Towards Human-Induced Vision-Guided Robot Behavior

Gabriel J. Ferrer
Department of Mathematics and Computer Science
Hendrix College
1600 Washington Ave.
Conway, Arkansas 72032

Abstract

An appealing alternative to tediously specifying robot behaviors in response to particular image features is to have the robot's behavior be induced by human decisions made when piloting the robot. This paper presents one promising approach to creating this alternative. A human pilots a camera-equipped robot, which builds a representation of its target environment using Growing Neural Gas (GNG). The robot associates an action with each GNG node based on what the human pilot was doing while the node was active. When running autonomously, the robot chooses the action associated with the node that is the closest match to the current input image. Preliminary results suggest that the approach has potential, but that subsequent alteration of the actions induced for some of the GNG nodes is important for acceptable performance.

Programming reactive behaviors (Brooks 1986) on a robot equipped with basic sensors such as touch sensors and sonars is reasonably straightforward and well-understood. Despite considerable progress in computer vision techniques, use of camera images to direct reactive behaviors remains tricky. Considerable effort is often invested in computer vision algorithms that are specialized for a particular environment. (Horswill 1994)

The goal of this work is to use machine learning to enable the reactive behavior to be induced by the actions of a human piloting the robot. The machine learning algorithm to be employed needs to meet the following criteria:

- It should be an online algorithm that learns incrementally as images are presented to it.
- The algorithm should operate quickly enough for the robot to be able to drive at a respectable speed, both when learning and when selecting actions based on learning.
- The learned representation should be comprehensible.
- The learned representation should be editable, to enable the programmer to compensate for errors that arise during the learning process.

The Growing Neural Gas (GNG) algorithm (Fritzke 1995) meets our criteria. The algorithm partitions its training inputs into clusters based on a distance metric. Each cluster is represented by a reference image. It adaptively determines the number of clusters based on the input images it receives.

As the robot is driven through its environment by a human operator, it builds a GNG network. This network associates an action with each node based on the choices made by the human pilot when the node is active. This choice can be subsequently changed if the programmer decides that the learned action is not appropriate. When running autonomously, the robot chooses the action associated with the node whose reference image is the closest match to the current input image. We found that this approach was effective for developing an obstacle-avoiding behavior.

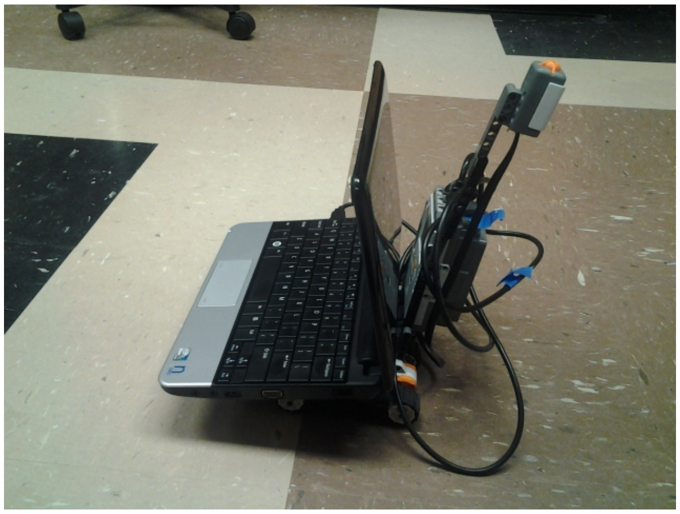

Figure 1: Portrait of the Robot

Learning the Environment

Unsupervised Learning

Unsupervised learning algorithms, such as k-means (Forgey 1965) (MacQueen 1967), the self-organizing map (SOM) (Kohonen 2001), and Growing Neural Gas (GNG) (Fritzke 1995), operate by partitioning their training inputs into clusters based on a distance metric. Each cluster is defined by a representative example of a subset of the input space. Each representative example is arithmetically derived (in an algorithm-dependent manner) from the training inputs. Because the clusters are defined by representative examples from the input space, it is possible to understand the learned representations by examining them directly.

Both k-means and self-organizing maps require the number of clusters to be fixed in advance. Growing neural gas adaptively determines the number of clusters based on its inputs. In the interest of allowing the inputs to determine the number of clusters, we selected GNG for this task.

