is performed for all cells. The exaggeration of the peak is necessary because the match score is typically very flat within the small search space, and without it the center-of-mass calculation would always pick a pose near the center of the search space (very close to the robot's current pose). The center-of-mass calculation is preferable to simply choosing the pose cell with the maximum score because the sparse sampling of the space (one pose per pose cell) can create additional noise, and sampling at a higher resolution would be computationally prohibitive for real time operation.

The two match functions examined in this work are designated the *binary match* (referred to in this paper as B), and the *product match* (referred to in this paper as P). For both functions, the short-term map is aligned with the long-term map according to the test pose currently being processed by the search. The evidence from each grid cell of the short-term map is compared to the evidence stored in the spatially-correspondent grid cell of the long-term map, and the score summed across all grid cells. Given the alignment for which the match score is to be computed, if C_L is the corresponding cell in the long-term map to the short-term map cell C_S, then we define the match score:

$$MatchScore = \sum_{all\ C_S} CellScore\big(C_{S_i}, C_{L_i}\big)$$

For each match function, the cell scores are determined as follows. The binary match function (B) compares the cells' evidence for simple agreement. It returns 1 if the cells agree occupied or agree empty, and returns 0 if they disagree or if either cell has no evidence (a value of 0):

$$CellScore_b\big(C_{S_i}, C_{L_i}\big) = \begin{cases} 1 & \text{if } C_{S_i} > 0, C_{L_i} > 0 \\ 1 & \text{if } C_{S_i} < 0, C_{L_i} < 0 \\ 0 & \text{otherwise} \end{cases}$$

The product match function (P) determines the degree of agreement, taking the product of the cells' actual evidence, each cell's evidence being a value between -1 (empty) and 1 (occupied). Cells in agreement produce a score in the range (0, 1], depending on the confidence of their individual evidence. Cells in disagreement produce a score in the range [-1, 0), and if either cell has no evidence, a score of 0 is produced:

$$CellScore_p\big(C_{S_i}, C_{L_i}\big) = C_{S_i}\, C_{L_i}$$

Early work with the continuous localization method revealed that the search space had large regions in which many registration poses resulted in the same match scores. This effect was suspected of causing the hill climber to give up early due to the inability to find a better neighbor in the search space, resulting in a non-optimal choice of pose. To counter this problem, interpolation (designated with I in the following text and graphs) can be performed on the long-term grid cells, such that the center of each grid cell retains its original evidence, but other locations within that grid cell have evidence values bilinearly interpolated with neighboring grid cells. When the search routine aligns the center of the short-term map cells with the long-term map cells, the interpolated evidence value of the long-term map is used for computing the match score. Small variations in pose (map alignment) can thus yield differing corresponding interpolated long-term map cell values and thus differing overall match scores.

To evaluate the various combinations, the same environment was used as in the first experiment, with the robot following the same square path and with pose error being measured at the same corner. Eight trials were conducted, with each trial being a unique combination of search routine, match function and interpolation. (In the following figures and discussion, each trial is designated by the combination of letters H, C, B, P, I indicating which of the above techniques are being used. For each trial, 5 runs were made (except CP and CPI which had 10 runs). Each run consisted of 40 measured points (40 laps), with the pose error measured as before.

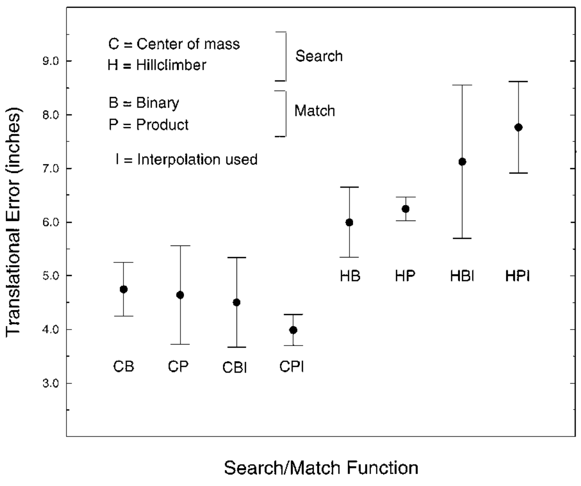

Figure 5. Different search and match functions compared.

Shown in Fig. 5 is the average pose error across all runs for each trial. Error bars indicate a 95% confidence interval. As a group, the center-of-mass combinations were significantly better ($p = .01$) than those using the hill climber. In all cases, the binary and product match functions performed equivalently. Being of roughly comparable computational cost, we have chosen the product match function (P).

The *CP* and *CPI* combinations did not have significantly different performance, nor did interpolation have any consistent effect overall. Interpolation's smoothing of the search space appears unnecessary when used with the center-of-mass search, which performs its own smoothing during the averaging process inherent to it. Since interpolation incurs additional computational cost without providing any additional benefit, the *CP* combination was selected for future work.

3.3. Related Localization Approaches

Yamauchi (1996) uses evidence grids to perform occasional localization by matching evidence grids. In that study, evidence grids are created for each specific "place" along the robot's path. When the robot revisited a specific place, it created a new evidence grid to match against the evidence grid for that location to correct its position.

An alternate search method by Lu (Lu and Milios, 1994) looks promising although it is intended for free-form scans without the use of evidence grids, and the effect of using it on the artificially rasterized data of evidence grids is an open question.

In an approach similar to that presented here, Schiele and Crowley (1994) compared grid matching to other localization methods that included detecting and matching edge segments in the evidence grids. Their work did not give quantitative results on matching evidence grids, nor did it examine various methods for matching or searching for poses. The work presented here seeks to determine the sensitivity of grid matching to changes in some of its fundamental parameters and determine suitable values for them.

