Reinforcement Learning in Sensor-Guided AIBO Robots

Nicole Birdwell
Department of Mechanical, Aerospace, and Biomedical Engineering
University of Tennessee
1512 Middle Dr.
Knoxville, TN 37996, USA
Email: cbirdwel@utk.edu

Scott C. Livingston
Department of Electrical Engineering and Computer Science
University of Tennessee
1508 Middle Dr.
Knoxville, TN 37996, USA
Email: sliving2@utk.edu

Itamar Elhanany
Department of Electrical Engineering and Computer Science
University of Tennessee
1508 Middle Dr.
Knoxville, TN 37996, USA
Email: itamar@ece.utk.edu

August 10, 2007

***Abstract*—A common challenge in autonomous mobile applications is maintaining controlled movement of an agent on unknown or unanticipated surfaces. The Tekkotsu application framework, built to facilitate research with the Sony AIBO robot, has basic walking functions but does not include any adaptive behavior. Through the use of an indoor vector-tracking system, we attempt to implement a reinforcement learning algorithm, Q-learning, to intelligently learn new surfaces and thus improve the existing walking mechanism.**

I. Introduction

With the advance of technology, wireless sensor networks are becoming more and more useful. These networks are used for a variety of purposes including technological advances in medicine, military, traffic, and manufacturing. As the demand for these networks increases, the need for more accurate and intelligent designs rises.

Benefits of wireless sensor networks incorporate ease of operation, simplified systems, compact sensors, and cost efficiency. Challenge arises in developing simplified modifications to wireless nodes and systems. Researchers and developers strive to design inexpensive and efficient networks that ultimately benefit society.

Sony's development of the AIBO robot eases the development of many experiments including those containing wireless sensor networks. The AIBO's agility and multi-faceted performance provide a flexible environment for research. Due to high demand, Sony released a low-level open source toolkit called OPEN-R to provide access to the AIBO hardware for software developers. The Tekkotsu application development framework [1], developed at Carnegie Mellon University (CMU), is built on top of OPEN-R to provide a richer and higher-level environment for programming the Sony AIBO. The implemented walking system, however, does not include any dynamic movement adjustments for various walking surfaces, and thus often fails to perform correctly on difficult surfaces, particularly those with low coefficients of friction. An attempt to solve this problem is presented by adding an intelligent feedback layer on top of the current walking system through a reinforcement learning algorithm, specifically Q-learning.

The general goal is to create an autonomous walking robot that is able to correct its step movements to follow a specified path. A program is developed to build a table of surface-dependent values intended to enhance the robot's decision making when choosing a specific action. The program uses a wireless sensor network to determine the location of a mobile robot. The mobile robot, a Sony AIBO (robotic dog), is programmed to walk in a line along a desired path. Reinforcement learning, a type of machine learning, creates a system of rewards based on an array of combinations for predefined states and actions. This information is used to aid the robot in following the desired path.

A long-term goal is to enhance the software to create a highly versatile program that will allow the robot to choose paths efficiently in unknown environments.

II. Laboratory Environment and Background Information

The Sony AIBO robot and Massachusetts Institute of Technology (MIT) Cricket wireless sensor nodes are conjunctively deployed to create an autonomous walking agent capable of correcting actions to strictly follow a desired path. The setup includes the Sony AIBO, Tekkotsu software, MATLAB software, and the MIT Cricket system.

A basic foundation for commanding the Sony AIBO robot is laid out in the Tekkotsu software developed at CMU [1]. Tekkotsu provides access to many high-level commands for the Sony AIBO including movement, visual, and behavioral controls which can be studied on the Tekkotsu website. Specifically for this project, Tekkotsu aids in controlling the walking parameters of forward velocity, angular rotation, and side-stepping of the robot.

