Spatial Learning for Navigation in Dynamic Environments[1]

Brian Yamauchi[2,3,4] and Randall Beer[5]

Abstract

This article describes techniques that have been developed for spatial learning in dynamic environments and a mobile robot system, ELDEN, that integrates these techniques for exploration and navigation in dynamic environments. In this research, we introduce the concept of adaptive place networks, incrementally-constructed spatial representations that incorporate variable-confidence links to model uncertainty about topological adjacency. These networks guide the robot's navigation while constantly adapting to any topological changes that are encountered. ELDEN integrates these networks with a reactive controller that is robust to transient changes in the environment and a relocalization system that uses evidence grids to recalibrate dead reckoning.

Footnotes

[1] Manuscript received . . .

[2] Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH, 44106, USA.

[3] Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC, 20375-5337, USA.

[4] Current address: Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA, 94306, USA. (email: yamauchi@robotics.stanford.edu, URL: http://robotics.stanford.edu/people/yamauchi/bio.html).

[5] Department of Biology and Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH, 44106, USA (email: beer@alpha.ces.cwru.edu, URL: http://yuggoth.ces.cwru.edu/beer/beer.html).

Figure Captions

Figure 1: ELDEN system architecture

Figure 2: Behavior arbitration

Figure 3: Nomad 200 mobile robot

Figure 4: ELDEN exploration trial (first 10 minutes)

Figure 5: ELDEN exploration trial (second 10 minutes)

Figure 6: Adaptive place network learned during exploration

Figure 7: ELDEN navigation from place 0 to place 13

Figure 8: APN initial state for navigation from place 0 to place 13

Figure 9: APN initial state for navigation from place 13 to place 0

Figure 10: ELDEN navigation from place 13 to place 0

I. INTRODUCTION

Mobile robots have potential applications in a wide variety of domains: from delivering packages in office buildings to driving vehicles on highways, and from performing reconnaissance on battlefields to acting as aides for the handicapped. If mobile robots are to realize their potential, they require certain abilities which they currently lack. In particular, they need the ability to operate robustly in changing environments. Some changes may take the form of moving people and vehicles; other changes may involve rearranged furniture or road construction; and still others may involve landslides or thunderstorms. Mobile robots need to be able to deal with these changes in order to function effectively in the real world.

Mobile robots also need the ability to explore and learn about their environments. In some cases, mobile robots may be provided with a map of their world, but this is not an altogether satisfactory solution for two reasons. First, the effort involved in creating a map that is sufficiently detailed to allow for robust and accurate navigation may be substantial, if not prohibitive. A second and more fundamental problem is that any changes present in the environment will introduce inaccuracies into the map, and these inaccuracies will accumulate over time to the point where serious errors may result. A package delivery robot that attempts to navigate through a rearranged office is likely to collide with a mahogany desk or a steel filing cabinet, and an autonomous vehicle that attempts to cross a washed-out bridge is likely to find itself in even deeper trouble. On the other hand, a robot that can explore and learn about the world will be able to alter its map to reflect the changes that it observes, and then use this modified map to navigate robustly through the world.

There are many different types of change that can occur within a dynamic environment: people may be standing or walking in front of the robot, doors may open or close, furniture may be rearranged, sensors may fall out of calibration. There are also many different techniques that can be proposed to deal with these changes, and some of these techniques may handle certain types of change flawlessly, yet be stymied by other types.

In order to provide a framework for discussing these different forms of change, we propose the following taxonomy for classifying environmental changes: First, changes may be *transient* or *lasting*. Transient changes are sufficiently brief that they can be handled reactively, without modification to the robot's internal state (for example: people walking past the robot). Lasting changes persist over longer periods of time and can only be handled through changes in the robot's internal state. Lasting changes may be *topological* or *perceptual.* Topological changes affect the paths that can be taken through the environment (for example: doors that open and close). Perceptual changes affect the robot's perception of the environment (for example: slippage in wheel encoders). A single change may have both topological and perceptual components. For example: rearranging the furniture in a room may affect the paths that the robot can take through the room, as well as the appearance of the room.

This article describes techniques we have developed for robust exploration and navigation in dynamic environments containing transient, topological, and perceptual changes. The specific contributions of this work are the following:

- Adaptive place networks: spatial representations that are constructed incrementally and constantly adapt to changes encountered by the robot, enabling robust navigation in the presence of topological changes.

- A relocalization technique that uses evidence grids to handle one type of perceptual change, the dead reckoning errors that occur due to wheel slippage.
- ELDEN (Exploration and Learning in Dynamic ENvironments): an integrated system that combines adaptive place networks and relocalization with a reactive controller capable of robust operation in the presence of transient changes.

ELDEN has been tested on a real mobile robot in a real-world environment containing transient changes (moving people), topological changes (rearranged obstacles), and perceptual changes (dead reckoning drift). In these experiments, ELDEN was shown to be capable of exploring and navigating robustly in the presence of all of these changes.

This paper also describes the path taken from early development in simulation to actual testing on a real robot. The availability of a simulator allowed for development of the initial system (ELAN: Exploration and Learning using an Adaptive Network) prior to access to the real robot and facilitated rapid transfer of the system architecture when the real robot became available. However, the differences between the simulated and the real-world environments also necessitated major changes in the method used for place recognition as well as minor changes in a variety of other subsystems. We discuss both the advantages and disadvantages of using simulation as a development environment, but our overall findings were clear: testing on real robots is absolutely necessary for establishing that any techniques or systems will work in the real world. ELDEN performs well in the real world, but ELDEN's predecessor, ELAN, did not, and it was only through experiments using real robots that we were able to develop the techniques that enable ELDEN to explore, learn, and navigate in dynamic real-world environments.

II. System Architecture

ELDEN (Exploration and Learning in Dynamic ENvironments) is a spatial learning system designed to allow mobile robots to explore and navigate within dynamic environments. ELDEN has three major components (indicated by the boxes in Figure 1): a low-level control system composed of reactive behaviors, an adaptive place network (APN) that learns the topology and geometry of the environment, and a relocalization subsystem that uses evidence grids to recalibrate the robot's dead reckoning.

Each of these subsystems is designed to handle a different type of environmental change. The reactive controller deals with transient changes, such as people walking in front of the robot, and guarantees that the robot will be able to operate even when its spatial representation does not accurately reflect the current state of the environment. The adaptive place network deals with topological changes and constantly adapts to reflect any new places in the environment that are encountered. The relocalization subsystem deals with one specific form of perceptual change, the slippage that accumulates in the wheel encoders and leads to errors in the robot's dead reckoning estimate of its position and orientation.

ELDEN operates in two different modes: exploration mode and navigation mode. During exploration, ELDEN's goal is to learn the spatial structure of the environment. During navigation, ELDEN's primary goal is to move to a particular location. However, ELDEN continues to learn during navigation, so that if the world has changed, ELDEN will be able to detect those changes and adapt its spatial knowledge to match the new world. Unlike many robot learning systems, there is no assumption that all spatial learning must be completed before any navigation occurs.

For example: when a robot is first introduced to a new environment, it may spend a certain amount of time exploring to determine the general structure of the environment, but without necessarily finding all paths through that environment. After initial exploration, the robot may be ordered to navigate to a particular sequence of destinations to perform specified tasks. During navigation, the robot will update its spatial representation to reflect any changes that it encounters. When the robot is not being ordered to a particular destination, it may spend its "free time" conducting further exploration. This exploration may lead the robot to find new, more efficient routes through the environment. More importantly, this process of continual learning allows the robot to maintain a dynamic representation of a dynamic world, constantly adapting its spatial knowledge to reflect the changing environment.

III. Reactive Controller

Behavior-based control is based upon the concept of using simple sensorimotor processes, operating in parallel, to enable robots to react quickly and robustly in unpredictable environments. Within this general paradigm, a wide variety of different architectures have been proposed [1] [5] [7] [8] [16] [17]. Like these other behavior-based approaches, ELDEN uses a set of reactive behaviors to control the low-level activity of the robot, but ELDEN differs in its methods for behavior activation and behavior arbitration.

ELDEN's behaviors are functions that map sensory inputs to output functions over the space of possible actions. Each output function is a sum of scaled Gaussians with variable mean, width, and height. So, instead of simply specifying a single desired action, each behavior can specify a number of desirable actions along with the priority (height) of that action and the tolerance (width) for deviations from those desired actions. Behavior arbitration consists of

summing all the output functions and selecting the action corresponding to the maximum in the quantized output space.