Many localization techniques rely on structures in the environment that can serve as landmarks, for example, vertical structures such as door posts and poles (Chenavier and Crowley, 1992), large planes (Horn and Schmidt, 1995), geometric beacons (Leonard, 1992), or regions classified by type (e.g., corridor, intersection, doorway, room) aided by assumptions about the geometry of such structures (Koenig and Simmons, 1998).

Using specific landmarks often requires the robot to perform special maneuvers in order to locate or recognize these landmarks (Bauer, 1995). In our work, such maneuvers are unnecessary. Because our method uses all available sensor data without the requirement of specific features in the environment, the robot can localize itself transparently while carrying out its assigned task.

4. Learning New Environments: Frontier-Based Exploration

In preceding sections we have presented a method for localization that requires a long-term map of the environment. In additional to localization, other robotic tasks also generally require some sort of map. In order to operate in previously unknown environments without assistance, the robot therefore needs the ability to explore and build maps autonomously.

We have developed an exploration strategy based on the concept of frontiers, regions on the boundary between open space and unexplored space. When a robot moves to a frontier, it can see into unexplored space and add the new information to its map. As a result, the mapped territory expands, pushing back the boundary between the known and the unknown. By moving to successive frontiers, the robot can constantly increase its knowledge of the world. We call this strategy *frontier-based exploration* (Yamauchi, 1997).

A process analogous to edge detection and region extraction in computer vision is used to find the

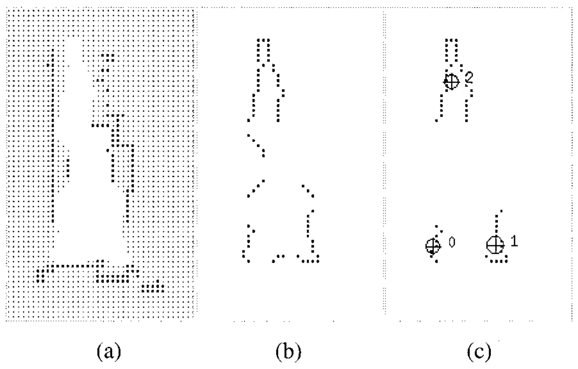

Figure 6. Frontier detection: (a) evidence grid, (b) frontier edge segments, and (c) frontier regions.

boundaries between open space and unknown space in the evidence grid. Any open cell adjacent to an unknown cell is labeled a frontier edge cell. Adjacent edge cells are grouped into frontier regions. Any frontier region above a certain minimum size (roughly the size of the robot) is considered a frontier. Figure 6(a) shows an evidence grid built by a real robot in a hallway adjacent to two open doors. Figure 6(b) shows the frontier edge segments detected in the grid. Figure 6(c) shows the regions that are larger than the minimum frontier size. The centroid of each region is marked by crosshairs. Frontier 0 and frontier 1 correspond to open doorways, while frontier 2 is the unexplored hallway.

Once frontiers have been detected within a particular evidence grid, the robot attempts to navigate to the nearest accessible, unvisited frontier. When the robot reaches its destination (or if the navigation routine determines that the robot cannot get to the frontier), it performs a sensor sweep using laser-limited sonar, and adds the new information to the evidence grid. The robot then detects frontiers in the updated grid, and navigates to the nearest remaining accessible, unvisited frontier.

We have demonstrated that frontier-based exploration can successfully map real-world office environments (Yamauchi, 1997), and that this technique scales well for use in multi-robot environments (Yamauchi, 1998). In relatively small environments, such as a single office, frontier-based exploration was capable of mapping accurately using dead reckoning for position estimation. However, for larger environments, dead reckoning errors would generate large errors in the generated maps. In the next section, we show how continuous localization and frontier-based exploration were integrated to allow accurate mapping of large environments.

4.1. Integrated Exploration and Localization

Frontier-based exploration provides a way to explore and map an unknown environment, given that a robot knows its own location at all times. Continuous localization provides a way for a robot to maintain an accurate estimate of its own position, as long as the environment is mapped in advance. The question of how to combine exploration with localization raises a “chicken-and-egg” problem: the robot needs to know its position in order to build a map, and the robot needs a map in order to determine its position. By integrating continuous localization and frontier-based exploration, we can solve this problem, allowing the robot to explore and build a map while maintaining an accurate estimate of its position (Yamauchi et al., 1998).

This works because the exploration strategy will only take the robot as far as the edge of its “known world,” such that about half of its sensors can still see the old, known environment, which can be used to localize, while its other sensors are extending the map into the unknown environment. Frontier-based exploration and continuous localization run in parallel. Whenever the robot arrives at a new frontier, it adds to the map of the environment and passes this map to continuous localization. Continuous localization uses this map of the known world as its long-term map. As the robot navigates to the next frontier, continuous localization constructs local perception maps based on the robot’s recent perceptions, and compares them to the long-term map to correct the robot’s position estimate. When the robot arrives at the new frontier, its position estimate will be accurate, and new sensor information will be integrated at the correct location within the map.

4.2. Effectiveness of Integrated Exploration and Localization

To measure the effectiveness of our combined system we conducted a set of experiments in a hallway environment (70 ft long). This hallway, like many of those in office buildings, is cluttered with obstacles, and also contains large alcoves.

We initially constructed a ground truth grid (Fig. 7) for a hallway environment by manually positioning the robot at viewpoints throughout the hallway and sweeping the robot’s sensors. This ground truth grid is used only to measure the accuracy of the learned map, and not as an aid to exploration or localization. The five Xs correspond to the robot’s starting locations for the exploration trials, and the four crosshairs indicate

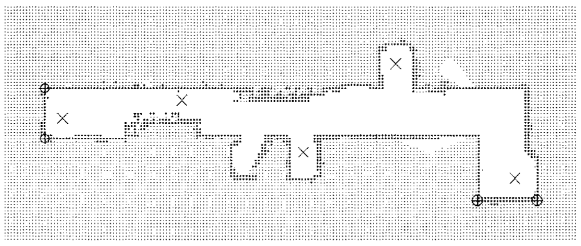

Figure 7. Ground truth evidence grid for hallway.