A MATLAB interface is used to simplify the deployment of

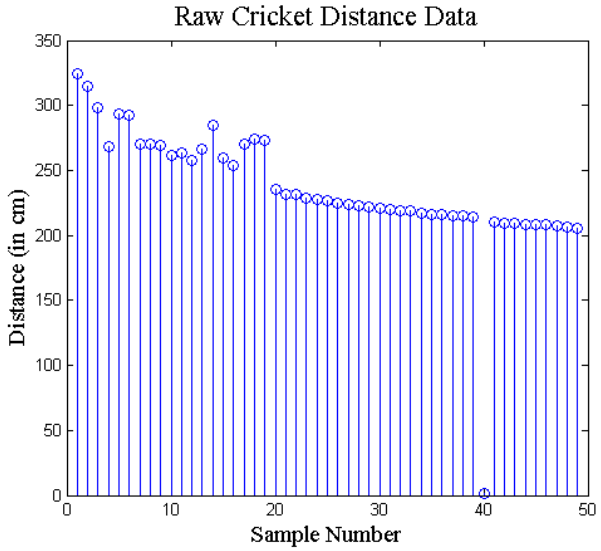

Fig. 1. Raw measurements of distance between a beacon node and a listener node.

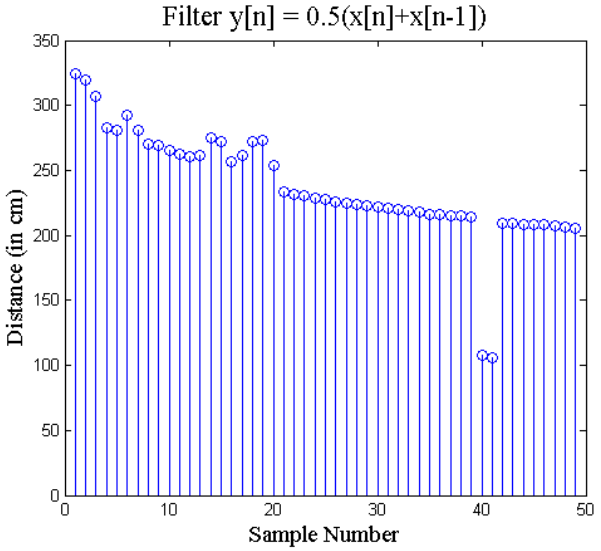

Fig. 2. Filtered measurements of distance between a beacon node and a listener node.

Tekkotsu. The interface, designed at University of Tennessee at Knoxville by Dr. Itamar Elhanany and students Richard Wunderlich, Travis Goodspeed, allow all aspects of the project to be run and analyzed entirely with the MATLAB software environment (produced by The MathWorks, Inc.). In short, a Java class is written to facilitate creation of a telnet connection through MATLAB to an extension of the Tekkotsu software, called a "behavior," which enables control of the walking parameters of forward, rotational, and perpendicular velocities [2].

The MIT Cricket system consists of wireless nodes that can act as either listeners or beacons [3]. The beacons operate by simultaneously emitting radio waves and ultrasonic pulses, which are received by the listener nodes. The difference in the speed of sound and the speed of light allows the distance of the beacon node to be determined by the listener node using the time delay between arrivals of the radio wave and ultrasound signals. This measured distance is accurate to about 1 cm for stationary nodes [4], though in testing this project it is found that distance error is usually about 3 or 4 cm for moving nodes.

The default arrangement of Cricket devices, that is stationary beacons and mobile listeners, can not be used in this setup since the listeners passively calculate distance measurements, while a component external to the AIBO requires live access to distance data. A modification of the Cricket system to allow tracking of a vector, consisting of two beacons, through space was designed by Richard Wunderlich and several others in [5]. Their vector-tracking system is based on trilateration, similar to GPS [6]. Operation involves three stationary listeners of known relative positioning acquiring distances of two mobile beacons and relaying this data to one of the listeners which is attached to a computer. Finally, software on the computer, such as MATLAB, can use this distance data and the known listener positions to determine coordinates for each of the two beacons, forming a vector. The paper that introduces this tracking system includes an argument that an active mobile architecture (moving beacons) helps meet the so-called simultaneity condition for trilateration, thus increasing position accuracy. Whether this improvement outweighs loss of some original Cricket design goals, particularly privacy, is addressed in [7]. However, for this application, the requirement of external computation is sufficient to select an active mobile architecture.

