Navigational Map Learning by Context Chaining and Abstraction

Arshan Dabirsiaghi and Goran Trajkovski

Cognitive Agency and Robotics Laboratory
Towson University, Towson, MD 21252, USA
arshan.dabirsiaghi@aspectsecurity.com

Abstract

Efficient navigation of one's environment is a fundamental requirement of a successful mobile robot. Ideally an agent's interactions with an unmarked environment should build reliable spatial relationship information without the aid of foreknowledge. Problems arise when different parts of the environment "look" similar to the agent, confusing the agent as to its true position. This problem has become known as 'perceptual aliasing'. In this research project this problem was approached by introducing and investigating the chaining of dynamic virtual landmark identification in the agent's environment. The learning method introduced, called "context chaining" is an extension of the Interactivist-Expectancy Theory of Learning (IETAL). Experiments with software agent simulations showed the success of this approach, measured by the number of steps required to reach the drive satisfier, cumulative memory size, and the number of surprises encountered.

Introduction

There is a well-established need for mobile robots to be able to navigate and learn unknown environments while unsupervised. Applications of such technology include search and rescue robots, assistance partners for handicapped persons, reconnaissance, etc.

Although there has been an overwhelming amount of research done in this area, there exist a number of overarching problems in the field. The most prominent problems among them are dynamic environments and perceptual aliasing (Thrun, 2003).

Dynamic Environments

A successful agent must be able to adapt to changes in its environment without suffering fatal or near-fatal setbacks. Realistic agents will not have the luxury of static environments. It is also not expected that an agent who assumes and receives a static environment will have the same efficiency as one who can internalize dynamic environments.

Perceptual Aliasing

Perceptual aliasing results when there is not enough difference in an agent's stimuli in separate sub-environments to avoid locale confusion. This creates problems in both topological maps as well as grid-based maps. In grid-based maps (sometimes called metric maps), locale confusion creates conflicting entry points into those confusing sub-environments. Topological maps are represented as planar graphs, with the nodes being locations and edges between them being the agent's actions. When an agent suffers perceptual aliasing, the edges between the nodes become inaccurate.

Related Research

There exist two major approaches when designing agent internalization mechanisms. The first is grid-based mapping, which is perhaps the most intuitive approach.

Grid-based Mapping

In grid mapping, an agent internalizes its environment as an n-space occupancy grid, with various attributes maintained per cell. This is a very popular approach and it's used in various ways, such as in multi-agent total map coverage algorithms (Taylor et. *al*. 2005). However, in its rawest form it suffers some serious drawbacks. Both the confusion of dynamic environments and the expense of map maintenance cause problems for agents using this approach, which usually use a combination of dead-reckoning and GPS to navigate through their environment. Work has been done to address temporal noise (either by dynamicity or other agent movement) in agent environments by decaying the grid and forcing periodic reorientation in order to weed out temporal noise (Yamauchi & Langley, 1997).

Topological Navigation

Topological navigation is a method of navigation by which subsections of an environment are classified under abstract types, e.g., "door", "desk", "towel rack". The environment can then be represented as a graph in which the nodes are the classified environment subsections, and the edges between those nodes are the agent's navigational actions.

Topological mapping is also very popular and can be used with fairly inexpensive equipment with good results (Huang & Beevers, 2004). Hybrid solutions involving topological and grid-based mapping have been researched (Savelli, 2003) where grid-based mapping is used to aid in agent localization in a topological map.

Landmark-based Navigation

Landmark-based navigation (Kuipers, 1987) is a subset of topological mapping. Rather than trying to internalize the

entire environment in topological form where the environment's objects are not necessarily easily classifiable, the agent maintains a list of "landmarks" as nodes in a graph, again with the agent's navigational actions acting as arcs between those nodes. Johns noted that this process goes on in humans unconsciously (Johns, 2002).

Problems with environment change also tend to confound landmark-based agents. As Tani noted (1998), landmarks used in traditional approaches tend to symbolize nodes in finite-state machines (again with the agent's actions representing the arcs between the nodes). When an expected landmark fails to appear (or fails to be recognized by the agent's top-down categorizer), the agent is left in a halted state, and its choice of action can be to either ignore the change or incorporate the change into the existing FSM. Ignoring the change is not suitable since the change may be legitimate or the agent may not be properly localized. Changing the existing FSM configuration creates problems when the agent encounters temporal noise.