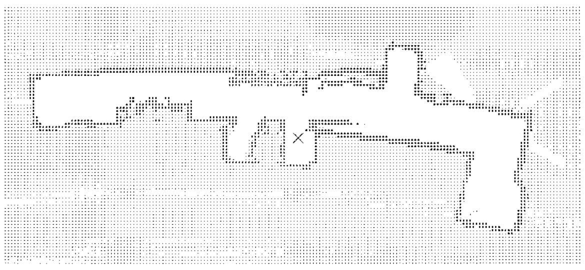

Figure 8. Map learned without localization.

reference points for measuring map error. By measuring the difference between the actual position of these reference points and the position of these points in the learned map, the amount of positional error incorporated into the learned map can be estimated. We refer to this metric as the reference point error for the learned map.

Our first set of trials measured the performance of frontier-based exploration without continuous localization. Five exploration trials were conducted from starting locations distributed throughout the hallway. Figure 8 shows a map learned with the robot starting at the position marked with the X. As the robot explored, positional error constantly accumulated. This particular map has a reference point error of 7.0 ft.

Frontier-based exploration without localization was successful at mapping the entire hallway in 60% of the trials. In the two unsuccessful trials, the positional error was sufficiently large to prevent further exploration. In the successful trials, the average reference point error for the learned maps was 7.9 ft.

Our second set of trials measured the system's performance using frontier-based exploration in combination with continuous localization. We used the same hallway environment, the same starting points for the robot, and the same ground truth evidence grid. Frontier-based exploration again directed the robot to explore the environment, but continuous localization

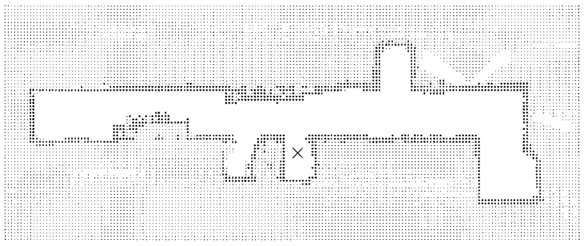

Figure 9. Map learned with localization.

also regularly updated the robot's position estimate as the robot explored.

Figure 9 shows the map learned with continuous localization enabled starting from the same initial position as in Fig. 8. This map has a reference point error of only 0.4 ft, which is equal to the width of a single grid cell. The entire hallway was successfully mapped in all of the trials, and the reference point error averaged over the five learned maps was 2.1 ft.

4.3. Related Exploration Approaches

While other systems have been developed for mobile robot exploration, they have been limited to constrained environments, e.g., where walls are either parallel or perpendicular to each other (Lee, 1996; Thrun and Bücken, 1996) and sufficiently uncluttered as to allow reliable line fitting to the sensor data (Thrun, 1998), or where the entire environment can be explored using wall-following (Mataric, 1992). Our system differs in being able to explore unstructured environments where walls and obstacles may be of any shape and orientation.

5. Learning Dynamic Environments: Adaptive Long-Term Maps

In addition to initial mapping of an unknown environment, we are also interested in learning and representing changes that occur after the robot has finished exploration but do not require complete re-exploration. These changes can include opened or closed doors, moved furniture, temporary storage of bulky items, and stationary people. To this end, we have extended the continuous localization algorithm to allow the long-term map to be updated with recent sensor data from the short-term perception maps, making the long-term map adaptive to the environment (Graves et al., 1997).

In the continuous localization process, after the mature short-term perception map is used to correct the robot's dead reckoning, the odometry correction is also applied to the short-term perception map itself. Its cells are then combined with the spatially corresponding cells of the long-term map using Bayesian updating (dashed lines in Fig. 3). The cells are weighted by a learning rate that controls the effect the short-term map has on the long-term map.

Because the short-term maps are constructed over time, the correction produced by the registration actually corrects the average error accumulated during that time. If there is systematic pose error (e.g., due to the mechanics of the robot), then the correction reflects the amount of error at some point in the recent past, not the current error. A small amount of error will therefore remain in the pose-corrected mature short-term map and will be added into the long-term map, causing a slow blurring of the long-term map.

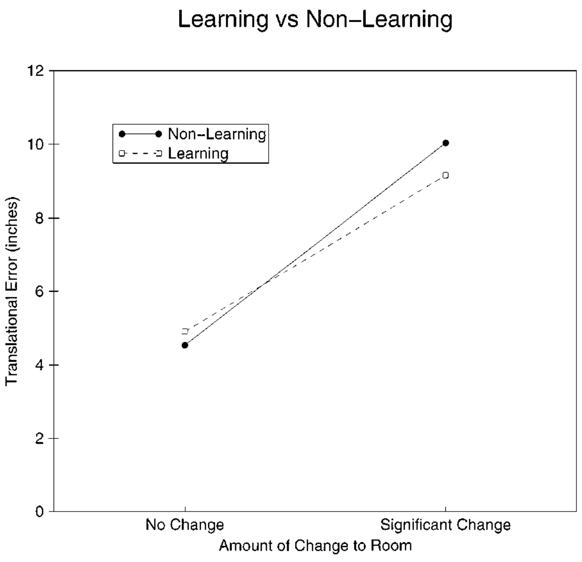

Figure 10. Effect of adaptive long-term map on localization.

5.1. Effectiveness of Adaptive Long-Term Maps

Two experiments were run to determine, for a changing environment, if the mean odometry error is comparable under both the learning and non-learning techniques, if the learning technique could provide accurate maps of the modified environment, and the effect of the slow long-term map blurring on localization.

