Curiosity in Time

Jenny Barry and Heather Jones
Computer Science
Swarthmore College

May 15, 2006

Abstract

We explore the idea of adding the ability to learn time dependent tasks into the Intelligent Adaptive Curiosity (IAC) learning scheme as presented by Oudeyer, et al. [2004, 2006]. We use a three layer architecture (IGE) consisting of a bottom-level IAC brain, an Elman network, and a top-level IAC brain. For both IAC brains we use the equilibrium Growing Neural Gas (GNG) algorithm to cluster the regions rather than k-means clustering. The bottom-level IAC brain takes as input the robot's sensors and motors and outputs actions, while the Elman network takes as input the regions of the lower-level IAC brain and attempts to predict the next region. The upper-level IAC brain takes the hidden unit activations of the Elman network and uses these to decide on actions for the robot. Our environment consists of a square where each corridor of the square has a different color scheme. Thus, each corridor is, in itself, predictable by the original IAC architecture, but the corners where the corridors join require time-dependence to predict. Upon running the experiments, we found that using GNG to cluster the regions allows the robot to learn the environment with only 20-25 regions, which is a significant improvement over the approximately 200 that would be formed by k-means clustering. However, while the robot was able to make accurate predictions in the corridors, it could not learn the time-dependence of the environment. We hypothesize that this occurred because the robot's behavior when controlled by IAC is too unstructured to be used as input to the Elman network. We conclude, therefore, that an Elman network may be helpful in introducing time-dependence to IAC, but only if trained off-line or if the lower-level brain produces more structured actions than those of IAC.

1 Introduction

In the traditional approach to robotics, a robot is engineered to perform a specific task or set of tasks. After the design of one robot, if a robot is needed to perform a different task, the design process must be repeated. As the tasks become more complex, however, designing a new robot for each new task becomes more expensive. Clearly, it would be advantageous if single robot could be designed that could learn many tasks, even those which were not anticipated when it was created. Because these unanticipated tasks may be structured very differently than any previously seen task, it is also helpful if this robot can develop its own representation of the tasks it is asked to perform. Furthermore, since a robot's physical abilities and perceptions may be nothing like those of its designer, engineering the robot's actions may result in far from optimal solution due to the designer's anthropomorphic bias; allowing the robot to develop its own representation of its world can avoid this problem. In a 2005 paper, Blank, et al. identified three crucial pieces that should be present in a developmental robot, abstraction, anticipation and self-motivation. Abstraction is necessary in order to condense the large volume of available sensor data into a compact form that the robot can use to make decisions. Anticipation is necessary to allow the robot to consider, when choosing an action, what is likely to occur if it chooses each action. Self-motivation is necessary to insure that the robot actually wants to learn. Many researchers have investigated one or two of these concepts, but few have attempted to combine all three in a single experiment.

Forming abstractions from sensorimotor information involves dividing up the sensorimotor space in some way. One method of doing this is a growing neural gas network, or GNG (Fritzke, 1995). In GNG, the two nearest nodes to the input vector are found. If no edge exists between them, a new one is added; if an edge does exist, its age is reset to zero. Then the nearest node is moved slightly closer to the input, and the nearest node's neighbors (those nodes connected to it by edges) are also moved toward the input, but not quite as far. A GNG network begins with two nodes placed randomly, and adds new nodes every X timesteps in the area of the network with the largest error. On each timestep, edges are aged; when an edge reaches the maximum age, it is deleted, and when a node is no longer connected to any edges it will be deleted. In this way, nodes are moved toward areas of the input space where input exemplars actually occur, and outliers are removed. Fritzke demonstrates the algorithm's success with two and three dimensional topologies, but clearly it can be extended to any number of dimensions. Provost, Kuipers and Miikkulainen used a modified version of GNG, called equilibrium GNG, that only adds nodes when the overall network error is above a certain threshold (2006). This ensures that when GNG has added enough nodes to accurately cover the input space, it will not continue to add extraneous nodes.

In cases where predicting what will be seen next can only be done if more information than just the current sensorimotor vector is available, a robot needs to have some kind of memory. This can be accomplished with an Elman network (Elman, 1990). An Elman network is essentially a feed-forward neural network with recurrent connections to a "context" layer. This allows the network to learn complicated time-dependent problems, such as the pattern of words in a sentence.

One example of an intrinsic motivation algorithm is Intelligent Adaptive Curiosity (IAC), developed by Oudeyer, et al. (2004). IAC divides its sensorimotor space into regions, each of which contains an "expert" which predicts what sensor values will be seen next, given the current sensor values and the chosen action. The prediction errors for past sensorimotor vectors in a region are stored with the region. Based on the increase or decrease in this error, the "learning progress" for the region is calculated. On each timestep, a list of candidate actions are generated. IAC then predicts which region it will end up in if a given action is taken, and based on this, chooses the action that will place it in the region with the most learning progress. Learning progress insures that the robot will focus on those aspects of its environment which are learnable but not yet learned. As Oudeyer et al. discuss, the use of learning progress has an advantage over simply focusing on regions in which the robot cannot predict what will come next: using the latter method a robot would be fascinated by unlearnably random features, such as "snow" on a television screen (2004). Every exemplar, each of which consists of a sensorimotor vector (the input vector) and the sensor vector seen in the next timestep (the output vector), is stored with its region. Thus, if IAC is run for a long period of time, the amount of memory it needs continues to grow. Oudeyer et al. ran IAC on a Sony Aibo robot which was constrained to a sitting position and could bite, bash or simply observe several objects in its environment. They found that the robot focused on different objects at different times, and the number of times which it successfully bit the one bitable object in its environment or successfully bashed the one bashable object tended to be greater in the later two thirds of their experiment, indicating that some sort of learning had occurred.

A number of researchers have explored the use of layered architectures in developmental robots. Blank, et al. use a resource allocating vector quantizer (RAVQ) governor to control how a prediction network is trained from sensor data as a robot performs a wall-following task (2005). They also use the output of a self-organizing map (SOM) as input to a prediction network in a goal-finding task. Nolfi and Tani suggest a hierarchy of alternating segmentation and prediction layers, where the representations formed on one layer become the input to the next (1999). The hope in constructing such multi-level architectures is that by building abstractions on top of abstractions the representational power of the robot will increase, allowing it to solve more complex problems.

2 Methods

2.1 Architecture

For this experiment, we wanted to combine the intrinsic motivation of IAC with the intelligent abstraction provided by GNG and the memory available from an Elman network. To do so, we first modified IAC to cluster its regions using a GNG. A diagram of how we incorporated GNG into IAC is shown in Figure 2.1. In the original IAC, regions were formed by starting with one large region and recursively splitting regions in two after a certain number of exemplars had been seen in that region. Oudeyer et al. use an algorithm that minimizes the variance of the output space points for the exemplars in each new region to decide how to classify exemplars after the split. In a previous experiment, we used k-means clustering to perform this split (Barry and Jones, 2006). Using a GNG network to form the regions has several advantages: GNG dynamically adds, deletes and moves nodes as needed to fit the input space, so it provides a more intelligent division of the input space than either of the two methods just mentioned, and because a region can be represented by the model vector for a GNG node, it is no longer necessary to keep all the exemplars ever seen. Also, in the original IAC experiment, the ”expert” in each region simply used the output vector of the nearest neighbor (of the current sensorimotor vector) as its prediction for the output of the current sensorimotor vector, while we use a neural network as an expert instead.

