Applying Predicted Information Gain to Physically Embodied Agents

Alexandra Nilles*
Complexity Sciences Center, UC Davis Physics Department
Advisor: Dr. Jim Crutchfield

September 30, 2014

Abstract

"Predicted Information Gain" (PIG) is an approach to maching learning, developed first by the Redwood Center for Theoretical Neuroscience, which builds a generalized internal model of agent-evironment interactions. This paper outlines how PIG was implemented with hardware from the Robotic Multiagent Development System (RoMADS). Most machine learning uses a specific task-based reward function, but the PIG approach is a more general model of learning and exploration. Our results show that PIG is effective at building consistent models, but for very simple environments, is not significantly more efficient than randomly chosen actions. Analysis was done on the resulting models and also on the accuracy of the algorithm's predictions about information gain. Suggestions are also given for future research.

Contents

*Physics Department, Colorado School of Mines, contact at alli.nilles@gmail.com

1 Introduction

In machine learning, most advances are made in training computers to perform specific tasks - tasks which can range from playing chess, to driving a car, to making us coffee in the morning. However, important research is also being done in the field of exploratory machine learning - investigating algorithms for agents that are exploring unknown environments. This idea has significant overlap with theoretical neuroscience, and this paper will draw heavily on research done at the Redwood Center for Theoretical Neuroscience at UC Berkeley [1].

The usefulness of such research in neuroscience is to understand and model how intelligent agents learn and make decisions when exploring new situations. Understanding this process could lead to advances in nearly all machine learning applications, as well as increasing our understanding of human thought processes.

In the Complexity Sciences Center at UC Davis, the research focus is on generalizing the idea of what it means to have structure in data, as well as how to describe this structure using computational techniques (such as epsilon machines). Physicists are nothing more than agents using their basic sensors and actuators to build tools to understand the universe, and generalizing a theory of this process contributes to all physics research. Information-theoretic machine learning also hints at the possibility of automating large parts of the physics research process. More prosaically, this paper will focus on how a specific learning algorithm (predicted information gain, or PIG) performs when applied to exploratory robots. These agents were tested with various environments, and options for extending this research were explored.

2 Overview of PIG

It will be helpful to define some terms before explaining PIG:

- Agent: the part of the system which is making decisions and collecting data.
- Action: how the agent interacts with its environment.
- State: what the agent is able to observe about its environment.
- Internal Model: the representation that the agent makes of its environment

The basic motivation of the PIG strategy is to choose actions that will maximize the information added to the agent's model of the world. To calculate this, we must define a probability distribution, which we will call Θ_{sa}. This is a distribution over all possible resulting states for an agent that is in state s and chooses action a. If we want to compare two of these distributions, noted as Θ_{sa} and $\hat{\Theta}_{sa}$, we can use the Kullback-Liebler divergence (D_{KL}) from information theory [3]:

$$D_{KL}(\Theta_{sa}||\hat{\Theta}_{sa}) := \sum_{s'=1}^{N} \Theta_{sas'} \log_2(\frac{\Theta_{sas'}}{\hat{\Theta}_{sas'}}) \tag{1}$$

s' represents states that can result after action a is taken in state s. The important characteristic of the KL-divergence is that as distributions become more similar, the ratio inside the log will approach one, and the divergence will approach zero. Thus, the KL-divergence between two identical distributions is zero, and is positive and larger as the distributions become more different.

The agent's internal model is the collection of all Θ_{sa} for all possible states and actions. To compare two internal models, we define the missing information (I_M), which is simply the sum of divergences over all states and actions:

$$I_M(\Theta||\hat{\Theta}) := \sum_{s,a} D_{KL}(\Theta_{sa}||\hat{\Theta}_{sa}) \tag{2}$$

It is important to note that internal models can only be compared if they have the same dimensions in state and action space.