The first experiment established a mean odometry error for each localization technique when the long-term map represented the true room configuration. The experiment was composed of one learning trial and one non-learning trial, each having eight runs. All runs used the same room configuration. Each run consisted of the robot beginning at a randomly determined pose and then wandering around the room randomly for fifty minutes, avoiding obstacles while continuous localization corrected its odometry. The robot stopped to allow recording of its internal odometry and true location at 1 min intervals. These paired pose readings allowed us to compute the error in the robot's odometry over the course of each run.

The second experiment tested each technique's ability to provide accurate localization when the a priori long-term map significantly differed from the robot's true environment. Before each run, eight objects (such as chairs, desks, etc.) were moved in the real world, though their positions in the a priori map did not change. Each object was displaced 30 in. in a random direction from its original position and then rotated a random amount between -30° and $+30^\circ$ from the original orientation. During each run the objects remained static. For each of eight distinct room configurations, one learning and one non-learning run were conducted. The learning rate was set to 10% (a weight of 0.1) and the random wandering scheme from the first experiment was used. Again, the robot was stopped each minute to record its internal odometry and true location.

The results are summarized in Fig. 10. Each data point in the graph represents the average of the eight runs for each of the learning (dotted line) and non-learning (solid line) trials. The data points on the left show the average translational error when the true room did not deviate from the a priori map, and the data points on the right show the experiments where the room differed significantly from the a priori map.

As expected, in cases where the room had no changes, continuous localization with learning performed no better than the non-learning version (4.91 in. of translational error compared to 4.54 in. of error).

In the second experiment, where the room differed significantly from the map, the learning technique's mean odometry error of 9.29 in. performed marginally better than the non-learning result of 10.02 in. of error.

In addition to the odometry data collected, the long-term maps used during the learning experiments were recorded at each update. These were used to produce an animation of the state of the long-term map while it was adapting to the environment in one of the learning

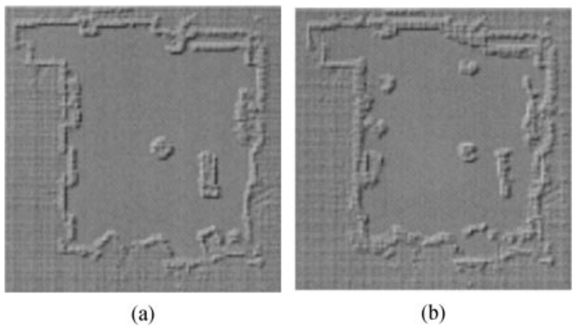

Figure 11. (a) Initial long-term map, and (b) Long-term map after several minutes.

trials with relocated objects. In Fig. 11(a), the initial frame of this animation is shown. This map is accurate for a room with no difference from the original map, but is incorrect for the current, altered room of this run. In Fig. 11(b), a frame from several minutes into the run shows how the map quickly converged to the true room configuration.

Three observations can be made from these results. First, the original, unmodified continuous localization technique performed much better than anticipated when the true room deviated from the a priori map used for localization. This robustness seems to be a result of the way registration is performed, which uses a match score that tends to ignore differences between the maps, and concentrates on similar regions. Note that in these experiments, furniture was moved, but not all of the furniture nor the walls of the room. If the walls of the room were moved, we would expect to see a corresponding error in the localization.

A second observation is that with learning, the map quickly adapts to the changes in the environment, and learns the correct room layout. In the animations, it is possible to see rapid changes in the evidence grid as the map adapts due to the new sensor readings.

The third observation, noticed from the animations, is that after a long period of updating the odometry, some noise would start accumulating in the long-term map because of errors in the content and registration of the short-term map (the short-term maps are only localized to an average error of about 5 in. as exhibited in Section 3.1). This effect was noticed only after long periods of updating (30 to 60 min after the map had adapted to reflect the relocated objects new positions). Despite the accumulated noise when learning, for each room configuration the robot's pose was kept to the same accuracy whether learning was employed or not.

6. Learning to Get Around: Trulla

While experiments up to this point used simple navigation schemes, we have extended our system to use Trulla, a propagation-based path planner (Hughes et al., 1992). Trulla uses a navigability grid to describe which areas in the environment are navigable (considering floor properties, obstacles, etc.). In order to integrate Trulla into our system, we note that Trulla's notion of a navigability grid is similar to our long-term metric map, providing an opportunity to use our common evidence grid representation to support navigation.

Trulla works as follows: beginning from the cell containing the goal, the neighboring cells are explored outward, and each is assigned its own subgoal. Each newly tested cell is assigned the closest subgoal of its already-tested neighbors, if that subgoal is visible from the new cell. If none of the neighbors' subgoals are directly visible, then the new cell lies around the corner of an obstacle, and the neighbor with the closest subgoal is itself assigned as the subgoal of the new cell. In this manner, the shortest paths to the goal are propagated out to all cells. Since each cell can only point to a closer subgoal, the paths that Trulla produces do not suffer from local minima. Once the subgoals are determined, each cell is assigned the direction to its subgoal, resulting in a field of vectors that point in the direction of the shortest path to the goal (see Hughes et al. (1992) for more details on Trulla).

We have replaced Trulla's navigability grid with our long-term map—cell occupancy probabilities are mapped to navigability values. As the long-term map adapts to changes in the environment, as described in Section 5, Trulla can update its paths in real time to reflect the robot's current knowledge about the world.

Figure 12(a) shows an example of a native Trulla navigability grid and the vectors to get from any grid cell to the goal, located in the upper, left-hand corner. Figure 12(b) shows the same area as represented by the long-term map. Figure 12(c) shows the vectors produced for the same goal after a change has occurred to the environment and the long-term map has been updated by continuous localization.