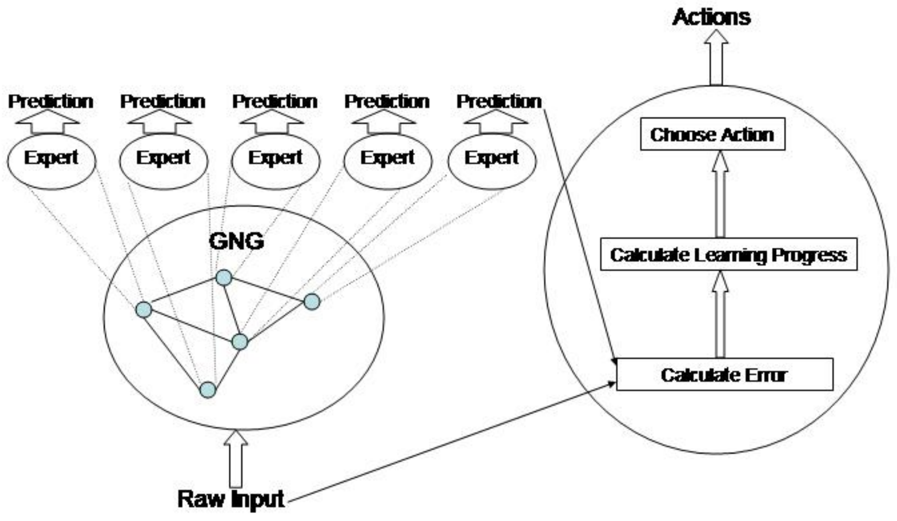

Figure 2.1: IAC modified to include GNG.

After modifying IAC, we added an Elman network which took as input the model vector of the current region. On top of this we added a second IAC/GNG combination, which took as input the activation of the hidden nodes of the Elman network. This complete architecture is shown in Figure 2.2. Because our architecture combines IAC, GNG and an Elman network, we dubbed it IGE. For the first level IAC, the input vector consisted of seven sensor values - four color sensors, forward, back, left and right, and three sonars, forward, left and right - as well as two motor values, adding to a total length of nine. The Elman network had input and output layers of size nine, and hidden and context layers of size four. The input vector for the second level IAC was four plus two motor values for a total length of six.

2.2 World

The task of learning any environment for which all features are not immediately visible is inherently time dependent. It is impossible for a robot to learn that continuing down a hallway will eventually lead to a turn and a differently colored hallway unless it has some memory of the first hallway by the time it gets to the second. To test our architecture, we constructed a world with four corridors with red and blue walls. A screenshot of the robot in this world is shown in Figure 2.3. To implement our color sensors,

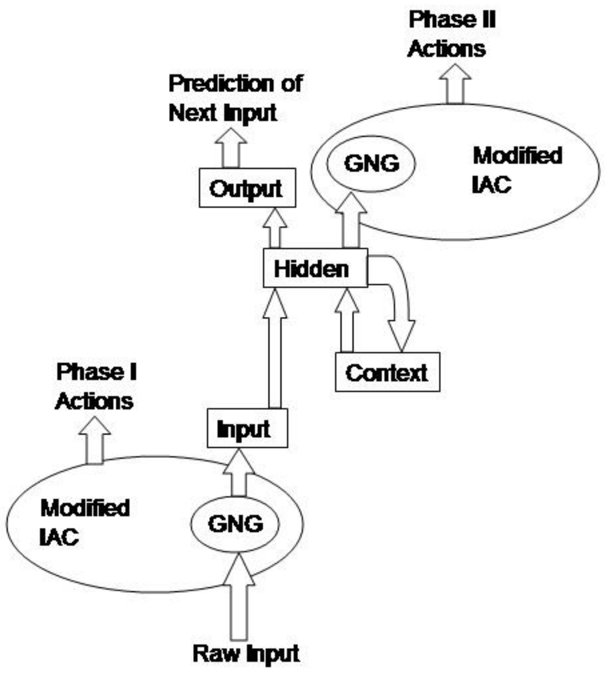

Figure 2.2: Architecture of IGE.

we used a camera with a 360 degree view, looked at the color of four points within this image and set a sensor value to one for red or zero for blue. In general this worked well, but occasionally in corners the image would be skewed such that two of the points picked from the image would be on the same wall. Also, because the camera occasionally returned an error, indicated by a value of 0.5 for all pixels in the image, we returned 0.5 for all color sensor values in this case. We allowed the robot to choose from five actions: move forward, move backward, turn left 90 degrees, turn right 90 degrees, or stay still. Due to an issue with the simulation software we used, the robot would occasionally get stuck inside or pass through walls. In order to prevent this from happening, we tested the robot's position on each timestep and bounced it back toward the center of the hallway if it was too close to a wall.

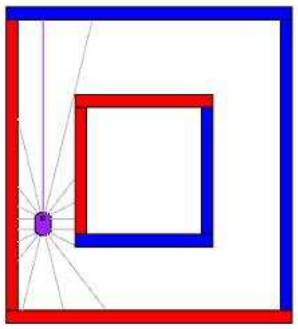

Figure 2.3: Simulated world.

2.3 Experiments

We ran each experiment for 10000 timesteps. At 1000 timesteps, after the GNG for the first level IAC had stabilized, we began training the Elman network. For the first 2000 timesteps, the robot's actions were controlled by the first level IAC, then control was shifted to the second level IAC. We also ran a simulation with a robot that randomly chose one of the five candidate actions on every timestep. This allowed us to determine if our robot was making choices that were significantly different from random. During each run, we collected data about the robot's position, its current region, its chosen action on each timestep, the regions formed thus far, and the error from each IAC and the Elman network. We also tried running this same experiment in a much smaller version of our world, however the results were not significantly different. The level of randomness for both IAC brains was 35%.

3 Results

We describe the results of our experiments. We ran many trials for this experiment, but the results were all similar so we present only the results and analysis for one representative trial. Because most of the data is too large to be put into tabular form, we mostly present graphs in this section. Occasionally, the graphs have had some analysis performed on them to make them easier to read, but in general we leave analysis for the next section. All trials were run for 10000 timesteps. We began training the Elman network after 1000 timesteps and passed control to the full IGE architecture after 2000 timesteps.

3.1 Position Data

We first just show the robot's location over 9000 timesteps. In Figure 3.1, the top row shows the IGE robot's position (time increases left to right). The bottom row shows the position of a robot running a brain that randomly picked one of the five possible actions. Each color/symbol pair in the graph corresponds to 1000 timesteps. For example, the red stars are the first 1000 timesteps in each subplot. The first subplot only shows the first 2000 timesteps, because this was the length of time the robot was controlled by only the lower-level IAC brain and not the full IGE architecture. Note that the random brain was run without the wall-bounce.

3.2 Monitor Data

We categorize how often the robot chooses each of the five actions (forward, backward, left, right, stay). To do this, over each 100 timestep period we collect the number of times in that period that the robot chose each action and plot the results in Figure 3.2. We do the same for the random brain in Figure 3.3.

3.3 Regions

We also wrote a script to allow us to visualize the regions that the lower-level IAC was forming (recall that the regions of the upper-level IAC correspond to hidden-layer activations of the Elman network and would therefore not be easy to visualize). For each region, we made a plot like the one in Figure 3.4.
In this figure, the robot is at (0, 0), positive y is forwards, negative x is left, and positive x is right. The length of the line corresponds to the sonar values - the longer the line the farther away the wall on that side. The color of the line is the color of the wall; if the value is not above 0.9 (red) or below 0.1 (blue), it is magenta. The star at (0,0) is the color of the back wall, as we have no sonar data in that direction. We also interpreted the motor values as one of the five actions and included that with the region. The example region, therefore, represents the robot not moving with free space on all sides, a blue wall ahead and to the left and a red wall behind and to the right. We did this same analysis for the full 24 regions at the 1900th timestep. We then looked at the GNG architecture to see which regions were connected. This is shown visually as arrows between connected regions in Figure 3.5.