Initially, three listener Cricket nodes are placed to form a coordinate axis on the ceiling enclosing an area of approximately 3 square meters, and two Cricket beacon nodes are attached to the moving Sony AIBO. To ensure precise time readings of radio and ultrasonic transmissions, the beacons must be pointed directly at the listeners. As the angle of deviation of the beacon's ultrasound transmitter from the listener's receiver increases, the error of beacon distance calculations also increases. In certain positions on the floor, the inaccuracy from AIBO-mounted beacon to listener is substantial, and regions exist where only two of the listeners can hear a beacon, thus preventing calculation of that beacon's coordinate. To address these problems, a fourth listener node is added for redundancy; overall, the four ceiling-mounted listeners form a 152 cm by 183 cm rectangle. Several additions to the original vector-tracking system software are made to handle the new fourth stream of distance measurements. After testing, this redundant node greatly reduces the presence of "dead spots".

Another problem, inherent to the Cricket system, is random noise in distance measurements. Major outliers, on the order of 20 cm from an expected coordinate, are easily detected and filtered. However, smaller variations in beacon-listener distances cause errors in trilateration results which are not easily distinguished from correct coordinates. To reduce the effect of this "coordinate noise," a filter is applied directly to raw beacon-listener distance data as they arrive to the external

Fig. 3. The laboratory setup for this project. Cricket listener nodes were attached on ceiling above testing surface.

controlling software. Figure 1 contains an example of consecutive readings for a specific beacon by one of the listener nodes. The cause of "coordinate noise" occurs during the first 19 measurements of the example; the major outlying distance at sample number 40 is easily detected. Filters tested to reduce noise are a running mean of the current and previous distances, a running mean of the last three distances, and a predictive filter based on linear progression of previous two distances. For this application, the results are best with the first filter, indicated in discrete-time by y[n] = 0.5(x[n] + x[n-1]) where x is the raw distance and y is the value used for processing. A figure of the result of running the samples of Figure 1 through this filter is in Figure 2. The effect of the noise in the first 19 measurements is somewhat reduced while allowing for sudden changes in walking movements to pass through the filter.

The AIBO robot "learns" how to walk on new surfaces through the implementation of Q-learning, a type of reinforcement learning. The general process involves the selection of an action based on the current state of the agent using a policy. The state in this setup consists of two components: orthogonal distance from the ideal walking path and orthogonal velocity relative to the ideal path. Thirty-five states are defined, with seven distance ranges and five velocity ranges. Five actions, which correspond to rotational velocity of the AIBO, are also defined. In Q-learning, the action-value function for state s_t and action a_t is $Q(s_t, a_t)$ defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

where s_t means the t^{th} state, α and γ are constants between 0 and 1, and r is the reward for a given state. After the detection of the current state, the Q(s,a) value for the previous state and the chosen action, which apparently led to the current state, is updated based on this formula. To maintain such values as the optimal Q function is approached, a table of Q(s,a) values is kept. In this paper, two policies are used: a reference and ε-greedy. A reference policy is useful to help an agent build a reasonable initial Q(s,a) table without directly relying on this table for action selection. The ε-greedy policy selects, for a given state s, the action a with maximum Q(s,a) with probability 1-ε (usually 0.8 to 0.9) or explores other actions (non-maximum) with probability ε (about 0.2 to 0.1). The general goal of this algorithm, as with many reinforcement learning methods, is to maximize long-term reward. A more detailed treatment of Q-learning and reinforcement learning in general can be found in [8].