Although the long-term map can adapt to somewhat rapid and persistent changes in the environment, very fast changes, such as a person walking through the room, will not appear in the long-term map. Paths generated by Trulla will avoid persistent obstacles but are not sufficient to prevent collisions with transient obstacles. In related work, Trulla has previously been

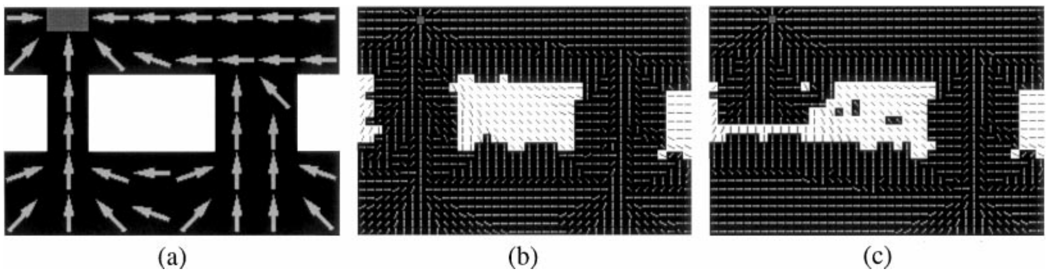

Figure 12. Paths generated by Trulla: (a) native representation, (b) paths using long term map, and (c) paths after adaption.

combined with reactive navigation to avoid collisions with unmodeled obstacles (Murphy et al., 1996). In the work reported here, Trulla is combined with Vector Field Histogram navigation to avoid transient obstacles and to perform reactive navigation.

7. Reactive Navigation: VFH

Vector Field Histogram (VFH) is a reactive navigation method that uses recent, local sensor perception to drive a robot towards a specified goal (Borenstein and Koren, 1991). It was chosen over other methods because of its performance, and because it uses a similar representation of the environment, making integration easier.

VFH uses the Histogrammic In-Motion Mapping (HIMM) method to construct an occupancy grid from sensor readings filtered through a simple sensor model. The area of the HIMM grid immediately surrounding the robot is divided into arcs, and for each arc an object density is computed as the weighted sum of the occupancy values of the grid cells contained by the arc.

Given a goal, VFH searches for the contiguous set of arcs with sufficiently low object density which best matches the direction to the goal. Because the method models the robot as a point object, the free path cannot be blindly followed—the robot's body would collide with the edges and corners of obstacles.

To compensate for this assumption, the HIMM grid is also used to compute a potential field. The resulting repulsion vector is added to the vector from the chosen set of arcs to provide a force away from nearby obstacles while generally heading in the chosen direction. The robot is steered in the direction of this summed heading vector.

In our integration, illustrated in Fig. 13, we replace the HIMM occupancy grid with our unifying evidence grid representation, specifically, the short-term perception map produced by continuous localization. The short-term perception map allows VFH to consider all

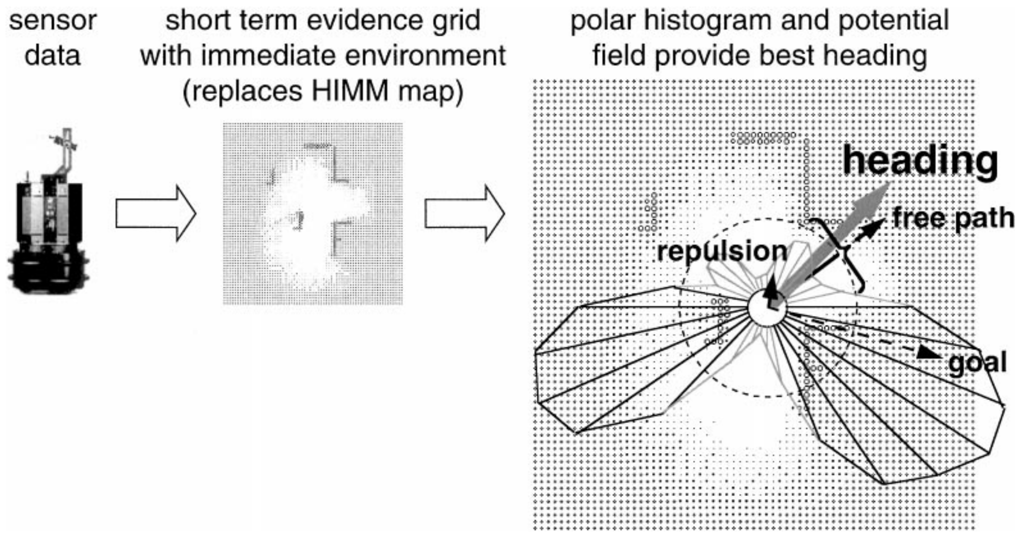
sensor data
short term evidence grid with immediate environment (replaces HIMM map)
polar histogram and potential field provide best heading
heading
free path
repulsion
goal

Figure 13. Integration of vector field histogram.

sensors, and yields a more consistent and less noisy picture of the robot's immediate environment.

8. Integrated Architecture

Figure 14 illustrates the complete architecture. When heading into an unknown environment, the robot autonomously maps the environment, producing the initial long-term map.[3] Continuous localization runs in parallel, regularly correcting the odometry of the robot. While continuous localization maintains the robot's odometry, it regularly produces the short-term perception maps and updates the long-term map, both of which are sent to a separate Map Server process. The Map Server allows the sensor-fused perceptions of the immediate environment to be shared among the various processes, reducing the sensor bottleneck and replicated sensor data gathering and fusion code.

The user (or possibly some other high-level process) specifies a navigation goal to Trulla, which consults the Map Server for the current long-term map and computes the vector field describing the best path from each cell to the goal. Trulla sends the vector field to VFH, which uses the robot's current position to index the vector field and get the direction to the goal. VFH then retrieves the short-term map from the Map Server, computes the object density and potential field, and steers the robot. VFH repeats this sequence until the goal is reached.