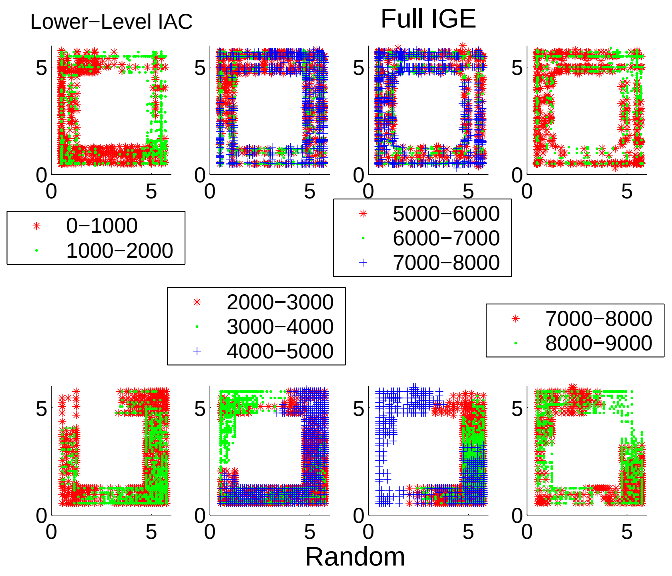

Figure 3.4: The robot's position in series of 1000 timesteps. Each color/symbol shows a set of 1000 timesteps. The top four figures are for an IGE brain (the figure on the farthest left is from the first 2000 timesteps before the full IGE brain takes over), while the bottom four are for a random brain run for the same amount of time.

3.4 IAC and Elman Error

Lastly, we present graphs of the error of the two IAC brains and of the Elman network over their lifetimes. Error is calculated as the sum squared difference between actual and prediction (note that this means that in general the highest possible error is greater than one). Because simply plotting the error is not very informative, we have also taken a moving average of the error with a window of 100 timesteps and plotted that over the plot of the error. In all of these figures, zero time corresponds to when that particular architecture began training. For the first level IAC brain, therefore, zero time is the beginning of the trial, for the Elman network, zero time is 1000 steps into the trial and for the upper level IAC brain, zero time is 2000 steps into the trial.

4 Analysis

4.1 Ability to Learn the Environment: Analysis of Lower Level IAC Error and Regions

An analysis of the lower-level IAC brain and the regions this brain formed show that the robot was able to learn something about its environment. First of all, we can clearly see from the moving average of the first-level IAC error, that the error in the robot's prediction drops dramatically over the first approximately 1000 timesteps. Of course, it does not drop fully because there is time-dependence in the environment that this architecture cannot predict.

We also see from the regions that the robot was able to correctly identify the binary nature of the color sensors. Of the 24 regions formed, 5 contained a magenta line, indicating an ambiguous color. Three of those, however, are entirely magenta and their color sensors are all approximately 0.5. We conclude, therefore, that these regions correspond to camera error. The remaining two regions are anomalous. They may be in the process of transitioning from camera error regions to color regions, but more likely, they are a result of the fact that the color data is coming from one camera. Therefore, right at a corner, the camera's cone can be wide enough that the robot will get two readings from the same wall and no reading from an adjoining wall. If the robot turns, it is possible for it to get a different color reading even

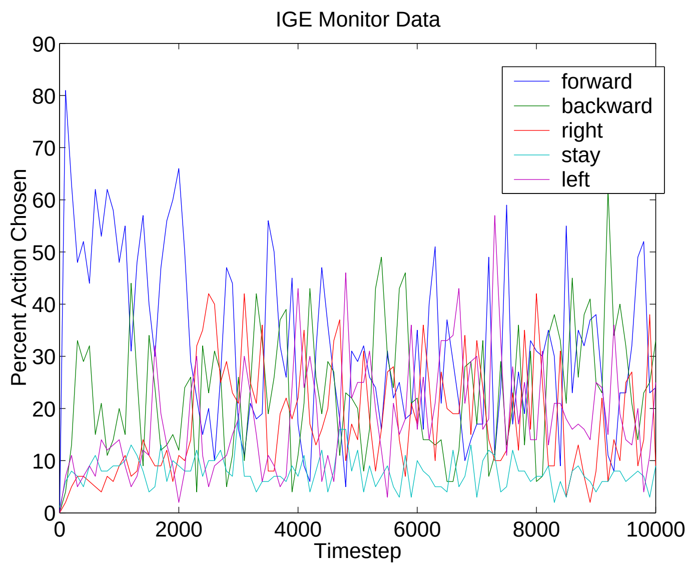

Figure 3.5: The number of times in periods of 100 timesteps the robot chose a certain action when running the IGE brain.

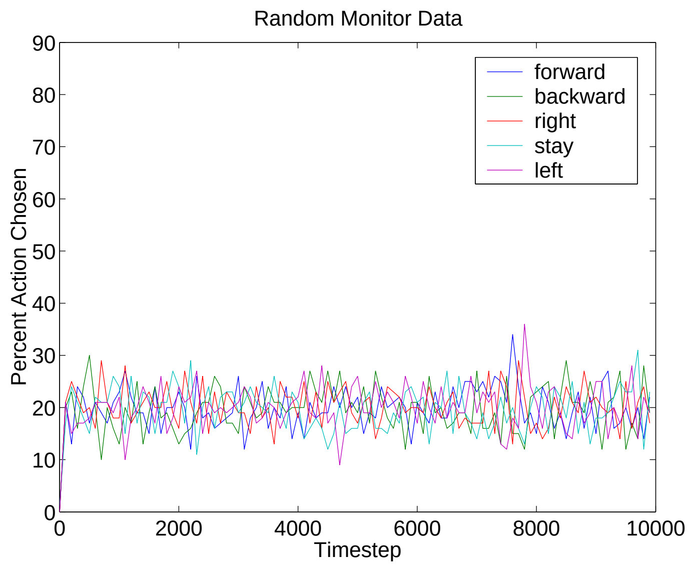

Figure 3.6: The number of times in periods of 100 timesteps the robot chose a certain action when running a brain that randomly chose actions.

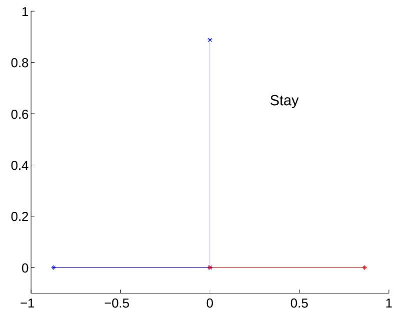

Figure 3.7: Example region (see text for explanation)

though it is technically in the same spot. This may lead to the confusion in the one wall we see in two of the regions. Because the experts for our regions are neural networks, these anomalous regions are not necessarily a problem; a neural network could easily learn that although it may see some exemplars with a red sensor reading to the right and some with a blue reading for the same sensor, different predictions should be made for each. The remaining regions are all regions that the robot would see in its traversal of the environment.

It is also interesting that so many of the regions have motor values corresponding to “stay,” especially considering how little the robot chose that action. This occurs because the sensorimotor vector contains seven sensors and two motors. Therefore, if we have two vectors in which all the sensors are the same, but the robot is moving in a different direction, the two vectors appear to be very close. The result is that the same sensor, different motor vectors get put into the same region and the motor values in that region end as an average over all the possible actions, which comes out to stay. Of course, since we are going forwards and backwards more often than anything else, we form regions for those too.

In addition, note that the robot formed only 24 regions due to the fact we are using equilibrium GNG to cluster regions rather than k-means clustering. By this timestep, k-means clustering would have approximately 200 regions. Since IAC needs to look through regions on each timestep, using GNG instead of k-means clustering is a significant improvement in time as well as in space.

We can also look at the clusters GNG formed. Once again, these appear reasonable. In general, especially in the two smaller clusters, regions with approximately the same sensor data but slightly different motors all wind up interconnected. The same is true in the larger cluster. The camera error regions provide a link between the unrelated sub-clusters in this cluster because their color data is all 0.5’s, meaning that it is right between the 1.0’s and 0.0’s of the other walls. Note that the camera error regions are all interconnected in the middle of the cluster as we would expect. The “top” of the cluster is all red walls on the right and behind and blue walls on the left and ahead. The bottom of the cluster is a little more confused because it includes the two regions that have one wall of ambiguous color. However all of the individual links are reasonable and the clusters are close to those a human might make.

The robot forms a reasonable number of sensible regions in this environment and, judging from the reduction in error of the IAC brain, is able to learn the non-time-dependent tasks. From the regions formed and the decrease in IAC error, we conclude that, as expected, the robot is learning the long corridors at its lower level.

4.2 Exhibiting Curiosity: Analysis of Monitor and Position Data

Looking at the results of the monitor and position data, we can see that the IGE architecture is not acting randomly. To make this clearer, we re-plot the monitor data from IGE and random together on the same plot in Figure 4.9.

