2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

A Robot in a Water Maze: Learning a Spatial Memory Task

Mark A. Busch, Marjorie Skubic, James M. Keller, and Kevin E. Stone

Abstract – **This paper explores several novel approaches to solve the Morris water maze task. In this spatial memory task, the robot must learn how to associate perceptual information with a particular location to aid in navigating to the goal. A Self-Organizing Feature Map (SOFM) is used to discretize the perceptual space. The robot must then learn to associate these perceptual states with an action used to navigate through the environment. Two navigational approaches are proposed. The first approach involves computing a probabilistic graph between SOFM nodes and then searching the graph to locate a path to the goal. The second approach uses temporal difference learning to learn the association between an SOFM node and an action that will direct it to the goal. The paper compares the effectiveness of these two approaches and discusses their respective utility.**

Index Terms – Morris water maze, Robot spatial memory, Self-organizing feature maps, Spatial learning

I. Introduction

Given the ease with which animals seem to learn and live in unstructured environments, it is reasonable to draw on biological systems as inspiration for creating robots that can also survive in unstructured settings. In this paper, our inspiration comes from an experiment in spatial learning that is typically performed using rats, called the Morris water maze.

The typical environment of the water maze is a pool of opaque water about 1 to 2 meters in diameter with colored visual cues situated around the pool [1]. A small raised platform is hidden somewhere in the pool, just under the water surface. In the water maze experiments, a rat is placed in the water and swims around the pool looking for an escape. In this case, the only escape is the hidden platform, which cannot be sensed from a distance. The rat can only sense the platform when it lands on top of it. After reaching the platform, the rat is plucked from its perch and once again placed in the water. Over time and with several trials, the rat learns quickly to locate the platform based on internal and external perceptual cues.

M. A. Busch is in the Computer Science Department of the University of Missouri, Columbia, MO 65201 USA (e-mail: mab9a5@mizzou.edu).

M. Skubic, J. M. Keller, and K. E. Stone are in the Electrical and Computer Engineering Department of the of the University of Missouri, Columbia, MO 65201 USA (email: skubicm@missouri.edu, kellerj@missouri.edu, kes25c@mizzou.edu)

The water maze problem represents an interesting study in spatial memory, spatial navigation, and learning. This experimental setting provides a useful tool in evaluating the learning of a spatial memory task and is sometimes used as an assessment of spatial learning in rats, e.g., in testing the effects of drugs [2]. Here, our intent is to use it as an assessment tool for investigating perceptual representations and comparing different spatial learning and navigation techniques developed for a mobile robot.

We are not the first to use the water maze experiment for simulated or physical robots. Brown and Sharp [3] used a simulated water maze environment to investigate a neural network-based model of the hippocampus. Neural firings caused the simulated rat to turn in the left or right direction by a certain angle. Results with their model matched the results of experiments with real rats.

Redish and Touretzky [4] investigated a computational model of the hippocampus that supports the dual mechanisms of self-localization and route replay, using a simulated environment. They hypothesized that 5 steps occur in rats learning the water maze: (1) an exploration phase in which the rat learns the environment, (2) self-localization, which allows the rat to determine its position in the water relative to the platform goal, using perceptual cues, (3) route learning, i.e., the storing of path information as the rat traverses along a route, (4) replay of routes during sleep, and (5) consolidation in which the dreamed routes are stored in long term memory.

Balakrishnan et al. [5] also tested a model using simulated animats that incorporated both the cognitive route map concept and self-localization, associating sensory inputs with the direction of the platform. Their animats performed similarly to real rats in the water maze.

Foster et al. [6] use a temporal difference learning approach to explore the mechanisms by which place cells in the rodent brain assist in navigation for a Morris-type water maze. Unlike our work, they do not address visual perceptual cues as an additional mechanism by which the rodent brain can navigate.