While VFH is steering the robot, continuous localization continues to correct odometry and produce short-term and adapted long-term maps. When a new long-term map is available, Trulla replans and sends the new vector field to VFH. When new vector fields or a new short-term map is available, VFH uses them to reactively navigate along the current path to the goal.

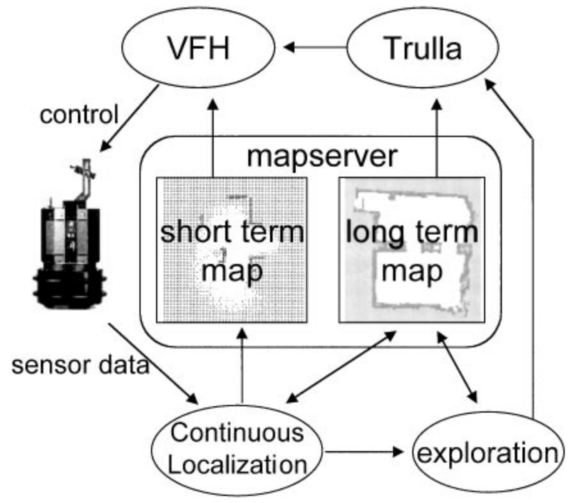

Figure 14. Architecture of integrated system.

8.1. Effectiveness of Integrated System

To demonstrate the capability of our integrated system to plan and navigate reliably in environments with unexpected changes, we conducted four experiments. All four experiments used an environment of two rooms separated by a common wall that contained two passages through which the robot could move from one room to the other. The robot was required to navigate from one room to the other starting with a long-term map learned through exploration. One of the passages was then changed (blocked or unblocked), requiring continuous localization to adapt the map and Trulla to replan accordingly, with VFH providing reactive navigation.

In the first two experiments, the system was given the long-term map shown in Fig. 15(a), with both passages open. However, the left passage was physically blocked as shown in Fig. 15(b). This was the "unexpected blockage" configuration. In the second two experiments, the robot was given the long-term map from Fig. 15(b), which showed the left passage blocked, but the environment was actually configured as shown in Fig. 15(a), with both passages open. This was the "unexpected opening" configuration.

Each room configuration was repeated using learning rates of 0.1 and 0.5. Ten runs were performed for each experiment, with varying start and goal locations chosen near the left side of the environment to

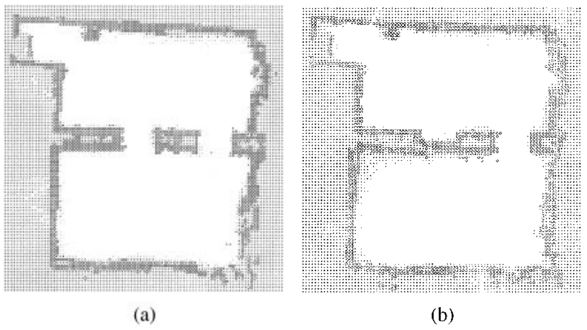

Figure 15. Initial room maps: (a) both passages open, and (b) left passage blocked.

ensure the robot would have an opportunity to sense the changes.

We expected that the higher learning rate would yield faster adaptation and replanning and more blurring around the edges of the map, while the lower learning rate would take longer to adapt but cause less blurring of the map edges. The match of the left passage area of the adapted map with the a priori map for the actual configuration (how well it learned the change) was expected to be roughly the same with either learning rate.

For the unexpected blockage experiments, the robot, as expected, planned a path through the left opening, which its map indicated was open. Approaching the blockage, VFH detected and tried to navigate around the blockage. Continuous localization accumulated evidence of the blockage and updated the long-term map. When the long-term map sufficiently represented the blockage, Trulla replanned its next path through the right passage, which VFH then followed to the goal. The run ended when the robot reached the goal. For the unexpected opening experiments, the robot planned a path through the right passage according to its map, unaware of the shortcut. As the robot passed by the closer opening on its way to the planned passage, sensor readings showing that the left passage was in fact open were obtained as chance permitted, and the long-term map updated. After one or more traversals past the opening, the long-term map indicated the left passage was open and Trulla planned a path through it as the shorter route.

In both the unexpected blockage and unexpected opening experiments, the runs continued until the robot actually traversed the unexpected opening. In the two unexpected blockage experiments, the change is considered learned when the planned paths change enough to cause the robot to follow a path through the right passage, even if the left passage is not completely blocked off in the long-term map. In the two unexpected opening experiments, the change is considered learned when Trulla can first plan a path through the opening in the current direction of travel which has a significant effect on the overall vector field, even if the robot's current position at that time causes it to instead follow a path through the right passage.

All runs were completed without any collisions. During one run of the unexpected opening experiment with learning rate 0.1, the robot's odometry was corrupted (due to a communication network error) and the robot was unable to complete the run. All results for that experiment are based on the nine successful runs.

Figure 16 shows the effectiveness of learning in terms of the match between the learned maps and the actual environment configuration as represented by the initial maps. Values shown are the percentage of cells in agreement—occupied, empty, or unknown. The average time to learn the change in the environment (as defined above) and the average error in the robot's pose (periodically measured during each run) are shown in Table 1.

The lower set of lines in each graph of Fig. 16 illustrates the percentage of matching cells in the local area around the left passage between the adapted map and the initial long-term map which included the change. Initially there is a low match score because the robot started with a map that did not match the environment, but the match improves over time as the long-term map adapts to the true state of the environment. Although the match score would ideally rise to 100%, it does not because of blurring and incomplete learning. The blockage is incompletely learned because the robot can only see the front until it replans through the alternate opening and passes to the rear of the blockage. The upper set of lines in each graph shows the match between the remainder of the adapted map and the initial long-term map. Before learning has had any effect the match is perfect, but over time the edges blur from the inaccuracies in pose.