Our Approach

The original approach in this paper represents new vectors for research in the greater scope of agent map learning. It combines a dynamic landmark navigation system with Trajkovski's IETAL principles (2007), with a few new ideas as the translational layer between the two. Our agent's (called Demeter) learning algorithm framework serves as the "glue" between IETAL and the general subset of approaches known as Dynamic Landmark Navigation.

As in previous studies using in IETAL projects (Trajkovski, 2003; Stojanov, 1997) we use an agent with a single motivational drive; hunger. The agent uses hunger to direct its navigational actions when its hunger drive is activated.

Agent Capabilities

The agent is presumed to be equipped with a camera and software capable of colored image processing, a compass, and a series of range-sensors. The range sensors are presumed to be fixed upon a mobile head, capable of pivoting in a full circle so as to accurately measure the distance from the agent to any other object within the sensor horizon, which is a simulation variable that can be tuned for different experiments. The sensors have 360 degrees of freedom. The agent's vehicular motor and other movement apparatus are not necessarily important for the simulation, other than to say that the agent is assumed to have 360 degrees of movement.

The camera is used only to differentiate in bottom-up fashion the difference between "food" and other obstacles.

Using the equipment just described the agent is capable of taking a relatively robust 2D "snapshot" of its current world state. This snapshot is essentially an array of 360 ranges of which each contain a distance measured in generic measurement units and a state value indicating the type of obstacle sensed (obstacle or drive satisfier). A current world state view is shown in the designer's view and agent's view in Figure 1.

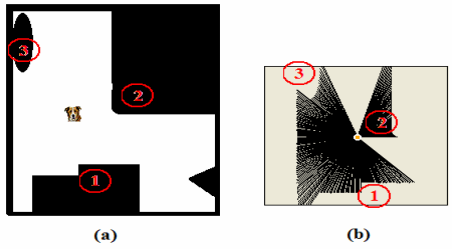

Figure 1: Agent Perspectives: a) The designer view, b) The agent's sensor-based snapshot

The snapshots are the basic tools used by the agent when trying to internalize its environment in the proposed learning method called "context chaining". Before going into the details of the learning method, details about the agent's biomimetic capabilities must be understood.

Biometric Capabilities

As is generally agreed on (Franz & Mallot, 2000; Stojanov, 1997), any theories dealing with intelligence should take into account biological systems since they are they are the only examples of "genuine intelligence". Although the biomimetic capabilities discussed in this section don't necessarily contribute much to the intelligence of the agent, the infusion of biological behavior should be considered at every layer possible.

The primary biomimetic capabilities of the agent are its ability to avoid collisions and its ability to go to an object with which it has line of sight. Before going into the details of these two operations, the nature of the agent's movement should be clarified. The agent's movements are not continuous, although they may appear so during the simulation. Its movement is a series of atomic events, consisting of an abstract script that is implemented differently depending on its current navigational mode. The general form of the movement script is like this:

1. Move in current direction at current "speed" (speed being the units traveled between re-evaluation)
2. Re-evaluate environment, mode, speed, and direction.

An important note about the agent's biomimetic capabilities is that they are considered inborn functions of the agent, not unlike how biological agents need no instruction on how to move to an object with which they has line of sight, or how to avoid collisions (although, embarrassingly enough, biological agents of the highest order still run themselves into walls, doors, trees, each other, etc.). This behavior is essentially instinctual. Thus, in the framework as well as in the code, these functions are logically separated from the proposed learning method, though the output of these functions feeds into the learning method to influence future decisions.

Collision-Free Navigation. The agent contains an inborn algorithm for safely moving around an obstacle-ridden environment without physically disturbing any of the obstacles. This is similar to Tani's robot (2004), although a simple bottom-up marker notifying the upper layer of an impending crash changes the agent's direction, instead of the potential-field method.

Drive Satisfiers and Object Tracking. Biological agents have "drives" which motivate the agent to choose actions. These drives are manifested as appetitive behaviors like hunger or the search for a mate, as well as their associated consummatory acts like eating and reproduction. Formal models of biological drives have been researched (Tinbergenn, 1951) and are generally considered hierarchal in nature. The research preceding this work focused on understanding an agent with a single internal drive which has been genetically programmed into the agent as a starting point with the goal of eventually considering agents with multiple, conflicting drives. The drive in our research, which is purely symbolic, is hunger. When the hunger drive is activated, the agent directs his attention towards navigating its environment in search of food.