Figure 2 shows a simplified example of behavior arbitration, using piecewise linear functions instead of Gaussians. Each of the behaviors specifies an array of values corresponding to the value of their output function at each quantized point in the output space (represented by the thin lines in the figure). In this example, the output space consists of the space of possible turns. The top behavior votes for a turn of 0 (i.e. moving straight ahead). The shape of the output function indicates that the behavior places a moderate priority on this action and has a wide tolerance for variation near this action. All of the array values are summed, and the output point with the greatest sum becomes the selected action. In addition, a small amount of random noise is added to the output space, allowing ELDEN to escape from situations where the outputs of the behaviors are nearly balanced and might otherwise result in oscillations.

IV. Adaptive Place Network

The adaptive place network (APN) provides a spatial representation and learning system for ELDEN that contains both metric and topological information about the structure of the environment. The APN is composed of place units, each of which corresponds to a region of Cartesian space, and place links, which represent the relationship between adjacent place units. The APN is always learning, during both exploration and navigation.

Formally, an adaptive place network $N = <P, L>$ consists of a set $P = \{p_1, p_2, \ldots p_n\}$ of place units and a set $L = \{l_1, l_2, \ldots l_m\}$ of place links. Each place unit is a 3-tuple $<i, x, y>$ containing the place index i and the Cartesian coordinates (x, y) of the center of the corresponding place region. Each place link is a 4-tuple $<i, j, \theta, c>$ where i and j are the indexes of the place

units connected by the link, θ is the approximate heading required to travel from place *i* to place *j*, and *c* is the confidence level that is associated with this link.

Initially, the APN is empty. At each timestep, the distance between the robot's current location (as determined using dead reckoning) and the nearest place unit is computed. If this distance is below a threshold, then the robot's current place index is set to the index of the nearest place unit. Otherwise, a new unit is created, with the Cartesian coordinates of the robot's current location, and the robot's current place index is set to the index of the new unit. In either case, the unit corresponding to the robot's current place index is referred to as the winning unit.

Note: In this discussion, one timestep corresponds to one update cycle for both the reactive controller and the adaptive place network. In all of the experiments described in this article, one timestep is approximately 0.1 seconds.

A place link is created the first time that ELDEN moves from one place to another (i.e. whenever a new unit is created, or whenever the winning unit at the current timestep is different from the winner at the previous timestep). Each link connects two places and stores the direction that the robot must travel to move from one place to the other, along with a confidence value indicating the robot's certainty that this link can be traversed. When the link is created, this direction is set to the robot's current orientation. Upon subsequent transits across this link, the direction is updated using the following rule:

$$\theta_{new} = \tan^{-1}\left(\frac{\lambda_{link}\sin\phi + (1-\lambda_{link})\sin\theta_{old}}{\lambda_{link}\cos\phi + (1-\lambda_{link})\cos\theta_{old}}\right)$$

where θ_{new} is the new link direction, λ_{link} is the link learning rate (0.5 in these experiments), ϕ is the robot's current orientation, and θ_{old} is the old link direction.

Links are treated as bi-directional, so if a link is initially traversed from place A to place B, and then it is later traversed from place B to place A in direction ϕ, the stored direction is updated using the opposite orientation ($\phi + 180°$).

During navigation, the location associated with a particular place unit is specified as the destination. A cost is associated with each link equal to the reciprocal of the link confidence value. Then, Dijkstra's algorithm [3] is used to find the shortest path from each place to the destination, and an orientation is associated with each link indicating the direction of travel towards the destination. After applying this algorithm, a single outgoing link is on the shortest path to the destination from each place unit. Behaviors within the reactive controller then orient the robot toward the direction of this link and move the robot forward (while also avoiding any nearby obstacles).

A link from place A to place B is considered traversed successfully, during either exploration or navigation, if at time *t*, the APN indicates that the robot is at place A, and at time *t+1*, the APN indicates that the robot is at place B. Whenever a link is traversed successfully, the new link confidence is increased to $\lambda_{link} + (1 - \lambda_{link}) c_{old}$ where λ_{link} is the link learning rate (0.5 in these experiments) and c_{old} is the old confidence. This allows the link confidence level to increase gradually (within the range 0 to 1) to reflect ELDEN's experiences in the world.

A link from place A to place B is considered unsuccessfully traversed during navigation, if at time *t*, the APN indicates that the robot is at place A, and at time *t+1*, the APN indicates that the robot is at place C (where C is equal to neither A nor B), or if the place network indicates that the robot is at an unknown location, or if the robot attempts to move from A to B for a certain period of time, and yet remains at A. These correspond to three different real-world situations that ELDEN may encounter. Instead of navigating to the correct location (B), sensor noise or

motor error may cause it to end up at a nearby location (C) instead. Or, steering around an unexpected obstacle, ELDEN may find itself in a completely new location. Or, the pathway may be blocked, and ELDEN may be unable to make progress. In any of these cases, the new link confidence is decreased to $(1-\lambda_{link})c_{old}$ where λ_{link} is the link learning rate and c_{old} is the old confidence.

Whenever the network changes, either through a change in link confidence levels or the addition of a new link or unit, the shortest paths toward the destination are redetermined. This allows ELDEN to continually adapt to new information that is acquired as it navigates. In particular, it allows ELDEN to deal with topological changes in the environment that may require substantially different paths toward the destination or the exploration of new territory.

V. Relocalization System

ELDEN's spatial learning system relies on reasonably accurate dead reckoning. However, over time, slippage between the robot's wheels and the floor results in errors, both in the position and orientation of the robot. The orientation errors tend to be the most serious because small errors in orientation can lead to large errors in translation at locations far from the origin of the coordinate frame.

The reactive controller can handle small errors (up to one meter) with no perceptible effect on performance. For example: if obstacles have been rearranged to move an opening one meter to the left, the navigation behaviors may vote for the robot to travel in the old direction of the opening while the obstacle avoidance behavior may vote for turning in the direction of the new opening. The arbitrator will then find a compromise that will move the robot through the new opening in a direction as close as possible to that of the old opening.

Larger errors do not affect the basic competency allowing the robot to move around while avoiding collisions with static and moving obstacles, but large errors may degrade navigation performance. In order to insure that the robot's estimated position remains close to the robot's actual position, we use a method based on matching evidence grids.

Evidence grids are a method for spatial representation developed by Moravec and Elfes [21]. A local region of space is subdivided into cells in a Cartesian grid, and each cell contains a value representing the estimated probability of occupancy. ELDEN uses a two-dimensional evidence grid with square cells that are 12 centimeters wide. Each sensor reading can be used to update cell occupancy probabilities using a Bayesian update rule based on a particular sensor model. For example: if a sonar with a 22.5 degree arc detects an obstacle at a range of 5 meters, then all cells on this arc at a distance of 5 meters should have their occupancy probability increased, while all cells within this arc at a distance of less than 5 meters should have their occupancy probability reduced. Since the laser rangefinder produces generally accurate and precise data, a simple point model is used for laser range data, increasing the occupancy probability for only the cell corresponding to the point returned by the laser.

Evidence grids have been previously used for mapping and path planning [9] [20], but in our system we introduce a new technique for using evidence grids to recalibrate dead reckoning. Evidence grids are constructed both before and after the error accumulates in the robot's dead reckoning and hill-climbing is used to find the transformation providing the best match between the two grids. This transformation is then applied to the robot's dead reckoning position estimate to determine the robot's actual location.

The first place unit in the APN is designated the robot's home location. Before starting exploration, ELDEN builds an evidence grid that represents everything that can be seen from the home location using the robot's sonar and laser rangefinders. This grid is constructed by taking 45 readings at one degree rotation increments from all 16 of the sonar sensors, then adding 36 sets of readings from the planar laser rangefinder (each containing up to 120 range values), rotated at 10 degree increments. This entire process requires approximately two minutes.

After exploring for a period of time, ELDEN uses its APN with its reactive controller to navigate back to the place region associated with its home location, and then uses dead reckoning to move to the coordinates associated with the center of this region. ELDEN then builds a second evidence grid, and the two grids are then brought into registration using a hill-climbing algorithm to search the space of possible translations and rotations. The hill-climbing algorithm starts with the null translation and rotation between these two grids, then takes steps in the space of possible translations and rotations in order to maximize the match between the two grids.