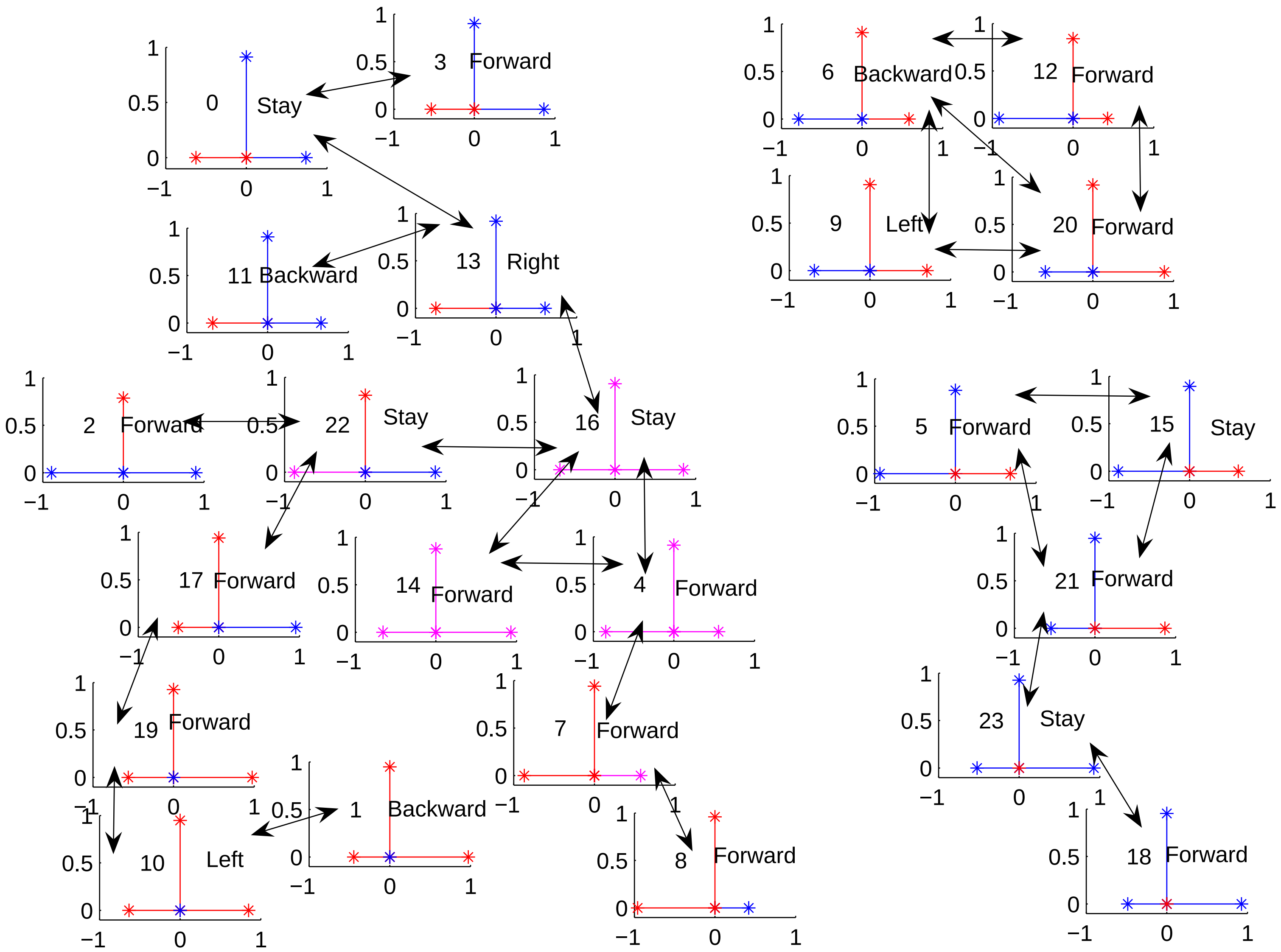

Figure 3.8: The regions formed by the lower-level IAC brain at the 1900th time step. The arrows show the connections made by the GNG algorithm.

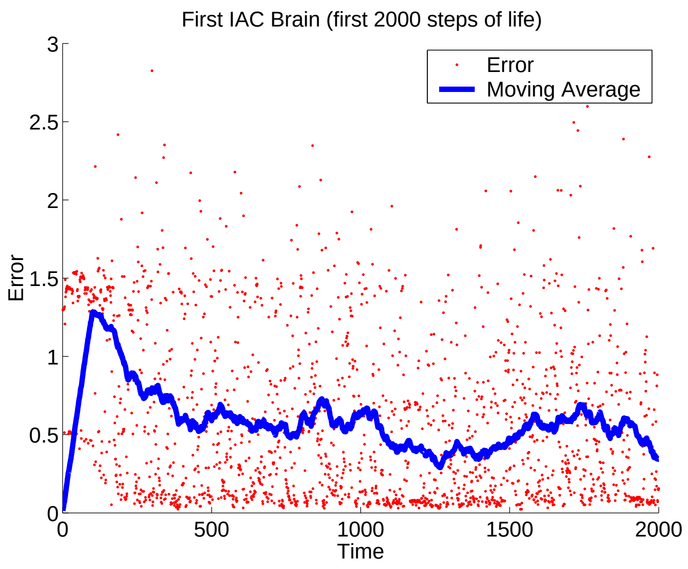

Figure 3.9: The error in the lower level IAC brain for the first 2000 timesteps.

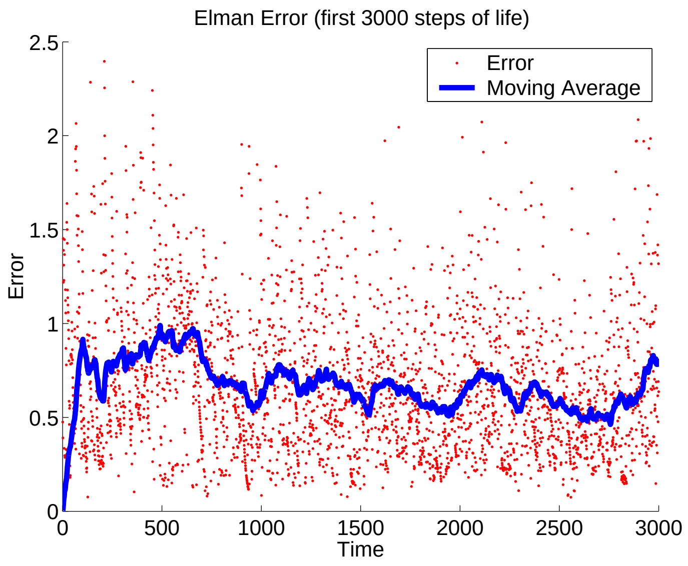

Figure 3.10: The error in the Elman network for the first 3000 steps of its life (note that this corresponds to steps 1000-4000 in the trial run).

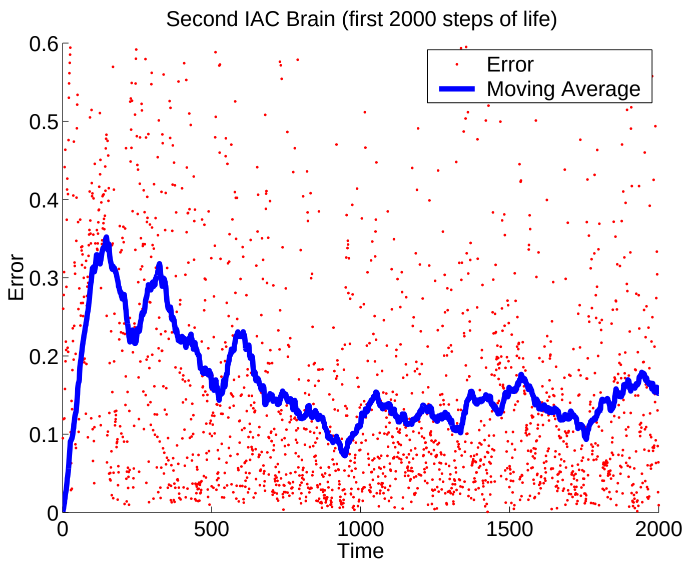

Figure 3.11: The error in the upper level IAC brain for the first 2000 timesteps of its life (note that this corresponds to steps 2000-4000 in the trial run).