As shown in Table 1, for a given learning rate, learning the blocked passage case was faster than learning in the unexpected opening case because the robot could gather a lot of sensor data while VFH was trying to navigate the blocked passage prior to the replanning. Learning that the passage was open took longer because it was dependent on getting occasional readings of the area while the robot followed its path through the other passage.

As expected, the learning rate had a significant effect on the ability to quickly adapt to changes. A higher learning rate results in a faster ability to learn the changes in the environment. In addition, there are no significant differences in the pose error as corrected by continuous localization.

By examining the differences across the 10 runs for each of the four experiments, we can examine the ability of the system to perform reliably and repeatably. As can be seen within each graph in Fig. 16, the difference among the runs was very small. The shape of the curves is almost identical, with the main difference being in the length of time required to notice the difference.

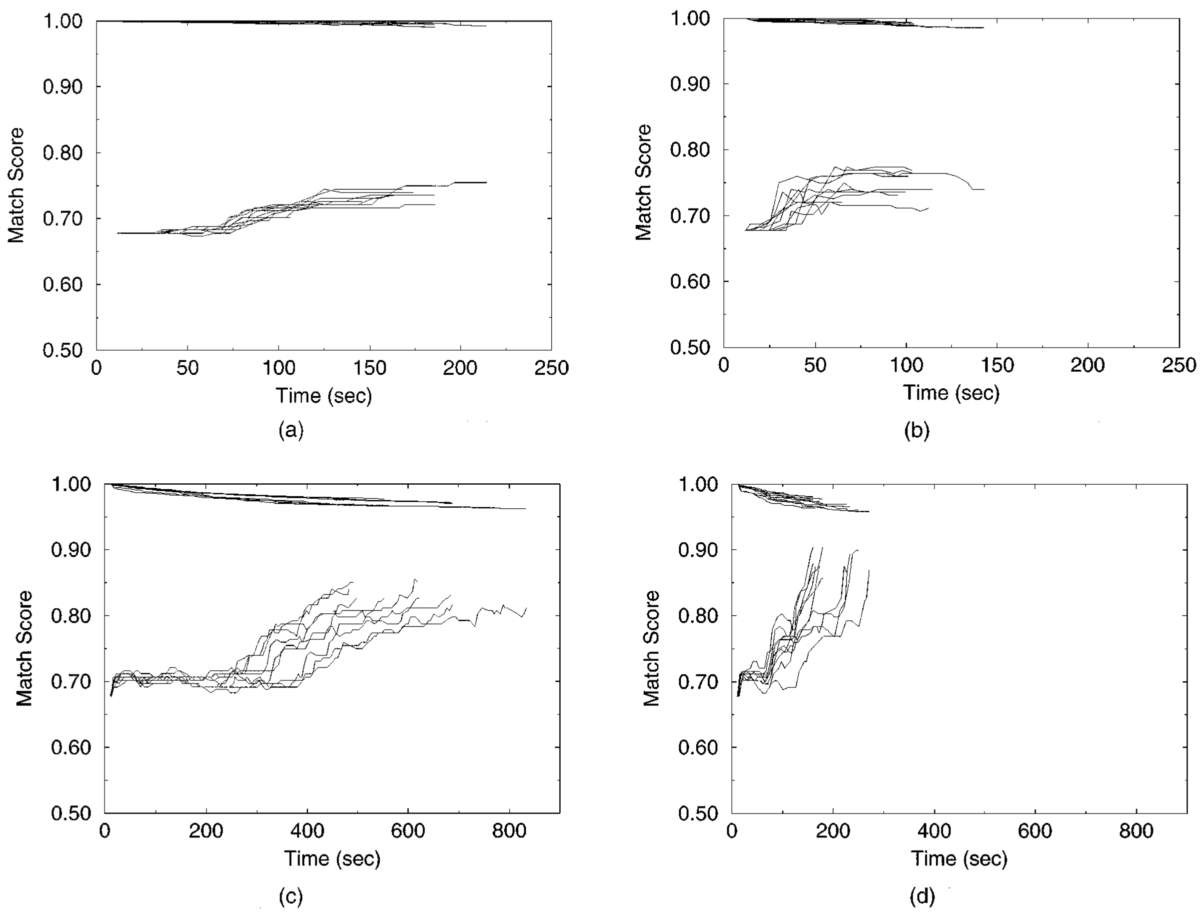

Figure 16. Effect of learning on long-term maps: (a) unexpected blockage, rate 0.1, (b) unexpected blockage, rate 0.5, (c) unexpected opening, rate 0.1, and (d) unexpected opening, rate 0.5.

9. Conclusion

We have created a system which allows robot can enter a previously unknown indoor environment, map that environment while maintaining accurate position information, and robustly plan and navigate within that environment. The system is designed to be adaptive to rapid changes in the environment. Using a unified representation for localization, exploration, reactive navigation and planning components enhanced the ability to integrate these components, allowing for more efficient data reuse. The common representation not only helped in integrating our own modules, but also made it easy to integrate the modules of other research groups.

Experimental results were presented for the effect of the learning rate on adaptation to changing environments, and also to show that the system performs reliably and repeatable. Work continues on a method for storing, identifying and using previously learned environments, using a topological representation for the overall world in which the robot works. In addition, we are enhancing the algorithms to eliminate an assumption that the robot is on level ground.

Table 1. Effects of learning rate: summary.

		Learning rate	
		0.1	0.5
Unexpected	avg time:	123 sec	46 sec
Blockage	avg pose error:	10.3 in	10.3 in
Unexpected	avg time:	493 sec	120 sec
Opening	avg pose error:	7.8 in	6.4 in

Acknowledgments

The authors would like to thank Michael Jones for his work in collecting data for some of the results reported here. Many thanks to the various reviewers who helped make this a better paper.