The actual "Predicted Information Gain" algorithm uses missing information as a way to quantify how much information will be gained by certain actions. When the agent is in state s, it constructs a different

hypothetical model for each action a and state s^* that could result from that state. For each action, the agent performs the calculation in Equation 3 to calculate the predicted information gain for that action.

$$PIG(a, s) := \sum_{s^*} \Theta_{sas^*} D_{KL}(\Theta_{sa}^{s \to s^*} || \Theta_{sa}) \tag{3}$$

This calculation is essentially the same as finding the missing information between the agent's current model and all the hypothetical models, except that each hypothetical scenario is weighted by Θ_{sas^*}, the probability of that hypothetical outcome actually occurring. In this way, the agent balances outcomes that are unlikely and high in information gain with outcomes that are very likely but thus low in information gain. This is related to the exploration vs. exploitation characterization of strategies in machine learning [4].

The agent chooses the action that maximizes PIG. This action results in the agent then observing one of the predicted states. The state-action-resulting-state combination is recorded as data, and this data is used to update the probabilities in the internal model of the agent. The PIG algorithm uses Bayesian updating to change the probability distributions in the agent's internal model. Bayesian updating is a method where the agent starts with a prior probability distribution, chosen by the designer of the agent, and that prior is updated by the data collected through successive actions. If a very inaccurate prior distribution is chosen, it can dramatically change the effectiveness of PIG [1]. Specific details of the prior choice and updating methods will be given in Section 3.2.

3 Implementation

3.1 Existing hardware

The goal of this project was to implement PIG with existing hardware from a previous project [2]. See Figure 1 for an image of the robot used. The robot has four binary light/dark sensors, one on each corner of the robot, pointing down. These sensors were used to interact with various environments, which consisted of dark lines printed on white paper. If the robot travels forward, and moves over a dark line, it is hardwired to stop. This means that the dark lines serve effectively as "walls" while the light paper serves as empty space that the robot can move through.

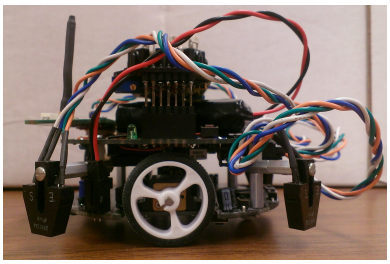

Figure 1: A photograph of the robot from the side. One of the wheels is visible, along with two of the visual sensors (the black objects in the same plane as the wheel).

The robots were pre-programmed with four possible actions: moving forward for a set length of time, turning 90 degrees to the right, turning 90 degrees to the left, and turning 180 degrees (about face). Because there are four binary sensors, there are sixteen possible states for the robot to be in, where a state is defined as a four-digit binary number representing the concatenated readouts of the sensors. Some of these states, such as all sensors reading black, are impossible or very unlikely, but for the sake of simplicity, all states were included in calculations.

3.2 Choice of Prior

Per the suggestions of the Redwood group, we began with a uniformly distributed prior probability distribution. That is, the robots started out by choosing actions based on the belief that any action, taken from any state, is equally likely to lead to any other state. We chose a prior distribution with the form of a Dirichlet distribution, as this has a closed-form solution for calculating the resulting transition probabilities conditioned by data. See the appendix of [1] for a more detailed analysis of different priors. Essentially, the Bayesian estimate for transition probabilities depends only on $F_{sas'}$, a count of the number of times the transition from s to s' under action a has occured; α, a parameter that can be tuned; and N_s, the number of neighboring states. The Bayesian estimate is given by:

$$\Theta_{sas'} = \frac{F_{sas'} + \alpha}{N_s\alpha + \sum_{s^*} F_{sas^*}} \tag{4}$$

In the limit of large amounts of data, this formalism approaches an intuitive ratio: the likelihood of a transition is the number of times a specific transition was observed, divided by the total number of transitions observed. Using this formula, we are able to update the internal model of the agent, and construct a model of state-action-state transition probabilities from the data that the agent observes.

3.3 Unique features of this implementation

There are several features of this project that differ from previous (computational) implementations of PIG. The implementation by Little & Sommer [1] examined three different environments - mazes, dense worlds, and 1-2-3 worlds. Mazes represent a spatial setup, where states represent a specific position in the maze. Dense worlds consist of states that can all transition to each other with different probabilities depending on the action chosen. 1-2-3 worlds are similar to dense worlds, but depending on the action, either one, two, or three states are accessible from each state (see [1] for a more detailed explanation).