All of these components work together to form a complete system that enables a walking agent, specifically the AIBO robot, to adjust to unknown surfaces. Because the AIBO cannot directly sense the direction it is facing, the ideal forward walking path is determined by acquiring several position vector samples and then creating a line in memory that represents the desired path. Thus, the accuracy of the system largely depends on the accuracy of this detected desired path. That aside, to understand the overall system, a somewhat simplified explanation of the acquisition and processing of a single detected vector is as follows. First, each of the two beacons attached to the AIBO broadcasts a radio frequency (RF) and ultrasound (US) signal pair. All four listeners hear this signal pair simultaneously and use it to calculate their individual distances to that particular beacon. Then, each listener not directly attached to a computer (three total) wirelessly (over an RF band) sends its distance data to the listener which is attached to a computer. As the distance data arrives, in the form of packets containing beacon-listener identifications and distance measurements, it is sent through a serial stream to the external computer. Next, the MATLAB script, which largely forms the heart of this system and which is listening to the serial port, receives this raw data and applies the filter to it. Once enough data measurements have arrived (at least three per beacon), the script performs trilateration to find the xyz-coordinate of each beacon, forming a vector. This coordinate is analyzed to determine the corresponding state of the agent. Using the history of previous state-action pairs and the reward detected for the current state, the table of Q(s,a) is updated. The selected policy is then applied to this state and an appropriate action is selected. This action, which corresponds to a rotational velocity, is sent to the AIBO using the MATLAB interface to Tekkotsu, and finally the AIBO adjusts its movement as commanded, repeating the selection process for the duration of the episode. Each iteration of this process, occurring for each coordinate reading and processing, takes approximately 0.61 seconds. This value represents the median time of a set of 1765 samples (mean is 0.72 seconds).

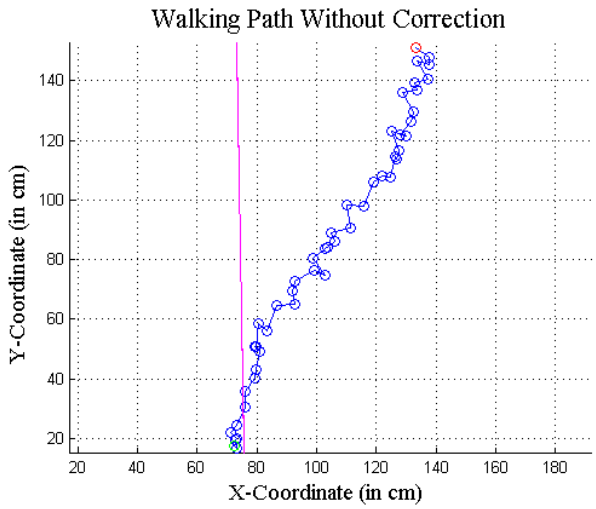

Fig. 4. The walking path of the AIBO without corrective actions. The circles represent sample coordinates while the solid line represents the desired path.

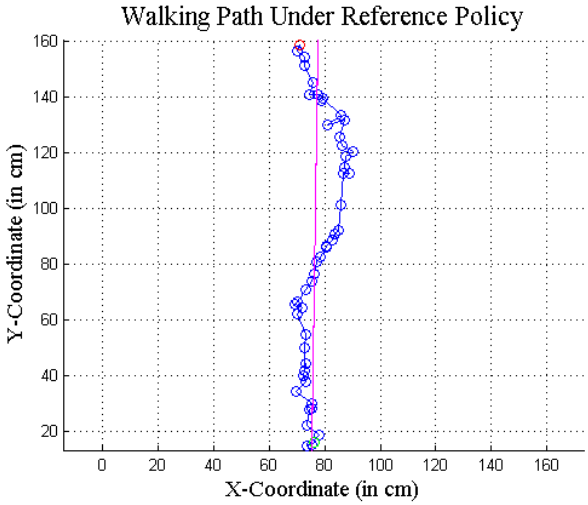

Fig. 5. The walking path of the AIBO while using the reference policy. The circles represent sample coordinates while the solid line represents the desired path.

