Unifying Exploration, Localization, Navigation and Planning Through a Common Representation[*]

Alan C. Schultz, William Adams, Brian Yamauchi, and Mike Jones
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, Washington, DC 20375-5337, U.S.A.
schultz@aic.nrl.navy.mil

Abstract

The 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.

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. In addition, this integrated system includes methods for adapting maps to allow for robust navigation in dynamic environments. As a result, a robot can 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 Section 3, we review our previous results in localization and exploration, along with our integration of these techniques and mechanisms to make them adaptive to changes in the environment. In Sections 4 and 5 we introduce the new components for reactive navigation and planning, and show how they integrate into the system using our representation. In the remaining sections, we 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 [7] as our spatial representation. An evidence grid is a probabilistic representation which uses Cartesian grid cells to store evidence that the corresponding region in space is occupied. 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 [1] The interpretation is then used to update the evidence in the grid cells using a probabilistic update rule. Evidence grids have been created that use different updating methods, most notably, Bayesian [7], and Dempster-Shafer [4]. In the results reported here, Bayesian updating is used.

[*]Submitted to the 1999 International Conference on Robotics and Automation. This work was sponsored by the Office of Naval Research.

[1]These sensor models may be learned or may be explicitly modeled. Our results use a simple, untuned, explicit model.

In this study, we use sonar sensors in combination with a planar structured light rangefinder. 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 marking the cells actually returned by the laser as occupied.

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 can be used for obstacle avoidance and for localization. The long-term maps are used to represent the environment over time, and can be used for navigation and path-planning.

3 Previous Results in Exploration and Localization

3.1 Learning Where You Are

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 [9].

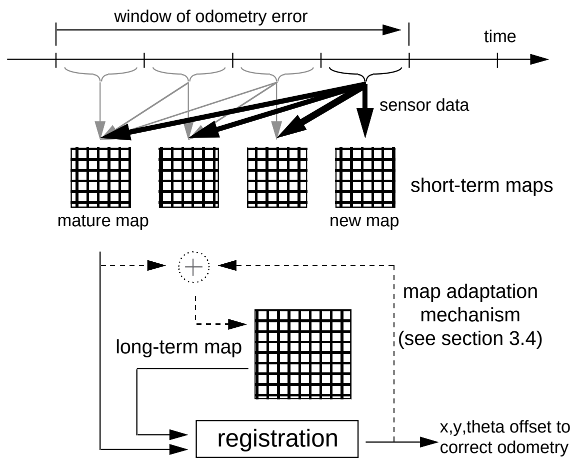


Figure 1: Continuous Localization

Continuous localization builds short-term 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 the next section we will describe how these long-term maps are created.) The results from this process are used to correct the robot's odometry.

Fig. 1 shows the process of continuous localization. The robot builds a continuous series of short-term perception maps of its immediate environment, each of which is of brief duration and contains only a small amount of dead reckoning error. After several time intervals, the oldest (most "mature") short-term map is used to position the robot within the long-term map by registering the two maps.

The registration process consists of sampling the possible poses within a small area around the robot's current estimated pose. For each tested pose, the mature short-term 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 short-term 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 short-term perception map is created. See [2] and [9] for more details, experimental results, and comparisons with other techniques.

3.2 Learning New Environments

In order for mobile robots to operate in unknown environments, they need the ability to explore and build maps that can be used for navigation. 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*[11].

A process analogous to edge detection and region extraction in computer vision is used to find the boundaries between open space and unknown space in the evidence grid. Any open cell adjacent to an unknown

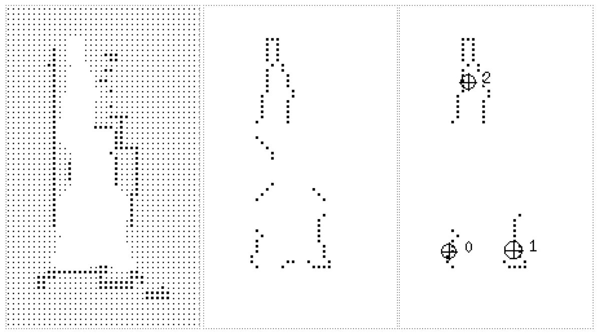


Figure 2: Frontier detection

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. Fig.2a shows an evidence grid built by a real robot in a hallway adjacent to two open doors. Fig.2b shows the frontier edge segments detected in the grid. Fig.2c 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 [11], and that this technique scales well for use in multi-robot environments [12]. 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 we integrated continuous localization and frontier-based exploration.