In our implementation, our state space combines some features from dense worlds and mazes. To a human observer, the setup looks like a very simple maze - for instance, one environment is a rectangle formed from black tape with all white inside. The robot navigates around inside this rectangle. If the robot were able to detect its position in space, this would be identical to the maze setup. However, all the robot can sense is which of its four sensors are touching a wall. In theory, there is nothing inherent about the sensors that stops any sensor state from being followed by any other sensor state in time, so our environment looks more like a dense world, where the 16 sensor states are all connected to each other.

The unique, and complicating factor of this implementation is that the probability of changing from one state to the other depends on the agent's position in space, which is a hidden environmental variable.

Another unique part of this implementation is the fallibility of physical hardware - for instance, the two wheels on the robot were controlled by separate motors, and if the motors weren't warmed up, one wheel would turn noticably faster than the other, causing the robot to turn instead of moving in straight lines. Sensors sometimes return false readings. These, and other hardware challenges, test the robustness of PIG to false data.

The final unique challenge with physical implementation is that we do not have a pre-designed "true model" to compare to the internal model generated by the agent, as the original Redwood paper did. Thus, we must find other ways to confirm that the generated internal models actually do correspond with reality.

4 Results

The robot, using PIG to control its actions, was tested many times. Three different environments were used - a rectangle, triangle, and circle. The environments were normalized by having the longest possible straight path within the environment be the same length for each. The robot navigated the white space bounded by black tape in these shapes. Usually, the robot ran for 20 minutes. Longer trials showed that the internal model ceased to change significantly with more data collection (see Section 4.2).

There were four main topics that I analyzed - first, what is a good way to represent the internal models generated by the robots? In general, these are three-dimensional matrices with 1024 entries (16 start states, 16 end states, 4 possible actions). A way to visualize this data and extract useful information needed to

be found. Second, I needed to confirm that PIG was producing converged internal models, and find a way to quantify how the models changed over time. Third, I chose to analyze the accuracy of PIG - does the amount of information that an action is predicted to gain the agent correspond to the amount of information that is actually gained? Fourth and finally, I analyzed the differences between models generated by different environments, to see if the agents are actually extracting information about the structure of their environment from the data they collect.

4.1 How to Represent Internal Models

I chose to represent the internal models graphically, by representing the sensor states as nodes and the transition probabilities as edges. If all data was included, these graphs would look like 16 nodes, with all nodes connected to all other nodes - in other words, not very easy to understand.

To trim the graph, first I noticed that many of the state/action combinations had not been visited very often, making the data learned from those states not very useful. By taking the state-action pair which was visited the most, and calculating a cutoff of 10% of that frequency, we can make an intrinsic measurement of frequent states. To trim the graphs, I dropped the states (nodes of the graph) that had a frequency of less than 10% of the maximum frequency. I also dropped edges that had a less than 10% probability associated with them. The resulting graphs looked like Figure 2. We can see in Figure 2 that only nine of the 16 states had high enough frequencies to be included. Also, many edges were dropped - for example, the states in the top left and right corners of the graph only have one incoming edge and no outgoing edges. This means that under the "turn right" action, either the likeliest resulting states were not visited very often, or the distribution between the 16 possible resulting states was uniform enough to drop each transition probability below 10%. It is important to note that each graph corresponds to only one action, so four graphs can be generated for each internal model.

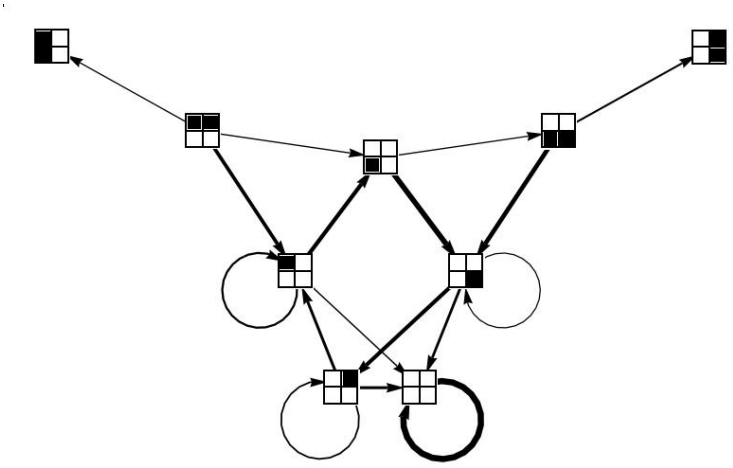

Figure 2: Graph of transitions for "Turn Right" action in the rectangle environment, with probabilities $< 10\%$ not shown, and infrequently visited states not shown. Edge weight is proportional to transition probability.