From these plots, it is clear that IGE chooses to go forward significantly more often than the random brain, especially in the first 2000 timesteps and never, in all 10000 timesteps does it choose to stay in the same place as often as random does. Even later in the trials, when the IGE data more closely matches random data, the spikes and dips in IGE are more dramatic than in random. This corresponds to a tendency in IGE to keep doing what its doing while it learns. If it is making learning progress going forward, it will go forward more often. In addition, right at the end (above about 6000 timesteps), it is probable that IGE has learned all it can and therefore not surprising that its behavior appears a little more random.

Looking at the position data, we see that, where the random brain almost never manages to traverse the entire square in 1000 timesteps, the IGE brain rarely fails to. In addition, although this is less marked, while the random brain spends an equal amount of time in each point in the environment, the IGE brain tends to spend more time at the corners.

We should also point out here the line that appears to develop down the middle of the square in the latter 5000 timesteps of the IGE brain. This is an artifact of our mechanism that keeps the robot in the square; if the robot gets too near a wall, it is "bounced" to a point farther away from the wall. Since the robot can only move a discrete distance in a timestep, after a while there are places it is very unlikely the robot will ever be. This is not seen in the random data because we did not run that brain with the wall-bounce.

Both sets of data show that IAC is exhibiting "curiosity" about its environment. It manages to explore the environment more fully than could possibly be accounted for simply by random action. In addition, it prefers, especially at first, to go forward. The effects of turning in a hallway are easy to learn, while the changing colors makes traversing the hallways by going straight more interesting. The fact that it does not often choose to stay in one place also shows curiosity; there is very little to be discovered by staying in one place. Lastly, the IGE preference for corners is another manifestation of its curiosity. The corners are the most "interesting" thing in the environment and therefore IAC spends the most time with them. Thus we show that IAC can exhibit curiosity not just in a situation in which the robot and environment are static, but also in a static environment where the robot is free to move.

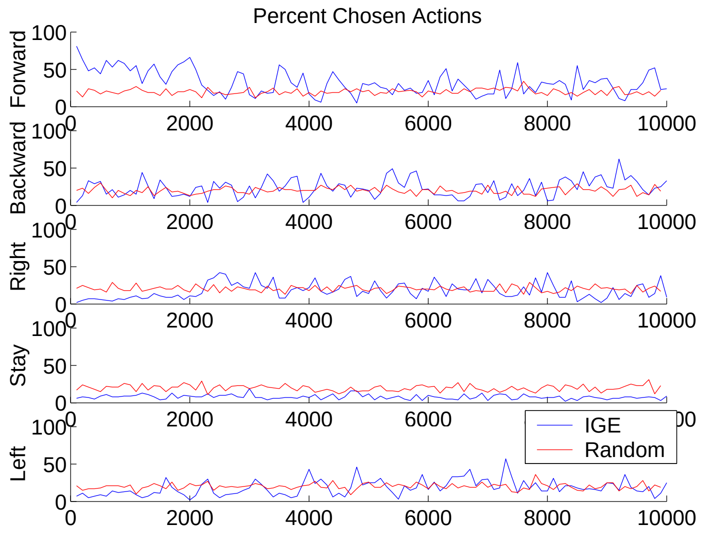

Figure 4.12: The number of times in 100 timesteps the brain chose a certain action. Each subplot shows the data for a different action.

4.3 Time-Dependence: Analysis of Second-Level IAC Brain and Elman Error

From the graph of the error of the second-level IAC brain, we see that it was able to learn something about the hidden activations of the Elman network. However, the Elman network does not appear to have learned the environment very well. It has a small initial decrease in error around 1500 timesteps (when the Elman network has been alive for 500 timesteps) and then possibly again at 2000 timesteps (Elman life 1000) when the second IAC brain takes over. However, these are not of the magnitude we should see if the Elman network was truly learning the environment.

In an attempt to discover how well the Elman network and the second-level IAC brain were able to learn the environment, we saved all the weights for the Elman network, the weights for the IAC experts and the IAC regions at 1900 timesteps (Elman network has been learning for 900 timesteps but the robot is still controlled by the lower-level brain) and 6000 timesteps (robot has been controlled by the full architecture for 4000 timesteps). We then loaded these and ran the robot once around the square in a clockwise direction starting from the lower left-hand corner of the square.

We saved the region the robot was in at each timestep and we show these regions by plotting the number of the region at the position of the robot in Figures 4.10 and 4.12. The regions are simply numbered in the order in which they are stored in the GNG, (which, for the most part, is the order in which they were created), and a guide to which region matches which number is given in figure 4.11 and 4.13.

As we can see from figures 4.10 and 4.12, the robot correctly identifies which region it should be in except in the left hallway. This was actually an on-going problem in all of our runs. For some reason, the robot failed to form a region that matched going forward in the center of this hallway well. However, although the color sensor data is not always correct, the sonar data does match in these regions.

Notice also that at the corners, the region the robot is in generally changes since it sees a different wall on one side. This is the time dependence of the task. In addition, in both tests, we can see that there are times when the robot is in the region corresponding to camera error. Camera error is the random part of the task, as it is impossible to predict when the camera will return an error.

We also include a figure that corresponds to the regions the robot was in for the second-level IAC brain at 6000 timesteps. As these regions correspond to hidden unit activations of the Elman net, they are not so easy to visualize, but we can still analyze the patterns of the regions.