3.3 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 [13].

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 building up the map in 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 short-term 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.

While other systems have been developed for mobile robot exploration, they have been limited to constrained environments, e.g. where all walls are either parallel or perpendicular to each other [5], [10] or where the entire environment can be explored using wall-following [6]. Our system differs in being able to explore unstructured environments where walls and obstacles may be in any orientation.

3.4 Dynamic Environments: Adaptive Long-Term Maps

We are also interested in explicitly modeling changes that occur in the world after the robot has finished exploration. This is useful for learning and representing changes in the environment. We have extended the continuous localization algorithm to allow the long-

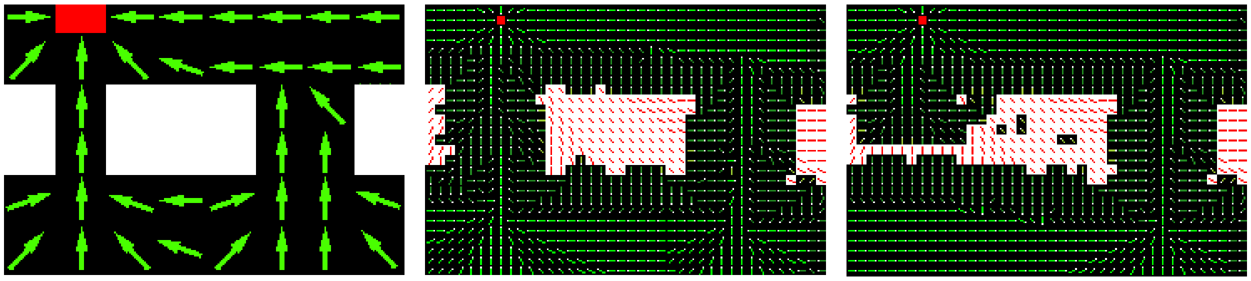

(a) native representation (b) paths using long term map (c) paths after adaptation

Figure 3: Paths generated by Trulla

term map to be updated with recent sensor data from the short-term perception maps, making the long-term map adaptive to the environment [2]. After the most mature short-term map is used to correct the robot's dead reckoning, the odometry correction from the continuous localization process is also applied to the short-term perception map, and then its cells are combined with the corresponding cells of the long-term map using Bayesian updating (dashed lines in Fig. 1). The cells are weighted by a learning rate that controls the effect the short-term map has on the long-term map.

Previous results demonstrated that this allows the robot to recognize and respond to changes in the environment without introducing excessive errors in odometry [2]. In the next section we will describe how this capability can be used in conjunction with planning to allow adaptation to changing environments.

4 Planning: Trulla

While previous versions of our system used a simple path planner, we have extended our system to use Trulla, a propagation-based path planner [3]. 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.

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 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 [3] 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 our long-term map adapts to changes in the environment, as described in Section 3.4, Trulla can update its paths to reflect the robot's current knowledge about the world. Trulla is capable of replanning quickly, and we have reached speeds in excess of one hertz.

Fig. 3a 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. Fig. 3b shows the the same area as represented by the long-term map. Fig. 3c shows the vectors produced for the same goal after a change has occured 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 combined with reactive navigation to avoid collisions with unmodeled obstacles [8]. In the work reported here, Trulla is combined with Vector Field Histogram navigation to avoid transient obstacles and to perform reactive navigation.

5 Reactive Navigation: VFH

Vector Field Histogram (VFH) is a reactive navigation method which uses recent, local sensor perception to drive a robot towards a specified goal [1]. It was chosen over other methods because of its performance and 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.