So, the agent will occasionally have its hunger drive activated (the time between drive satisfaction and drive activations is simulated- the agent simply reappears elsewhere supposing a random amount of time has passed). In accordance with instinctual behavior observed in biological agents, the agent contains an inborn algorithm for moving toward a differentiable object in space. This ability resides above the collision-free navigation layer, so that the agent will not collide with any of the obstacles in the environment on its way to reach that object. This capability is used to allow the agent to navigate to a drive satisfier it senses.

Context Chaining

Creating Hot Spots. When the agent is initially navigating the environment, its algorithm in pseudo code form is as follows:

```
Move(CurrentDirection)
LoadSensors()
If SenseDriveSatisfiers() = True

TemporarilyRememberThisHotspot(CurrentWorldS
tate)
   If GoToDriveSatisfier() = SUCCESS
      MoveTempToPermanentMemory()
      RandomlyPlaceAgentOnMap()
   End If
End If
If OnCrashCourse()
   CurrentDirection :=
GetRandomCollisionFreeDirection()
End If
```

The term "hot spot" is used to represent a spot where the agent can sense (hitherto referred to as "seeing") drive satisfiers (sometimes referred to as "food"). The SenseDriveSatisfiers() function represents the *bottom-up* tasking inborn to the agent. If a drive satisfier is detected, the agent switches modes and feeds its current input to a new task handler (in this case, the GoToDriveSatisfier() function).

The agent would like to remember this "hot spot" for the future. Given that, the agent stores a current snapshot of it in temporary memory. The agent then attempts to move and "obtain" the food, where "obtaining" means the agent is capable of physically touching the food. If the agent is able to reach the food successfully, it moves the hot spot already stored into permanent memory. The agent may not always be successful in obtaining food that it sees, since its girth may prevent it from navigating through impeding obstacles, such as bars or fences.

The fact that the agent is constantly searching for an existing hot spot (or cue spot, as discussed later) represents the *top-down* tasking of the agent. When the agent has a collection of world states that have led it to drive satisfaction in the past it will constantly and proactively scan for those world states, and any matches will provoke a switch in the agents mode to a following algorithm discussed later. In order to understand that process flow, the algorithms the agent will use to detect the landmarks in the environment need to be discussed.

Transpositioning. Chronologically, the next element in the learning process is where transpositions are first used. The agent uses "transpositioning" to compare its immediate world state to predict the layout of world states near it in all directions in order to see if any "hot spots" or "cue spots" are near it. As discussed by Stojanov (1997), biological agents use visual frames to predict and expect their next world state. If a biological agent is operating in an intimately known environment and finds something incongruent with the expected world state, the agent registers a surprise emotion (something which comes into play later, in the core of the learning method). The transpositioning process is a simple geometric transformation on the current world state, permutated in different directions and distances around it. The agent compares all those permutations to the stored "hot spots" or "cue spots" (the union of which are referred to as the collection of the agent's world states) using a matching algorithm (although chronologically there are no cue spots in the cognitive map yet).

The agent compares these transpositions to the "hot spot" it found before. If the agent finds a favorable comparison between that hot spot and a transposition, it transfers control to a FOLLOW function (discussed at length later) and passes it the world state the agent wishes to go towards. The first issue dealt with before continuing is how to effectively compare world states to one another.

Comparing World States. Comparing world states is an essential and tricky process that only found success after extensive trial-and-error. Comparisons happen quite frequently during the agent's traversals (after every step, up to thousands of times). An accurate way of "comparing" one snapshot of the world-state to another was needed.

The basic premise of the COMPARE function created to fill that need is that it returns an exponential regression fit between two data sets (world-states). Differences between the two data sets is raised to β and accumulated. The typical use case for this function is to compare a known snapshot or world-state with a transposition of the current world-state.

There are two constants used in this function, α (alpha) and β (beta). Alpha is a threshold variable which dictates the maximum difference between two world states while still considered a "match". Once the accumulated difference reaches that point, more accumulation is irrelevant so the function returns a value greater than α so as to indicate comparison failure (α plus one).

Beta is an experiment variable tuned for precision by exhaustive trial-and-error. It is the control mechanism for how "loose" the agent is with comparing hot spots and cue spots. If the variable is too small, the agent will make many mistakes when finding "matches", but it will be able to "see" world states from relatively far away. Conversely, if the variable is too large, the agent will be too sensitive to any differences in world states, and will be unable to match world states unless practically already at them. However, it will make practically no mistakes aside from any related to perceptual aliasing.

The value of β for the experiments within was chosen to complement the matching model used to match saved world-states that have limited sensor information available. A non-static matching model was required, and this is explained in the next section.

