Work in Progress - Enhancing Reinforcement Learning Class Curriculum using a Matlab Interface Library for use with the Sony AIBO Robot

Travis Goodspeed, Richard Wunderlich, Itamar Elhanany
Department of Electrical and Computer Engineering
University of Tennessee
Knoxville, Tennessee 37996
travis@utk.edu, rwunderl@utk.edu, itamar@ieee.org

Abstract - **In an effort to mitigate the inherent complexities of embedded robotics programming, a novel Matlab-based interface library with matching firmware is presented. The platform was developed at the University of Tennessee for use in a course on reinforcement learning. Using this software interface, students were able to easily implement machine learning schemes and apply them to the Sony AIBO robot, a flexible platform with diverse capabilities. The behavior development process for the AIBO is traditionally limited to programming in a C++ environment. To avoid the overhead of such tedious programming, the Matlab interface was designed to allow the user to both implement functions and behaviors already developed for the AIBO, as well as to construct new behaviors using existing components as building blocks. Moreover, since the interface relies on a bidirectional network socket connection, the user is offered full control of the robot from any (remote) networked computer.**

Index Terms - machine learning curriculum, reinforcement learning, robotics labs.

INTRODUCTION

Reinforcement learning (RL) is an important discipline within the field of machine learning, which focuses on systems that learn by interacting with their environments [1]. Beyond the theoretical components of a typical RL course curriculum, an imperative aspect of learning about RL systems pertains to obtaining hands-on experience with robotic platforms that help clarify the course material. The Sony AIBO robot has recently been extensively used in machine learning courses. However, the programming interface that is typically used when working with the AIBO is limited to a C++ environment. This results in students spending a substantial amount of time on programming the interface to the robot rather than focusing on machine learning algorithms.

The field of embedded robotics has long been an advanced topic in education due to the complex and varied nature of the subject. Students are required to not only understand how the hardware works, often at the logic level, but must also be fluent in high-level programming or even assembly language. Moreover, a working knowledge of the robot's electromechanical systems is beneficial as means of properly governing aspects of the physical control. Only in recent years have embedded robotic systems grown to include such advanced technology as wireless network connections, large flash memory storage, and 32-bit processors that were previously reserved for high-end computer platforms.

This paper describes a Matlab-based library that was designed for use in a course on RL. The goal of the effort was to offer a seamless interface to the Sony AIBO robot from the Matlab programming environment. The library enabled students to read the robot's sensory information (including buttons and visual signals) as well as issue it control commands. The embedded wireless networking capabilities of the AIBO facilitated convenient investigation of various machine learning techniques from a remote PC workstation. This streamlined the development of student projects by allowing them to avoid the hassles of traditional embedded development, particularly the cumbersome testing cycle of recompiling, writing to the flash memory, and rebooting the robot. Instead, students simply connect to the robot over the wireless network and execute Matlab scripts. That way, an error in the code can only terminate the client socket, not the robot run-time environment, such that only Matlab has to be restarted. This reduces hardware crashes to a minimum. Individual statements may be executed in the Matlab console to test their impact instantly, without requiring a reboot procedure. In the event of misbehavior, a student may use Matlab's built-in debugger or dump the value of a global variable with no greater effort than he or she would with any other scripts. Further, the script may be debugged with a physical robot, while with traditional C++ AIBO development, debugging may be more easily performed using a simulator.

CLIENT ARCHITECTURE

The client is, in essence, any application capable of sending data over a network socket connection. We have employed Matlab to perform this function for several reasons. First, it allows for complex loops and various programming logic functionality to be coded easily using a single script or multiple scripts, which allows for code modularity and reuse. Second, Matlab has the unique ability to visualize data in a vast variety of easily-accessible ways. Using the built-in functions and toolbox add-ons, graphs and images can be created with a minimum amount of effort. Finally, Matlab is a convenient choice as virtually all engineering students have been exposed to it, often as early as in the freshmen year.

Our system is built as a small extension to Tekkotsu [2], an open source development framework for the Sony AIBO. We have added a class to Tekkotsu's AIBO firmware to allow for servo motor positions to be set over the network, and we have constructed a complimentary client in Matlab, with hooks to Tekkotsu's own Java client classes. The client system currently consists of two major components: robot vision and motion control.