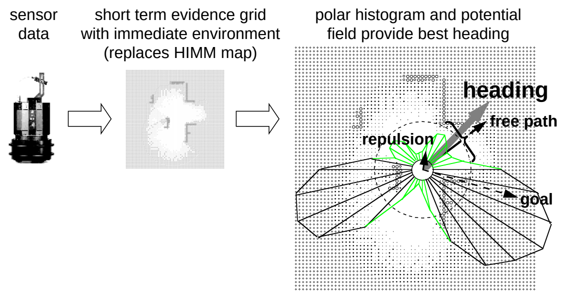

sensor data
short term evidence grid with immediate environment (replaces HIMM map)
polar histogram and potential field provide best heading
heading
free path
repulsion
goal

Figure 4: Integration of Vector Field Histogram

In our integration, illustrated in Fig. 4, we replace the HIMM occupancy grid with the short-term perception map produced by continuous localization. The short-term perception map allows VFH to consider all sensors, and yields a more consistent and less noisy picture of the robot's immediate environment.

6 Integrated Architecture

Fig. 5 illustrates the complete architecture. When heading into an unknown environment, the robot autonomously maps the environment, producing the initial long-term map [2]. 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.

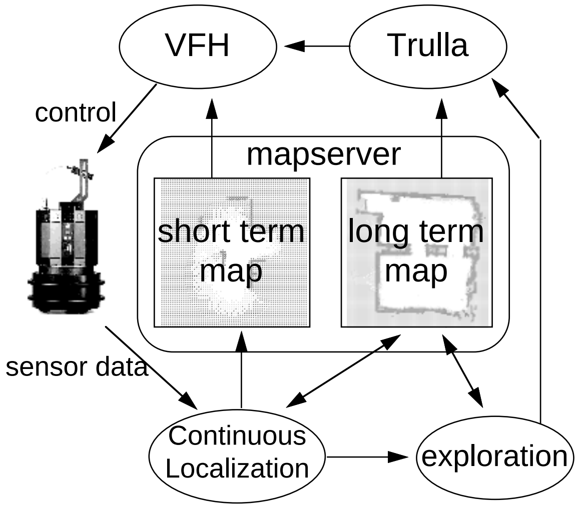


Figure 5: Architecture of integrated system

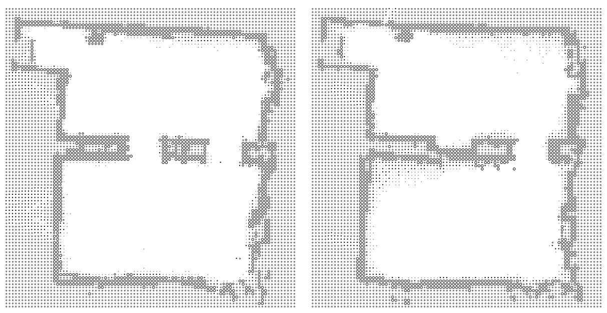
The user (or possibly some other high-level process) specifies a navigation goal to Trulla, which consults the Map Server for the current long-term map and computes the vector field describing the best path from each cell to the goal. Trulla sends the vector field to VFH, which uses the robot's current 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 seqeunce 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 reactive navigate along the current path to the goal.

[2]We are currently extending the system to recognize previously explored environments in which case the map is simply retrieved.

7 Experiment Design

To demonstrate the capability of our integrated system to plan and navigate reliably in environments with unexpected changes, we conducted four experiments with a Nomad200 mobile robot. All four experiments used an environment characterized by a wall between two rooms with one or two passages in which the robot could move between rooms. 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.



(a) both passages open (b) left passage blocked

Figure 6: Initial room maps

8 Results

In the first experiment, the system was given the long-term map shown in Fig. 6a, with both passages open. However, the left passage was physically blocked as shown in Fig. 6b. This was the "unexpected blockage" configuration. In the second experiment, the robot was given the long-term map from Fig. 6b, which showed the left passage blocked, but the environment was actually configured as shown in Fig. 6a, with both passages open. This was the "unexpected opening" configuration.

Each experiment 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 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 fuzziness or 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 experiment, 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 experiment, 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.