Unfairness of Static Models When Comparing Sensor Readings. Suppose the theoretical function COMPARE returned the accumulated difference between each sensor value in two states. The accumulated difference was returned and compared to a threshold value in order to determine a match. This function could be written as seen in the boolean equation $\sum (Ai-Bi)^{\beta} < \alpha$, where A is a saved world state and B is the agent's current world state. Ai represents the ith sensor value in A. This formula is inherently unfair because the function favors world-states with limited sensor information.

Consider an environment where the agent's range readings are nearly non-existent because only a few items are in the range horizon, at a far-away distance. Consider a second environment in which the agent is extremely close to 2 objects; so close in fact that practically all its range readings are fairly small, representing another scenario where range data is limited. For this case, the total sum of the sensor readings is limited, whereas the first case represented a situation where the number of readings itself was limited.

Either way, the sum of B's range readings (sometimes called the 'sensor sum') is small, perhaps around the size of or smaller than α. Consider then the possibility that the agent is comparing itself to the "hot spot" world-state A that also had limited sensor readings available. It is plain to see that as the sum of the two world-states to be compared gets smaller, the easier it will be to successfully "match" using this algorithm. As $\sum$ Ai and $\sum$ Bi diminish, COMPARE(A,B) tends to 0.

To combat this, a non-static model for comparison was designed originally for this work. The proposed solution is a fairness model that incorporates the sum of the sensor readings available when making a match.

The value of α (match threshold) becomes dynamic in the model. The total sensor sum for a snapshot is used as an input to decide what α should be. In situations where enough sensor information is available (when the sum of sensor readings is >= α), then α will be higher (indeed, the value returned will be α itself). In situations where sensor information is limited, α will be lower. The actual function to produce α is shown here, along with a graph of the function in Figure 2:

```
FUNCTION GETALPHA
GIVEN WORLDSTATE SavedWorldState
 SensorSum := SavedWorldState.GetSensorSum()
 SensorSum := SensorSum / 1000
 RETURN MIN (α, 2^(SensorSum-6))
END FUNCTION
```

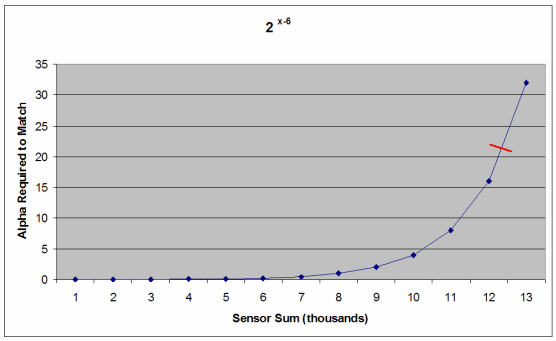

Figure 2: The red line represents the maximum value for α

The COMPARE function works in constant time, so not much work has been put into optimizing it. However, late in the project it was realized that an optimization is sorely needed. An agent with a "fully loaded" memory (an agent whose memory is not expanding any longer) has visually noticeable slowdown when searching for a path. Fortunately, an agent with a "fully loaded" memory is not searching for very long in an environment it can conquer. And once an agent finds a path (also called a "chain") it doesn't do all the normal comparisons; it only compares its environment towards the next step in the chain. However, the slowdown is enough to affect simulation data and that must be resolved. A promising improvement upon this algorithm is proposed in the Future Work section.

Cue Spots. Once the agent has found a "hot spot", it has an average confidence in its ability to lead it to food, and when its drive is activated it will begin searching for that hot spot (and any others it has come across). If the agent thinks it sees a "hot spot", it performs a function similar to that of creating a hot spot (this function also looks for cue spots since they will also lead the agent to food).

The algorithm for creating a new cue spot uses the same variable α (alpha) as in the COMPARE function. The agent stores the location it "sees" the hot spot world state from in temporary memory and only commits it to permanent memory if it turns out to be right, because the hot spot led the agent to food. The cue spots (which along with hot spots make up the set of saved world states) represent the topological element of context chaining. They are essentially links in a chain created by the agent from food to the rest of the environment. This technique possesses the inherent benefit that the agent does not need to store or maintain as much information as a grid-based mapping approach would.