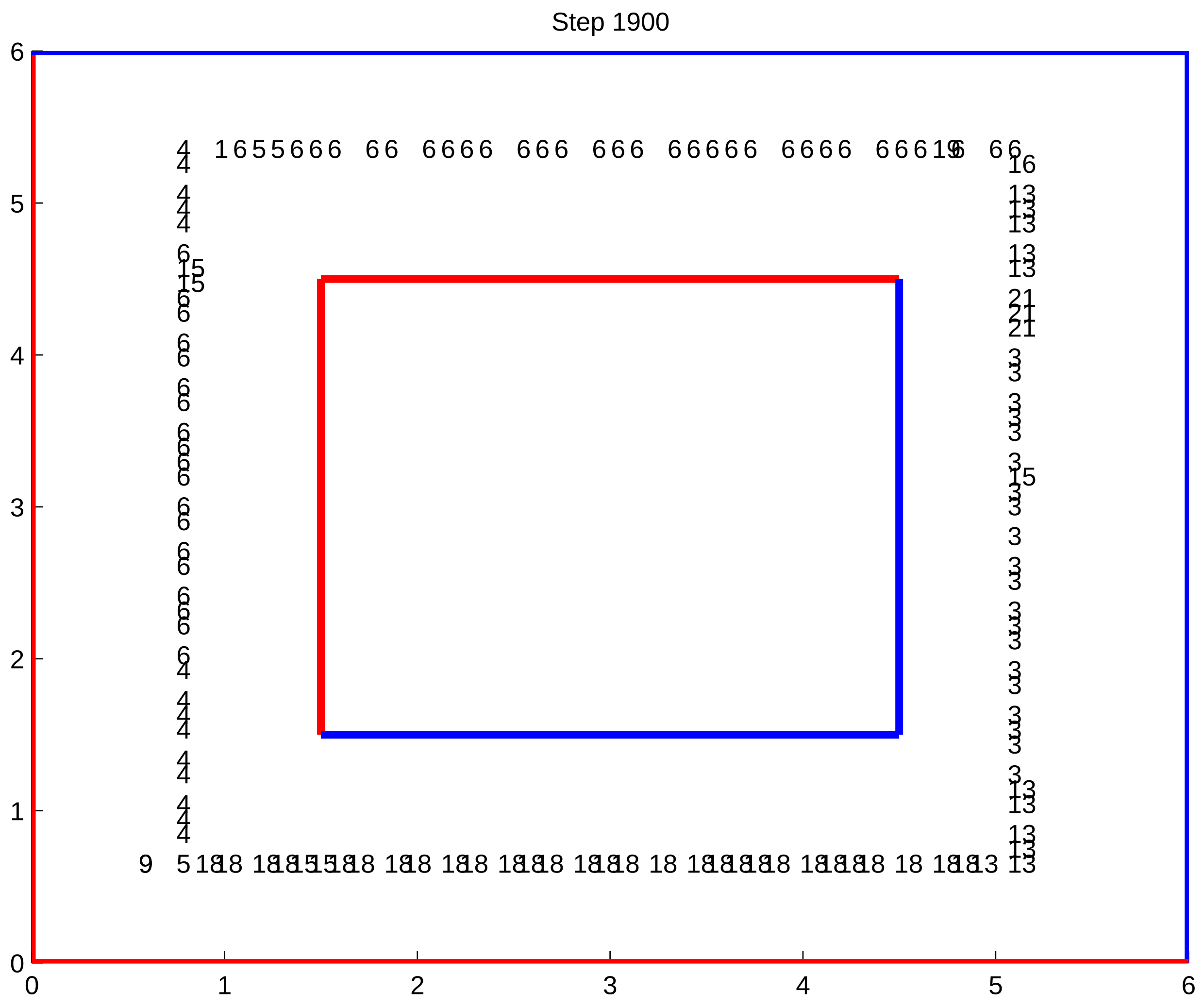

Figure 4.13: The robot was run once clockwise around the entire environment using the weights and regions from the 1900th time step. The regions it placed itself in at each timestep are shown in this figure (the bottom hallway is 18 if that is hard to see).

We do not see exactly what we might hope to see in Figure 4.14. Were the robot learning time dependence, the regions in the hallways should all be the same (as they are) but then, right before the corner should the regions should change, indicating that the robot is about to see a change in color. Instead, we see a pattern similar to the lower-level IAC brain regions, indicating that the Elman network has learned the long hallways but not learned to predict the corners.

We can also plot the error in the IAC brains and the Elman network as the robot traverses the square. We plot the error in the lower level brain for the 1900th timestep and the upper level brain for the 6000th timestep since this corresponds to which brain was controlling the robot. The robot turned (i.e., saw a corner) every 35 timesteps. These plots are given in Figures 4.15-17.

Clearly the largest error for the IAC brains and the Elman network occurs at the corners. We would expect that at 1900 timesteps the lower-level IAC brain would be unable to predict the corners. This is precisely what we see in Figure 4.15. Along each hallway, the error is approximately constant, but at the corners there are large spikes in error. In addition, we note the spikes at the end of the first hall and the middle of the third - these correspond to camera error and will always be unpredictable.

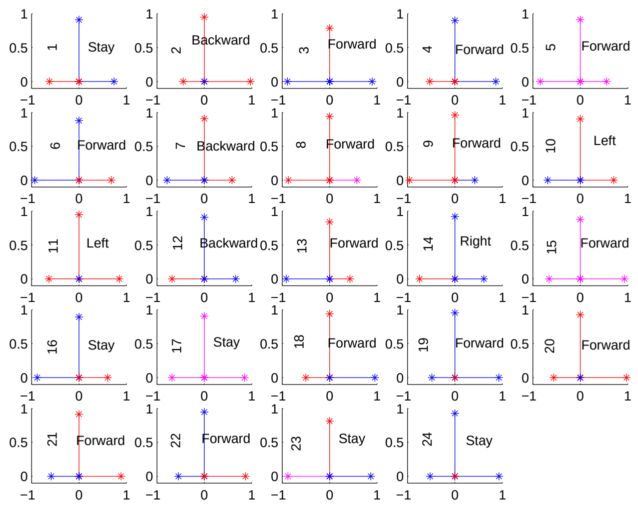

Figure 4.14: Region/Number key for timestep 1900.

We had hoped that at 6000 timesteps the second-level IAC brain would be able to predict the corners. The graph of the error of this brain, however, shows otherwise. It is similar to the graph of the lower-level brain, in that it is flat in hallways and spikes in corners, indicating that the second-level brain was *not* predicting corners correctly.

Looking at the error in the Elman network, we find an explanation for the failure of the second-level IAC brain to predict the corners. From the graph, it is clear that the Elman network itself never learned the pattern of the world. Since the Elman network would have to be able to predict the corners in order for the IGE architecture to do so, we conclude that the IGE architecture is unable to learn this time dependence because the Elman network was unable to do so.

4.4 Reasons for the Failure of the Elman Network

The Elman network may fail to learn the environment for a number of reasons. The most obvious of these is that our environment was simply too large. Even if the robot went straight down the hallway, the Elman network would get 25 identical inputs before seeing something different. In addition, the 35% randomness of the IAC brain might also make it hard for the Elman network to learn.

In order to address this problem, we shrank our world so that a hallway took only approximately 10 timesteps to traverse. We also gave the lower-level IAC brain 3000 steps to train and trained the Elman for the last 2000 of those steps. In addition, once we began training the Elman network at 1000 timesteps, we lowered the percent of random actions to 5%. We then ran the robot once around the environment using the data from 7000 timesteps and a graph of the Elman network error at is given in Figure 4.18. Note that now corners occur every 10 timesteps.

In Figure 4.18, we see spikes at 10, 20, and 30 timesteps showing that the Elman network was unable to learn even this smaller, less random world. Therefore, it is likely that the size of the world was not actually the issue. Rather we need to consider how the lower-level IAC brain works.

The IAC brain attempts to find places of interest in the environment. Therefore, it is going to spend more time in "interesting" locations, such as the corners and less time in the corridors. The result of this is that the Elman network does not get sequences that are distributed well. It sees a lot of turning at the corners and a little moving down the corridors, but very little moving down a corridor, turning a corner, and moving down another corridor, which is the kind of input it needs to learn. It is possible, therefore that the curiosity of IAC is keeping the Elman network from fully learning the world.

