Learning Leg Movement Patterns Using Neural Oscillators

Patrick McDowell and Theresa Beaubouef
Southeastern Louisiana University
Dept. of Computer Science and Industrial Technology
Hammond, LA 70402 USA
985-549-2189

{pmcdowell, tbeaubouef}@selu.edu

ABSTRACT
This paper discusses an approach to learning in order to find the joint movement patterns of a legged robot. In particular, we concentrate on a movement exploration technique based on patterns generated by a neural oscillator. The current stage of development and project status are presented along with a philosophy and implementation plan.

Categories and Subject Descriptors
I.2.9 [**Robotics**]

General Terms
Algorithms, Design

Keywords
Robotics, Neural Oscillators, Learning.

1. INTRODUCTION
At our university we are currently pursuing a robotics research program focused on providing robots with the ability to adapt and learn as necessary in order to keep their goals in target. We have created a research platform, the Large Actuator Test Bed (LATB) from which we can do basic leg motion and gait learning research as shown in Figure 1.

The LATB robot consists of a four wheeled utility cart with a set of 2 degrees of freedom legs mounted to it. Each leg has two degrees of freedom (thigh and calf sections). The muscle for the robot is provided by linear actuators, each with 20 pounds of linear force, each with a 1 foot range of motion. The system is controlled by a laptop and a SD 84 servo controller card. The entire system is monitored using a pc and a remote connection.

Basic functioning of the systems hardware and software components has been tested and various fundamental control issues have been identified [4]. The current stage of development includes the ability to move individual thighs and calves, monitor position of the joints, walk in a straight line a specified number of steps, and make right and left turns. A sequence of commands, such as "walk forward 4 steps, turn to the right, and walk forward 3 steps" is also possible. These motions use sensor feedback to determine the joint positions and coordination, but there is no feedback-based optimization. The goal of the LATB is to facilitate the development of low-level learning systems for leg motion and gait coordination. The cart was selected in order to simplify the problem; balance is not issue, thus leaving the focus on learning how to find the best motions and the coordination of them in order to move the cart in the desired direction. That being said, the cart does offer some nice features: it is easy to transport, mounting computers and control boards is trivial, and it is cost effective.

Figure 1. The Large Actuator Test Bed (LATB).

This paper details activities associated with the LATB. It specifically focuses on our philosophy, the resulting proposed techniques of using neural oscillators to explore possible solutions, and holding the parameters for these in a working memory. A learning technique for later interpolating between viable solutions is briefly discussed. This paper is organized as follows: the background section briefly describes selected learning techniques for leg movement and gait coordination, the approach section describes our philosophy and the resulting neural oscillator based approach, and lastly the future work describes issues with the project and the directions in which we plan to head.

2. BACKGROUND
Many successful walking robots have been assembled by engineers and scientists in recent years. The popularity of robots and in particular walking robots has grown tremendously, as evidenced by the explosion of toys such as Aibo, RoboSapien, FemiSapien, etc. [1]. There are robot soccer competition divisions for Aibo robots and for bipeds. Most of these robots,

ACM '10, April 15-17, 2010, Oxford, MS, USA.

especially the toys, come programmed with fixed gaits in which the robots shuffle along. While some of them learn, they usually don't learn gait and leg movement parameters in the sense that an animal would. Instead they concentrate on learning responses to their owners' actions, or have a learning mode in which the owner can train the robot's limbs through a sequence of positions that the robot can commit to memory and repeat. We next briefly describe work done by researchers in the area of learning and coordinating leg motions.

Anytime Learning [7] describes a particular way in which a learning system can be integrated with an execution system. The basic idea is that the execution system and the learning system run together continuously. The learning system contains a simulation of the environment in which the execution module operates. Its job is to continuously test strategies in the simulation so it can identify and develop improved strategies that it can then provide to the execution module. Meanwhile, the execution module is operating in the environment collecting relevant parameters about performance of the strategies and any changes in the operating environment. This information is used to improve the fidelity of the simulation. Parker used a variant of Anytime Learning, punctuated Anytime Learning, to develop gaits for hexapod robots and to evolve team behavior in box pushing tasks [7]. The term punctuated means that the learning system is not in the robot with the execution system. Since it is not close at hand, it does not have immediate access to sensor data, so the simulation in the learning system is not continually kept in sync with the environment. Punctuated Anytime Learning is a modification that compensates for the lack of feedback, precise sensors, and high computational power in the robot. In Parker's system, he used a Genetic Algorithm (GA) to create the leg movements in tiny steps. At any one moment a leg could move either forward, backward, up, or down. The GA was used to construct a string of these "moments" that resulted in a gait for his hexapod. The key advantage that Parker exploits is that the simulation can reject non-viable solutions, running at simulator speed, which is orders of magnitude faster than the physical robot. Not only does this strategy save time, it saves on battery power for the robot and wear and tear on the hardware components. One disadvantage is that he needs an accurate simulation system; if the environment or the robot changes, the simulation must be modified accordingly.

