KEER2010, PARIS | MARCH 2-4 2010
INTERNATIONAL CONFERENCE ON KANSEI ENGINEERING AND EMOTION RESEARCH 2010

COMPARISON OF LEARNING ON THE DESIGN OF A KANSEI ROBOT TESTBED FOR UNDERSTANDING HUMAN-MACHINE INTERACTION

Frederick Livingston[a], Edward Grant[a], and Gordon Lee[*b]

[a] *Department of Electrical and Computer Engineering; North Carolina State University; Raleigh, NC 27606, USA*

[b] *Department of Electrical and Computer Engineering; San Diego State University; San Diego, CA, 92182, USA*

ABSTRACT

The design of autonomous controllers for mobile robotic systems continues to be a challenge, due to the complexities and uncertainties of not only the robotic systems but also the environment. In this paper, we present preliminary work on the development of a KANSEI robot testbed; in particular, a simple neural network controller in combination with human reaction monitoring and post processing is developed. Currently, computer vision and a joystick are used but additional sensor platforms will be developed to capture human reaction in interacting with the robot for a collaborate activity. The validity of the approach is tested experimentally on a mobile robot with vision capability. Results show that the using human reaction to capture robot images provides a feasible architecture for designing mobile robot controllers.

Keywords: *robot testbed, neural network control*

Corresponding author; Email: glee@kahuna.sdsu.edu

1. INTRODUCTION

I hear and I forget; I see and I remember; I do and I understand

…ancient Chinese proverb

Research continues to flourish in the development of colonies of miniature robotic platforms that are inexpensive and yet robust, computationally powerful enough to be autonomous while displaying certain desired behaviors [1-7]. To be truly useful, comprehensive development tools and robust controller designs must be available for robot colonies and, in particular, modules for supporting rapid controller construction through evolution in robot simulation [8] and experimentation [9]. Further, to be true partners in performing a task, humans working with robots must share common preferences, strategies and goals; we believe this can only be accomplished through a KANSEI engineering approach.

In the area of control designs for mobile robotic systems, Boada and others [10] developed a method based upon reinforced learning to teach robot simple skills such as moving to a goal or contour following. Yibin and others [11] also employed reinforced learning and in particular Q-Learning for path following and obstacle avoidance. In [12], Kim and others used fuzzy rules and simple cooperative learning methods to design controllers for a colony of robots. Further Barate and Manzanera [13] used genetic programming to produce a vision-based obstacle avoidance algorithm for mobile robots.

Humans can play a role not only in collaborating with robots but also in teaching robots. Several researchers continue to investigate the roles of the human and the machine in human-robot interactions. For example in [14], Demiris provides an excellent overview of several approaches for generating action recognition and predicting *intent* for human-robot interactions. The approaches are classified as descriptive (characterizing patterns under constraints to generate actions) or generative (formulating a set of variables to link causes with observed data). Further architectures such as HAMMER [14], which generates internal models and uses robot learning by imitation, and a pipeline information system, based upon a situation awareness model [15], have been developed for intent recognition. An interaction debugging tool has been developed which may be used in analyzing the data collected in human-robot interactions [16].

In this paper, as part of a larger effort to understand the process of how humans can teach robots to learn, we focus on a simple method that integrates neural networks and reinforced learning through human interaction with the robots. The Center for Robotics and Intelligent Machines (CRIM) at North Carolina State University has

focused on the design of intelligent controllers for mobile robot colonies for many years. Further the CRIM has developed an experimental testbed and several simulation tools to assist with the development of the control designs as well as with the evaluation of these methods. The CRIM approach is a generative one whereby genetic algorithms are employed to capture behavior characteristics and supervisory evolutionary methods are used to train the robot controllers. Previous controller designs have been developed using acoustic analysis [17], rule base controllers and evolutionary algorithms [18]. The acoustic analysis controller navigates the robot to the goal by triangulating the distance and direction of the sound source using an array of microphones. The robot will then navigate to the goal using classical motor control algorithms. An evolutionary neural network algorithm was previous implemented to successfully navigate the robot to its goal. The evolutionary algorithm used reinforcement learning where high-level task performance feedback is applied to the evolution of controllers for autonomous mobile robots. Evolutionary controller design involves many generations of lengthy training. The purpose of this research is to develop a "breeder training" algorithm where a human is used to teach the robot to accomplish its goal. This is believed that this approach can produce results similar to the evolutionary neural network controller but in a faster time period.