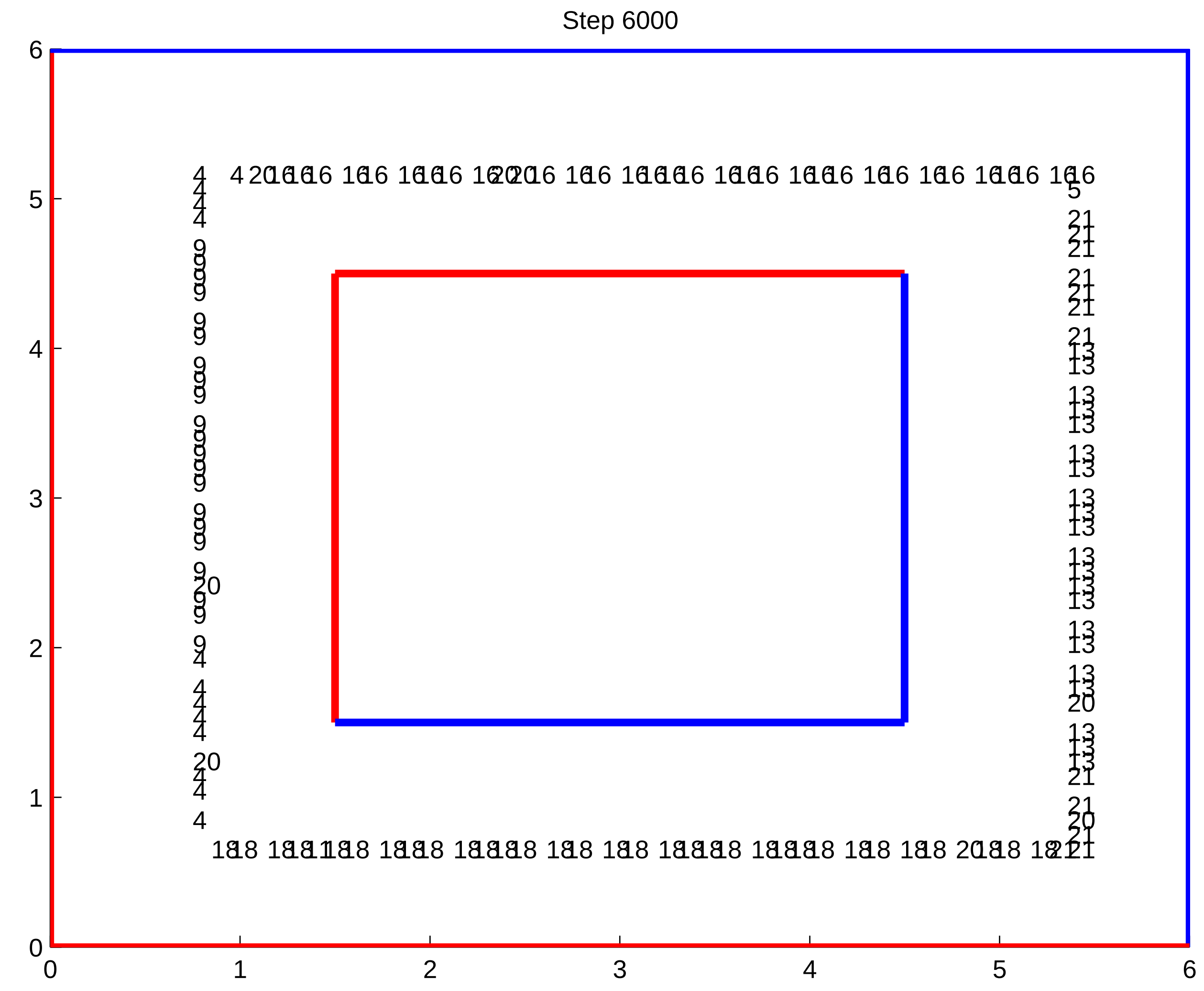

Figure 4.15: The robot was run once clockwise around the entire environment using the weights and regions from the 6000th time step. The regions it placed itself in at each timestep are shown in this figure (the bottom hallway is 18 and the top is 16 if that is hard to see).

To circumvent this problem, we could try training an Elman network offline and then put that network, ready-made, into our architecture. However, that starts to put our own biases into the architecture. Rather than letting the robot explore for itself, we are telling it where to go to train the Elman network How are we to determine where the robot should go and what it should see? We are starting to manipulate the world in unrealistic ways.

Therefore, what we need is either a method of exploration for the robot that gives more structured coverage of the environment while not losing the concept of "curiosity" entirely or a method of learning time dependence that does not required such structured input. The combination of pure curiosity and Elman networks has not led to an architecture that can learn time-dependence in a dynamic world.

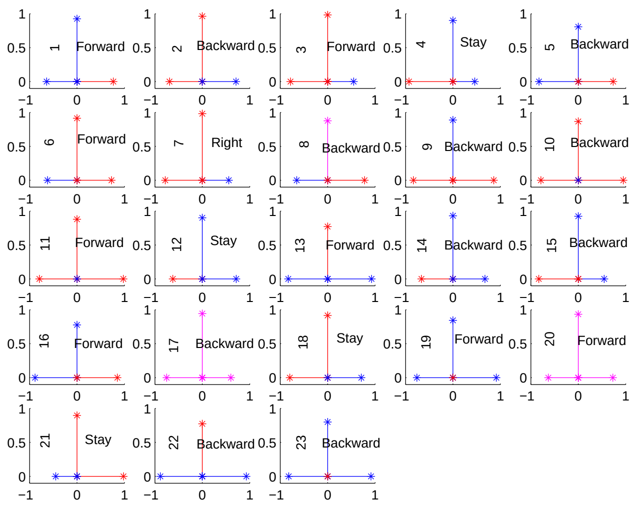

Figure 4.16: Region/Number key for timestep 6000.

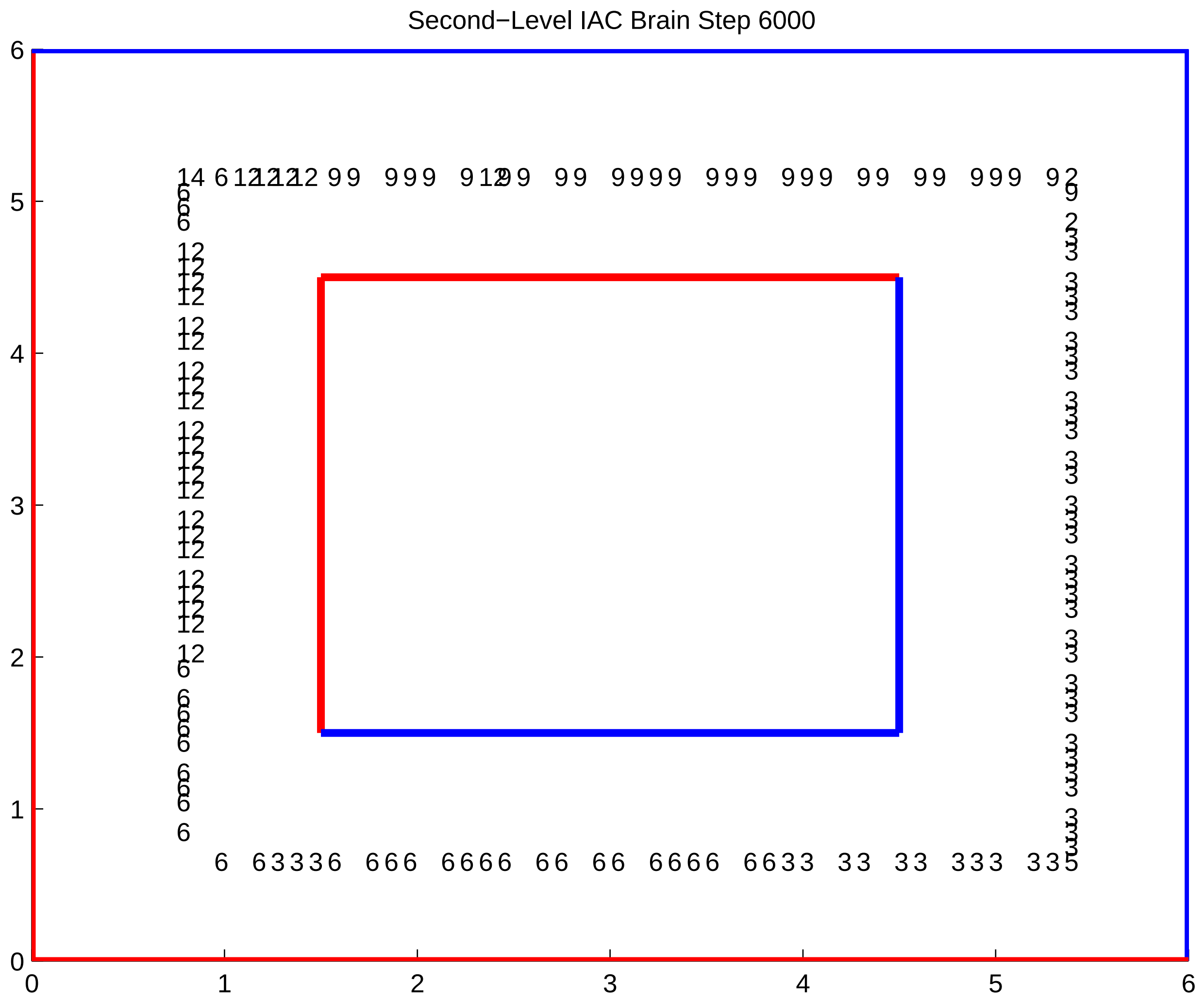

Figure 4.17: The robot was run once clockwise around the entire environment using the weights and regions from the 6000th time step. The regions it placed itself in at the second level IAC brain for each timestep are shown in this figure

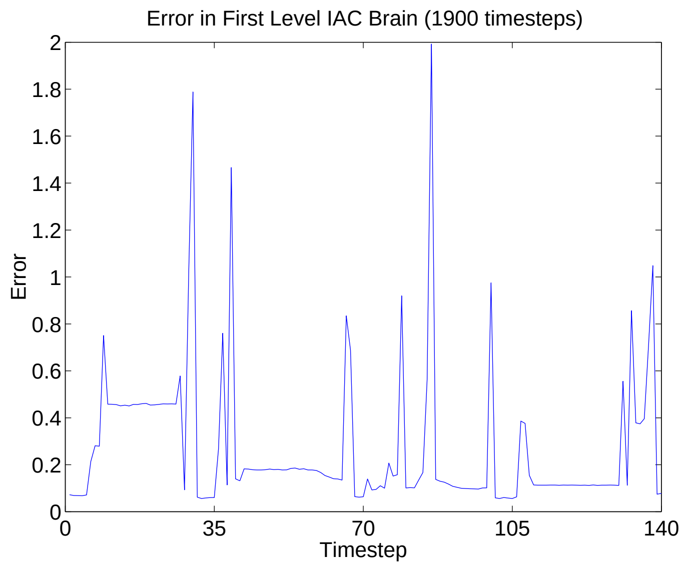

Figure 4.18: The robot was run once around the entire environment using the weights and regions from the 1900th time step. The error of the prediction of the first-level IAC brain is shown. Corners occur every 35 timesteps.