Growing Neural Gas

Growing Neural Gas is an artificial neural network that is trained using an unsupervised learning algorithm. The network is an undirected weighted graph. In this implementation, both the network inputs and the network nodes are 2D grayscale images. When an input is presented to the network, the Euclidean distance between the input and each node is calculated. The node with the shortest distance relative to the input becomes the active node.

Each edge connects two nodes that are active in response to similar inputs. The edge weight reflects the frequency with which the pair has been responsive in tandem; it is reset to zero whenever this occurs. Large weights denote weak connections that are eventually purged. Nodes who lose all their edges are purged as well.

Training In each iteration of training, a single input is presented to the network. The network is then adjusted as follows:

- Identify the nodes that are the closest and second-closest matches to the input.
- Adjust edge weights.
- Update error and utility values.
- Create a new node.
- Purge edges and nodes.

In the training process, two nodes are identified: the node with the shortest distance to the input, and the node with the second-shortest distance. If an edge is present between them, its weight is reset to zero; otherwise, a new edge is created between them with a weight of zero. The weights of all other edges adjoining the winning node are increased by one.

The values of each image pixel p_{xy} for the winning node and each of its neighbors are updated as follows relative to each input pixel q_{xy}:

$$p_{xy} = p_{xy} + \alpha(q_{xy} - p_{xy})$$

The learning rate parameter α is significantly larger for the winning node in comparison to the lower value used for the neighboring nodes.

Error and Utility Each node maintains *error* and *utility* values. The purpose of the error value is to identify nodes that, while they are frequently the best matching node for a variety of inputs, are still not very close matches. These nodes, then, represent subsets of the input space that are not adequately covered by the current set of nodes. The purpose of the utility value is to identify nodes that are distinctive relative to their neighbors. Nodes with high utility are frequently much closer matches to many inputs than their neighbors.

On each training iteration, these values are updated as follows:

- For the winning node:
  - Increase the error for the winning node by the Euclidean distance to the input.
  - Subtract the distance to the winning node from the distance to the second-best node. Add this difference to the utility.
- For each node n:
  - Reduce the error and utility values using a decay constant β ($0 < \beta < 1$) as follows:
    - $error_n = error_n - \beta \times error_n$
    - $utility_n = utility_n - \beta \times utility_n$

Creating New Nodes An integer parameter λ controls the creation of new nodes. The frequency of node creation is inversely proportional to lambda; low values imply frequent introduction of new nodes. The purpose of adding new nodes is to better represent subsets of the input space that are inadequately represented by the current node set.

Every λ iterations, a new node is created as follows:

- Find the node m with the largest error value in the network.
- Find its neighbor n with the largest error value among all of m's neighbors.
- Create an image by averaging the corresponding pixel values of m and n.
- Create a new node p using:
  - The averaged image
  - The mean of the errors of m and n
  - The maximum of the utility values of m and n
- Break the edge between m and n.
- Add an edge of weight zero between m and p, and another between n and p.
- Divide the error and utility values for each of m and n by two.

Purging Edges and Nodes On each iteration, every edge whose weight exceeds a specified limit is purged. If any node has all its edges purged, that node is purged as well.

In addition, for every node the *utility ratio* is calculated (Fritzke 1997). The largest error of any node, e_{max}, is determined. The utility ratio for each node n with utility u_n is $\frac{e_{max}}{u_n}$. The node with the single largest utility ratio is the *most useless node*. If its ratio exceeds a parameter k, it is purged.

Training and Programming

Our robot is equipped with a controller that allows a human pilot to drive it around its environment. As the robot is driven around, the first two images it acquires become the first two nodes of Growing Neural Gas. Each subsequent image acquired is used for one iteration of training the GNG network.

Training ends upon a signal from the pilot. The GNG network is then saved for later use.

We modified the GNG learning algorithm as follows. Whenever a node is the winning node, there are the following three possibilities, resolved as follows:

1. When no action has yet been assigned to the node, the current action is assigned.
2. When the action assigned to the node is identical to the current action, nothing needs to change, as the current assignment is confirmed to be satisfactory.
3. When the action assigned to the node is different from the current action, we have a "conflicting opinion" from the human pilot that must be resolved. This conflict is embedded in the GNG network itself, as a single node is covering inputs that trigger distinct responses from the pilot. The conflict is resolved as follows:
   (a) Create a new GNG node, setting its representation to be the current input image.
   (b) Create an edge of weight zero between the new node and the winning node.
   (c) Assign the current action to the newly created node.

A further modification to the algorithm is that any node to which an action has been assigned is immune from purging.