Several things are notable about this type of representation. First of all, we can do some "sanity checks" to make sure that the model makes physical sense. For example, since the robot turns right in place, we would expect that if the robot is in the state with all sensors white (meaning it is not touching a wall), and it turns right, it should remain in that state. This is what we see in the lower right hand corner of Figure 2, where the all-white state returns to itself with high probability. The rest of the transitions pass similar sanity checks - if the robot has one or two sensors on a wall, and rotates, there are certain resulting states that are expected to follow and we see that the graph reflects these.

It is important to note places where the graph shows the imperfections of an embodied robot. For

example, we would expect more symmetry in the graph for a perfect robot. Not all of the states with one sensor activated connect to each other with equal weights as we would expect, and not all of them connect to the state with no sensors activiated. The physical robot has assymetries and stochastic influences which are often reflected in the graphs of the internal models. Further analysis of the differences between graphs will be done in Section 4.4.

4.2 Convergence of the Internal Model & Algorithm Efficiency

As another check, I analyzed whether the models generated by PIG converged over time. PIG would not be a very useful algorithm if the choices of actions caused the model to constantly change. To confirm convergence, the missing information between models was calculated after every 50 actions. For example, the missing information between the starting model and the model after 50 actions was calculated, as well as the missing information between the models at 50 actions and 100 actions, etc. The results of this analysis for all three environments is shown in Figure 3.

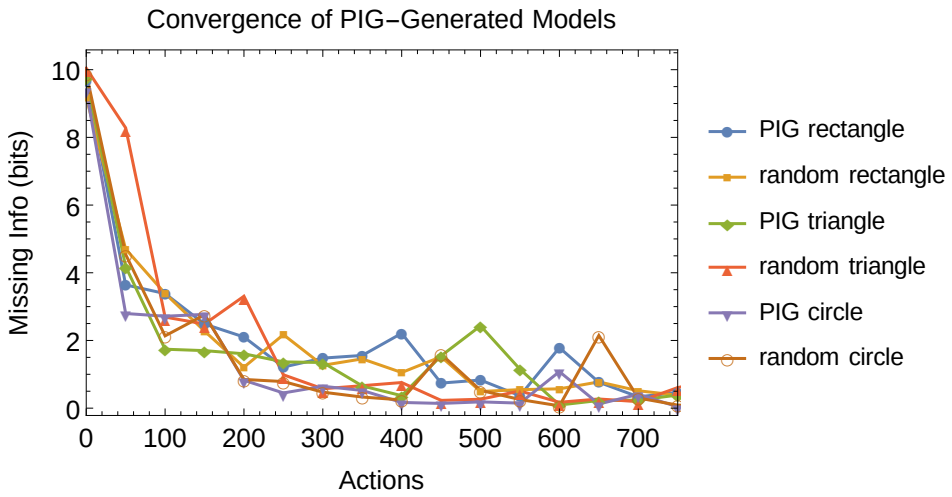

Figure 3: Comparison of successive missing information for different strategies and environments.

The figure shows that missing information between successive models decreases quickly over time (700 actions corresponds to roughly 10 minutes of letting the robot explore). These results were averaged over 3 runs for each scenario, and so provide a rough idea of the convergence behavior.

The lines labelled "random" indicate that a random controller was used to choose actions for those runs. As can be seen, the random controllers create models that converge at about the same rate as the PIG strategy. Ideally, PIG would be choosing actions more effectively than random, and thus would create models that converged more quickly. However, this lack of performance gain is to be expected for our specific configuration. As mentioned in Section 3.3, our state space, with all-to-all connection corresponds to the "dense worlds" analyzed by Little & Sommer [1]. Figure 4, taken from the work by the Redwood group, shows that for dense worlds, PIG did not outperform a random strategy (far left graph). Our environments were similar to dense worlds, except that the transition probabilities between states depend on the robot's position in space, thus making this an even more difficult environment to navigate than the dense worlds described by Little & Sommer. Robots using PIG in environments that had fewer hidden variables and that were less similar to dense worlds would most likely see more advantages from the PIG algorithm.