Krichmar et al. [7] used a physical robot to perform a dry variation of the water maze experiment, in an effort to investigate a neural model of the brain. A rectangular environment was set up as the "water" area and a circle of reflective paper (sensed by an IR sensor) comprised the hidden platform. The motivation of their work was to study a complex model of the brain that simulated the nervous system and investigate the effects of the learning process. A color camera was used to input perceptual information; the robot turned left and right to obtain multiple camera views. Odometry was used

to provide heading information. A BEOWULF cluster of 12 Pentium IV computers was used for the brain. Results showed that the robot learned to navigate to the platform after about 8 trials. The best way to apply this change to the final version of the compliment is to gauge asunder the grunge movement. It is also important to fix the first value and allow the other to vary. Consider these modifications when examining the final product

In this paper, we modelled our environment on Krichmar's work, although our experiments are done in simulation to test the concept. Some further distinction should be made between Kritchmar's and our experiments. Firstly, unlike Kritchmar, we do not use any odometry information to guide the robot. This significantly handicaps the speed with which the robot learns and forces the robots to infer odometry from only the change in perceptual cues. In addition, our experiments use a random starting location to force the robot to comprehend, in some meaningful way, the entire water maze environment.

Section II describes the environment setting and the robot configuration. In Section III, we describe a technique used to discretize the perceptual space using a self-organizing feature map. Next, we present two novel methods of performing the water maze task. The first method uses a graph technique to represent traversal through the space. Given a goal location as represented by a node, a graph search is performed to find the route to the goal. The second method uses temporal difference learning to learn the association between a perceptual node and the desired robot command to direct the robot to the goal node. The two methods are evaluated and compared using a set of experiments, as presented in Section V. We also investigate different resolutions in the discretization of the perceptual space. Concluding remarks and future work are included in Section VI.

II. The Water Maze Setting

The environment in which the robot will navigate was designed to emulate the environment of water maze experiments done with rodents and particularly Krichmar's experimental environment. The Player/Stage robot simulator [8] was used to simulate the environment along with the robot's navigation. The basic layout of the environment can be seen in Figure 1. The environment consists of an 8 x 10 meter room in which a hidden circular platform with a radius of 0.8 meters is located. The robot is considered to be on the hidden platform when its center of mass is within 0.8 meters of the center of the platform. On the walls are 18 panels of various widths and colors. The robot must navigate to the hidden platform using only the perceptual information pertaining to the colored panels on the walls. This setup varies from Krichmar's in that during his experiments the robot also had explicit access to heading information. For our experiments, it is anticipated that the robot will infer its heading from the perceptual information. In addition, the robot can sense when it is directly on top of the goal platform but cannot see it from a distance.

The simulated robot has three cameras each with a 60º field of view. The cameras are positioned as shown in Figure 2. In addition to the cameras, the robot has a laser range finder for obstacle avoidance and an infrared sensor to detect the goal platform.

The robot has five potential actions available at any time step: hard left, left, forward, right, and hard right. These moves involve a rotation of -30º, -15º, 0º, 15º, and 30º respectively followed by a forward translation of 0.5 meters.

When the laser range finder determines that the wall is within 0.4 meters of the robot, an obstacle avoidance procedure is activated. This involves the robot stopping, turning in place away from the wall until its front directional axis is 30º beyond parallel with the wall, and then choosing an action based on its updated perceptual state.

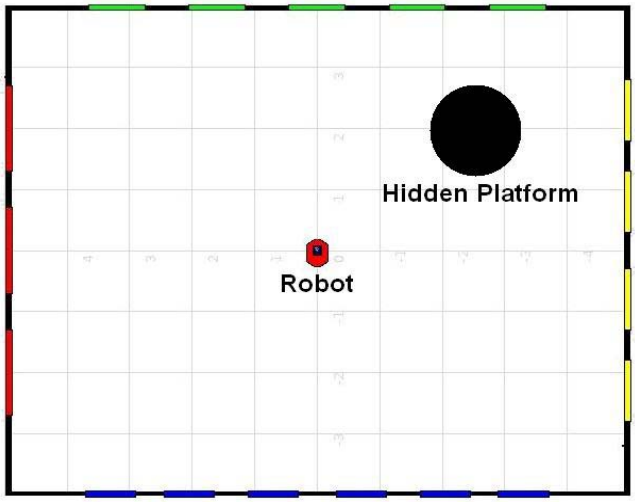

Fig. 1. The water maze environment inspired by Krichmar's work. The gridlines correspond to one meter. There are a total of 18 panels on the environment's walls that the robot uses for navigation: 5 green panels on the north wall, 4 yellow panels on the east wall, 6 blue panels on the south wall, and 3 red panels on the west wall.