When the controller executes, as each image is acquired it is presented to the GNG network. (Nodes that never received action assignments are not included.) The action specified for the winning node is immediately executed.

Experiments

Configuration and Training

The experimental goal was to train the robot to be familiar with a particular room, such that it could wander the room without hitting anything. The robot is a Lego Mindstorms NXT. To enable image processing, a Dell Inspiron Mini Netbook equipped with a webcam was placed atop the robot to control it via a USB cable. The configured robot is shown in Figure 1. The human pilot chose one of the following actions on each time step: `FORWARD`, `SPIN_LEFT`, and `SPIN_RIGHT`.

The webcam images were acquired at a size of 640x480 pixels. Each image was averaged to produce a grayscale image upon acquisition, with each pixel value ranging from 0 to 255. Each grayscale image was scaled down to 160x120 prior to being applied as an input to the GNG network. The first two GNG nodes are the first two images acquired. This configuration enabled images to be acquired and processed and learning to occur at about 15-16 frames per second. Given a robot velocity of 17 cm/s, it processes about one frame per centimeter of travel. In the execution phase, the processing rate dropped to about 7 frames per second, which is still fast enough for respectable performance.

The GNG algorithm was parameterized as follows:

- Maximum edge age: 100
- Learning rate (winning node): 0.05
- Learning rate (neighboring nodes): 0.0006
- Maximum utility ratio (k): 4
- Iterations per node creation (λ): 200
- Error decay (β) per iteration: 0.0005

Most of the parameters were taken directly from the experiments described by (Holmstrom 2002). Since the robot processes one frame per centimeter of travel, the λ value of 200 was selected to ensure the creation of a new node after, at most, two meters of travel.

The GNG network was trained by driving the robot around the room for several minutes. The trained network had 74 nodes by the completion of the run.

Action Selection

For 50 out of the 74 reference images (67.6%), the induced action for the node reflected the intent of the pilot. Figure 2 is a typical example of a node labeled for forward motion. The floor tiles are blurred together, and the walls and cabinets are found exclusively on the side of the image.

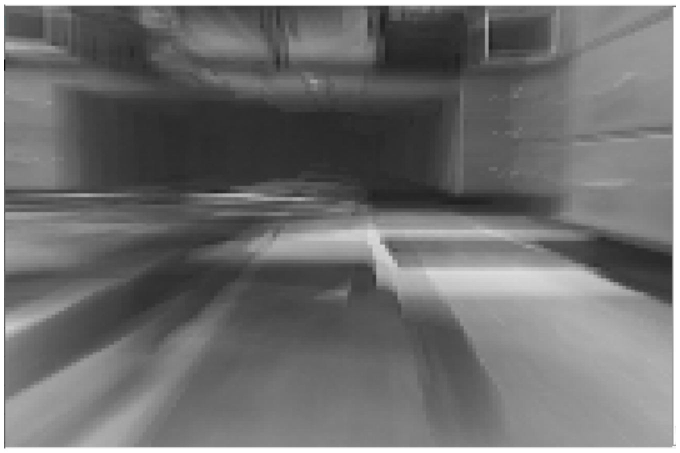

Figure 2: `FORWARD` Node

Figure 3 is a typical example of a node labeled for turning. The cabinet is a dominant image feature. Rotating to the left would orient the robot towards the clearly visible floorspace.

In several cases, two very similar images were assigned different actions. In these cases, the partitioning of the input space given by the algorithm did not match the partitioning intended by the pilot. A typical example is given in Figure 4. The top image was assigned a `SPIN_LEFT` action, while the bottom image was assigned `FORWARD`. For some images that match the bottom reference image, there is enough space to move forward; in other cases, there is not. Consequently, the action for the bottom image was changed to `SPIN_LEFT` after the training phase was completed.