Maes and Brooks [3] describe an algorithm which allows a behavior based robot to learn based on positive and negative feedback, thus eliminating the need for the programmer to solve all the potential walking problems ahead of time. The end goal was to control overall robot functionality by selecting the appropriate behaviors from a "library" of behaviors, defining the connections to the sensors and actuators, defining the positive and negative feedback functions, and then letting the behaviors learn from experience when to become active. As with all behavior based robots the algorithm is mostly distributed. Each behavior tries to find out, based on feedback, when it is appropriate for it to be active. That is, in what situations is a particular behavior's operation relevant? Also, what are the conditions in which its operation becomes reliable? Said another way, when is the behavior's operation consistently associated with positive feedback? Based on feedback, each behavior is able to determine its own relevance and reliability in the context of a given situation.

Using this algorithm the six legged robot "Genghis" successfully learned to walk using swing forward behaviors for each leg, avoiding negative feedback produced when it fell on its belly, and maximizing positive feedback generated when it moved forward. Genghis learned the tripod gait, keeping three legs on the ground at any one time, the middle leg on one side and the front and back leg on the other side. This is not a trivial task; neither the negative nor the positive feedback is associated with any single behavior, but with the coordinating of the behaviors in a distributed system.

The work that Parker [7] did is fundamentally different from that of Maes and Brooks [3] in that his system constructed the reach and pull motions for the legs and coordinated them. His Anytime Learning system integrated several discrete leg movements in a continuous pattern of motion that resulted in a coordinated gate. The Genghis robot had some low level behaviors such as "lift leg", "reach", "pull", etc. whose priority of activity was sorted out and coordinated by the algorithm of Maes and Brooks. Each of these approaches was successful, but each is fundamentally different from our approach in that we assume that because of mass and momentum, physical things like legs prefer to move in patterns. The next section describes the rational for these ideas and our current state of implementation and our overall design.

3. PROPOSED APPROACH

Our approach is based on the desire to be able to learn what needs to be learned onboard the robot with as little a-priori information as possible. If the robot is armed with a goal and has little or no knowledge of itself or its environment, it must learn. In order to learn how to coordinate actions with environmental inputs (sensor or other means) some knowledge of the results of various actions must be present. In lieu of pre-programmed information the robot must acquire the knowledge by a teacher or some other means.

One way to gain knowledge is to explore the environment, both internal and external. By internal, we mean that the robot asks itself "What are my capabilities?" The results of internal exploration would answer questions concerning actuator and limb range of motion, speed that the robot can move, etc. External exploration refers to learning the effects that a robot's actions have on the environment, and/or its position and orientation in the environment. After some period of exploration time, using the robot's goal and the knowledge of the robot's (or agent's) surroundings and the effects of the robot's actions acquired during the exploration, a strategy to achieve the goal can be formed.

This approach, an explore/learn cycle, is fundamental to many learning methods, including Sutton and Bartow's Q-Learning technique [5]. One of the keys to this approach is to be able to pick up usable knowledge during the exploration period. In Q-Learning, the exploration is random, which has been shown to effective, especially in game playing situations such as Tesauro's Backgammon system [8]. This random search for information is functionally similar to the genetic algorithm's search that Parker used when coordinating gaits for hexapods. It works well because in both cases, the bulk of the searching was done inside a computer environment. Systems that don't search using a computer environment need to optimize their searches. For example, Maes and Brooks [4], provided their robot with the reach, lift, and pull behaviors. That way the robot was able to avoid a myriad of non-viable leg movements (in essence trying to

find good reach, lift and pull movements), and instead concentrate on coordinating its behaviors to form an effective gait.

Earlier work [5] has shown that in non simulation/game board situations, the method of exploration is crucial. In that work a robot learning to follow a sound source initially used a random search. Inevitably it would wander far enough from away from the source that its sensors could not discern if the source was to the left or right of it. The conclusion was that if the exploration is random, the robot could end up making a set of random movements during the exploration period that give it no usable information. To solve the problem, a "random but purposeful" search was used. The goal was known during the exploration period but no association was made between actions and sensor readings other than if the goal was getting farther away, a random change to the actions was made.

Key to our proposed approach is that many movements used by animals, such as walking, swallowing, throwing a ball, etc, are based on patterns. During activities such as walking or swallowing the pattern repeats itself over and over. The neural oscillator has been widely used to model these patterns and to control leg and body movements [2]. Typically the interconnections of the neural oscillator are produced by solving systems of differential equations, or by an optimization technique such as a GA. In our work, we plan to use neural oscillators to generate candidate patterns for the robot to test, and then from these to select the ideal solutions. Figure 2 below illustrates a typical neural oscillator.