III. Experimental Results

As touched on in the explanation of Q-learning in the "Background Information" section, training the walking agent for a given surface is performed by initially using a training reference policy to hasten convergence of values in the Q(s,a) table and then switching to an ε-greedy policy in which the agent strongly favors (with 0.8 to 0.9 probability) maximizing Q(s,a) from its current state. After the ε-greedy policy is selected, entries in the agent's Q(s,a) table approach, at least theoretically, their real values, thus optimizing reward and, accordingly, the walking behavior.

As a control, twenty walking episodes are recorded with no corrective action, which means a simple forward velocity is sent alone to the Tekkotsu software of the AIBO robot. A sample plot of such movement can be seen in Figure 4, where each circle represents a coordinate read by the vector-tracking system. The agent gradually deviates from the desired forward path, indicated by a solid line. Along with results for reference policy and ε-greedy policy performance, which will be explained in the next two sections, Table 1 contains the maximum, median, and mean orthogonal distances from the desired path during all 20 episodes. Also, Table 2 lists portions of orthogonal distances which are less than 1 cm, 3 cm, and 5 cm from the ideal path. Comparisons are made after explanations of the two policies are given.

A. Reference Policy

In Q-learning, the goal of a reference policy is to aid development of values in the agent's Q(s,a) table while preventing the agent from wandering too far from ideal behavior. In this experiment, the ideal behavior is to walk in a straight line on an unknown surface. Because there are 35 states and 5 actions in our setup (thus, 175 combinations), a chart mapping specific states to actions would be cumbersome to view. A simple equation cannot be found, so a series of "if" statements is chosen instead.

	Control (no policy)	**Reference Policy**	**ε-greedy Policy**
Mean	-2.01 cm	-3.09 cm	3.89 cm
Median	-0.0852 cm	2.11 cm	2.06 cm
Maximum	99.1 cm	34.7 cm	62.6 cm

TABLE I

Statistics for 20 episodes under each of three arrangements: with no policy, and with either the reference or ε-greedy policy. Statistics relate to orthogonal distance measurements of AIBO from desired path.

	Control (no policy)	**Reference Policy**	**ε-greedy Policy**
Less than 1 cm	13.4 %	14.9 %	12.1 %
Less than 3 cm	35.6 %	38.3 %	27.9 %
Less than 5 cm	49.3 %	54.4 %	37.7 %

TABLE II

For each policy, the percentage of orthogonal distances from desired path which are less than some value. These distances were acquired during a series of 20 episodes performed for each policy.

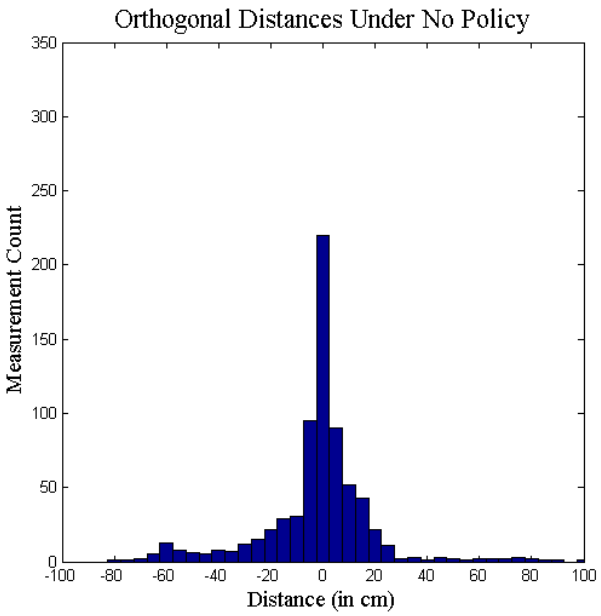

Fig. 6. Histogram of orthogonal distances read during 20 episodes with no corrective actions.