The CRIM has developed a computationally powerful colony of small mobile robots [17-18]. These robots are called EvBots from EVolutionary roBOTs. Each robot is 9 in. wide by 12 in. long by 8 in. high and is constructed on a two track treaded wheel base. Each robot is equipped with a PC/104 based onboard computer with an X86 software compatible 32-bit CPU core operating at 133 MHz. For communication between robots in the colony, each is linked to the Internet via a Linksys Wireless Ethernet PC-Card. A custom Linux distribution derived from Debin is used as the operating system and is capable of executing high-level programming languages. The robots are linked to one another and to the Internet via a Linksys wireless network access point that can support up to 21 devices. Each robot also supports video data acquisition (up to 640x480 live motion resolution) through a USB video camera mounted on each robot. A photograph of the fully assembled EvBot II is shown in Figure 1. Using this mobile robot colony, we are able to test several control designs, including the approach presented here.

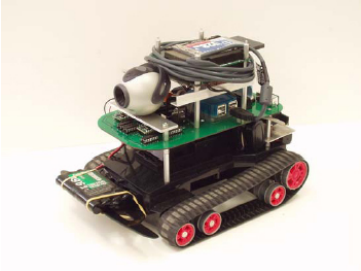


Figure 1: Mobile Robot Platform

2. THE OVERALL KANSEI ARCHITECTURE

Recently, several researchers have focused on the application of *kansei* in robotics. For example, Hashimoto [19] suggests a new area in which robots and human can interact as partners. In [20], Sato et. al. employ kansei in designing neural network controllers for a manipulator arm. For the over-arching architecture for our human-robot interaction model, we plan to employ the Multi-Agent Mind Model suggested by Tauchi et. al. [21]. In Tauchi's work, the mind of a human can be modeled by multi-agents: (a) stimulus processing agent (SPA); (b) knowledge processing agent (KPA); (c) emotion processing agent (EPA); and (d) response processing agent (RPA). One can associate each agent with classical robot components or software strategies: (a) stimulus processing agent – sensor information; (b) knowledge processing agent – knowledge databases; (c) emotion processing agent – heuristics; and (d) response processing agent – control rules. Each of these agents interacts with and is stimulated by the world environment.

Thus, a robot as an equal partner with a human in a collaborative activity should have similar functions as does a human. While one may be a teacher (the human) in the beginning of a task scenario, the other (pupil) can gain information and develop kansei through working with the human – *learn by doing*. The architecture we propose is based upon the Tauchi model [21] but with a modification as shown in Figure 2. The EPA block receives mental images from other robots based upon instincts, including human instincts during training, and outputs mental images to the learning mechanism; the KPA block receives mental images from its knowledge base built from other robots as well as itself from previous tasks and outputs mental images to the learning mechanism; and the SPA block receives stimuli from other robots and the environment and outputs mental images to the learning mechanism. The learning mechanism takes all of the mental images and produces actions – in the future, we plan to investigate the use of knowledge amplification [23], amongst several techniques, for the learning.

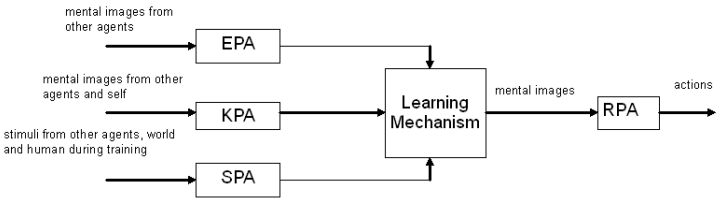


Figure 2: System Architecture

In [22], we focused on the KPA and SPA. In particular, we have developed several mobile robots and have investigated robot behavior in a colony. Each robot consists of a data structure that stores the robot's current position, orientation, sensor input readings, and actuator output. The overarching approach is an evolutionary one where three different controllers were studied (rule-based, random and neural networks). Human intervention can be interjected during the training sessions and the simulation environment is marked by m by *m* planar grids in which each grid element is either solid or space. We simulated learning cooperatives and then transferred the information to the actual experimental testbed; in our tests, we used a tournament game playing between two teams of robots.