III. Perceptual Discretization

To simplify the problem of navigating in the environment, the perceptual space was discretized. This allowed the robot to create a mapping between a state and the appropriate action at that state. A Self-Organizing Feature Map (SOFM) [9] was used to discretize the unmanageably large number of perceptual configurations into a small and manageable number of perceptual states.

Only the pixel heights of the colored panels in the simulated camera viewpoints were presented to the SOFM for discretization. Each camera has the potential to witness any of the 18 colored panels and consequently the vector for each camera consists of 18 bins. If the camera observes a panel, its pixel height is recorded for that bin; otherwise a value of zero is recorded. It is important to note that when viewing a panel of a particular color the robot does not know which of the

panels of that color it is viewing. Accordingly, the heights of panels viewed are placed into the first appropriate bin of that particular color. For example, a camera can potentially view three red panels. Therefore, there are three bins in the camera vector that can potentially hold the pixel heights of red panels. If only two red panels are in the image generated by a camera then the height of the leftmost viewed panel is recorded in bin one and the height of the rightmost viewed panel is recorded in bin two. Bin three is given a value of zero to represent that the camera did not observe a third red panel. Because there are 18 bins per camera and 3 cameras per robot, there are a total of 54 features per perceptual vector.

To create the SOFMs for these experiments, the perceptual vectors for 10,000 random locations and orientations within the environment were used as the training data. An example of an 8 x 8 SOFM generated with this data can be seen in Figure 3. To illustrate the resolution of perceptual discretization, one node of a 20 X 20 SOFM along with its corresponding area in the water maze is given in Figure 4.

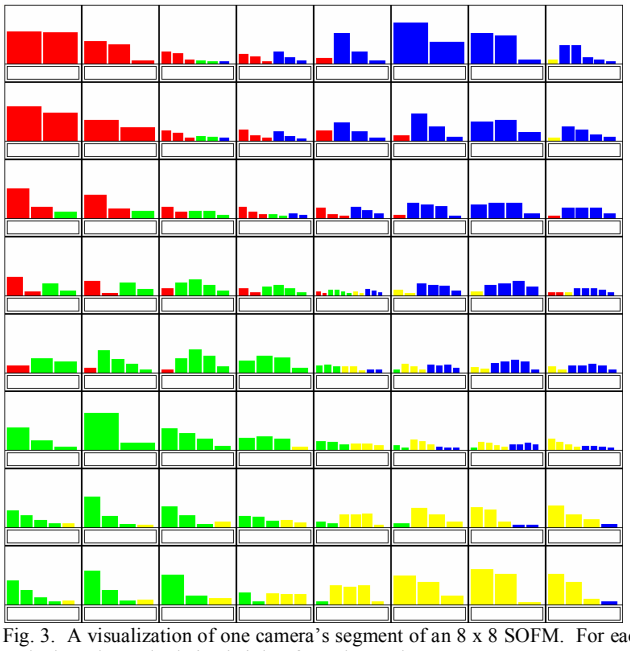

Fig. 3. A visualization of one camera's segment of an 8 x 8 SOFM. For each node the color and relative height of panels are shown.

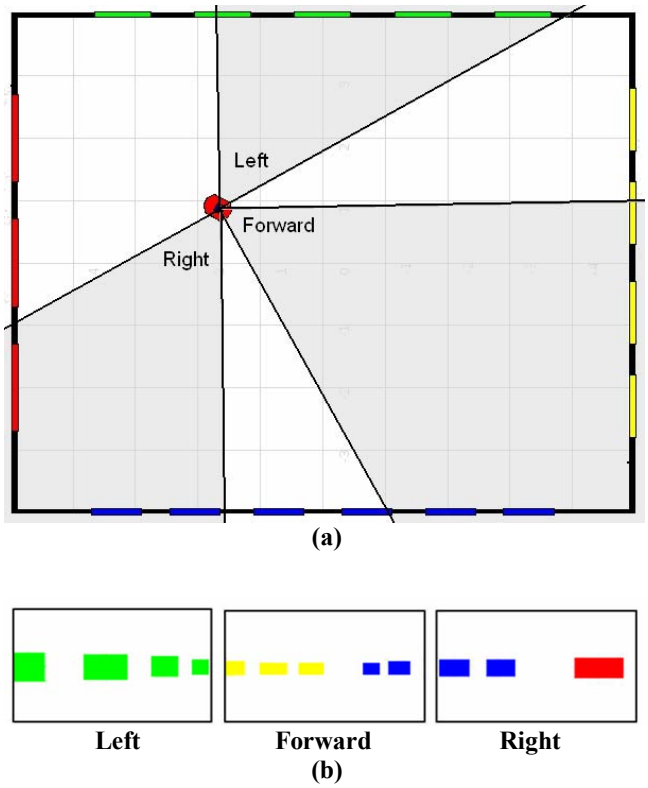

Fig. 2. (a) The configuration of the robot's cameras (b) The corresponding images obtained from the cameras.