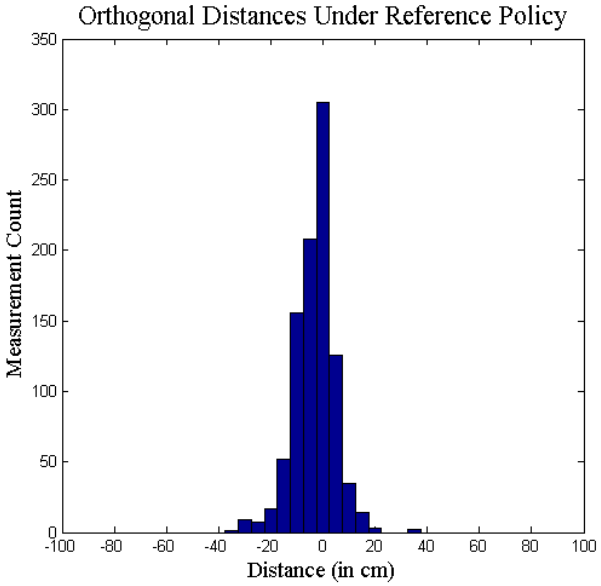

Fig. 7. Histogram of orthogonal distances read during 20 episodes operating with the reference policy.

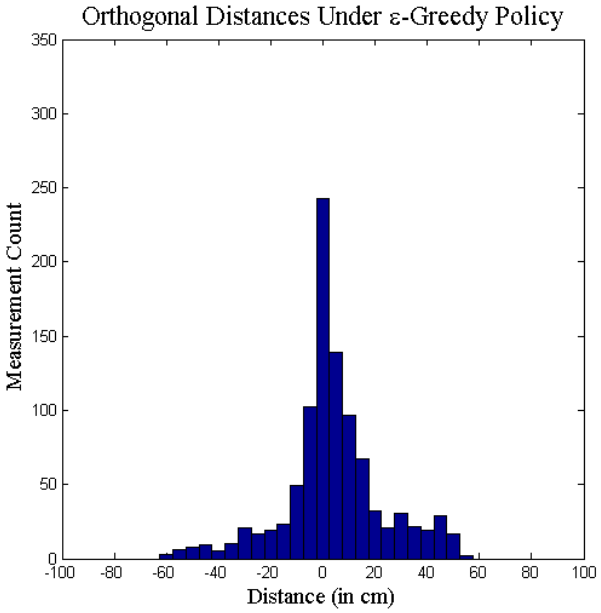

Fig. 8. Histogram of orthogonal distances read during 20 episodes operating with the ε-greedy policy.

Based on state and action definitions explained in "Background Information" section, the pseudocode for this process is listed here:

```
if current_pos = 7 and current_ov > 2
    current_rv := 1
else if current_pos = 1 and current_ov > 2
    current_rv := 5
else if current_pos == 4 or (current_pos != 4
        and current_ov < 3)
    current_rv := 3
else if (current_pos > 3 and current_ov > 2)
        or (current_pos < 5 and current_ov < 3)
    current_rv := 2
else if (current_pos < 5 and current_ov > 2)
        or (current_pos > 3 and current_ov < 3)
    current_rv := 4
```

where current_pos is the orthogonal distance portion of the current state (an integer 1 through 7), current_ov is the orthogonal velocity portion of the current state (an integer 1 through 5), and current_rv is the selected action (an integer 1 through 5 representing different rotational velocities). This operation has the general effect of choosing greater rotational velocities for greater distances from the original desired path, unless the desired path is already being approached at a large enough speed. Figure 5 contains a sample AIBO movement plot for an episode with the reference policy. The plot demonstrates that the AIBO takes corrective actions when deviating too far from the desired forward path. Overall performance of any corrective policy in this project relies heavily, if not completely, on correct initial selection of the desired path. For example, if the detected desired path is perpendicular to the heading of the agent due to vector-tracking noise, the visually observed walking performance will be very poor though the action-selection policies are performing correctly.