Results when playing the game of capturing the flag [8] have been reported elsewhere and it was noted that interesting and unexpected scenarios have been observed. For example, in some instances, without even teaching the robots, we noticed that one group of robots evolved the strategy of dividing up into smaller groups whereby one group would try to block the opposing team while the other group would try to capture the flag. In this paper, we continue with our efforts by focusing on modifying the neural network controller.

3. DEVELOPING THE NEURAL NETWORK CONTROLLER

To begin the development of the mental function architecture, we employ an artificial neural network as the controller (response processing agent) for the robot. This is now discussed.

3.1. Neural Network Inputs Processing

The EvBot's USB camera is used to capture a 640 by 480 pixel image of its environment as shown in Figure 3a. A range-finding system was created to identify object types in addition to distance. This vision system takes advantage of fixed geometric elements with the physical environment to calculate the ranges and angles of obstacles and goals. The walls, goals and environment flooring are known fixed colors. Using this information, the vision system can distinguish between the following object types: obstacles, red goal, environment flooring, and environment walls. For each

pixel, the Sum of the Squared-Errors (SSE) is computed from known RBG (Red, Green, and Blue) values from the four distinct object types using (1). The pixel is identified as being part of the object with the lowest SSE. Figure 3b displays the robot environment after the object identification process. The goal and wall obstacles are visible only after the object identification step. All other objects were converted to NULL pixels. The wall obstacles and goals are of a constant known height. The distances can be calculated from an image by determining the relative width of the colored bands within the image. The vertical sum of pixels, □p, of each object type is calculated and stored in a separate array covering the horizontal spread of the image. These numerical arrays are then fed element by element through a simple distance formula (2) to produce the final vectors of ranges **d** for each object type where **H** is the physical height of each object type and **K** is an empirically derived constant. **H** and **d** are in length units (inches). The final form of the data is (for each object type) a vector of numbers spanning the horizontal spread of the original image, where each number represents the distance of the closest object of that type in that direction. If no object is detected, the maximum sensor range is returned.

$$SSE_{DesiredObject} = \frac{1}{2}\sum^{r,g,b}[Pixel_{Actual} - Pixel_{Desired}]^2 \quad (1)$$

$$d = K\frac{H}{\sum p} \quad (2)$$

Both range and angles of detectable objects are reported over a spread of 48 degrees centered on the forward direction of the robot frame reference. From the 640 by 480 pixel images, a vector of 480 range values is produced for wall obstacles and goal objects shown in Figure 3c. In this figure, the black lines represent the wall ranges and the red lines represent the gold ranges. The controllers are only provided with the resulting numerical data vectors. All distances, angles, and object types must be learned by the neural networks. Object data vectors are always presented to the networks in the same order so a particular scalar input resulting from the distance of an object type in a particular direction will always be presented to the same input. Using all 480 elements from both the obstacle and goal object as inputs into the neural network would produce large size networks with very long training times. To reduce the size of the networks, the inputs were sampled so that 32 elements from both object types were used to reduce the number of inputs to 64 shown in Figure 3d.

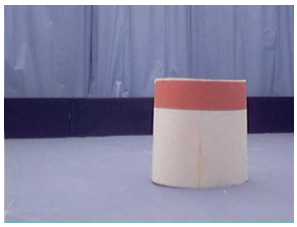
3.2. Human Reaction Monitoring

A server/client interface, shown in Figure 5, has been developed to maneuver the Evbot II remotely. This software makes use of TCP sockets and the 802.11 wireless interfaces to synchronize captured images and motor commands between the mobile robot and the Mobile Robot Network Manager Client. The client makes use of Microsoft Direct X libraries to capture the input from the game controller's devices. In

this research, an aviator style joystick with optional force feedback was used to wirelessly maneuver the robot.