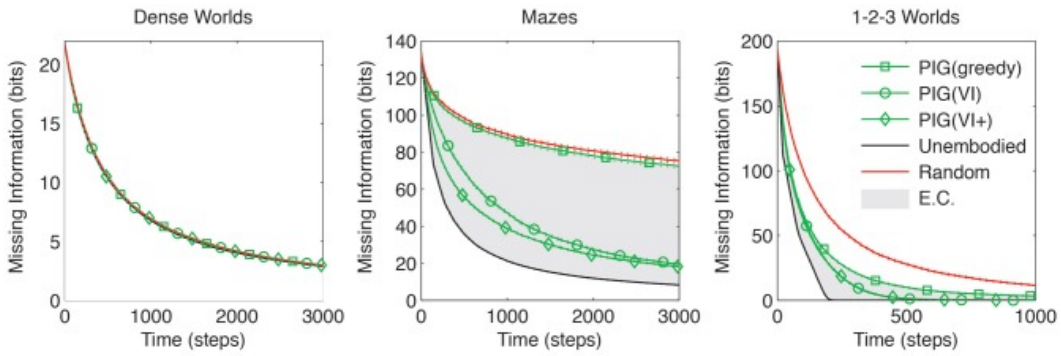

Figure 4: Figure from [1] comparing model convergence for different environments and strategies.

4.3 Accuracy of Information Gain Predictions

Predicted information gain, as calculated in Equation 3, is essentially the expected value of the information gain of all possible outcomes of an action. Because of this, we would expect that if predictions are being made well, the PIG of an action and the actual information gained by that action would tend to be similar. Actual gained information is calculated as the missing information between the model before the action and the model after. Figure 5 shows the result when predicted and actual information gain are plotted against each other.

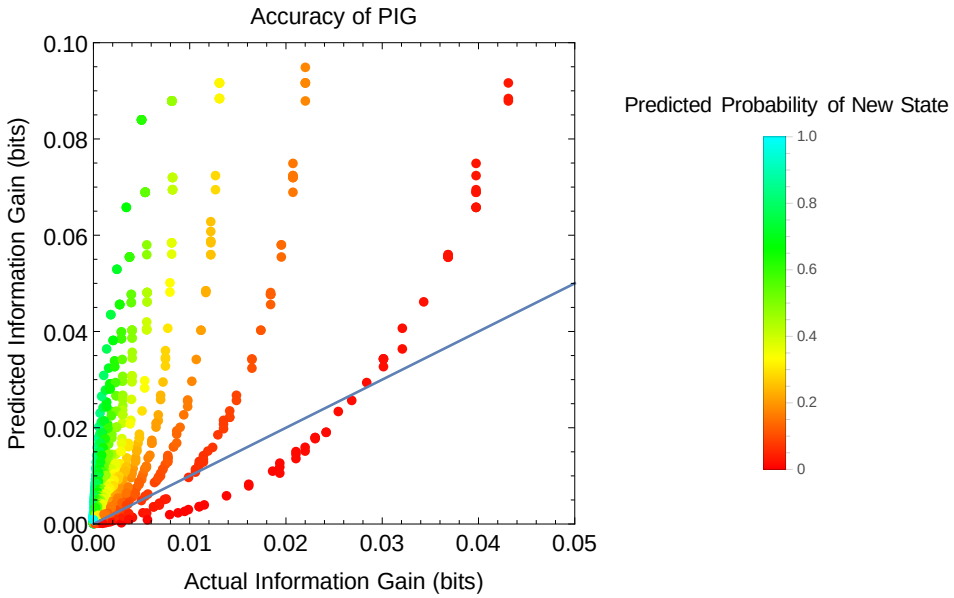

Figure 5: Predicted and actual information gain per action, with the probability of the actual resulting state shown.