Tables 1 and 2 list statistics from 20 episodes with the reference policy of orthogonal distances relative to desired path. It is found generally that use of the reference policy leads to better results than operation without any corrective actions, referred to as "basic walking." Histograms of orthogonal distances for "basic walking" and reference policy episodes are in Figure 6 and 7, respectively. The histogram for "basic walking" operation has a wider spread than that of operation

under the reference policy, indicating improved walking behavior with this policy.

B. ε-Greedy Policy

After sufficient training episodes are performed with the reference policy, action selection is changed to the ε-greedy policy. Sufficiency depends on repeatable accuracy and precision of episodes, which, once achieved, indicate a strong table of Q(s,a) values. In this project 200 training runs are performed before switching to the ε-greedy policy. For a brief explanation of ε-greedy policy, see Background section above. Initially, ε is set to 0.2 to allow for more frequent exploration to previously unknown states and actions. This on occasion leads to poor performance as the AIBO robot wanders substantially from the desired path after repeated selections of random actions for non-maximum Q(s,a) values. However, since reward is greatest in a narrow band about the desired path and decreased with each state further away from this band, such deviating exploration is only temporary as the agent soon returns to maximizing Q(s,a) and thus takes actions toward the desired path. After approximately 40 episodes, ε was reduced to 0.1 in order to hasten convergence of the Q(s,a) table to actual values. The training is stopped after a total of 295 runs on the same surface.

Tables 1 and 2 list statistics based on the last 20 episodes using the ε-greedy policy. Also, Figure 8 is a histogram of orthogonal distances for these episodes. Visual inspection of Figures 6 and 8 reveals the histogram for ε-greedy has a greater density of coordinates near the desired path than the histogram for no policy (called "basic walking"). However, the performance of episodes with the reference policy seems to be better than that of ε-greedy as the reference policy succeeded in forcing the AIBO closer to the desired path. The tables of orthogonal distance also suggest the superiority of the reference policy.

To test the ability of the agent to adjust to a new surface given this project setup, 30 episodes are performed using the same Q(s,a) table from the original surface and the ε-greedy policy. The new surface is the "white surface," and the original surface is the "black surface." Interestingly, since the AIBO walking software uses the forearms of the front legs (made of smooth plastic) and the feet of the rear legs (made of a rubber-like material), the rear legs have better traction on the white surface than on the black surface, while the front legs have better traction on the black surface than on the white surface. However, the rear legs have better traction than the front legs overall. To aid in analysis, several statistics for the orthogonal distances of each episode are compared across episodes. The statistic of minimum for absolute value of orthogonal distance has a correlation coefficient of -0.3800, which indicates the table of Q(s,a) values is approaching correct values for the new surface. Also, the statistics of mean and median have correlation coefficients of -0.3628 and -0.3484, respectively, which also suggest Q(s,a) values are adjusting to the peculiarities of the new surface. However, the magnitudes of these correlation coefficients are small and thus may not indicate a trend. For comparison, taking similar statistics for the last 72 episodes with ε-greedy policy on the black surface, maximum and mean correlation coefficients have magnitudes less than 0.16, and minimum and median correlation coefficients have magnitudes less than 0.09. Given the random error of approximately 3 to 4 cm for vector-tracking and the attempt to closely follow the desired path, these correlation parameters reveal the variation of detected coordinates within a band about the desired path. The AIBO is typically within 15 cm of the desired path on the black surface, indicating the Q(s,a) table is near optimal values for that surface.

IV. Conclusion and Future Work

The reference policy performed with more precision than the ε-greedy policy on the original learning surface. Yet the benefits of the ε-greedy policy exceed the reference policy when operating on unknown surfaces. While the reference policy can potentially lead to more precise results, the ε-greedy policy is more efficient in adapting to new environments. Because of the on-board “knowledge” in the AIBO, the ε-greedy technique has the absolute advantage over time-consuming modifications to the reference policy for each new surface.