Cue spots are attached to other cue spots, resulting in an eventual chain of cue spots to a hot spot which points to food. Since cue spots can have an arbitrary number of child cue spots, the cue spots have a tree-like structure. A very high-level visualization of an agent's cognitive map could be represented as in Figure 3:

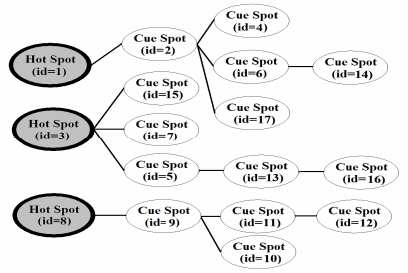

Figure 3: The agent's cognitive map of the environment

Note that agent doesn't know the distance between its current spot and the cue spot or hot spot that it "sees". The cue spots are stored in a tree form data type. The physical distances between nodes are not maintained. To reiterate the agent's independence from any global dependencies, the agent does not maintain a coordinate system to locate the cue spots or hot spots, and only relies on his world state transposition and comparison functions.

The agent does keep track of how many times it has found a hot or cue spot. Also, associated with each hot and cue spot is the number of times the hot or cue spot has induced a surprise (in that it was either "seen" but not reachable or reachable but not successful in leading to the agent to food). These simple attributes, when joined with information from surrounding world states can be used to provide context to an agent.

Perceptual aliasing results when an agent is not able to localize itself accurately based on its current stimuli. Recognizing situations when the agent should or shouldn't rely on stimuli is the first step to overcoming perceptual aliasing. This ability is accomplished in Demeter by using IETAL principles. More specifically, the emotion associated with each saved world state (represented by the number of surprises resulting from using that world state the and overall confidence associated with it) allows the agent to develop confidence in a path through the nodes in its cognitive map using the confidence in individual nodes.

The confidence associated with the individual nodes of the cognitive map gathered during interaction will affect a "big picture" confidence in certain paths through the graph. As the agent navigates the environment, it will be surprised by locales that look very much like one another. Sets of individual nodes that the agent loses confidence in represent sub-environments that are too similar to other sub-environments to rely on. The nodes representing the entry paths will always have a high confidence since, when found and followed, they reliably take the agent to a drive satisfier. Those are the nodes that the agent will learn to rely on. Again, due to its interactions the agent learned and continued to reinforce expectations. Therefore, there exists an emotional context in the agent's choice which is provided by the agent's experiences.

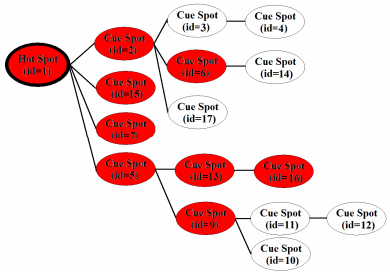

Figure 4: Subset of an agent's cognitive map which shows unreliable (red) and reliable (white) nodes

In Figure 4, it is shown that the subset of nodes {2,5,6,7,15} and {9,13,16} represent locales that are perceptually identical to other sections of the environment. The entry points, however (the sets {3,4}, {14}, {17} and {10,11,12}) are perceptually unique, and therefore reliable for following. In actuality the confidence of the implementation is far more granular, and in the experiments the confidence in each node is visualized as the deepness of the color of the node. A highly unreliable node will be true red, while a normally unreliable node will be pink.

This functionality represents the agent's behavior of learning when and when not to rely on stimuli to direct the agent to satisfiers. If the agent is following a path then it does not concern itself with the confidence of the nodes it is passing through. It is irrelevant if the node it is currently following is unreliable because for the agent to be on the path at all it must have followed a reliable node somewhere deeper in the tree. Over time, all the entry points that are confusing will prove they are unreliable and cause the agent to lose confidence in them. As such, the agent won't begin a path planning process at one of those nodes.

This is the type of emergent behavior that follows with Brooks' (1986) assertion that complex behavior (coping with perceptual aliasing) can result from a simple system (creating expectations for certain landmarks/world states).

Learning Controls. The agent has a few controls in place in order to make the learning algorithm work. For one, the agent has a mechanism for preventing two extremely similar world states to be saved to permanent memory by comparing all new world states to the saved world states. Any saved world state whose difference from a saved world state less than β is not moved to permanent memory. The parameter β is a threshold variable for determining an absolute match; the difference acceptable for an agent to be 100% confident that two world states are the same.

Also the agent has a deadlock prevention mechanism. If an agent were to find a world state that prompted the misleading of the agent, it would continue re-discovering and failing until it completely diminished any confidence in the world state in question. The immediate response might be, "If that world state was misleading, how come you don't want to eliminate it anyway?" The fact is that any world state could be so important in another part of the environment that reducing its confidence that dramatically could be a fatal blow for the agent.