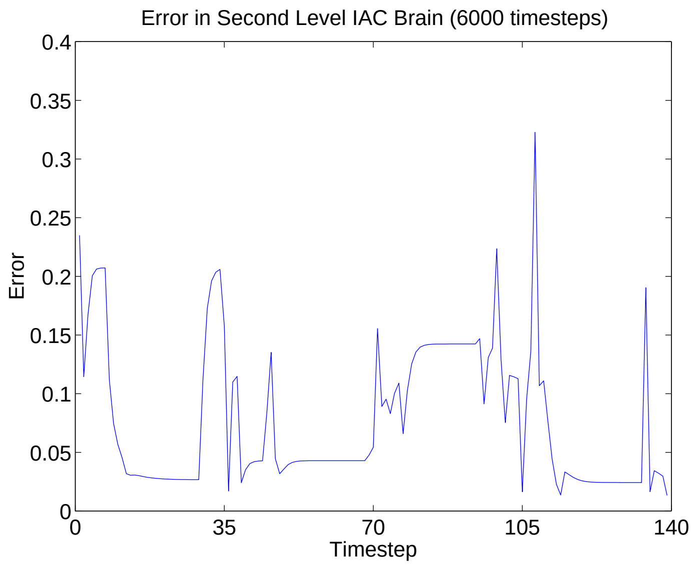

Figure 4.19: The robot was run once around the entire environment using the weights and regions from the 6000th time step. The error of the prediction of the second-level IAC brain is shown. Corners occur every 35 timesteps.

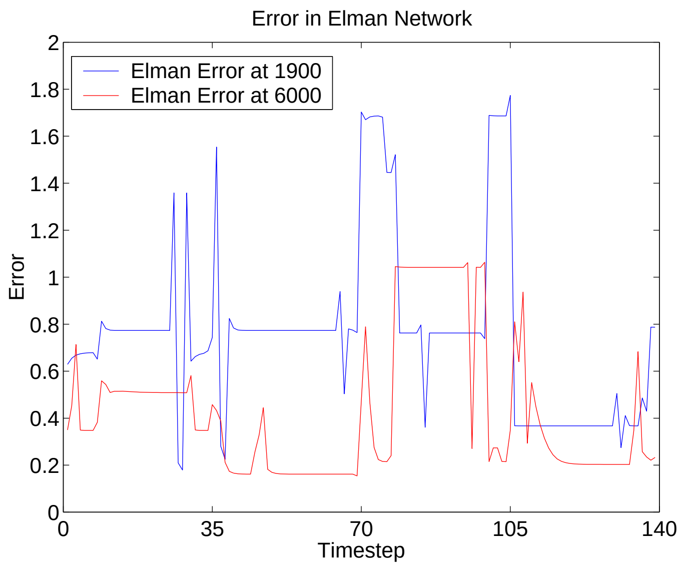

Figure 4.20: The robot was run once around the entire environment using the weights and regions from the 1900th and 6000th time step. The error of the prediction of the Elman network for both of these runs is shown. Corners occur every 35 timesteps.

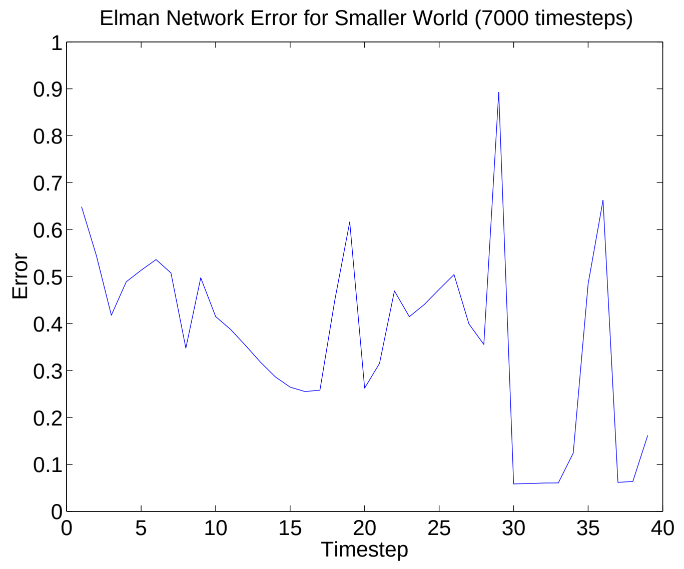

Figure 4.21: The robot was run once around the entire smaller environment using the weights 7000th time step. Corners occurred every 10 timesteps. The error in the prediction of the Elman network is shown.

5 Conclusion

Our results for the combination of IAC with GNG are very encouraging. The regions formed by the GNG are meaningful, and there are far fewer regions than would have been formed by k-means clustering. It is also clear that the behavior of the IAC/GNG is significantly different from random. Our results for the full IGE architecture are a little less clear. Although the error in the Elman network does decrease, both when it first begins training and when the second-level IAC takes control, it never got as low as we would expect if the network had actually learned the environment. In addition, the error spikes at the corners when the robot was driven around the world indicate that the Elman network did not successfully learn these features. There are several reasons why this might have occurred. The time sequences present in the world may have been too long for an Elman network to learn (it took about 35 timesteps to travel from one corner to the next). Alternatively, the Elman network used in this experiment may simply have simply been too small, with only four hidden and four context nodes, to learn this world. Also, because the both IAC brains had a choice of five different actions, and 35% of the time that choice was made randomly, the robot may never have moved in such a way that the Elman network could see a time sequence corresponding to driving directly down a hall and around a corner. The second level IAC brain is able to form meaningful regions, to the extent that within a given hallway it tends to stay in the same region, however, since the Elman network never learned the time-dependent aspects of the environment, the higher-level IAC cannot act in a time-dependent manner. Perhaps if the training for this architecture was altered such that the first layer IAC was trained, then the Elman network was trained while the robot was driven along a specified path, then the second layer IAC was trained, IGE would be more successful. There are certainly many ways in which the parameters of our architecture could be adjusted that might make it more successful. This experiment has clearly shown that, in the construction of any multi-layer architecture, the way in which the behavior of one layer affects the performance of subsequent layers must be very carefully considered.

6 References

Barry, J. and H. Jones (2006) Midterm paper, Adaptive Robotics course, Swarthmore College.

Blank, D., et al. (2005) Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture. Cybernetics and Systems, 36 (2).

Elman, J.L. (1990) Finding Structure in Time. Cognitive Science, 14, pp. 179-211.

Fritzke, B. (1995) A Growing Neural Gas Network Learns Topologies. In Tesauro, G., Touretsky, D. S., and Leen, T. K. (eds), Advances in Neural Information Processing Systems 7, Cambridge, MA. MIT Press.

Nolfi, S. and Tani, J. (1999) Extracting Regularities in Space and Time Through a Cascade of Prediction Networks: The case of a Mobile Robot Navigating in a Structured Environment. CScience, 11 (2).

Oudeyer, P.-Y., et al. (2004) The Playground Experiment: Task-Independent Development of a Curious Robot. American Association of Artificial Intelligence.