Chapter 1

Introduction

1.1 Intelligent motion control

All animals, from insects to people, are endowed with a sophisticated range of behaviors that allow them to interact appropriately with their world. They are capable of complex sensory processing, versatile execution of coordinated movements, and (in higher animals) planning of actions that will accomplish some task. Their behavior is *motivated* or goal driven rather than just being purely reactive. Their behavior is *adaptive*, in the sense that it can be automatically adjusted to suit unanticipated variations in their unstructured environments. And most importantly (in higher animals), those skills that they were not born with they can *learn*. The intelligent control problem is to design robots (or in general, automatons) that have some of these properties.

The problem of intelligent control is wide ranging, involving many disciplines of science and engineering. This thesis is concerned with only one aspect: the *motion control problem*—that is, how to make a robot move in a dynamic and coordinated way within its environment. Of course a useful robot will also need sensory processing and planning capabilities, but these problems will not be considered at all here.

In the emerging science of autonomous robots, the motion control problem is often ignored. This is because most existing robots use wheels for locomotion, so the problem is reduced to achieving a desired robot direction and forward speed, which is easily solved with current control technology. But the dynamical aspects of intelligent control are becoming more important. Consider the problems which would benefit from intelligent motion control:

- Controlling a wheeled autonomous robot where the mass-to-desired-speed ratio is high enough that the robot's dynamics play an important part in its trajectory, so that simple position-reference control strategies can not be used.
- Controlling a legged robot to achieve stable locomotion, where the desired leg motions are not known.
- Controlling a mechanically imperfect robot arm to achieve high speed placement of the end-effector. Mechanical imperfections such as flexibility in the links and slip in the drive-train are hard to deal with using conventional motion controller hardware.
- In general, the control of an arbitrary nonlinear mechanical system (such as an inverted pendulum on a cart) so that its performance is optimized in some manner.

Learning is an essential component of intelligent control. Consider that a non-learning controller must be carefully constructed to suit the environment: either a good system model must be constructed and a controller derived analytically from it, or a controller must be built with a number of unknown parameters which are fine tuned by hand as the system operates. Both approaches are very time consuming for all but the simplest systems. In learning-control the unknown controller parameters are automatically adjusted on-line as the system operates. This has the added advantage that the controller can compensate for unanticipated changes in the controlled system or environment.

The problem of learning motion control is encompassed by the highly developed field of *adaptive control* [51]. The two fields have similar goals but differ in scope and execution. A traditional adaptive controller tends to adjust a small number of parameters for linear systems to achieve optimal performance. Typically a learning process adjusts the parameters slowly through some least squares technique. Adaptive control can be highly robust, so there are many industrial applications. In contrast, a typical "intelligent" motion controller tends to output the control signals directly from a neural network (or similar device). A much larger number of parameters are adjusted by learning. These techniques are much more experimental and much less robust, although they have the potential to outperform more traditional approaches.

In this thesis a novel biologically motivated learning-control method called the "FOX" algorithm is developed and tested. It is shown that FOX is useful by itself for adaptive nonlinear optimal control. A "toolbox" approach to intelligent controller design is advocated, and it is shown that FOX is useful as a generic *learning component* in intelligent controllers.

1.2 Thesis overview

The remainder of this chapter gives some general background on autonomous robots and learning. The next three chapters look at three separate aspects of intelligent motion control. Chapter two gives some background information on how biological motor control is achieved in humans and other animals. This is important for two reasons: first, the FOX algorithm is a model of the cerebellum, a part of the brain that modulates movement. Second, one of the organizing principles in the brain is that neural circuits in the brain stem modulate reflex-implementing circuits in the spinal cord—a principle that is useful when applied to the design of robot controllers.

Chapter three looks at the CMAC neural network on which FOX is based. This chapter is largely a tutorial, although the performance of the CMAC is analyzed in some detail.