The match quality is given by:

$$M = \sum_{\forall x,y} \begin{cases} 1 & \text{if } \dfrac{p_{x,y} - p_0}{p'_{x,y} - p_0} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where M is the match quality, p_0 is the base cell occupancy probability, $p_{x,y}$ is the occupancy probability of cell (x, y) in the original evidence grid, and $p'_{x,y}$ is the occupancy probability of the cell (x, y) in the second evidence grid.

All cells in both grids are initialized to p_0 (0.5 in these experiments) before the sensor readings are used to update the occupancy probabilities. The match algorithm steps through both grids, comparing the cells corresponding to the same Cartesian location. If two corresponding cells are both more likely to be occupied than the base probability, then the match quality is

increased. If two corresponding cells are both less likely to be occupied than the base probability, then the match quality is increased. Otherwise, the cells have no effect on the overall match score.

The hill-climbing stepsize is initially set to 30 cm in translation, 1.0 degrees in rotation and is halved when a local maximum is reached, in order to more precisely locate this maximum. When a local maximum is reached with a stepsize of 3.75 cm and 0.125 degrees, the search is stopped and the resulting maximum is output as the transformation between the two evidence grids. This transformation is then used to recalibrate the robot's dead reckoning. Following this procedure, ELDEN can continue to explore.

VI. Robot Implementation

ELDEN was implemented using a Nomad 200 mobile robot (Figure 3) at the Navy Center for Applied Research in Artificial Intelligence (NCARAI) at the Naval Research Laboratory. The Nomad 200 is a wheeled robot with a zero-turning radius. The robot has 16 sonar range sensors and 16 infrared range sensors spaced evenly around the base. The infrared sensors are used to detect objects less than 40 centimeters away, and the sonar sensors are used to detect objects at greater distances. The robot also has a planar laser rangefinder which can collect high resolution range data in a ten degree arc. All of these sensors are mounted on a turret which can rotate independently of the base. In addition, both the steering and drive motors have encoders which provide the Cartesian location of the robot with a resolution of 0.254 cm (0.1 inches), though actual accuracy depends on the slippage between the robot's wheels and the floor.

VII. Exploration Experiments

In order to test ELDEN's performance in real-world environments, experiments were performed in the NCARAI robot laboratory, an indoor area, 14 meters by 8 meters, containing moving people, boxes, chairs, desks, bookshelves, carts, and other obstacles. In the exploration trials, ELDEN explored for ten minutes, navigated back to its home location, relocalized, and then explored for another ten minutes.

Figure 4 and Figure 5 show the actual path taken by the real robot during a typical exploration trial. These paths differ because the initial heading of the robot is set randomly. The robot is represented by the circle, with the line in the circle indicating the robot's orientation at the end of the run. The line trace represents the robot's estimated dead reckoning position recorded during the trial. Fixed obstacles are represented by the black rectangles reflecting the actual position of the obstacles in the room. In addition, this environment also contained people moving through the room past the robot.

During this exploration, ELDEN constructed an adaptive place network to represent the places visited and their topological connections. Figure 6 shows the APN learned during this trial. Place units are represented by vertices in this graphs. Place links are represented by lines between adjacent place units. Note that ELDEN was able to determine when it had revisited a previously explored location, as multiple passes over the same part of the environment resulted in only a single place unit for each region of space, rather than multiple overlapping units created for each visit.

VIII. Navigation Experiments

To test ELDEN's ability to navigate using this learned APN (Figure 6), ELDEN was directed to travel from one location to another in the presence of transient changes (moving people) and topological changes (rearranged obstacles). In these tests of over 30 navigation runs, ELDEN was highly robust to both types of change.

A typical navigation trial is shown in Figure 7, where ELDEN was directed to navigate from its home location to place 13 (in the lower right corner of the figures), but the environment was rearranged, with a large box blocking the minimum-cost path to the destination. The initial state of the APN is shown in Figure 8, with the robot at its home location (place 0) and the arrows on the links indicating the minimum-cost path to place 13. This path starts at place 0 and passes through places 33, 32, 18, 17, 16, 15, 14, 11, and 12 before reaching place 13. (This path was chosen over the path passing through place 1 because these links had been traversed more often and, as a result, had higher confidence values.) The large box was placed blocking the path between place 16 and place 15.

Figure 7 shows the path actually taken by ELDEN. Initially, ELDEN attempts to take the minimum-cost path, but the obstacle avoidance behavior prevents the robot from colliding with the large box. Instead, ELDEN detours away from the path, turning to its right (left in the figure) while continuing to advance. On this detour, ELDEN visits two new places, and adds two corresponding place units to the APN (places 52 and 53, see Figure 9), and ELDEN also decreases the confidence associated with the link from place 16 to place 15. After arriving at place 33, ELDEN circles around to attempt to follow the initial path again. Since the large box

completely blocks the path from place 16 to place 15, this second attempt is also unsuccessful, and ELDEN again reduces the confidence associated with the link connecting these two places.

At this point, the cost associated with this link has increased to the point where an alternate route from place 16 to place 13 has a lower cost than the route passing through the link from place 16 to place 15. This alternate route starts at place 16 and passes through places 17, 18, 19, 32, 33, 0, 1, 2, 3, 4, 8, 9, 10, 11, and 12 before reaching place 13. This route is chosen over the route that passes through the link from place 16 to place 52 because this new link has only been traversed once and has only a moderate confidence while the links on the alternate route have been traversed more than once and, as a result, have higher confidence values. In the process of attempting to follow this alternate route, ELDEN discovers a shortcut from place 18 to place 33 through a previously unvisited place (place 54) which is added to the APN. Taking this detour, ELDEN is able to successfully navigate to the destination, as shown in Figure 7.

After arriving at the destination, ELDEN is told to navigate back to its home location. Since the APN has learned that the link between places 15 and 16 is not reliable, ELDEN does not attempt to travel back through this route. Instead, ELDEN follows a path that starts at place 13 and passes through places 12, 11, 10, 9, 8, 4, 3, 2, and 1 before reaching place 0. The links on this path are shown in the APN in Figure 9, and the actual path taken by ELDEN is shown in Figure 10. Following this path, ELDEN was able to navigate successfully back to its home location.

IX. Relocalization Experiments

In order to evaluate ELDEN's relocalization capability, we first conducted experiments to measure the quantitative effects of encoder slippage in this environment. In each of these trials,

ELDEN explored for ten minutes, returned to its home location (place 0) using its APN, and positioned itself at the stored Cartesian location of the place center using dead reckoning. This location also corresponds to the origin of the Cartesian coordinate frame used for positioning.

The error between this position and the actual place center was calculated by manually aligning the robot with markings on the floor indicating the position and orientation of the place center, and then reading the encoders to determine the offset. In these trials, encoder slippage typically resulted in a translation error of 15-25 centimeters and an orientation error of 5-20 degrees. The translation error (at the origin) is relatively insignificant, but the orientation error is substantial. An orientation error of 20 degrees at the origin results in a translation error of over 3 meters at a position 10 meters from the origin.

This relocalization subsystem was then used in the exploration trials as described above. During these trials, the relocalization subsystem was able to correct this error to within 3-6 centimeters and 2-3 degrees. Relocalization was also tested following the navigation trial described in Figure 7. In this experiment, the grid constructed during relocalization was substantially different from the initial grid, due to the fact that one of the boxes (previously in the lower left section of the figure) had been moved to block the path between place 16 and place 15 (in the lower right section of the figure). Despite this environmental change, the relocalization subsystem was still able to correct the dead reckoning error to within 8-10 centimeters and 2-3 degrees.

X. From Simulation to Reality

Prior to developing ELDEN, we developed another system for exploration, spatial learning, and navigation in dynamic environments. This system, ELAN (Exploration and Learning

using an Adaptive Network), was developed using the Nomadic Robot Simulator, a simulator that was designed to model the Nomad 200 as closely as possible for the purpose of robot software development. This simulator includes models for sensor noise, specular reflections from sonar rangefinders, and dead reckoning slippage.