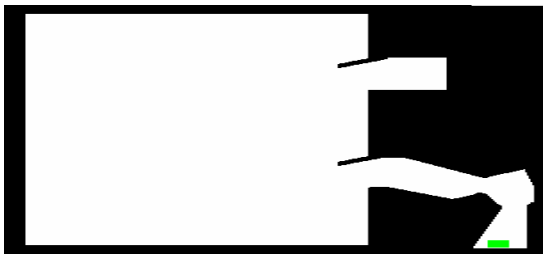

Figure 5: An environment with 2 similar sub-environments

Consider the environment in Figure 5. There are two doors on the east wall. One leads to a twisty path which leads to food, and one opens straight into a wall. If the agent were to find food using the door and then diminish confidence the world state that looks like a door because the top door persistently led to failure in one time period, the agent would fail completely.

Binding Cue Spots together

The agent contains a FOLLOW algorithm which serves as the glue that binds links in a cue spot chain. It is the topological traversal algorithm which makes an agent capable of traveling from its current environment to another known environment using a chain of saved world states and all the capabilities and algorithms described previously. Like many things, the idea is simple but ends up being complex in its implementation. The algorithm is mainly a loop that uses the COMPARE function and transpositioning to decide how large a step in what direction would result in the most congruence with the target world state. The algorithm will decrease its step size when it realizes it overshot or missed its target. The realization may be bogus however, because the agent will not know the difference between "missing" and "not really there". So, every time a bad step is taken (meaning the previous difference to the world state was less than what exists currently) the step size is decreased by half. Once the step size reaches γ, a simulation variable, the FOLLOW algorithm has officially failed.

A failed FOLLOW means that the world state the agent was currently looking for was unreliable and that all the previous world states that led it to that point are also untrustworthy. Because of that, the FOLLOW algorithm recursively decreases the confidence associated with the cue spots in the chain upon failure. Conversely, if the FOLLOW was successful the agent recursively increases the confidence associated with those same cue spots.
This modification of confidence is what gives the agent the versatility needed to survive in dynamic environments. Even if the agent reaches the complete minimum confidence in a world state, the agent will increase its confidence in that node if it is used in a path that leads the agent to food. The minimum confidence value in any node is determined by the simulation variable ε (epsilon). If a node reaches that level, then it is no longer sought after when the agent is looking for a path. If a node reaches that ε but is used in a successful path later it will be restored to $\varepsilon+1$, which re-enables the world state.

If the agent were to completely diminish its confidence in certain world states and never look back before an environment change required the agent to watch for those spots, the agent would perish. The ability to lend itself to surprise hurts the agent in static environments, but that same flexibility is its saving grace in dynamic environments.

The FOLLOW function is recursive, so upon successful completion FOLLOW calls itself on the parent of the current goal world state. Once there is no parent of the goal world state, the agent should be in sight of the food. If the agent does reach food eventually, the agent increments the confidence level in each of the chain links (nodes) used. Upon failing it decreases the confidence level in those same links.

Data and Conclusions

A number of experiments were conducted in various environments in order to test the validity of the learning framework. They were conducted using different environments, but with the same simulation variables. The time taken to reach the satisfier, the number of surprises, and the cumulative memory size was recorded during each simulation, and the average case of three trials is presented.

Dynamic Environment

The Dynamic Area environment seen in Figure 6a provided a number of valuable elements for research. It

provided the agent with many dynamic obstacles to avoid. It was also used as a "control" environment to prove that the framework was capable of succeeding in environments that had little to no perceptual aliasing as well as abundant stimuli throughout the environment.

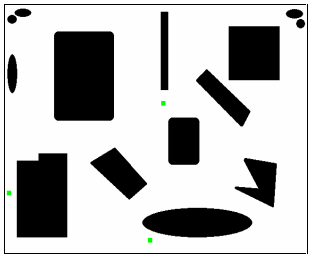

Figure 6a: The Dynamic Area environment

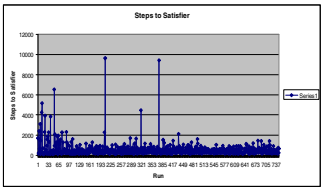

Figure 6b: Steps to satisfier for the Dynamic Area

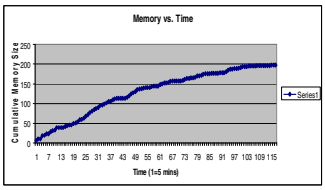

Figure 6c: Memory size over time for Dynamic Area

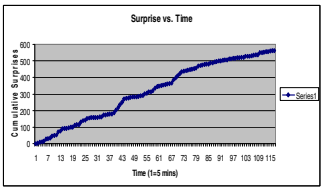

Figure 6d: Surprise over time for Dynamic Area

The agent was successful in all areas of data recorded in this simulation. The reduction in steps needed to reach the satisfier was drastic, indicating a quick internalization of the environment (Figure 6b). Also, the agent's cognitive map was shown to be complete by the quick leveling of the agent's memory size (Figure 6c). As is shown in the surprises, it agent learned the reliability of the nodes in the cognitive map very well (Figure 6d).