Figure 5 shows that the agents tend to be over-confident in their predictions - the blue line is the identity, where we would expect the average of the points to lie. Instead, predicted information gain is consistently higher than the actual information gain. However, by showing the probability of the actual resulting state, we can see that many of the actions resulted in a state with low probability - indicating that the data that the agents get from taking actions is highly unpredictable. This is because the environment is a hidden variable - the same state and action combination can result in very different outcomes depending on where the robot is in space, something that is unknown to the agent.

The apparent structure of the graph is a result of the discreteness of our setup - as seen in Equation 4, the transition probabilities are updated discretely, and information gain is calculated using these transition probabilities.

The other interesting feature of this graph, not apparent here, is that over time while the robot is running, both PIG and actual information gain decrease toward zero. This occurs as the model becomes more accurate, and as additional data reveals less surprises about how the robot interacts with its environment.

To analyze whether any actions were more inaccurate than others, or if any actions were correlated with information gain, I also plotted a similar graph but colored the points corresponding to what action was taken. The results can be seen in Figure 6.

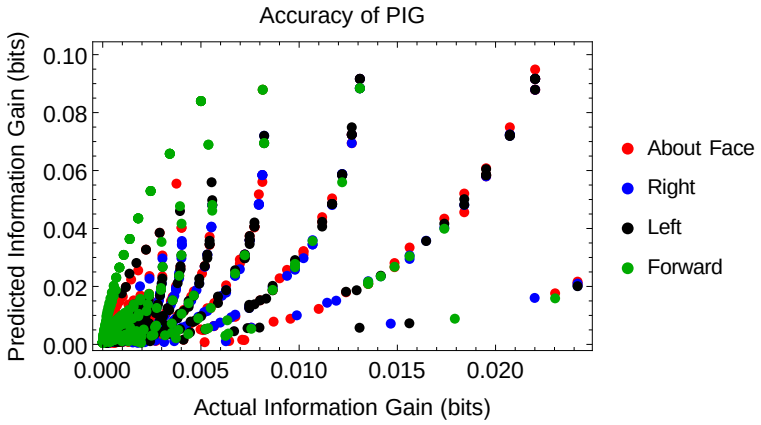

Figure 6: Predicted and actual information gain, colored by action.

There is not much correlation between action and information gain. The "forward" action seems to group at high and low information gain, possibly because the forward direction was the one most affected by hardware errors (the robot would tend to curve to the right if not warmed up).

4.4 Comparisons of Environments

The last analysis done, one which is not specific to PIG, was to analyze the structure of the calculated internal models. Reassuringly, the internal models reflect many common-sense behaviors. For example, we can compare the graphs for the "forward" action in three different environments.

In Figures 7, 8, and 9 there are some shared characteristics. For example, if one or both of the front sensors are activated (the top two squares are black), this means that the robot's front is up against a wall. In these cases, we would expect that if the robot went forward, the robot's front would still be up against the wall. This is what we see - in all three environments, the most likely transitions (thickest edges) are the ones corresponding to this scenario.

The differences between the graphs reflect the features of the environments. In Figure 7 (rectangle), seven states met the frequency cutoff (state-action combination visited more than 10% of the maximum state-action frequency), and the included states are the ones where one sensor, no sensors, or the front or back two sensors are touching the walls. In Figure 8 (triangle), we see that the state with the back two sensors against a wall is missing, meaning it was not visited frequently enough to be visualized in the graph. In Figure 9 (circle), even the state with the front two sensors is missing. These differences in frequently visited states reflect the properties of the environments - since the robot is configured to move at 90 degree angles, it is well suited to environments like the rectangle, with walls at 90 degree angles. The triangle environment, with walls at 60 degrees to each other, is a bit harder for this robot to navigate, and the circular environment with curved walls makes it very difficult for the robot to get more than one sensor on the wall. We can see this reflected in Figures 7-9: the difficulty of exploration is evidenced by how the rectangle internal model has the most states (seven nodes in the graph), compared to the triangle model with six nodes and the circle model with five states. This indicates that for more "difficult" environments,

state frequencies are clustered and the robot is not able to explore other states as easily. Since PIG is fundamentally an exploration algorithm, we would hope that a large number of the possible states would be visited frequently and thus represented in the graph.