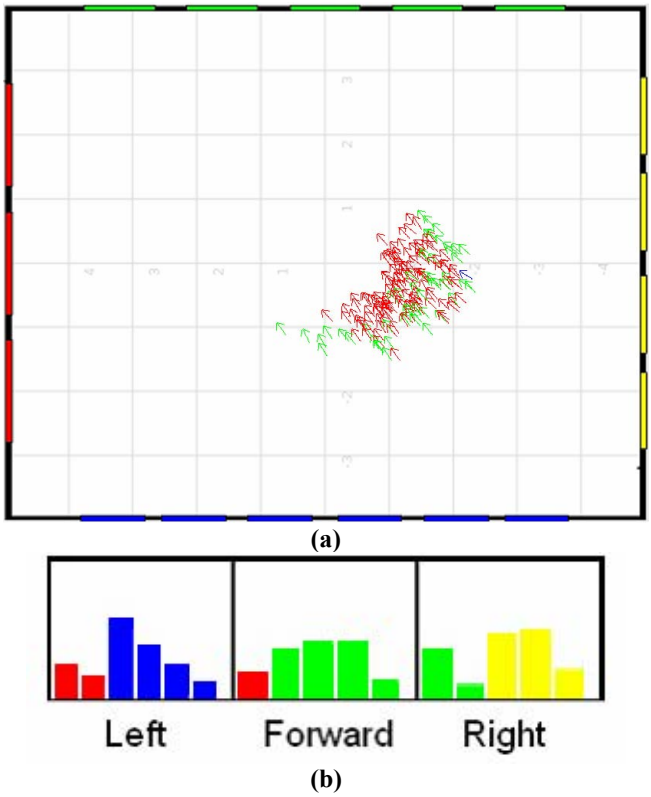

Fig. 4. An example of a node from a 20 x 20 SOFM. (a) The location and orientation of samples in the water maze environment that correspond to this SOFM node. The arrow direction shows the orientation for that sample and in this example generally points in the northwest direction. The color of the arrow denotes how closely the sample point perceptual vector matches the prototype vector, with red being the closest followed by green and then blue. (b) The prototypical relative heights and colors for each of the 3 cameras at this SOFM node.

IV. Navigational Approaches

Once the perceptual space is discretized, an appropriate action must be determined given the current state of the robot. The SOFM node that most closely resembles the robot's current perception represents the current state of the robot and is determined using Euclidean distance between feature vectors. The two approaches used for these experiments are explained in this section and are experimentally compared in Section V.

A. An Attributed Probabilistic Graph Search

This approach is trained offline using information from SOFM node transitions to create a directed graph. The nodes of this graph correspond to the nodes of the SOFM. Each edge of the graph has an attribute that corresponds to one of the five possible actions: hard left, left, forward, right, and hard right. The value of an edge from node A to node B with attribute C gives the probability of traversing from an area in the water maze which corresponds to SOFM node A to the area corresponding to node B by performing action C.

After collecting the SOFM node transition data, the graph is populated with edge values that represent the ratio between the recorded frequency of transitions from node A to node B via action C and the total number of transitions from node A. An example of a small segment of this type of probabilistic graph is given in Figure 5.

When recording the transitions for creating the graph, an extra "goal" node that does not directly correspond to any SOFM node is created. This "goal" node is considered to be the result of an action if the robot immediately moves to the hidden platform as a result of that action. In essence, the perceptual information of being on the platform supersedes any other perceptual information.

To decide what actions to take from the current position to reach the "goal" node the robot employs a graph search from the current node to the "goal" node. In particular, a modified version of Dijkstra's Algorithm [10] was implemented. Dijkstra's Algorithm assumes that when edges are concatenated to make a path, the resulting cost of that path is the summation of the costs for each of the edges. However, for a probabilistic graph, the resulting path cost should be the product of the costs of the edges. The only other alteration to Dijkstra's Algorithm is that the robot needs to find the highest probability of reaching the goal not the shortest cost path. To reconcile this difference the multiplicative inverse of the probabilty for an edge is used for the graph search. Once the path with the highest probability of reaching the "goal" node is determined, the robot performs the action that is attributed to the first edge of that path.