Notes

1. In the Bayesian method, a value of 0 indicates that being occupied or being empty are equally likely, and this value is generally used as the *priors*. In the Dempster-Shafer method, not having enough evidence can be explicitly modeled.
2. These sensor models may be learned or may be explicitly modeled. Our results use a simple, untuned, explicit model.
3. We are currently extending the system to recognize previously explored environments in which case the map is simply retrieved.

References

Bauer, R. 1995. Active manoeuvres for supporting the localisation process of an autonomous mobile robot. *Robotics and Autonomous Systems*, 16:39–46.

Borenstein, J. and Koren, Y. 1991. The vector field histogram—fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288.

Chenavier, F. and Crowley, J. 1992. Position estimation for a mobile robot using vision and odometry. In *Proc. IEEE Int. Conf. on Robotics and Automation*, IEEE: Nice, France, pp. 2588–2592.

Graves, K., Adams, W., and Schultz, A. 1997. Continuous localization in changing environments. In *Proc. of the 1997 IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, IEEE: Monterey, CA, pp. 28–33.

Horn, J. and Schmidt, M. 1995. Continuous localization of a mobile robot based on 3D-laser-range-data, predicted sensor images, and dead-reckoning. *Robotics and Autonomous Systems*, 14:99–118.

Hughes, K. and Murphy, R. 1992. Ultrasonic robot localization using Dempster-Shafer theory. In *1992 SPIE Stochastic Methods in Signal Processing and Computer Vision*, invited session on applications for vision and robotics.

Hughes, K., Tokuta, A., and Ranganathan, N. 1992. Trulla: An algorithm for path planning among weighted regions by localized propagations. In *Proc. of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Raleigh, NC, pp. 469–475.

Koenig, S. and Simmons, R. 1998. Xavier: A robot navigation architecture based on partially observable Markov decision process models. In *Artificial Intelligence and Mobile Robots*, Kortenkamp, Bonasso, and Murphy (Eds.), AAAI Press, pp. 91–122.

Lee, W. 1996. Spatial semantic hierarchy for a physical robot. Ph.D. Thesis, Department of Computer Sciences, The University of Texas, Austin.

Leonard, J., Durrant-Whyte, H., and Cox, I. 1992. Dynamic map building for an autonomous mobile robot. *The International Journal of Robotics Research*, 11:286–298.

Lu, F. and Milios, E. 1994. Robot pose estimation in unknown environments by matching 2-D range scans. In *Proc. IEEE Computer Vision and Pattern Recognition*, IEEE: Seattle, pp. 935–938.

Mataric, M. 1992. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312.

Moravec, H. and Elfes, A. 1985. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 116–121.

Moravec, H.P. 1988. Sensor fusion in evidence grids for mobile robots. *AI Magazine*, 61–74.

Murphy, R., Hughes, K., and Noll, E. 1996. An explicit path planner to facilitate reactive control and terrain preferences. In *Proc. of the 1996 IEEE International Conf. on Robotics and Automation*, pp. 2067–2072.

Schiele, B. and Crowley, J. 1994. A comparison of position estimation techniques using occupancy grids. *Robotics and Autonomous Systems*, 12:163–171.

Schultz, A. and Adams, W. 1998. Continuous localization using evidence grids. In *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, pp. 2833–2839.

Thrun, S. 1998. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71.

Thrun, S. and Bücken, A. 1996. Integrating grid-based and topological maps for mobile robot navigation. In *Proc. of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, pp. 944–950.

Yamauchi, B. 1996. Mobile robot localization in dynamic environments using dead reckoning and evidence grids. *Proc. IEEE Int. Conf. on Robotics and Automation*, IEEE: New York, NY, pp. 1401–1406.

Yamauchi, B. 1997. A frontier-based approach for autonomous exploration. In *Proc. of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 146–151.

Yamauchi, B. 1998. Frontier-based exploration using multiple robots. In *Proc. of the Second International Conference on Autonomous Agents (Agents'98)*, Minneapolis, MN.

Yamauchi, B., Schultz, A., and Adams, W. 1998. Mobile robot exploration and map-building with continuous localization. In *Proc. of the 1998 IEEE International Conference on Robotics and Automation*, Leuven, Belgium, pp. 3715–3720.

Alan C. Schultz is a computer scientist in the Adaptive Systems Group at the Navy Center for Applied Research in Artificial Intelligence, part of the Naval Research Laboratory in Washington, DC, where he leads several research projects. His research is in the areas of evolutionary robotics, machine learning and evolutionary computation. Mr. Schultz earned his Master's of Science degree in computer science at George Mason University in 1988, and a B.A. in Communications from American University in 1979. Mr. Schultz is a member of IEEE, IEEE computer Society, AAAI, and ACM.

William Adams received his B.S. in computer engineering from Virginia Tech in 1991 and his M.S. degree in electrical and computer engineering from Carnegie Mellon University in 1994. He is currently a research scientist at the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory in Washington, DC, working in many areas of mobile robotics including computer vision, mapping, localization, planning, navigation, and multimodal robot interaction combining speech, gestures, and PDA devices.

Brian Yamauchi is a research associate at the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory. He earned his B.S. in Applied Mathematics and Computer Science from Carnegie Mellon University, his M.S. in Computer Science from the University of Rochester, and his Ph.D. in Computer Science from Case Western Reserve University. He has conducted research and development in mobile robotics the Jet Propulsion Laboratory, Kennedy Space Center, and Hughes Research Laboratories. His current research interests include mobile robot exploration and map-building and multirobot cooperation. Previously, he has published research on place recognition and adaptive navigation in dynamic environments.