We chose to develop in simulation for a simple reason: the period of time during which we had access to the real mobile robots was limited (two months), and it would have been impractical to design, implement, and test the entire system over this period of time. The use of the simulator allowed us to design the overall system architecture and implement most of the underlying software prior to this period.

ELAN is similar to ELDEN in terms of its overall system architecture, use of reactive behaviors for dealing with transient changes, method of behavior arbitration, use of a topological/metric map for spatial representation, and use of variable-confidence place links to deal with topological changes. The primary difference between these two systems is in their methods of localization. Where ELDEN uses dead reckoning in combination with a relocalization subsystem, ELAN's learning scheme was designed to allow the robot to constantly update its position estimate based on both dead reckoning and a stored range image for each place. This would have allowed ELAN to use its range sensing to automatically correct for small errors in dead reckoning without explicitly relocalizing.

In ELAN's adaptive place networks, place units are initially added to the network in a manner similar to that used in ELDEN. In addition, competitive learning is used to associate both the dead reckoning position and the range input perceived at each location with the corresponding place units. At each timestep, the activation of each place unit is computed as an inverse

Gaussian function of the difference between the current sensor input and the sensor input associated with the corresponding unit. During navigation, activation is also propagated across place links from currently active units toward other units on the path to the destination. If all activations are below a cutoff threshold, then the current location does not resemble any of the stored locations, so it is assumed to be a new location, and a new place unit is created, associated with the current sensor input. Otherwise, the place unit with the greatest activation is treated as the winner, and all of its associated sensor values are changed to more closely resemble the current sensor input. This allows the associated sensor values to adapt over time to changes in the appearance of places.

Initial experiments were performed in a simulated office environment containing both transient changes (moving obstacles) and topological changes (doors that opened and closed). In these experiments, all of the aspects of the system worked effectively: the use of reactive control for dealing with transient changes, the incremental construction of the APN during exploration, the use of the topological/metric structure of the APN for navigation, the use of variable-confidence links for path replanning in the presence of topological changes, and the place recognition system.

When we transferred ELAN from the simulator to the real robot, we discovered that, with minor modifications, most of the system's techniques worked well in the real world—with one major exception: the approach used for place recognition. During exploration, ELAN was able to construct an APN correctly representing the general structure of the environment, but at certain places in this environment, ELAN encountered a serious problem with perceptual aliasing. The similar appearance of these places, in terms of the robot's sonar and infrared sensors, resulted in

confusion as to the robot's actual location. As a result, spurious place links were constructed between non-adjacent places.

Perceptual aliasing occasionally occurred during simulation, but was substantially more common in the real world. We believe that this was due to the different characteristics of dead reckoning error in simulation and in the real world. In simulation, each dead reckoning translation integration step included noise of +/-20% (from a uniform random distribution), but no attempt was made to model rotational slippage. On the real robot, translation integration error tended to be small, but rotation integration error tended to be large. This had two effects on ELAN's place recognition. First, as described in Section IX, a large orientation error generates a large error in Cartesian position at locations far from the origin. This reduces the ability of the robot to use dead reckoning to disambiguate places that have similar range images. Second, orientation error tends to "blur" the stored range images due to errors in the estimated angle for each sensor reading. This reduces the ability of the robot to disambiguate places that appear similar if rotated through angles less than or equal to the magnitude of the orientation error. This can actually make localization using range images less accurate than using dead reckoning alone, since the robot may confuse places that appear similar but are separated by large distances.

In practice, we found that ELAN could navigate reasonably well for short periods of time (5-10 minutes), but that perceptual aliasing could result in substantial errors over longer periods of time. We also found that relying exclusively on dead reckoning tended to result in more accurate localization than the combination of dead reckoning and range images.

The results of these experiments led us to focus on accurate dead reckoning as a means of avoiding the problems associated with perceptual aliasing. The relocalization methods described

earlier in this article were our approach to dealing with this problem and providing accurate localization for both exploration and navigation.

In our opinion, simulation can be a powerful tool for software development. The use of simulation allowed us to build a much more capable system than would have been possible if we had been required to build the entire system from scratch during our two months at NRL. Many of the components of the system developed in simulation were transferred to the real robot with little or no modification, allowing us to focus our efforts on the parts of the system that actually required modification to work in the real world.

However, simulation is no substitute for real-world experimentation, and, as many researchers have argued [4] [24], only experiments with real robots can conclusively validate that any proposed technique will work in the real world. Compared to many of the simulators used in robotics research (which often assume error-free sensing), the Nomad Robot Simulator provides a relatively realistic domain. However, no simulation is a completely accurate model of the real world, and while it is possible to correct simulation inaccuracies with improved models, it is often impossible to know which inaccuracies are important to correct until experiments are performed with real robots. The use of real robots allowed us not only to discover these problems, but to develop solutions, and to confirm that these solutions work in the real world.

XI. Related Work

Mataric [18] has developed a mobile robot learning system that combines a behavior-based reactive controller with a distributed representation that reflects the topological structure of the environment. Mataric's system also uses the reactivity of the behavior-based controller to deal with transient changes. ELDEN differs in being able to deal with topological

changes by modifying the variable confidence links connecting the place units. In addition, ELDEN uses a more flexible behavior arbitration system that allows compromise between commands from different behaviors. ELDEN's behavior arbitration system is similar to that of the DAMN behavior architecture developed by Rosenblatt [14]. Similar methods have also been used in fuzzy controllers, where fuzzy control rules correspond roughly to behaviors, and defuzzification is analogous to behavior arbitration (for example, see Saffiotti, Ruspini, and Konolige [22]).

Kuipers and Byun [12] have developed a spatial learning system that identifies distinctive places, as defined by a set of pre-defined criteria (e.g. equal range readings in three directions), and links these places with edges specifying transit behaviors that take the robot from one place to another. ELDEN differs both in its method of place learning and in its ability to explore and navigate within dynamic environments.

Miller and Slack [19] have developed a system that combines reactive behavior with mapmaking through the incremental construction of vector fields. The obstacle fields decay over time to allow for the possibility of changes in the environment. ELDEN differs both in its spatial representation and in its explicit strategies for dealing with topological changes. In the system developed by Miller and Slack, local changes in the world will result only in local changes to the robot's motion, but in ELDEN, local changes (such as a closed door) can trigger significant path replanning at the global level.

Engelson [10] has developed a system for passive mapping using a representation containing both topological and geometric information. This system includes techniques for recognizing places, modeling location uncertainty, and correcting map errors. However, this

system has only been tested in simulation. ELDEN differs in integrating active exploration and navigation behaviors and in being implemented and tested on a real robot.

Connell and Mahadevan [6] have developed a system that learns the locations of doorways and intersections as it moves down hallways in a dynamic office environment. Their system also uses reactive behaviors for low-level control, but their system assumes that doorways and hallways intersect at 90 degree angles. ELDEN differs in integrating exploration behaviors with the learning system and in being able to learn spatial representations with arbitrary geometric and topological form.

Kurz [13] and also Walker, Hallam, and Willshaw [25] have developed systems that use competitive learning for place recognition within topological networks, similar to the original networks used in ELAN. Neither system, however, includes methods for dealing with topological changes.

Elfes [9] and Moravec [20] have used evidence grids for mobile robot navigation, but their approach differs widely from our own. In their systems, evidence grids are used as the primary representation of space, and traditional search methods (such as A* and relaxation) are used to find paths through the environment at the level of individual grid cells. ELDEN uses evidence grids, but this use is limited to relocalization. ELDEN's primary spatial representation is its adaptive place network, which explicitly models the topology of the environment and also models the uncertainty of traversability between adjacent places, in contrast to evidence grids, which model uncertainty at the level of individual grid cell occupancy.

Schiele and Crowley [23] have developed a relocalization method based on matching evidence grids using a Kalman filter. This approach has been used to localize a mobile robot

based on sonar data, using both raw occupancy probabilities and line segments extracted using a Hough transform. However, this approach has only been tested in a static environment. ELDEN's relocalization system differs in using a hill-climbing approach for matching which our experiments have demonstrated to be robust to both transient changes (moving people) and perceptual changes (rearranged obstacles).

Other approaches have also been suggested for robot spatial learning, ranging from finite-state machines [2] to topological maps based on gateway locations [11] to quadtrees constructed using tactile sensing [27]. Some of these systems include provisions for dealing with sensor error and uncertainty. However, all of these systems have been developed to explore static environments rather than dynamic environments.