It may seem tempting to follow the attributes of the entire path; however, it is very unlikely that the robot will actually follow the exact path specified due to uncertainty in the system. To illustrate this point, consider that although self transitions occur in about half of the sample transitions, they are inconsequential when attemping to find the optimal path. Therefore, with each edge of the optimal path the probability of that path decreases by at least a factor of 2 and more often by a factor of 10. On a typical path of length 3, the probability will often be at least around 0.001. This implies that the optimal path found should not be used as a strict roadmap to the goal but rather as a gauge of the general direction to the goal from the robot's current position. For these reasons, the robot recalculates the optimal path after each move and then moves accordingly.

B. Temporal Difference Learning

The second approach utilizes the Working Memory Toolkit [11] developed at Vanderbilt University to learn a mapping between a SOFM node and an action taken by the robot. The toolkit maintains the action preferences of the robot for each node of the SOFM. The initial behavior of the robot is determined by the random initialization of the toolkit's preferences.

The other crucial feature of the Working Memory Toolkit is its usage of Temporal Difference (TD) learning [12]. By rewarding and punishing the robot during training, the system

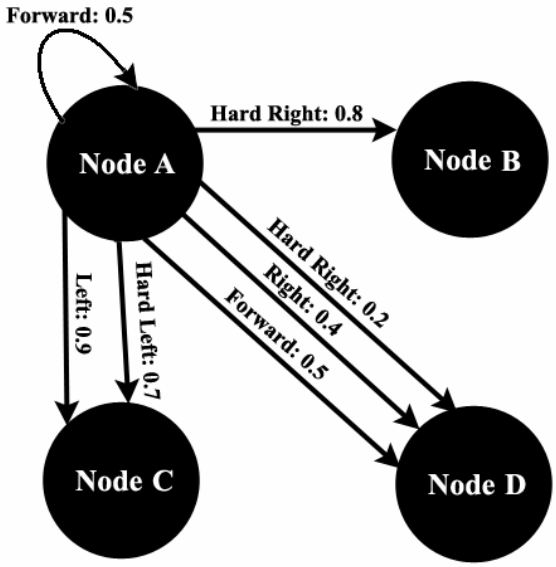

Fig. 5. A section of a probabilistic graph. This graph conveys that if the robot is at node A and performs the hard right action there is a 0.8 probability of immediately moving to node B and a 0.2 probability of immediately moving to node D.

can mold its behavior. A training episode lasts until the robot either finds the goal or 50 moves have been made. For these experiments, the robot was punished if it did not find the platform by the end of the episode or if obstacle avoidance was engaged. The robot was rewarded when it located the platform. After each episode, the action preferences for each SOFM node experienced are updated, and the robot uses these new preferences for the next episode.

An additional important aspect of this system is the concept of a delayed reward. The robot knows very little of the status of its current episode until the episode is finished. Because the goal area is only 2.5% of the total environment area, the robot will often require a long time to locate the goal during the initial learning phase when its actions are

determined by the initial randomness. This type of "needle-in-a-haystack" problem significantly slows the learning process.

V. Experiments

A. Comparing the Two Navigational Approaches

This experiment varies the number of samples used to initialize both the graph search and the temporal difference methods and compares the average number of actions needed to reach the goal from a uniformly distributed random starting location. The maximum average number of moves needed per episode is 50 since both the graph search and the temporal difference method will be stopped after 50 actions. Also, for this experiment a SOFM size of 20 x 20 was used. A SOFM of this size allowed for a fairly fine discretization of the perceptual space. Further analysis of SOFM configurations can be found in the next experiment.

For the graph search, the number of node transition samples was limited at several points from 0 to 100,000. After training the probabilistic graph with this data, the robot attempted to find the goal for 1000 episodes. The average number of moves taken per episode for each of the sample sizes is graphed in Figure 6.

For the temporal difference method, the number of moves taken during the training phase was limited to predefined points between 0 and 100,000. After the system learns for the set number of moves the learning is turned off and the performance is tested for 1000 episodes. The results of this process are also summarized in Figure 6.