Chapter four introduces a control method called "feedback-error" (FBE). FBE was originally conceived by Kawato [60] as a model of human motor control, but in the form presented here it is simply an adaptive control methodology. FBE is important because it is a starting point and useful comparison for the FOX controller.

In chapter five the theory of the FOX controller is explained. A theoretical consideration of the adaptive control problem (using a CMAC) is used to derive the FOX algorithm. Aspects of the three previous chapters are pulled together to show how FOX overcomes the deficiencies of FBE, and how it is a model of the cerebellum. An efficient implementation of the FOX algorithm is described that achieves high speed by overcoming several practical problems.

In chapter six, FOX is put to the test. First a design methodology for FOX-based systems is described. Then FOX is successfully tested on four problems: controlling a simulated linear system, controlling a model gantry crane, balancing an inverted pendulum on a cart, and making a wheeled robot follow a path. In the process various design issues are explored.

In chapter seven it is shown that FOX can be a useful component in the controllers for more complex

robots: a simulated hopping monoped, and a simulated walking biped. FOX modules in these robots learn various parameters that fine tune the movements of their hard-wired controllers, in a manner analogous to the cerebellar modulation of spinal cord reflexes in human movement. Note that the CDROM that accompanies this thesis contains several MPEG and AVI format movies that have been created to illustrate various aspects of robot motion and training.

Finally, in chapter eight a list of unsolved problems and possible future work will be presented, and the main conclusions of the thesis will be summarized.

1.3 Thesis contribution

It will be seen later that the FOX controller adds an eligibility value to each CMAC weight. This by itself is not a new concept [67], but FOX is unique in several respects:

- FOX's eligibility values can be vectors, not just scalars. Vector eligibility values have not been used in any existing eligibility-based reinforcement learning techniques, to the author's knowledge.
- It is shown that the best eligibility update equations can be derived from easily measured dynamics of the controlled system. This allows a practical FOX design methodology to be created. Previous studies [68] have only provided loose guidelines for updating scalar eligibilities.
- A highly efficient FOX implementation is given which overcomes the need to update each weight's eligibility at each time step. This is more sophisticated, faster and more useful than previous approaches such as [48].
- FOX has learned to control the classically "hard" inverted pendulum system. Many other approaches have also solved this problem, but the FOX solution is particularly elegant.
- A simulated biped robot is taught to successfully walk—this is not new, as others have demonstrated successful biped locomotion. The novel aspect is the controller architecture that was used, where FOX modules were used as learning components.

1.4 Background on robots and learning

Before describing the CMAC and the FOX algorithm some general background information on learning methods and autonomous robot design will be presented. Some basic material on neural networks, training and generalization is given in Appendix A. Virtually every autonomous robot designer uses a different combination of controller paradigm and learning method. As a result the field of autonomous robots is very broad, but shallow, with few unifying threads. Many working systems have been demonstrated but no "killer paradigm" has been found. The many current techniques used are drawn from so many areas that it is sometimes difficult to make meaningful comparisons between different systems.

1.4.1 Why is intelligent control useful?

Intelligent control is necessary for autonomous robots which must survive in unstructured environments without continuous human guidance. Examples include robot sentries, intra-office delivery robots, and robot tour guides. Unlike the typical factory robot they may have to negotiate environments which are complex, changeable, full of obstacles and possibly hostile.

Intelligent control is also necessary when there is a time lag between operator commands and robot execution. A example is NASA's six-wheeled robotic rover, Sojourner, that was placed on the surface of Mars in July 1997 as part of the Mars Pathfinder project [74]. The rover had to be partly autonomous because of the large communication delay between Earth and Mars[1]. The rover navigated between waypoints specified by an earth bound operator, avoiding obstacles along the way.

Intelligent control is useful in telepresence applications, where a human operator issues high-level commands from a remote location and the robot complies using its own behavioral resources to implement the lower-level subtasks that are required.

Finally, a relatively new and speculative application of intelligent control is to control virtual actors. For example, in some situations an animator would prefer to be able to ask a virtual actor to "walk into the room and sit down" rather than specifying the detailed motions required.