XII. Conclusions and Future Work

Mobile robots need the capability to explore and navigate in dynamic environments if they are to be useful in a wider range of real-world applications. In this article, we have described a number of techniques that we have developed to deal with different types of dynamic change. Reactive behaviors can provide robust low-level control in the presence of transient changes such as moving people. Adaptive place networks can be constructed incrementally during exploration and can guide navigation in the presence of topological changes, such as rearranged obstacles. Evidence grids can be used in combination with hill-climbing to deal with one particular form of perceptual change, the dead reckoning error resulting from wheel slippage.

ELDEN is an integrated system that combines a reactive controller, an adaptive place network, and a relocalization system to provide robust exploration and navigation in the presence of transient, topological, and perceptual changes. ELDEN has been implemented and tested on a

real Nomad 200 mobile robot and has demonstrated the capability to explore, learn, and navigate in a real-world environment containing moving people and rearranged obstacles.

ELDEN currently has two primary limitations. First, ELDEN's reactive exploration strategy is inefficient because it incorporates no mechanism for guiding exploration towards uncharted territory, and as a result, time is wasted revisiting previously explored locations. Second, since ELDEN relies upon dead reckoning for localization, it cannot solve what Engelson has termed the "kidnapped robot problem" [10]. If the robot is picked up and moved blindly (without sensor input) to an arbitrary location, it cannot self-localize.

Future research will be focused on developing new strategies for exploration. A directed exploration strategy is currently being developed based on the concept of frontiers [26]. A frontier is an open arc of space visible from a particular place. Frontiers are detected using the robot's laser rangefinder, which provides accurate, high-resolution range data. A list of frontiers is stored with each place unit in the APN. When the robot is at a location containing unexplored frontiers, it uses its reactive behaviors to orient toward that frontier and advance (avoiding any transient obstacles that it may encounter in the process). When the robot is at a location that does not have any unexplored frontiers, a spreading activation search is used in the adaptive place network to find a nearby place that does have unexplored frontiers, and ELDEN's navigation system is used to guide the robot to that place.

Future work is also planned on developing a more general form of place recognition to provide ELDEN with the ability to localize from an initial unknown location. For example: Langley and Pfleger have demonstrated (in simulation) that case-based learning techniques can be used to identify places using evidence grids [15]. If these techniques can be extended to the real

world, then a similar approach could be integrated with ELDEN's adaptive place networks. A local evidence grid could be stored with each place unit, as each place is explored. Then, if ELDEN needs to localize from a completely unknown location, it could do so by generating a new evidence grid and finding the best match between that grid and the stored grids.

XIII. Acknowledgments

We thank Hans Moravec for the use of his evidence grid code, Jim Slater of Nomadic Technologies for providing the simulator/development environment, and Bill Adams for implementing the hill-climbing algorithm for the relocalization system. We also thank Alan Schultz, John Grefenstette, Frank Pipitone, and Pat Langley for their useful comments on this research. We also thank the anonymous reviewers for their comments on this article.

References

[1] R. Arkin, "Motor schema-based mobile robot navigation," *International Journal of Robotics Research*, Vol. 8, No. 4, pp. 92-112, 1989.

[2] K. Bayse, T. Dean and J. Vitter, "Coping with uncertainty in map learning," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 663-668, 1989.

[3] J. Bondy and U. Murty, *Graph Theory with Applications.* New York: Elsevier, 1976.

[4] R. Brooks, "Elephants don't play chess," in *Designing Autonomous Agents*, P. Maes, Ed. Cambridge, MA: MIT Press, pp. 3-15, 1991.

[5] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, pp. 14-23, 1986.

[6] J. Connell and S. Mahadevan, "Rapid task learning for real robots" in *Robot Learning,* J. Connell and S. Mahadevan, Eds. Boston, MA: Kluwer Academic, pp. 105-139, 1993.

[7] M. Dorigo and M. Colombetti, "Robot shaping: Developing autonomous agents through learning," *Artificial Intelligence,* Vol. 71, No. 2, pp. 321-370, 1994.

[8] M. Dorigo and U. Schnepf, "Genetics-based machine learning and behaviour based robotics: A new synthesis," *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 23, No. 1, pp. 141-154, 1993.

[9] A. Elfes, "A sonar-based mapping and navigation system," *IEEE Journal of Robotics and Automation*, Vol. 3, No. 3, pp. 249-266, 1987.

[10] S. Engelson, *Passive Map Learning and Visual Place Recognition.* Ph.D. Thesis, Department of Computer Science, Yale University, 1994.

[11] D. Kortenkamp, *Cognitive Maps for Mobile Robots: A Representation for Mapping and Navigation.* Ph.D. Thesis, Electrical Engineering and Computer Science Department, University of Michigan, 1993.

[12] B. Kuipers and Y. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Journal of Robotics and Autonomous Systems,* Vol. 8, pp. 47-63, 1993.

[13] A. Kurz, "Building maps based on a learned classification of ultrasonic range data," *Proceedings of the IFAC Workshop on Intelligent Autonomous Vehicles,* New York: Pergamon, pp. 191-196, 1993.

[14] D. Langer, J. Rosenblatt, and M. Hebert, "A behavior-based system for off-road navigation," *IEEE Journal of Robotics and Automation,* Vol. 10, No. 6, pp. 776-782, 1994.

[15] P. Langley and K. Pfleger, "Case-based acquisition of place knowledge," to appear in *Proceedings of the Twelfth International Conference on Machine Learning*, San Mateo, CA: Morgan Kaufmann, 1995.

[16] P. Maes, "A bottom-up mechanism for behavior selection in an artificial creature," *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pp. 238-246, Cambridge, MA: MIT Press, 1990.

[17] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artificial Intelligence,* Vol. 55, No. 2, pp. 311-365, 1992.

[18] M. Mataric, "Integration of representation into goal-driven behavior-based robots," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, pp. 304-312, 1992.

[19] D. Miller and M. Slack, "Global symbolic maps from local navigation," *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pp. 750-755, 1991.

[20] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine,* Vol. 9, No. 2, pp. 61-74, 1988.

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

[22] A. Saffiotti, E. Ruspini, and K. Konolige, "Blending reactivity and goal-directedness in a fuzzy controller," *Proceedings of the IEEE International Conference on Fuzzy Systems,* pp. 134-139, 1993.

[23] B. Schiele and J. Crowley, "A comparison of position estimation techniques using occupancy grids," *Journal of Robotics and Autonomous Systems*, Vol. 12, pp. 163-171, 1994.

[24] T. Smithers, "Taking eliminative materialism seriously: a methodology for autonomous systems research," *Proceedings of the First European Conference on Artificial Life*, Cambridge, MA: MIT Press, pp. 31-40, 1992.

[25] A. Walker, J. Hallam, and D. Willshaw, "Bee-havior in a mobile robot: The construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment," *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1451-1456, 1993.

[26] B. Yamauchi, *Exploration and Spatial Learning in Dynamic Environments.* Ph.D. Thesis, Department of Computer Engineering and Science, Case Western Reserve University, 1995.

[27] A. Zelinsky, "A mobile robot exploration algorithm," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 6, pp. 707-717, 1992.