So we can see that the resulting internal models reflect not only the structure of the environment, but also the sensor and actuator limitations of the robot. Additionally, the structures of the internal models are not unique to PIG - in the limit of large numbers of actions, the PIG-generated and random-action-generated models converge. The PIG strategy is slightly better at accessing unlikely states earlier, so when randomly-generated and PIG-generated models are compared, the PIG models usually include at least one more state than the randomly-generated models.

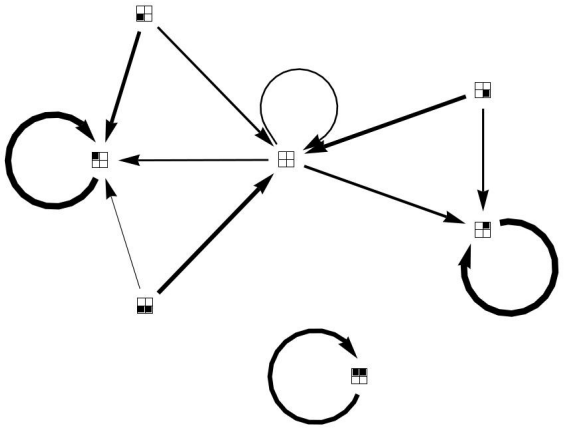

Figure 7: A graph of the "Forward" action state transitions, for a rectangle environment.

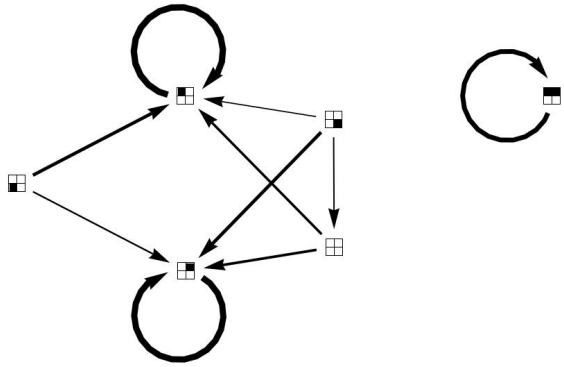

Figure 8: A graph of the "Forward" action state transitions, for a triangle environment.

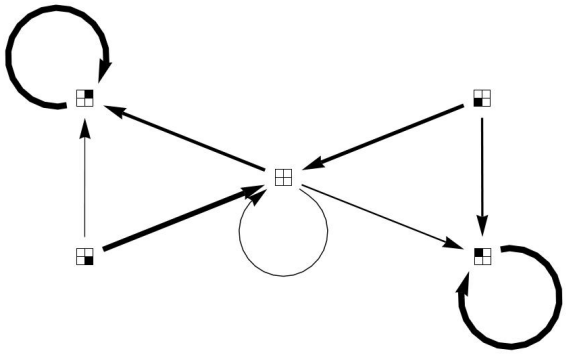

Figure 9: A graph of the "Forward" action state transitions, for a circular environment.

5 Future Investigations

There are several avenues of research that could be continued from this point. They include:

- **Combining PIG with other objective functions or behavior models:** PIG is an effective strategy for modelling learning in a new environment. However, as the agent explores more and builds a good model of its environment, the average information gain of each action decreases and the agent no longer has a strong driving force to choose one action over another.

  To better model how agents actually explore environments (from a neuroscience perspective), a second-stage learning algorithm could be included, where when information gain per action decreases below some threshold, a different strategy is implemented.

  One idea for a second-stage learning algorithm is to have the agent make predictions about what state sequences would result from given start states and action sequences. These predictions could vary in length, and the relationship between prediction length and prediction accuracy could be determined (for example, the agent might be able to make predictions three actions in the future with 75% accuracy, but accuracy drops to 25% when predicting four actions ahead in time). The accuracy of predictions would be a measure of how unpredictable the environment is (in our case, with static environments, this means how much the environment varies in space). The agent could then modify things about its strategy to maximize accuracy of predictions. This could mean that the agent changes its objective function to skew toward "exploitation" of its environment, by repeating state-action combinations more, or it could mean that the agent modifies its sensors, actuators, or memory.

- **Using PIG with more advanced sensors or actuators:** Part of the reason that PIG was not more efficient than random action choice was because the agent had very limited sensory information about a hidden environment. An agent that was able to keep track of where it is in space would explore environments very efficiently, as shown by Little & Sommer. One way to improve the performance of PIG would be to decrease how hidden the environment is to the agent, by adding more sensing capabilities.