In a few cases, the reference image is so ambiguous that it is difficult to give it a meaningful action assignment. In Figure 5, which was assigned `FORWARD`, floor pixels are dominant. However, there is a ghostly image of a wheeled chair in the foreground as well. When assigned as a turn, the robot often makes spurious turns in the midst of an open area. When assigned as forward motion, the robot is very

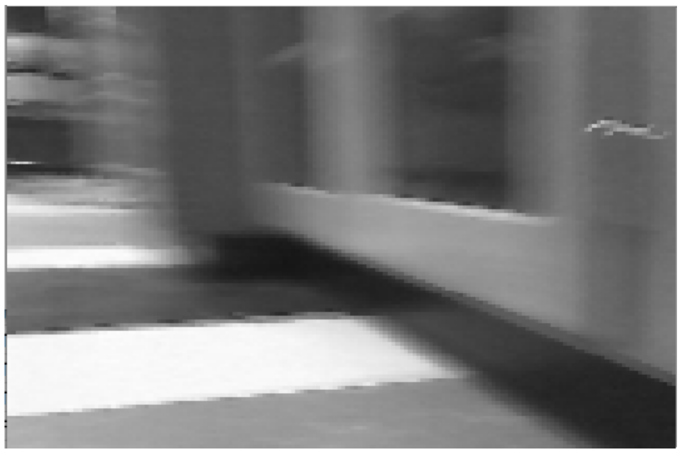

Figure 3: SPIN_LEFT Node

susceptible to collisions with wheeled chairs. This node and some other problematic nodes were handled by manual removal from the GNG.

A final problem that arose periodically was oscillation between nodes. The top image in Figure 6 has a SPIN_LEFT action; the bottom image is SPIN_RIGHT. The robot was stopped after it entered an oscillation between these nodes that it could not escape.

Observed Robot Behavior

Figure 7 gives the duration and percentage of forward motion for each of ten runs. Runs A1-A3 were based on a controller for which 21 out of 74 actions had been altered after training. All runs except A3 ended by a movement-stopping collision. As Run A3 was terminated as a result of the oscillation described in Figure 6, Runs B1 and B2 reflect the changing of all turns to be right turns. Runs C1-C3 reflect a similar controller after the removal of an ambiguous node. That particular node had been implicated in the collisions that ended the B runs. Runs D1 and D2 reflect the removal of four additional nodes found to be problematic during the "C" runs. The "Human" value denotes the percentage of forward motion for a single run of a human piloting the robot, maximizing forward motion while avoiding obstacles. No duration was provided for the human, as no movement-stopping collision occurred when manually piloting the robot.

The early runs show respectable performance but with some clear flaws. Runs A1 and A3 ran for over a minute without collisions, but with forward motion below 50%. Run A2 managed excellent forward motion, but its early collision reflects the fact that it moved forward a bit too often. The decision to interpret all turning actions as right turns improved forward motion but did not improve the durations. The removal of ambiguous nodes significantly increased duration at the cost of decreased forward motion.

The robot's failure to reach human levels of forward motion can be attributed to two factors. First, the manual replacement of several FORWARD actions with turns introduced an extra level of caution to the robot relative to the actions of the pilot. This was especially noteworthy for the situations reflected in Figures 4 and 8. Second, the robot occasionally performed spurious turns as the result of encountering ambiguous nodes.

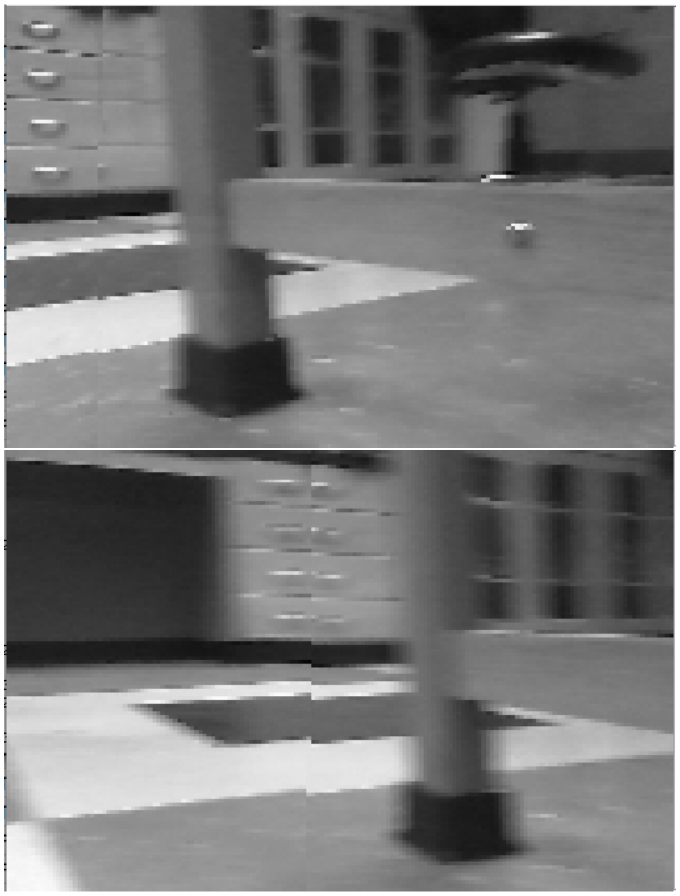

Figure 4: Mismatched Actions

Analysis

Characteristics of Nodes with Changed Actions

In analyzing the 24 nodes that had to be changed to reflect human intent, we made the following observations:

- Out of the 74 nodes in the GNG, there were 23 nodes with only one neighbor. We refer to these as "singletons", and they were all generated as a result of the conflict-resolution strategy described earlier.
- Each of the four nodes with more than one singleton neighbor had to have its action changed. Only one of these nodes was left in the GNG for the final two "D" runs.
- Four of the 23 singleton nodes required an action change.
- Nine nodes had one singleton neighbor. Six of these required an action change.
- Out of the remaining ten nodes with changed actions, six of them had a strong humanly-detectable visual resemblance to a neighboring node with a different action. We were not able to devise an algorithmic test that corresponded to this human-intuitive relationship.

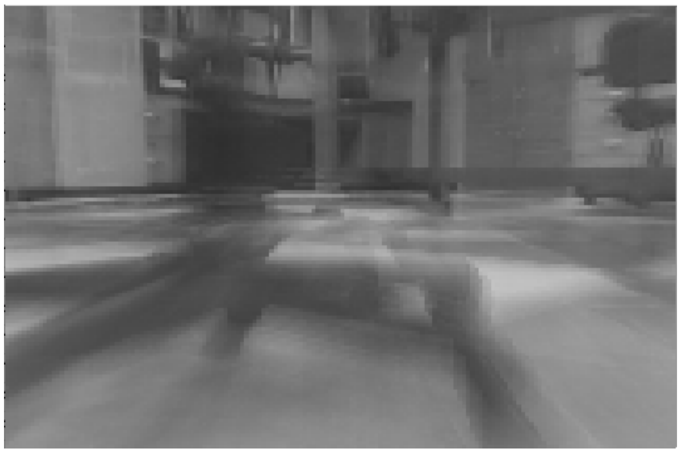

Figure 5: Ambiguous Node

- The last four nodes were all neighbors with at least one other node in need of an action change. In all cases, this node was one of the four nodes that had multiple singleton neighbors.

Human Performance

The robot did not always learn the lesson intended by the human pilot. We saw one example of this earlier, in the discussion of Figure 4. A related issue can be seen in Figure 8. When approaching the cabinet, the pilot turned briefly. After the brief turn, there was enough space available to enable some brief forward movement, prior to executing another turn. This movement pattern was reflected in the structure of the GNG. As a practical matter, there was not enough distance between the nodes to justify the forward motion, so the middle node was changed to a turn.

Further Development

The results of this preliminary study are very promising. About two-thirds of the learned actions were usable without further editing. The success of the robot in performing its task provides evidence that this formulation of GNG can properly partition the input space to correspond reasonably well to human intent. The decision to use a clustering algorithm proved helpful, in that the learned behaviors were able to be debugged and improved.

Our agenda to continue developing this technique is as follows:

- Our initial results provide evidence that singleton nodes and their neighbors are strong candidates for manual action changes. We plan to modify the user interface to present the singleton nodes and their neighbors for programmer examination at the start of the customization process, in order to get the programmer started with nodes that are most likely in need of modification.
- As ambiguous nodes are a significant problem, we are experimenting with a new feature that will allow the programmer to transform the reference image of a node to more closely resemble one of its neighbors. This may allow the programmer to smooth away ambiguities in the reference image.