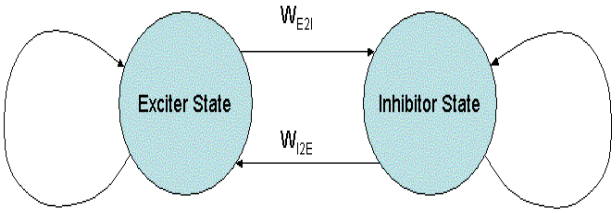

Figure 2. Basic 2 state oscillator. After the output of the system has reached the maximum value control moves to the Inhibitor state, after the final value is multiplied by the Exciter to Inhibitor weight W_{E2I}. The reverse happens in the Inhibitor state.

Neural oscillators can be considered to be a type of recurrent neural network. In general a recurrent network is different from a normal network in that its elements can store their states. In a normal network, the output is a product of the input's values, a set of weights, and a transfer function. So there is a one to one mapping from the input set to the output value. This is not the case with an. A neural oscillator usually consists of an excitatory and inhibitory neuron pair closely linked by interconnecting weights. The essential thing to remember is that the interaction of the two neurons produces a repeating pattern. Its intensity and frequency are typically regulated by the "tonic" input and can be modified by reflexive feedback. Neural oscillators are different from other networks and transfer functions in that they produce a range of values for a constant input. Most commonly used neural networks and transfer functions will produce the same output each time given a particular input. That is, there is a direct mapping from inputs to outputs. With neural oscillators, changing the tonic input changes the range of the output and/or the frequency of it. There is not a one-to-one mapping of input values to output values, but it could be said that there is a high correlation between input values and the characteristic pattern produced over time by the neural oscillator.

The neural oscillator is advocated because it produces patterns that are "natural" to physical systems. Because physical systems, like leg parts, have weight and inertia, they don't lend themselves to starting and stopping in square wave or jittery patterns. Our philosophy is to avoid the forward/backward searching done in the earlier Anytime Learning [7] system, and also to avoid having to write our own "reach" and "pull" routines (as done by the Maes and Brooks [3] system), by generating patterns that are natural to the equipment being used. By doing so, the solution space of the problem will be dramatically reduced.

The strategy is as follows:

1. Using a neural oscillator, find the ranges of motion of the robot's leg components.
   W_{E2I} and W_{I2E} weights can be set to values greater than one. This will produce an oscillating pattern that will increase with time. When the robot's low level software detects actuator stoppage the search ends.
2. Find the thigh and calf positions in which the foot can contact the ground.
   By varying parameters (tonic, gain, etc.) the waveform shape can be sculpted so that the foot moves to and through the peak of the pattern slowly (so that it can gently contact the ground). Do this for discrete steps throughout the range of motion of the limb.
3. Using the information collected from the above step, create a two dimensional table of thigh versus calf positions in which the ground was located.
   Using this table, leg-reach and leg-pull parameters can be generated and fed into the neural oscillator to generate a variety of different patterns. By selecting parameters in which the ground is contacted, the robot can then move itself. For values that in-between those in the table, an interpolation scheme can be developed, or the table can be learned by a feed forward neural network.

4. CONCLUSIONS AND FUTURE WORK

We have presented a case for using the neural oscillator as means of generating possible solutions for the control of a robot's leg actuators. We have developed a neural oscillator that is easily tunable with 3 parameters, and lends itself to reflexive inputs. Our next step is to test the neural oscillator system in simulation, and then transfer the test onto the LATB.

5. ACKNOWLEDGMENT

This research is supported by Louisiana Board of Regents LEQSF (2007-10)-RD-A-27.

6. REFERENCES

[1] http://www.robotbooks.com/

[2] Ijspeert, A. J., “A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander”, *Biological Cybernetics* 84, pp 221-348 (2001)

[3] Maes, P. , Brooks R. A., "Learning to Coordinate Behaviors", AAAI, Boston, MA, August 1990, pp. 796—802

[4] McDowell, P , Beaubouef, T , “Maintaining Control of a Robot's Limbs Using the Bakery Algorithm”, CCSC-MS 2010

[5] McDowell, P., Petry, F., Bourgeois, B., “Robot Control in Dynamic Environments using Memory-Based Learning”, *Springer Verlag*

[6] Mitchell, T. ,“Machine Learning”. McGraw-Hill, 1997, pp 367-386

[7] Parker, G. B.. “Punctuated Anytime Learning for Hexapod Gait Generation”, Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland, October 2002

[8] Tesauro, G., “Temporal Difference Learning and TD-Gammon”, *Communications of the ACM*, Vol. 38, No. 3, March 19