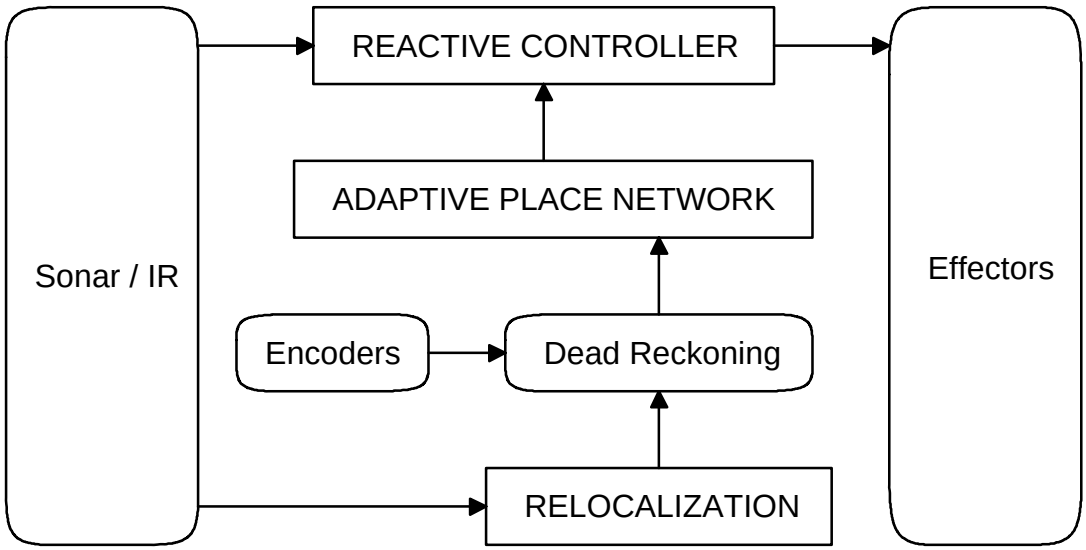
REACTIVE CONTROLLER
ADAPTIVE PLACE NETWORK
Sonar / IR
Effectors
Encoders
Dead Reckoning
RELOCALIZATION

Figure 1

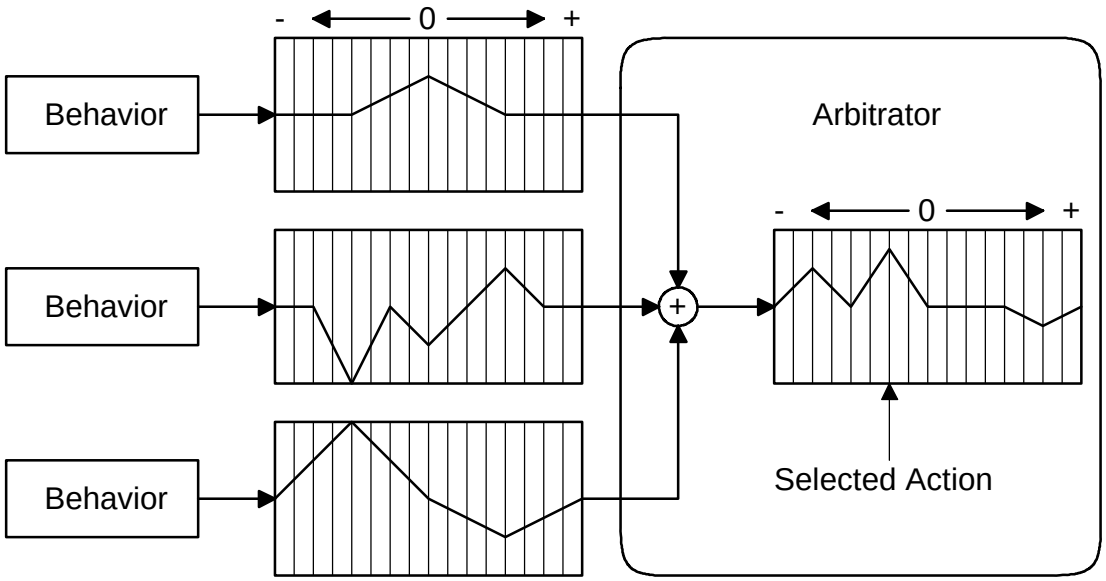
-
0
+
Behavior
Behavior
Behavior
Arbitrator
-
0
+
+
Selected Action