I. Robot Vision

Streaming image data is read from the AIBO's camera by way of the provided class from Tekkotsu's Java library. Using the latter, images are read from the AIBO directly into an array in Matlab. At that point, any number of image processing techniques can be employed to extract data from the image. The robot need only acquire the image and send the data over the network, shifting the computational load to a remote PC.

II. Robot Motion Control

Motion is handled by a simplistic networking class, which abstracts details of the network connection away from Matlab. For motion control, commands are executed on the robot immediately as they are received. Although a few English commands exist, such as "stand" to make the robot stand, the interface was primarily designed to send arrays of joint values to the robot. As a subsequent step, the interface was expanded to allow for the AIBO's built-in walking engine to be accessed from Matlab. The programmer can then command the robot to move at a specific forward, strafe, and rotational velocity. Moreover, access to the three head-positioning servos was added to facilitate the collection of image and infrared ranging data from the sensors located in the robot's head.

SERVER ARCHITECTURE

The server-side programming was done in C++ using the OPEN-R software development environment [3] provided by Sony and the Tekkotsu framework. Tekkotsu contains methods for accessing video over the network and managing behaviors, so the bulk of the server code was already available. A behavior is Tekkotsu's closest analog to a process. Example behaviors include classes that follow a ball or, in our case, adjust the robot's position in response to network-directed commands. TCPReceiveBehavior is a C++ class compiled into Tekkotsu, which manages the reception and execution of commands coming in over the wireless network connection. New commands can easily be added to the list of available functions by editing the TCPReceiveBehavior code. Eventually, the functionality of this behavior should be such that every important AIBO function can be accessed from the Matlab client, effectively eliminating all C++ coding from the development process.

PRELIMINARY RESULTS: SAMPLE COURSE PROJECT

As an exercise in machine learning, a project was developed to apply reinforcement learning techniques to solve a robot-balancing problem. The AIBO is moved through different side-to-side leaning positions in order to balance it laterally on a tilting platform. The robot's camera is used to determine its position by observing a vertical line in its field of view. Balancing is achieved both with a classical control technique (PID controller) and with a reinforcement learning method utilizing eligibility traces for state value updates. Both schemes were tested on the same physical setup.

The PID controller proved to be more consistent, and generally performs better than the RL agent. The latter is trained for about 30,000 steps prior to taking measurements. The agent learned fairly well how to balance, and very often chose the correct action for a particular position. However, as evidenced by the results, the agent does go through periods of low reward where it seems to have difficulty identifying a good equilibrium position. Reduction of the state space led to a more stable functionality.

FUTURE WORK

Future work will involve a revision of the example client scripts and addition of new commands to the server behavior. A rewrite of the motion interpreter is also under consideration, to deliver a more regular transaction language. We have also been experimenting with scripting on the AIBO itself, and toward that end we ported a LISP interpreter prior to implementing the present system. We abandoned the idea when it became apparent that Matlab would be easier for the students to use. However, it may be superior for industrial use as it maintains the robot's autonomy. We expect the system to be used in future course offerings, starting Fall of 2007, at which point a formal release should be available on the Machine Intelligence Lab website [4].

CONCLUSION

A streamlined method of software development for the Sony AIBO robotics platform has been implemented, allowing students to easily develop real-time robotics applications. The approach accelerates program development by dramatically shortening each testing cycle and allowing for debugging during live operation with a real robot rather than a simulator. A course focusing on reinforcement learning was offered in Fall 2006 to both undergraduate and graduate students of Electrical and Computer Engineering at the University of Tennessee. Successful use of the software library was demonstrated as students utilized it when working on their final course projects. Consequently, the projects were completed in four weeks with the majority of student time spent on the topics of the course rather than on tedious coding and debugging of communication and control software.

REFERENCES

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

[2] *Tekkotsu* information available at : http://www.cs.cmu.edu/~tekkotsu/

[3] *Sony AIBO SDE* information available at: http://openr.aibo.com/

[4] *UTK Machine Intelligence Lab* webiste : http://mil.engr.utk.edu