Big Circle

The Big Circle environment seen in (Figure 7a) provided the agent limited perceptual aliasing challenges. All areas except the center-middle provided the agent with stimuli. However, an agent's vertical movements did not alter the stimuli dramatically, causing some locale confusion. The environment lends itself to successful navigation using only the agent's collision-free navigation mode, so the learning mechanism was hoped to make the average case and best case results congruent.

The agent was successful in all areas of data recorded in this simulation. The reduction in steps needed to reach the satisfier did not reduce as drastically as others because the agent's biomimetic capabilities are sufficient to reach the satisfier comparatively quickly (Figure 7b). Also, the agent's cognitive map was shown to be complete by the quick leveling of the agent's memory size (Figure 7c). As is shown in the surprises, it agent learned the reliability of the nodes in the cognitive map very well (Figure 7d).

Perceptual Aliasing with Prison

The Perceptual Aliasing with Prison environment (seen in Figure 8a) is the ultimate challenge for the agent. It involves wall-following, perceptual aliasing and a "prison"-type sub-environment in the northwest quadrant that is very difficult to escape.

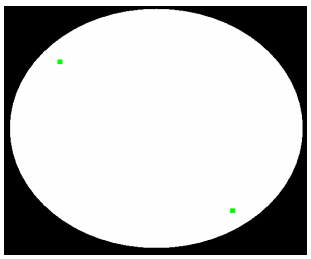

Figure 7a: The Big Circle environment

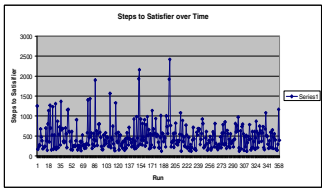

Figure 7b: Steps to satisfier for the Big Circle

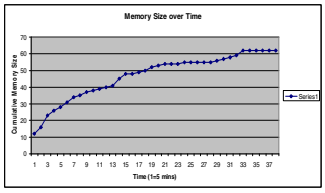

Figure 7c: Memory size over time for Big Circle

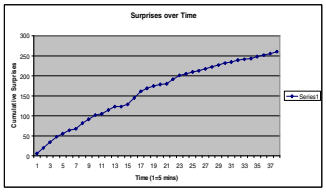

Figure 7d: Surprise over time for Big Circle

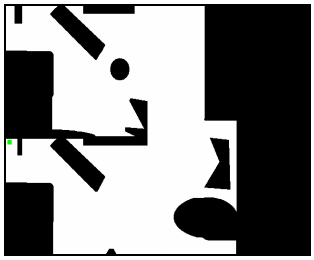

Figure 8a: The Perceptual Aliasing environment

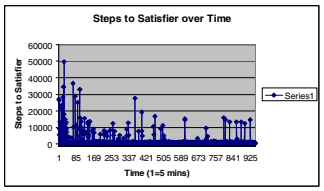

Figure 8b: Steps to satisfier for the Perceptual Aliasing

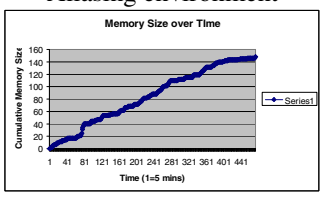

Figure 8c: Memory size over time for Perceptual Aliasing

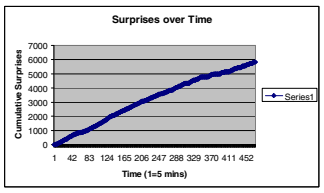

Figure 8d: Surprise over time for Perceptual Aliasing

The agent successfully internalized this environment as well. The steps required to reach the satisfier decreased considerably, though the agent sometimes got stuck in the prison sub-environment, resulting in anomalistic spikes (Figure 8b). The agent's memory size was shown to be complete, although the internalization process took nearly twice as long as other simulations due to the prison environment (Figure 8c). The cumulative surprises show that the agent never fully to learn what landmarks to rely

on after an extended internalization period (Figure 8d). This showed that some nodes were as successful in leading the agent to food as they were in leading it astray, creating a precarious balance of uncertainty.