The two major sources of error are noisy Cricket readings and incorrect selection of the desired path. Because of jarring movements in the AIBO, the two beacons attached on the walking agent often do not directly face listener nodes, leading to signal power attenuation that results in random variation measurements unable to be easily filtered. Results of any given episode rely heavily on correct detection of the forward direction of the AIBO. If this direction is flawed, due to noise in the vector-tracking system, the desired path is not properly established. Even though this path is followed according to system specifications, the path itself is wrong and thus the operation of the system as a whole is erroneous. To reduce the probability of poorly establishing a desired path, the median of 9 samples of the AIBO vector is acquired before walking begins. Still, deeply flawed paths are chosen, though not often.

Future work on this project includes increasing the number of Cricket listener nodes to reduce the presence of “dead spots,” where coordinates cannot be read. Also, this reinforcement learning system could be applied to the action of sidestepping, that is walking perpendicular to the direction faced by the AIBO. Further testing of response of the ε-greedy policy to an irregular obstacle would be useful since most natural terrains include many irregularities.

The results of this project can be applied generally to any situation involving an autonomous agent maneuvering through an unknown terrain where coordinate information is available. It is also readily useful to anyone interested in utilizing the subsystems of this project (Tekkotsu, Cricket, and so on) to control an AIBO. Future work is simplified when employing this walking methodology to achieve intelligent behavior.

This setup incorporates more flexible control software

aiding the AIBO's ability to walk along a desired path. While some reference policy results show minimized deviation from desired path, the ε-greedy policy has greater advantage in its ability to explore unknown states. The versatility of this project design lays the foundation for future work. Several different platforms are combined into a single system entirely accessed through MATLAB. This facilitates rapid implementation and testing while allowing for conversion to other software languages.

V. Acknowledgment

The authors of this paper wish to acknowledge the National Science Foundation for providing the grant and the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville (UTK) for making this opportunity possible.

Also, special thanks is given to the helpful members of the Machine Intelligence Laboratory (MIL) at UTK and Dr. Itamar Elhanany for mentorship and academic guidance.

Finally, this project would not be possible without the Cricket Location-Support System from MIT, Tekkotsu software from CMU, and TinyOS from the University of California at Berkeley, all of which are well-supported and open source.

References

[1] Tekkotsu application development framework, Carnegie Mellon University, http://www.cs.cmu.edu/~tekkotsu/index.html.

[2] T. Goodspeed, R. Wunderlich, I. Elhanany, "WIP: Enhancing Reinforcement Learning Class Curriculum using a Matlab Interface Library for use with the Sony AIBO Robot," to appear in the ASEE/IEEE 2007 Frontiers in Education Conference, October, 2007.

[3] N. B. Priyantha, A. Chakraborty, H. Balakrishnan, "The Cricket Location-Support system," Proc. 6th ACM MOBICOM, Boston, MA, August 2000.

[4] H. Balakrishnan, R. Baliga, D. Curtis, M. Goraczko, A. Miu, N. B. Priyantha, A. Smith, K. Steele, S. Teller, K. Wang, "Lessons from Developing and Deploying the Cricket Indoor Location System," available at http://cricket.csail.mit.edu/, November 2003.

[5] R. Wunderlich, V. Mahoney, and Z. Liu, "3-D Vector Tracking Using Cricket Motes," class project for ECE 599-012, Department of Electrical Engineering and Computer Science, University of Tennessee at Knoxville, May 2007.

[6] M. Brain and T. Harris, "How GPS Receivers Work," September, 2006, http://electronics.howstuffworks.comgps.htm.

[7] A. Smith, H. Balakrishnan, M. Goraczko, N. Priyantha, "Tracking Moving Devices with the Cricket Location System," Proc. 2nd USENIX/ACM MOBISYS Conf., Boston, MA, June 2004.

[8] R. S. Sutton and A. G. *Barto, Reinforcement Learning: An Introduction*, MIT Press, 1998.