Figure 2

Figure 3

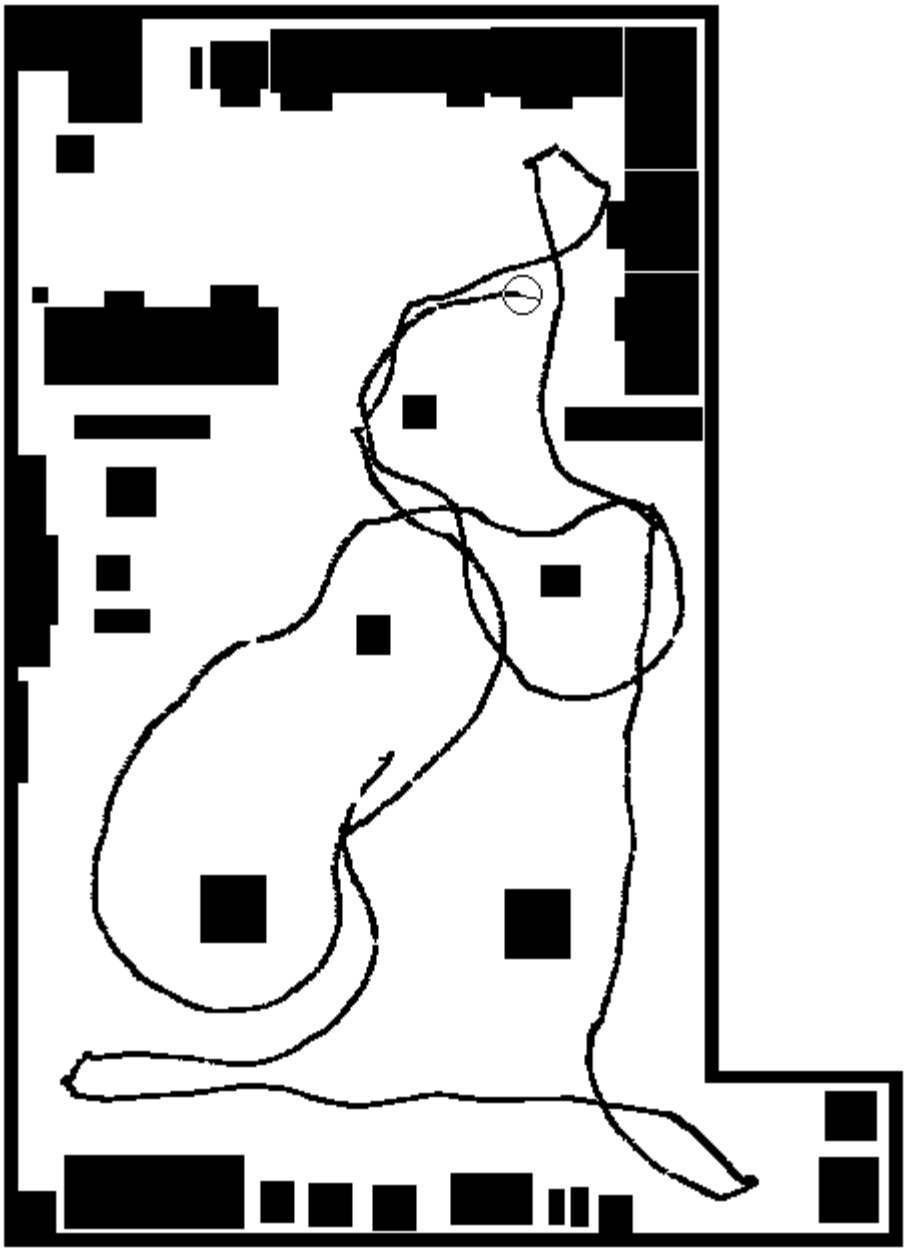

Figure 4

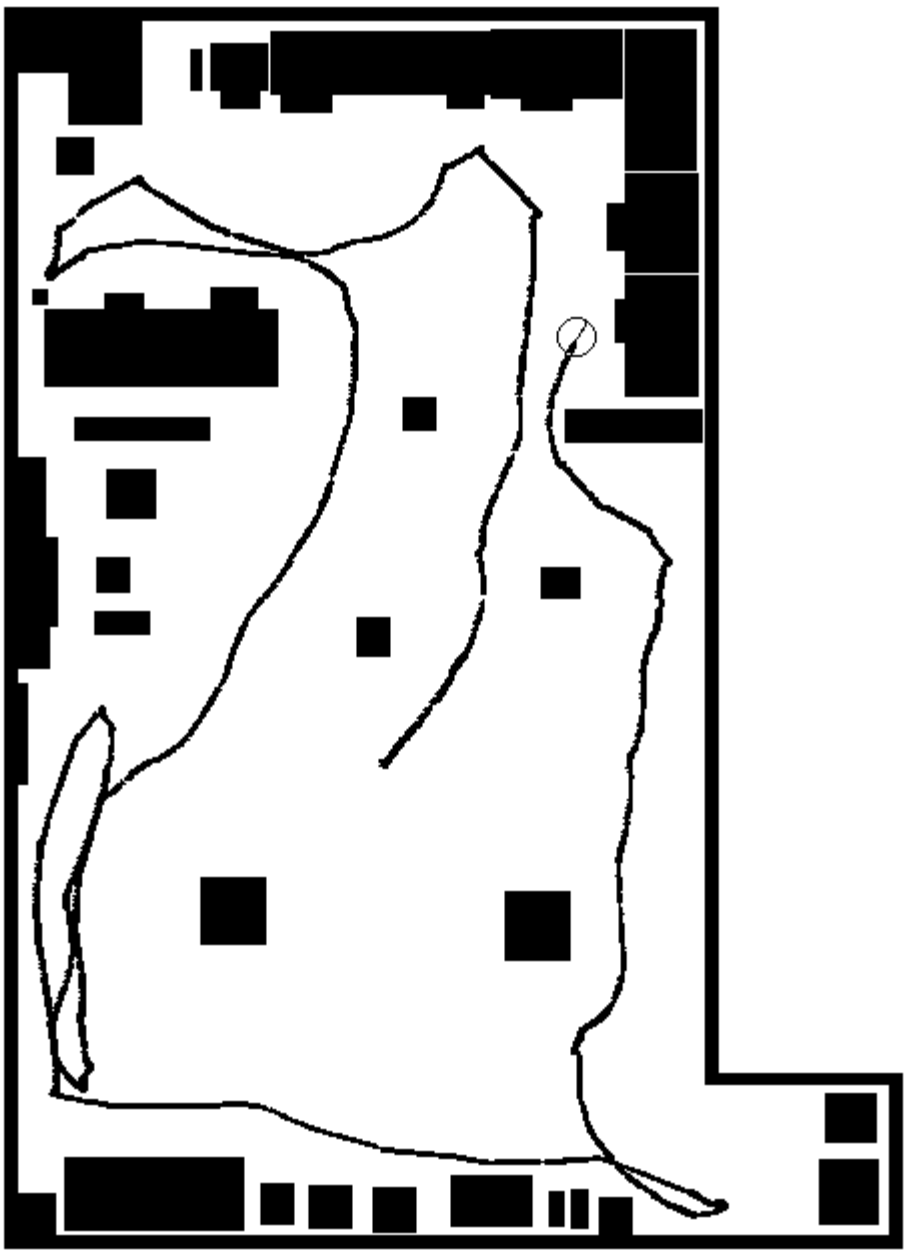

Figure 5

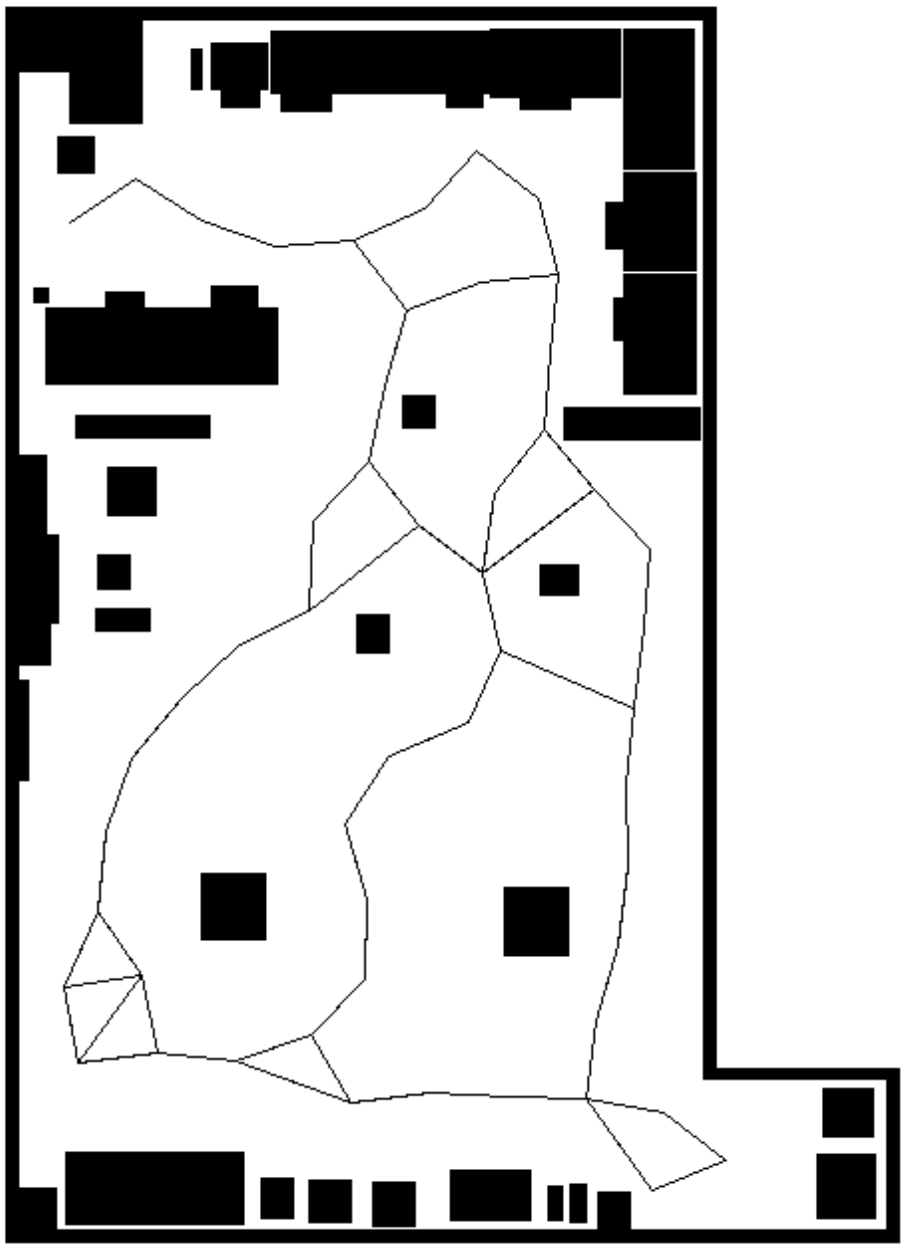

Figure 6