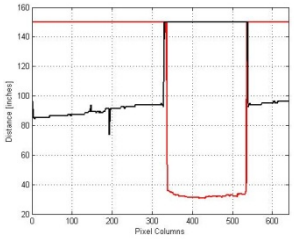
The client establishes a connection with the server running onboard the EvBot II and continually requests images. A 24 bit RGB raw frame is captured from the USB webcam and compressed using jpeg standards. The image is compressed to minimize data transfer. It was experimentally determined that the compression is needed to successfully synchronize the image at 1 to 2 frames per second depending on client processing speed and network limitations. After the synchronization, the range finding emulation system is used to produce the neural network inputs. It was determined that the compression had very little impact on the calculation of the goal and obstacle ranges.

The joystick magnitude and direction is captured and decoded to the robot motor commands. A slight backward joystick movement will slowly reverse the robot while a forceful forward joystick movement will result in forward robot behavior. The motor command is sent to the robot for execution and contains the following elements: m1pwm, m2pwm, m1dir, m2dir, where m1pwm is the pulse width modulation (pwm) value for motor 1, and m1dir is the corresponding direction in which the motor should turn. The user indicates that the robot is at the goal by pressing any button on the joystick. The motor pwm and direction values are stored along with the corresponding goal and obstacle ranges in a spreadsheet. The client exports the spreadsheet using the csv (comma-separated values) format so it can be easily imported to data mining software such as the Matlab Neural Network Toolbox, The University of Waikato WEKA, or the authors' back-propagation neural network library.

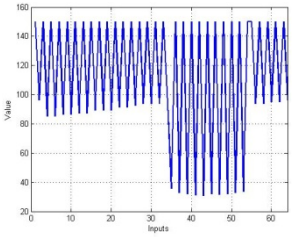


(a) Environment captured by Mobile Robot

(b) Processed Image



(c) Obstacle/ Goal Range



(d) Neural Network Inputs

Figure 3: Neural Network Preprocessing

3.3. The Neural Network Architecture

For completeness sake and to introduce the notation used in this paper, a brief summary of the artificial neural network (ANN) architecture is presented. In the work presented here, it was sufficient to use back-propagation, although more sophisticated methods may be employed if necessary.

At each layer, the output of a unit j is calculated as:

$$O_j = f\left(\sum_{i=1}^{k} O_i \, W_{ij} + \theta_j\right) \qquad (3)$$

where O_i and O_j are the output of nodes i and j, w_{ij} is the connection strength from node i to node j, and θ_j, is the internal threshold of node j. The function f(x) is the neuron evaluation function selected as a sigmoid function here. A linear function then transforms the input value into continuous ranges of value.

In the back-propagation phase, the weights and the internal thresholds are modified using the errors between the desired and actual outputs. More specifically, they are adjusted by the following equations:

$$\Delta\omega_{ij}(t+1) = \eta\, O_i \delta_j + \alpha \Delta\omega_{ij}(t) \quad (4)$$

$$\Delta\theta_j(t+1) = \eta\, \delta_j + \alpha \Delta\theta_j(t) \quad (5)$$

where η is a small positive constant called the learning rate, δ_i is the error signal at node j, and α is the momentum coefficient that determines the effect of past learning on the current weight changes. If node j is at the output layer, the error signal δ_j is calculated as:

$$\delta_j = O_j \left(1 - O_j\right)\left(O_j^d - O_j\right) \quad (6)$$

where O^d_j is the desired output for node j. If node j is at any of the hidden layers, the error signal is calculated as:

$$\delta_j = O_j \left(1 - O_j\right) \sum \delta_i\, \omega_{ij} \quad (7)$$

The feedforward phase and the back-propagation phase are repeated until the errors are below some desired threshold. One may also employ a generalized adaptive neural fuzzy inference system (GANFIS) which combines ANNs and fuzzy inference [24] for control.

3.4. Input/Output Mapping

Due to the unclear relationship between actuator signals and goal and obstacle range, developing a rule base controller may become complex. A machine learning algorithm is applied to reduce the complexity and to automatically map relationships between inputs and outputs. In many cases, the actuator signals that will result in a good expression of a complex behavior are generally not known. Hence, it is not possible to explicitly formulate an error back-propagation training scheme to train controller structures. In this particular case, an expert controller (human) performs the complex behavior using the joystick for the robot to reach its goal. Future research will add other stimulus processing agents as well as heuristics.