As would be expected with no training, both the graph search approach and the TD approach yield similar results with an average episode length of 36.2 and 36.4 respectively. With relatively little experience, of about 1000 samples, the graph search does considerably better than the TD approach. This can be attributed to the ability of the graph search algorithm to be able to generate a path to the goal with very little data. This path will be far from optimal; however, the first action on that path most often points in an approximately correct direction. Because the robot only uses the first action of the path, this limited information may be sufficient. The TD approach does not have the capability to find a path through the SOFM nodes and therefore requires a more complete picture of the SOFM node to action mapping to efficiently reach the goal. As the number of samples approaches 10,000, the TD approach actually outperforms the graph search. This relationship will be investigated in future experiments. Given the different utilities of the two examined approaches, it is reasonable to assume that some combination of the two may yield improved results, as proposed by Redish and Touretzky [4] and Balakrishnan et al. [5].

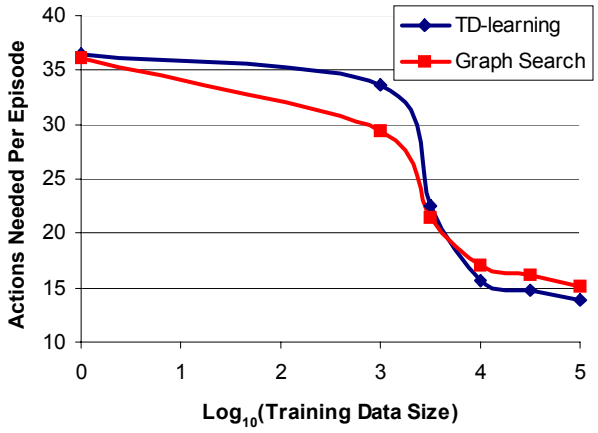

Fig. 6. A comparison of the two navigational approaches.

B. An Analysis of SOFM Size Efficiency

This experiment was designed to determine the optimal number of nodes for the SOFM. Six sizes of SOFMs were tested: 4 x 4, 8 x 8, 12 x 12, 14 x 14, 16 x 16, and 20 x 20. A database of 300,000 perceptual vectors from the water maze was used to determine the variation in the x and y water maze coordinates for the perceptual vectors that mapped to a specific node. For each SOFM size, the average variation over all the nodes was computed. Any node that had no perceptual vectors map to it was discounted from the computations. In addition, the area encompassed by each SOFM node was computed by calculating the convex hull of the vectors that map to that node. With that information, the average area of a node given a SOFM size was computed.

The results for these tests, given in Table I, are understandable considering that as the SOFM size increases the nodes should increase their specificity as relating to area within the water maze. Further study is needed to determine if this extra specificity actually aids the robot's navigation. Figure 7 illustrates the area covered by a typical 8 x 8 SOFM node. When compared with the coverage of the 20 x 20 SOFM node from Figure 4a, the 8 x 8 node clearly has a much larger range of positions and orientations.

TABLE I
The Effect of SOFM Size on Node Coverage

SOFM Size	σ_x	σ_y	Average Area (m^2)
4 x 4	1.84	0.79	24.1
8 x 8	1.26	0.62	13.3
12 x 12	0.96	0.50	8.41
14 x 14	0.99	0.55	7.82
16 x 16	0.85	0.49	6.18
20 x 20	0.81	0.36	4.36

σ_x and σ_y are the average standard deviations in meters of the x and y coordinates for nodes of this sized SOFM.

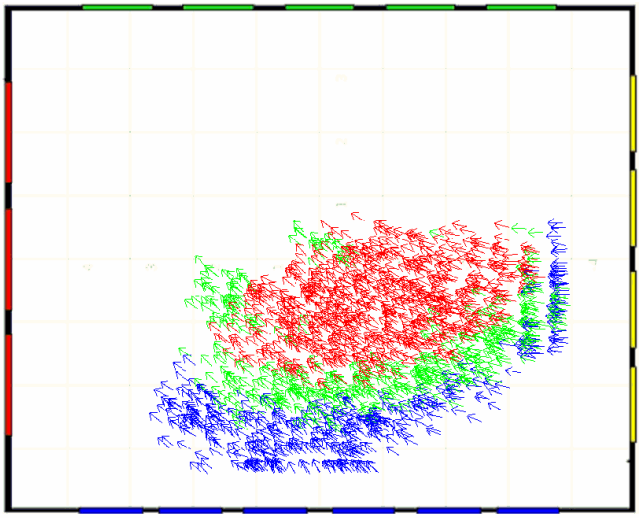

Fig. 7. An example of the positions in the environment that map to one node of an 8 x 8 SOFM. The arrow direction shows the orientation for that sample and in this example generally points somewhere between west and north. The color of the arrow denotes how close the sample point perceptual vector is to the prototype vector, with red being the closest followed by green and then blue. Notice the difference in area of coverage, due to SOFM size, from Figure 4a.