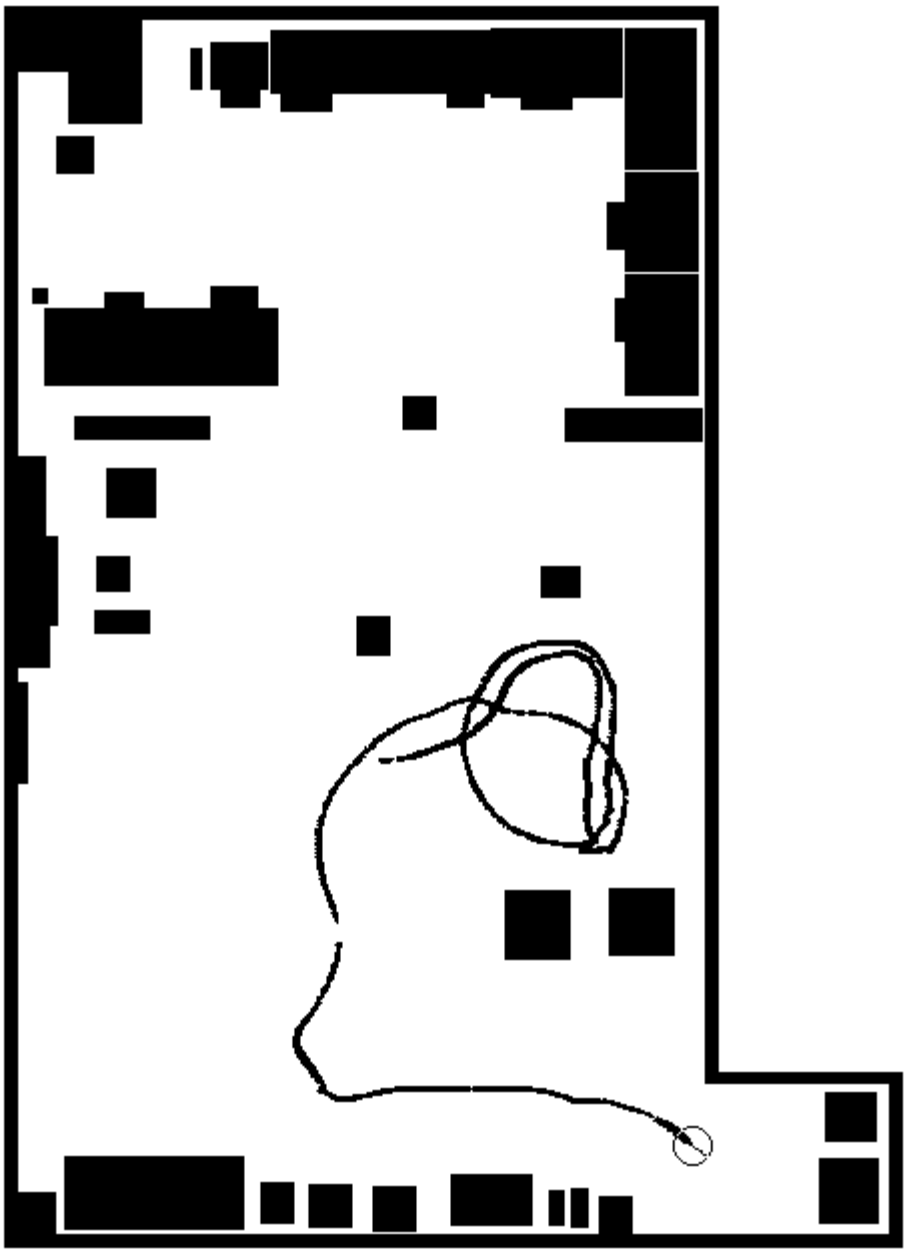

Figure 7

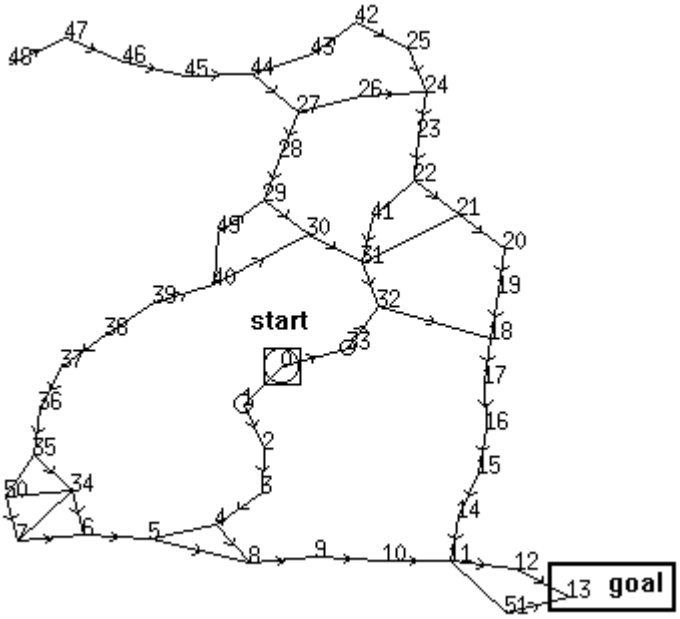

Figure 8

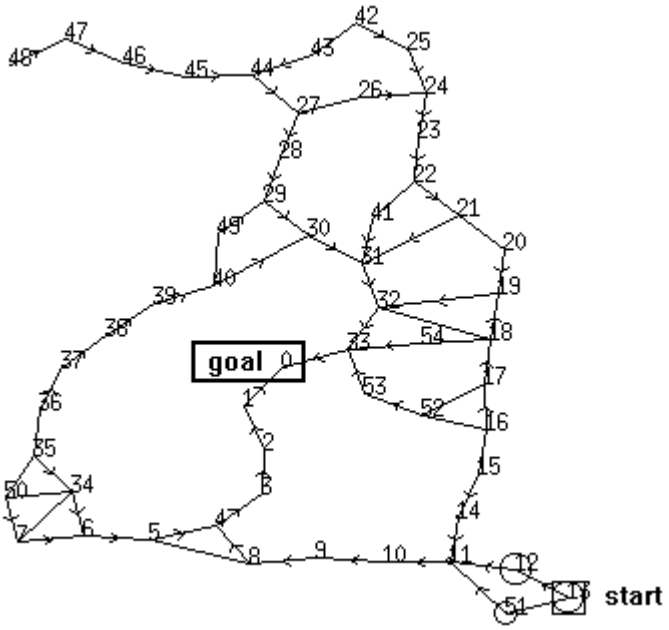

Figure 9

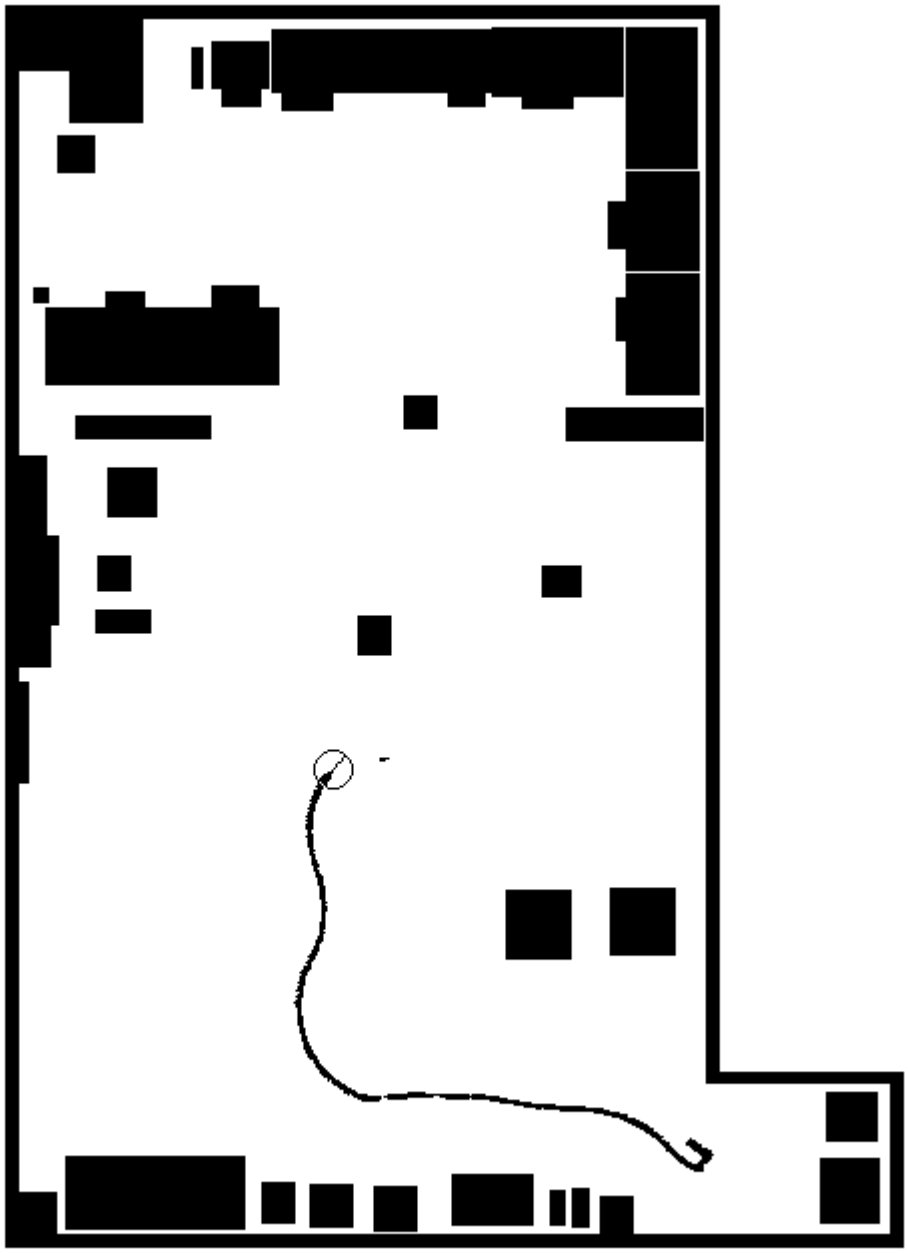

Figure 10