The training set is created by randomly initializing the robot location and orientation in its environment. The human operator then navigates the robot to its goal. This procedure is repeated to create a training set of 117 samples. A neural network architecture is used on the training samples to determine the weight values to produce the desired motor commands. In this architecture, all of the neurons are connected to each other in a brute force fashion with a variable number of hidden neurons and a variable number of output neurons and types. The initial weights in the network were initialized to a random small number and the inputs were normalized to prevent the network from saturation during training. The inputs were normalized by dividing all

input values by the maximum sensor range value. Eleven training runs were performed to produce the desired input/output mapping. Table 1 displays the results of the effects of varying neuron types and number of hidden neurons on the input/output mapping. The multi-output training took approximately 24 to 72 hours to propagate through 100,000 epochs, while the single output architecture reaches its training goal in 5 to 7 hours.

The results of the training reveal the complexity of mapping all five outputs at once. The single output architecture produced a much lower SSE than the multi-output architecture, regardless of the number of hidden neurons and type of output neurons. The linear output neurons also seem to produce slightly better mapping than neurons using the sigmoid function.

3.5. Post Processing

Post processing of the network outputs is kept to a minimum. The outputs are rounded to the nearest integer and the thresholds kept at minimum and maximum values to meet the motor control specifications. Post processing affects the network SSE using the training samples. A summary of these results are shown in Table 2. For the sigmoid and multi-output networks, the post processing may hinder performance; but for single linear output networks, performance is enhanced.

4. EXPERIMENTAL RESULTS

An "Autodrive" program was implemented onboard the mobile robot. This software recreates the neural network using weights from the training samples. This software interacts with the range-finding system to locally detect wall and goal ranges. The system then sends the ranges through the neural network to predict the desired motor command. This command is sent directly to the motor to navigate the robot. This process is repeated until the neural network output predicts a goal. Figure 4 displays the robot behavior using the "Autodrive" algorithm. In this experiment, the robot was initialized at a position where the goal was slightly visible; then the robot successfully navigated to its goal. Figure 5 shows the Network Management Client User Interface for the experiments. In other experiments, the robot behaves in the similar manner if the goal was initially visible.

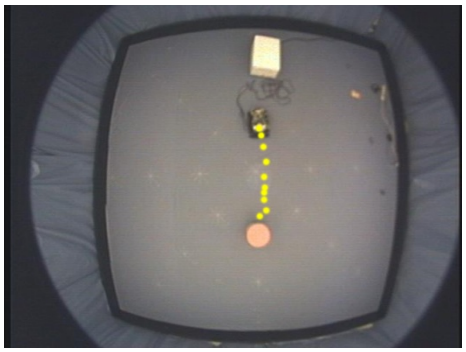


Figure 4: Path Tracking View from Overhead Camera

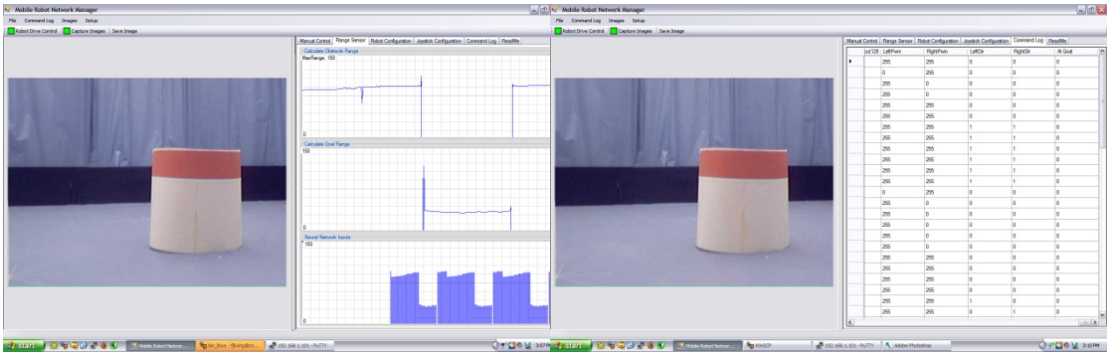


Figure 5: Mobile Robot Network Manager Client User Interface

5. CONCLUSIONS

In this paper we present a different approach to vision-guided mobile robot navigation. Instead of using a rule-based controller, a neural network controller was trained, mapping a human-operated joystick command to capture robot images. Using a back-propagation gradient descent training algorithm, a controller was developed to successfully navigate the robot to its goal. Results show that this simple yet effective approach produces reasonable performance. Future research involves characterization of the newly designed controller to the rule base controllers and evolutionary algorithm controllers previously designed as well as the development of additional sensors for both human and robot in order to capture how the mental functions of humans operate in a goal-oriented scenarios which can then be transferred to mobile robots leading to enhanced human-robot interaction.