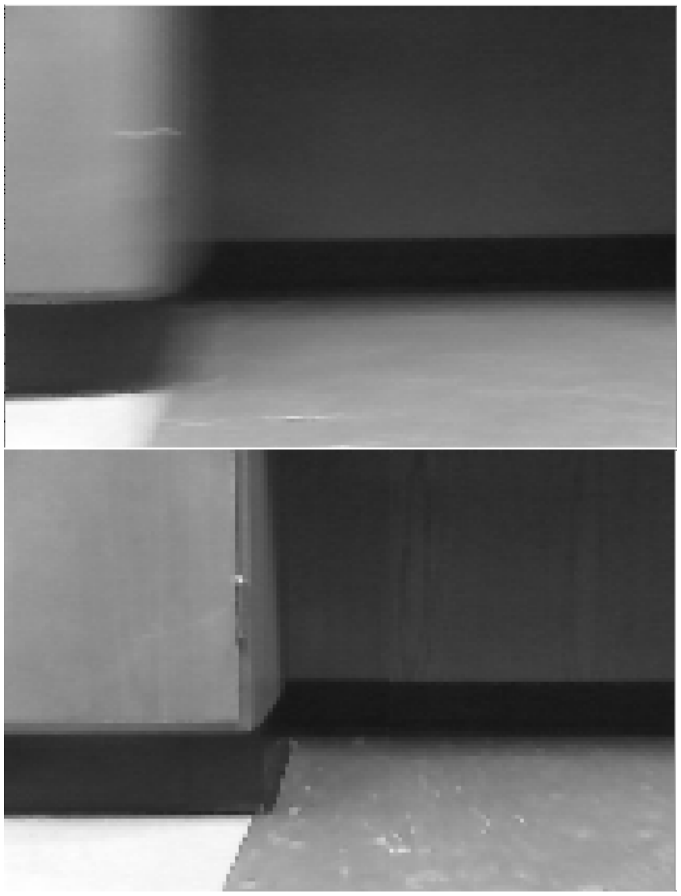

Figure 6: Oscillating Actions; the top image is a left turn, while the bottom image is a right turn

- Since there were some node relationships where the human intuition of image distance diverged from that given by the Euclidean distance metric, we plan to investigate alternatives to Euclidean distance that exploit higher-level image features, provided that such alternatives can be made to interact properly with the GNG learning algorithm. One example of such an alternative is the bag-of-visual-words representation employed in certain types of appearance-only SLAM models (Cummins and Newman 2010).
- It is possible that additional ambiguities could be resolved by using color information. In effect, the current scheme uses only the Intensity element of the HSI image model. Using the Hue and Saturation values has the potential to improve the clustering.
- We plan to create an implementation for the Lego Mindstorms EV3, using a webcam connected via its USB port. We plan to distribute the EV3 implementation in an open source manner. This will enable a wider community to experiment with this technique.

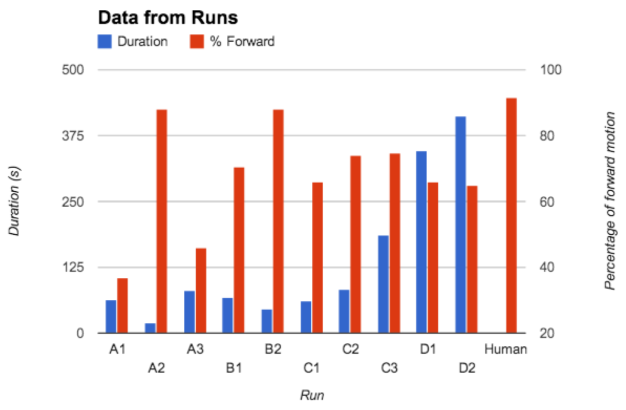

Figure 7: Duration/Forward Motion

- We plan to teach a robotics course in Fall 2014 employing the EV3 implementation. This will help us assess the degree to which the underlying behavior creation/improvement methodology is effective. It will also provide the opportunity to experiment with this technique in several different problem spaces.

Related Work

This project is an extension of our previous work applying Growing Neural Gas for behavior creation (Ferrer 2014). In our earlier variation, the human pilot's actions were not recorded. The human was then responsible for specifying the action for every single GNG node. The goal of this work was to reduce the effort necessary for action specification by exploiting the knowledge of the pilot's action selections. Doing this required introducing the action-conflict resolution mechanism into the learning process.

Growing Neural Gas has been applied to robotic systems for numerous purposes beyond that proposed in this paper, including localization (Baldassarri et al. 2003) (Yan, Weber, and Wermter 2012), modeling the physical structure of the environment (Kim, Cho, and Kim 2003), (Shamwell et al. 2012), and control via gesture recognition (Yanik et al. 2012).

A representative example of a system that uses artificial neural networks to simplify the programming of reactive behaviors is (Cox and Best 2004). In their system, the programmer specifies motor settings for various combinations of sensor values that a simulated robot encounters. A multi-layer perceptron is employed to generalize the motor settings to sensor combinations that had not been previously encountered. The simulated sensors are three infrared sensors and two bump sensors. It is not at all clear how this approach would scale to the much greater complexity of visual input. Our work, by exploiting the inherent intelligibility of the reference images of the GNG clusters, provides the programmer with meaningful abstractions of the visual input for which actions can then be coherently specified.