Fig. 7 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 unkown. The average time to learn the change in the environment (as defined above) and the average error in the robot's pose (peri-

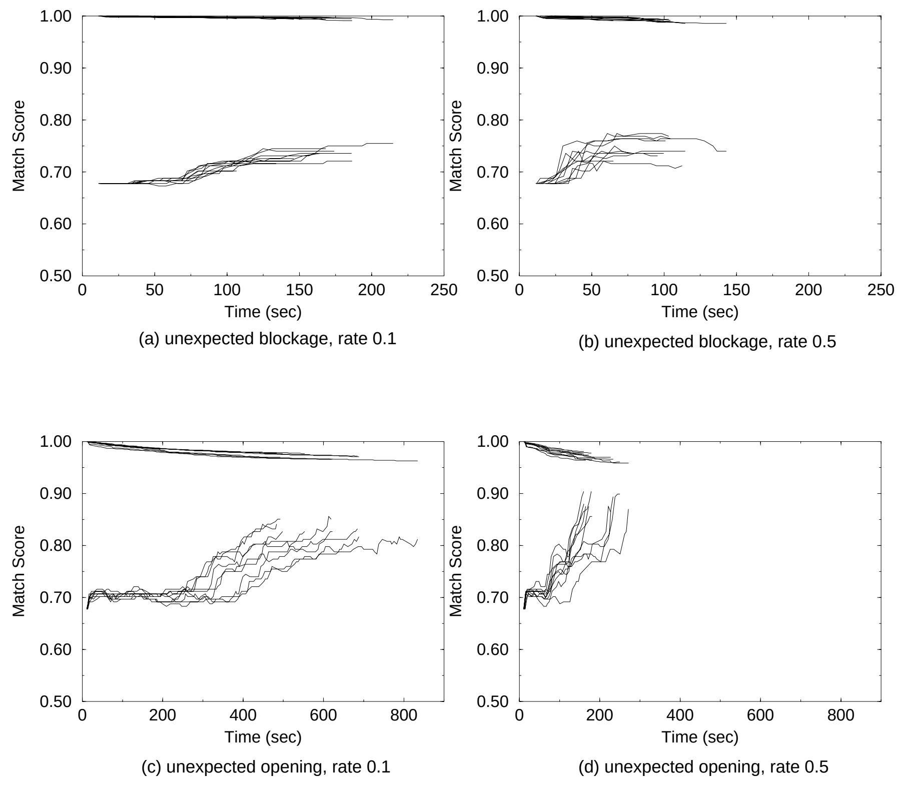


Figure 7: Effect of learning on long-term maps

odically measured during each run) are shown in Table 1.

The lower set of lines in each graph 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 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 percent, 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 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.

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

Table 1: Effects of learning rate: summary

By examining the differences across the 10 runs for each of the four experiments, we could examine the ability of the system to perform reliable and repeatably. As can be seen within each graph in Fig. 7, 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.

9 Conclusion

We have created a system where a 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.

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.

References

[1] **Borenstein, J. and Koren, Y. (1991).** "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots," IEEE Transactions on Robotics and Automation, IEEE: New York, 7(3):278-288.

[2] **Graves, K., Adams, W., and Schultz, A., (1997).** "Continuous Localization in Changing Environments," proc. of the 1997 IEEE Int. Symp. on Computational Intelligence in Robotics and Automation, IEEE, Monterey, CA, July, 28-33.

[3] **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, 469-475.

[4] **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.

[5] **Lee, W. (1996).** "Spatial Semantic Hierarchy for a Physical Robot," Ph.D. Thesis, Department of Computer Sciences, The University of Texas at Austin.

[6] **Mataric, M. (1992).** "Integration of Representation Into Goal-Driven Behavior-Based Robots," IEEE Transactions on Robotics and Automation, 8(3), IEEE: New York, 304-312.

[7] **Moravec, H. and Elfes, A., (1985).** "High Resolution Maps From Wide Angle Sonar," In Proceedings of the IEEE International Conference on Robotics and Automation, 116-121.

[8] **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 Conference on Robotics and Automation, 2067-2072.

[9] **Schultz, A. and Adams, W. (1998).** "Continuous localization using evidence grids," In Proc. of the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium, 2833-2839.

[10] **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, 944-950.

[11] **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, 146-151.

[12] **Yamauchi, B., (1998).** "Frontier-Based Exploration Using Multiple Robots," In Proc. of the Second International Conference on Autonomous Agents (Agents'98), Minneapolis, MN.

[13] **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, 3715-3720.