Further, one approach that may be employed to generate learning is the use of a KASER [23] which has shown promise as a heuristics-based architecture, using randomization and symmetry for learning. This architecture also allows the building of a knowledgebase (KPA) and possibly feedback to the EPA as more experience is

gained (more robot runs are performed). These tools can be integrated into the testbed to improve the transference of intelligence between human and robot.

6. REFERENCES

[1] D. Filliat, J. Kodjabachian, and J. A. Meyer, "Incremental Evolution of Neural Controllers for Navigation in a 6 Legged Robot", Sugisaka and Tanaka, editors, *Proc. of the Fourth International Symposium on Artificial Life and Robotics*, Oita Univ. Press, 1999.

[2] N. Jakobi, "Running Across the Reality Gap: Octopod Locomotion Evolved in a Minimal Simulation", *Proc. of the First European Workshop on Evolutionary Robotics: EvoRobot'98*, 1998.

[3] D. Floreano and S. Nolfi, "Adaptive Behavior in Competing Co-evolving Species", *Mantra Technical Report*, LAMI, Swiss Federal Institute of Technology, Lausanne, 1997.

[4] D. Floreano and F. Mondada, "Evolution of Homing Navigation in a Real Mobile Robot", *IEEE Transactions on Systems, Man, Cybernetics Part B: Cybernetics*, Vol. 26 No. 3, pp. 396-407, 1996.

[5] D. Floreano and S. Nolfi, "Adaptive Behavior in Competing Co-evolving Species", *Mantra Technical Report*, LAMI, Swiss Federal Institute of Technology, Lausanne, 1997.

[6] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics", F. Moran, A. Moreno, J. Merelo, and P. Chacon, Editors, *Advances in Artificial Life:* Proc. 3rd European Conference on Artificial Life, pp. 704-720. Springer-Verlag, Lecture Notes in Artificial Intelligence 929, 1995.

[7] R.A. Watson, S.G. Ficici, J.B. Pollack, "Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots", *Robotics and Autonomous Systems*, Vol. 39, No. 1, pp 1-18, April 2002.

[8] A. L. Nelson, E. Grant, G. J. Barlow, and T. C. Henderson. "A Colony of Robots Using Vision Sensing and Evolved Neural Controllers." *Proc. of the IEEE Conference on Intelligent Robots and Systems*, Las Vegas, NV. October 2003.

[9] E. Grant, L. Mathos, G. J. Barlow, A. L. Nelson, K. Luthy, B. Levedahl, and G. Lee, "Evolutionary Neural Controllers for Mobile Robot Colonies", *Proc. of the World Automation Congress*, ISSCI, Seville, 2004.

[10] M .J. L. Boada, R. Barber, V. Egido, and M. A. Salichs, "A Continuous Reinforcement Learning Algorithm for Skills Learning in an Autonomous Mobile Robot", *Proc. of the 28th Annual Conference of the IEEE Industrial Electronics Society*, Sevilla, pp. 2611-2616, 2002.

[11] L. Yibin, L. Caihong, and Z. Zijian, "A Q-Learning Based Method of Adaptive Path Planning for Mobile Robots", *Proc. of IEEE International Conference on Information Acquisition*, Weihai, China, pp. 983-987, 2006.

[12] J. H. Kim, J. B. Park, H. S. Yang, and Y. P. Park, "Generation of Fuzzy Rules and Learning Algorithms for Cooperative Behavior of Autonomous Mobile Robots(AMRs)", *Proc. of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery*, Changsha, China, pp. 1015-1024, 2005.

[13] R. Barate and A. Manzanera, "Learning Vision Algorithms for Real Mobile Robots with Genetic Programming" *Proc. of the 2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, Edinburgh, pp. 47-52, 2008.