Conclusions

Advantages. Overall the research demonstrated that an unsupervised agent using IETAL principles could internalize unmarked environments, and mitigate the damage done by perceptual aliasing.

The internalization is shown by the drastic reduction in steps taken to reach a satisfier in Figures 6b, 7b, and 8b. These reductions indicate the agent's reaching of functional equilibrium with the environments tested.

The completeness of the cognitive maps is demonstrated by the logarithmic formation of the cumulative memory size as shown in Figures 6c, 7c, and 8c.

The surprise data found in Figures 6d and 7d, representing data from the Dynamic Area and Big Circle simulations, are considered successful as they follow a logarithmic model. The Perceptual Aliasing/Prison surprise data in Figure 8d also fit the logarithmic model, but the learning curve was much steeper due to all the challenges presented to the agent in the environment. However, as hoped, the agent created "prison escape" path which led from the prison-type sub-environment (which was not localizable on sensory input alone) through the environment to the prison's perceptual counterpart environment which contained food. This proves that the agent learned not to rely on the perceptually aliased sub-environments, and used the unique entry and exit paths out of those environments to form a single, reliable node tree which consistently led the agent to food.

The biomimetic capabilities of the agent also proved very successful. A circumstance during the simulations was not encountered in which the agent could see a satisfier and not be able to reach it. The algorithm allowed the agent to navigate around obstructions in order to reach food, including when only a partial view of the satisfier was seen. The agent also never got "stuck" anywhere, thanks to his inborn sensory capabilities. Although these were not central goals of the research, they provided independent value to it.

Disadvantages. The cumulative surprises did not reach optimal drop-off in Figure 8d, due to a few factors. First, the slowdown resulting in the simulation prevented the deadlock prevention mechanism from working properly. The timeout central to the mechanism was frequently eclipsed when the agent was hunting for the same two spots repeatedly, amounting to a deadlock.

Second, the agent would infrequently hunt around a world state and not be able to make the match despite being extremely close. The technique for measuring difference in two world states relies too much on the existence of overlap in sensor readings.

Finally, he agent's need to maintain some faith in poisonous spots (which is represented by the propagating increment in confidence for the whole chain every time a spot is used to lead the agent to satisfier).

References

Brooks, R. A. "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, RA-2, April, 1986.

Franklin, Stan and Graesser, Art. "Is it an Agent, or just a Program?" *Proceedings of the 3rd International Workshop on Agent Theories, Architectures and Languages*. Springer-Verlag, 1996.

Franz, Matthias and Mallot, Hanspeter. "Biomimetic Robot Navigation*". *Robotics and Autonomous Systems*, No. 30, pg. 133-153, 2000.

Kuipers, Benjamin and Byun, Yung-Tai. "A Robust, Qualitative Method for Robot Spatial Learning". *Proceedings of the AAAI*, 1987.

Savelli, Francesco. "Addressing perceptual aliasing in topological map-building: economical use of metric information in a cognitive framework". Ph.D. Thesis, Universita degli Studi di Roma, "La Sapienza", 2003.

Stojanov, G. "Expectancy Theory And Interpretation of EXG Curves in the Context of Biological And Machine Intelligence". Ph.D. Thesis, SS Cyril and Methodius University, 1997.

Tani, Jun. "An Interpretation of the 'Self' From the Dynamical Systems Perspective: A Constructivist Approach". *Journal of Consciousness Studies, 5 (5-6),* 1998.

Tani, Jun. "Model-Based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective". *IEEE Trans. System, Man and Cybernetics (Part B), Special Issue on Learning Autonomous Robots*, Vol. 26, No. 3, 1996: 421-436.

Thrun, Sebastian. *Robotic Mapping: A Survey*, 2003.

Trajkovski., G.: "MASIVE: A Case Study in Multiagent Systems". *Intelligent Data Engineering and Automated Leaning*, LNCS 2412, Springer-Verlag, 2003: 249-254.

Trajkovski G, Stojanov G., Davani D., Williams A.: "Investigating Learning in Biologically-Inspired Artificial Agents: The POPSICLE Experiment", *14th Annual Midatlantic Conference on Artificial Intelligence and Cognitive Science*, Cincinnati, April 12-13, 2003.

Trajkovski, G. *An Imitation-Based Approach to Modeling Homogenous Agents Societies*, Idea Group Publishing, 2007.

Yamauchi, B., Langley, P. "Place recognition in dynamic environments." *Journal of Robotic Systems*, 14 (2) 1997: 107-120.