VI. Concluding Remarks

This paper presented several methodologies that can be used to simulate the proposed mechanism by which rodents learn a mapping between perception and location. These methodologies included using SOFMs to discretize the perceptual space, using a probabilistic graph-searching algorithm for path planning, and using temporal difference learning to learn a mapping between a SOFM node and an action. The main rationale for using temporal difference learning is its ability to adapt to new situations such as if the platform was moved during trials.

The two navigational approaches of probabilistic graphs and TD learning were compared and it was found that each is suited to a slightly different situation. It was found that the probabilistic graph algorithm outperformed the temporal difference approach with relatively little training; however, with additional training the temporal difference approach tended to be the more successful algorithm. This relationship lends itself to further experimentation. Future work will also include more exhaustive analysis into the effect that SOFM size has on the convergence times of the navigational training process and the number of samples needed to generate an effective SOFM. These experiments will aid in the ultimate goal of implementation on a physical robot, which will allow further testing of the learning efficiency of the working memory approach.

ACKNOWLEDGMENTS

This work has been funded by the U.S. National Science Foundation under grant EIA-0325641.

References

[1] R. Morris, p. Garrud, J. Rawlins, and J. O'Keefe, "Place navigation impaired in rats with hippocampal lesions," *Nature*, vol. 297, 1982, pp. 681-683.

[2] C.V. Vorhees, T.M. Reed, M.R. Skelton, and M.T. Williams, "Exposure to 3,4-methylenedioxymethamphetamine (MDMA) on postnatal days 11-20 induces reference but not working memory deficits in the Morris water maze in rats: implications of prior learning," *International Journal of Developmental Neuroscience,* 2004, vol. 22, no. 5/6, pp. 247-259.

[3] M.A. Brown, and P.E. Sharp, "Simulation of spatial learning in the morris water maze by a neural network model of the hippocampal formation and nucleus accumbens," *Hippocampus*, 1995, vol. 5, pp. 171-188.

[4] A.D. Redish and D.S. Touretzky, "The role of the hippocampus in solving the Morris water maze," *Neural Computation*, 1998, vol. 10, no. 1, pp. 73-111.

[5] K. Balakrishnan, R. Bhatt, and V. Honavar, "A Computational Model of Rodent Spatial Learning and Some Behavioral Experiments," In *Proceedings of the Twentieth Annual Meeting of the Cognitive Science Society*, 1998, Madison, WI.

[6] D.J. Foster, R.G.M. Morris, and Peter Dayan, "A Model of Hippocampally Dependent Navigation, Using the Temporal Difference Learning Rule," *Hippocampus*, 2000, vol. 10, pp. 1-16.

[7] J.L. Krichmar, D. A. Nitz, J.A. Gally, and G. M. Edelman, "Characterizing functional hippocampal pathways in a brain-based device as it solves a spatial memory task," In *Proc National Academy of Science USA*, 2005, vol. 102, pp. 2111-2116.

[8] Gerkey, B.P., Vaughan, R.T. and Howard, A. 2003. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In Proc. IEEE Intl. Conf. Advanced Robotics, Coimbra, Portugal.

[9] Kohonen, T. 1990. The self-organizing map. Proc. IEEE 78, 1464-1480.

[10] E. W. Dijkstra. 1959. A note on two problems in connexion with graphs. In Numerische Mathematik. 269–271.

[11] J.L. Phillips and D.C. Noelle, "A Biologically Inspired Working Memory Framework for Robots," in *Proc. of the 27th Annual Meeting of the Cognitive Science Society*, Stresa, Italy, July 2005.

[12] Sutton, R.S. 1988. Learning to predict by the methods of temporal differences. Machine Learning, vol. 3, pp. 9-44.