[14] Y. Demiris, Y., "Prediction of Intent in Robotics and Multi-Agent Systems", *Cognitive Processing*, Vol. 8, pp. 151-158, 2007.

[15] K. Gold, "An Information Pipeline Model of Human-Robot Interaction", *Proc. of the 4th ACM/IEEE International Conference on Human-Robot Interaction, HRI'09*, pp. 85-92, 2008.

[16] T. Kooijmans, T. Kanda, C. Bartneck, H. Ishiguro, and N. Hagita, Norihiro, "Interaction Debugging: An Integral Approach to Analyze Human-Robot Interaction", *Proc. of the 2006 ACM Conference on Human-Robot Interaction*, pp. 64-71, 2006.

[17] L. Mattos, "The EvBot II: Enhanced Evolutionary Robotics Platform Equipped with Integrated Sensing for Control", in *Electrical Engineering Report*, North Carolina State University: Raleigh. pp. 194, 2003.

[18] A.L. Nelson, "Competitive Relative Performance and Fitness Selection for Evolutionary Robotics", in *Electrical Engineering Report*, North Carolina State University: Raleigh. pp. 188, 2003.

[19] S. Hashimoto, "KANSEI Robotics to Open a New Epoch of Human-Machine Relationship - Machine with a Heart", *Proc. of the IEEE International Workshop on Robot and Human Interactive Communication*, pp. 1, 2006.

[20] T. Sato, M. Hashimoto, and M. Tsukahara, "Synchronization-Based Control Using Online Design of Dynamics and Its Application to Human-Robot Interaction", *Proc. of the IEEE International Conference on Robotics and Biomimetics, ROBIO,* pp. 652-657, 2007.

[21] E. Tauchi, K. Sugita, G. Capi, and M. Yokota, "Towards Artificial Communication Partners with Kansei", *Proc. of the International Conference on Advanced Information Networking and Applications, AINA,* vol. 2, pp. 688-692, 2006.

[22] Nelson, A.L., Grant, E. and Lee, G., "Using Genetic Algorithms to Capture Behavioral Traits Exhibited by Knowledge Based Robot Agents", *Proc. of the ISCA Int'l Conference on Computers in Industry and Engineering*, San Diego, 2002.

[23] S.H. Rubin, G. Lee, W. Pedrycz, and S.C. Chen, "Modeling Human Cognition Using a Transformational Knowledge Architecture", *Proc. of the IEEE Intl. Conference on System of Systems Engineering (SoSE*), Monterey, CA, 2008.

[24] G. Lee, E. Grant, and P. Tang, "On the Design of An Adaptive Neural Network Fuzzy Inference Controller Using Heuristics and Predictive Tuning", *Proc. of the IEEE International Conference on Distributed Human-Machine Systems*, Athens, 2008.

Table 1: Network Training

Net	Hidden Neurons	Hidden Neuron Type	Outputs	Output Neuron Type	SSE	Epochs
1	32	Sigmoid	All	Sigmoid	341.414	100000
2	10	Sigmoid	All	Sigmoid	191.404	100000
3	128	Sigmoid	All	Sigmoid	215.305	100000
4	256	Sigmoid	All	Linear	10021.6	100000
5	256	Sigmoid	m1pwm	Linear	0.00590	100000
6	256	Sigmoid	m2pwm	Linear	0.00600	15159
7	128	Sigmoid	m1dir	Linear	0.17625	100000
8	64	Sigmoid	m2dir	Sigmoid	4.97000	24639
9	128	Sigmoid	m2dir	Linear	0.00267	400376
10	32	Sigmoid	Goal	Sigmoid	0.90518	100000
11	64	Sigmoid	Goal	Linear	0.82893	100000

Table 2: Post Processing Effect on Error

Network	SSE	POST PROCESSED SSE
1	341.4136	349.0000
2	191.4040	214.5000
3	215.3054	216.5000
4	10021.5660	10123.0000
5	0.00590	0.0000
6	0.00600	0.0000
7	0.17625	0.0000
8	4.97000	6.0000
9	0.00267	0.0000
10	0.90518	1.0000
11	0.82893	1.0000