  It would also be possible that the effectiveness of PIG would change if the robot had access to more actions - for instance, if the angle of the right and left turns was variable, it might be easier for the robot to build models of environments which have non-orthogonal components.

- **Including more memory, and using the Chinese Restaurant Process:** Another way to improve PIG effectiveness in a highly hidden environment would be to build memory into the algorithm. One way to do this, which I have done some preliminary tests with, is to encode multiple states into one "state" that is then what the PIG algorithm uses to make calculations. For example, if the current state of the agent is state 0 (binary sensor value 0000), and the previous state was also state 0, these two binary representations can be concatenated into a new state. If we started with 16 possible

sensor states, we would have 256 states with one time step of memory built in. The computational complexity of PIG scales exponentially with the size of the state space, and the size of the state space scales exponentially with the number of time steps - this is clearly not optimal for fast computation. If the decision is made to include the actions that caused transitions between states (which is very useful information), the memory scales even faster.

One possible solution to this problem of computational complexity is to use the Chinese Restaurant process [5], which allows the elimination of the prior internal model. Instead, the Chinese Restaurant process starts with one known (observed) state, assumes the state space is infinite, and always has a nonzero probability of discovering a new state. This means that when PIG is calculated, the agent will calculate the PIG based on all known states, but also the PIG of discovering a new state. Thus, instead of calculating PIG for an entire space of all possible state combinations, many of which the agent may never actually visit, PIG is only caculated for the known states + 1.

- **Exploring different environments:** Additionally, the effectiveness of PIG depends heavily on the environment. PIG should be especially effective over a random controller when the environment contains states that are difficult to get into without specific planning. For example, a rectangle environment with a small "trapping state" like a small square cut into one of the walls could produce interesting results for our agents. However, this would likely only work with increased memory or sensor sophistication, in order to increase the chances of the agent finding the trapping state in the first place and being able to deliberately return to it.

6 Conclusion

In conclusion, while this physical implementation of PIG did not show significant efficiency gains over a randomly-controlled explorer, there are several significant conclusions to be drawn. First, the degree to which the environment is hidden from the agent strongly determines the efficiency of PIG, as seen both in this study and the one by Little & Sommer. This can be seen especially when predicted information gain is plotted against actual information gain, the result of which indicates that the agent's predictions are often wrong due to its lack of knowledge about where it is in the environment. However, PIG did successfully create a more accurate model than the prior distribution that the robot started with, and the accuracy of predictions increased over time.

The second main conclusion to be drawn is that the structure of the internal models depends both on the actual structure of the environment and on the limitations of the agent's sensors and actuators. An agent that has four corner sensors in a square configuration, and that can only turn in multiples of ninety degrees, will always have an easier time exploring an environment with lots of orthogonal features.

Finally, we can conclude that in order to increase the utility of an information-theoretic algorithm like PIG, we must give our agent better ways to gain information about its environment. This could take the form of more advanced sensors, more control over the actions the agent can take, and/or increased memory. Of these three, I think that increased memory is the most interesting and most likely to produce interesting results without needing significant hardware changes to the robots. The problem of how computational complexity scales with memory / state-space size is a significant one, but could potentially be overcome using the Chinese Restaurant process. The field of information-theoretic machine learning objective functions is an important one, and one that has interesting applications both in neuroscience and physics.

References

[1] D. Little and F. Sommer, "Learning and Exploration in Action-Perception Loops." Frontiers in Neural Circuits, 22 Mar 2013, doi: 10.3389/fncir.2013.00037

[2] RoMADS project, Complexity Sciences Center, UC Davis. http://csc.ucdavis.edu/ dynlearn/dynlearn/robots.htm

[3] T. Cover and J. Thomas, "Elements of Information Theory." 2nd ed, Wiley-Interscience, Hoboken, NJ. 2006.

[4] Coggan, M. Exploration and Exploitation in Reinforcement Learning. In: Fourth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '01). Shonan International Village Yokosuka City (2001).

[5] Blei, David. COS 597C: Bayesian nonparametrics lecture notes. Princeton University, 21 Sept 2007.