(Touzet 2006) employs self-organizing maps (Kohonen 2001) to build models of a physical robot's environment and the effect of performing actions in certain states. The desired behavior is then specified in terms of targeted sensor values. For the range sensors they employed, specific windows of target values are specified in order to induce the target behavior. When using computer vision as a sensor, creating such windows of target values is impractical. The GNG cluster reference images provide a usable alternative.

(Adam Coates and Ng 2008) describe a different approach to developing a robot controller based on human demonstration. They employ a human expert to pilot a remote-controlled helicopter multiple times through a target trajectory. The helicopter then autonomously traverses that target trajectory. Our work addresses a different problem. The work of (Adam Coates and Ng 2008) shows how to learn a complex trajectory given relatively simple sensor inputs. Our work shows how to learn a relatively simple trajectory given complex sensor inputs. The differences in the underlying data structures employed reflects this.

(Leonetti et al. 2012) describe a method for automated generation of finite state machines as robot controllers. As our use of machine learning is aimed at making complex sensor data accessible, and their use of learning is to create controllers given a formal model of a problem domain, there is possible synergy between them. Our work could help reduce a visual input stream into a finite set of states upon which the controller-learning architecture could operate.

Conclusion

We have shown that Growing Neural Gas can be employed for inducing robot behavior modeled on that of a human pilot. As an online learning algorithm with a humanly understandable representation, a human can further alter the behavior of the learned controller when its behavior in practice is not satisfactory. It runs sufficiently fast for real-time processing, both when learning and when selecting actions.

Significant work remains to be done to improve upon the results of this preliminary study. The research agenda from here includes further modifications to the GNG learning algorithm as well as experimentation with a larger number of problems and a wider variety of domains.

References

Adam Coates, P. A., and Ng, A. Y. 2008. Learning for control from multiple demonstrations. In *Proceedings of the International Conference on Machine Learning*.

Baldassarri, P.; Puliti, P.; Montesanto, A.; and Tascini, G. 2003. Self-organizing maps versus growing neural gas in a robotic application. In Mira, J., and lvarez, J. R., eds., *IWANN (2)*, volume 2687 of *Lecture Notes in Computer Science*, 201–208. Springer.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2(10).

Cox, P. T., and Best, S. M. 2004. Programming an autonomous robot controller by demonstration using artificial neural networks. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, 157–159.

Cummins, M., and Newman, P. 2010. Appearance-only SLAM at large scale with FAB-MAP 2.0. *International Journal of Robotics Research*.

Ferrer, G. J. 2014. Creating visaul reactive robot behaviors using growing neural gas. In *Proceedings of the 25th Modern Artificial Intelligence and Cognitive Science Conference*, 39–44.

Forgey, E. 1965. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics* 21(768).

Fritzke, B. 1995. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, 625–632. MIT Press.

Fritzke, B. 1997. A self-organizing network that can follow non-stationary distributions. In Gerstner, W.; Germond, A.; Hasler, M.; and Nicoud, J. D., eds., *Proceedings of the Seventh International Conference on Artificial Neural Networks: ICANN-97*, volume 1327 of *Lecture Notes in Computer Science*, 613–618. Berlin; New York: Springer-Verlag.

Holmstrom, J. 2002. Growing neural gas: Experiments with gng, gng with utility and supervised gng. Master's thesis, Uppsala University, Department of Information Technology Computer Systems Box 337, SE-751 05 Uppsala, Sweden.

Horswill, I. 1994. Visual collision avoidance by segmentation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 902–909. IEEE Press.

Kim, M. Y.; Cho, H.; and Kim, J. 2003. Obstacle modeling for environment recognition of mobile robots using growing neural gas network. *International Journal of Control, Automation, and Systems* 1(1).

Kohonen, T. 2001. *Self-Organizing Maps*. Springer, 3rd edition.

Leonetti, M.; Iocchi, L.; ; and Patrizi, F. 2012. Automatic generation and learning of finite-state controllers. In *Proceedings of AIMSA 2012: The 15th International Conference on Artificial Intelligence: Methodology, Systems, Applications*.

MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. In *Proc. of the Fifth Berkeley Symposium on Math. Stat. and Prob.*, 281–296.

Shamwell, J.; Oates, T.; Bhargava, P.; Cox, M. T.; Oh, U.; Paisner, M.; and Perlis, D. 2012. The robot baby and massive metacognition: Early steps via growing neural gas. In *ICDL-EPIROB*, 1–2. IEEE.

Touzet, C. 2006. Modeling and simulation of elementary robot behaviors using associative memories. *International Journal of Advanced Robotic Systems* 3(2):165–170.

Yan, W.; Weber, C.; and Wermter, S. 2012. A neural approach for robot navigation based on cognitive map learning. In *IJCNN*, 1–8. IEEE.

Yanik, P.; Manganelli, J.; Merino, J.; Threatt, A.; Brooks, J. O.; Green, K. E.; and Walker, I. D. 2012. Use of kinect depth data and growing neural gas for gesture based robot control. In *PervasiveHealth*, 283–290. IEEE.

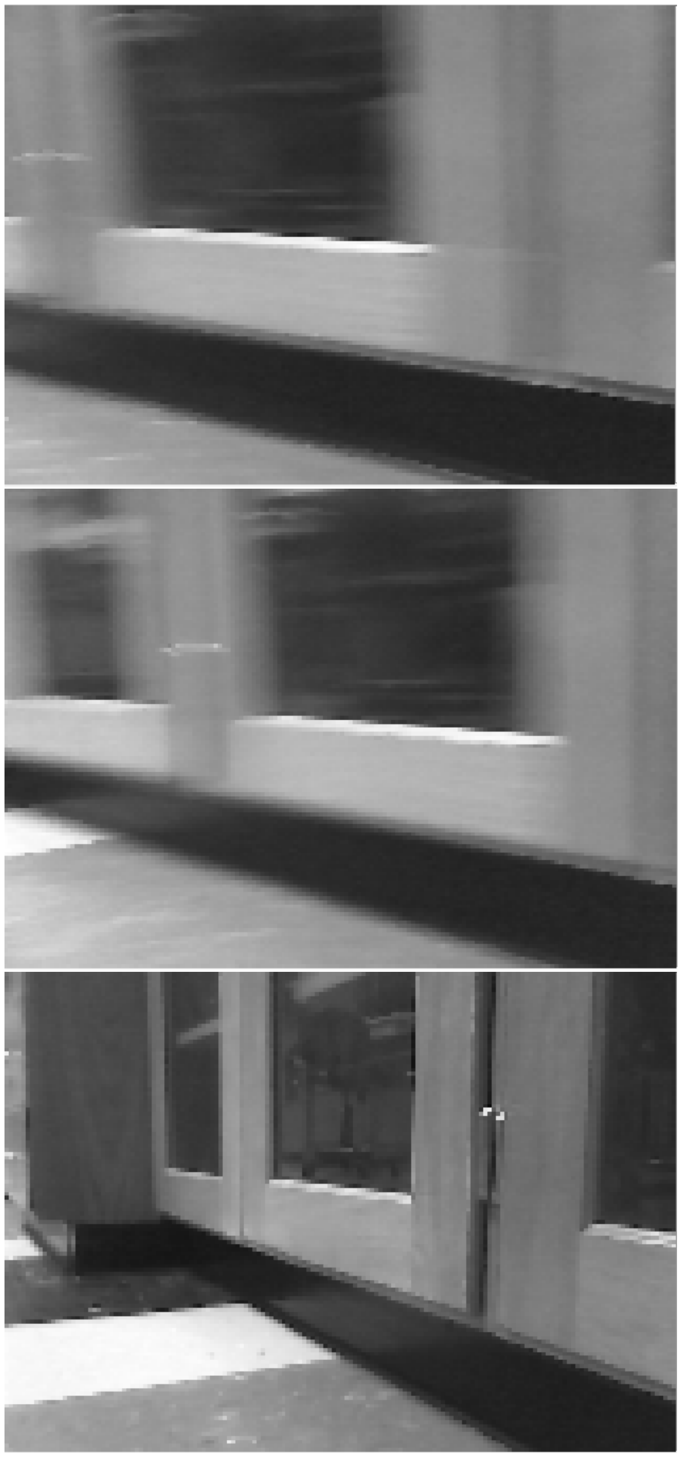

Figure 8: Mixed-up human actions; the top and bottom images were learned as turns, while the middle is a forward move