1.4.2 Modular behavior and the break with classical AI

The classical artificial intelligence (AI) approach to controlling autonomous systems is to break the problem up in to functional modules such as sensory perception, environmental modeling, planning of actions and execution of those plans [19]. Various techniques from the AI toolbox are used within each module, such as knowledge representation schemes, goal directed searching and reasoning [121]. However, it has been argued that classical AI can not produce systems that are robust in real-world environments [13]. AI "intelligence" is notoriously narrow and brittle and is highly sensitive to the representation schemes used. AI knowledge is encapsulated in symbolic abstractions, and therefore is inflexible to deviations in the real world. AI systems do not seem to have much "common sense".

An alternative paradigm for the design of intelligent controllers is that of "modular behavior". The idea is that the controller contains modules which each perform some simple task oriented function. The overall behavior of the robot emerges from the interaction of these modules, rather than being specified explicitly.

For example, Brooks' subsumption architecture [19, 20] consists of modules containing state machines and timers that each implement some simple behavior. State machine outputs can be combined so that the output from one suppresses that of another. In this way higher level modules take over when the lower levels are inadequate to the task. The subsumption architecture has been used to make robust controllers for wheeled and legged robots, though the networks have to be carefully constructed to get the desired behavior.

Another example is Beer's artificial insect [13, 15, 14]. It is a hexapod robot that is controlled by an artificial nervous system made up of about 80 biologically realistic neurons. The neural network is designed rather than trained: it contains groups of neurons dedicated to particular tasks (such as leg control), which interact to achieve overall coordinated movement. The robot is capable of walking with various gaits and performing simple tasks such as wall following and food finding.

These two examples, though different in implementation are similar in principle: they are designed from the bottom up, by adding units that interact with the existing structure to create new behavior. Such systems are constructed especially to survive in their environments. They are not smart in the sense of classical AI—that is they do not perform high level planning and problem solving. Instead they are smart in the sense that simple animals like insects are smart: able to robustly survive in and interact appropriately with their environments.

[1] The time delay is between 6 and 41 minutes.

1.4.3 Neural network control

A lot of research has been done on using feed-forward neural networks[2] as the adaptive component in a learning controller [86, 128]. The network weights can be adjusted using the backpropagation algorithm (a gradient descent approach, see Appendix C), genetic algorithms [8], or various stochastic search algorithms (for example, Alopex [124] and statistical gradient following [129]). Supervised training is usually performed using error signals derived from the system's performance error, although other approaches which transfer expert information from a rule base are common. For example, Handleman [43] trains a CMAC from a "knowledge base" to control a planar two-link manipulator. A similar approach is implemented in [107], and [52] uses a fuzzy rule base.

Several control approaches have been developed which perform training on the system with its controlling neural network unfolded over discrete time (see Appendix B). Backpropagation through time [95, 99] propagates error information backwards through time, and forward propagation algorithms [130] do the reverse. Such algorithms can also train recurrent neural network controllers that have their own dynamical properties. These algorithms have been generalized to continuous systems [96, 50] and made more efficient in various ways [24]. See Appendix B and [97] for more information.

Miller [82] has extended the backpropagation through time approach so that error information is also propagated through a custom-built central pattern generator (CPG). Judicious choice of the CPG circuit can improve the performance and stability of learning simple motor tasks.

Although theoretically elegant, forward and backward propagation approaches are ill suited to practical on-line control (see Appendix B). Others have used a more successful analytical control-theory approach to train a neural network so that it becomes an inverse (in some sense) of the system being controlled [102, 108, 89, 7].

1.4.4 Reinforcement learning (RL)

Reinforcement learning [55] (RL) is a description used in the literature for several different learning paradigms. In fact it more properly refers to a general class of problems, those in which the success or failure of a system is indicated by a scalar reward (or penalty) signal. The system must try to select its actions such that the total future reward is maximized. In the most difficult form of this problem the reward signal is a simple binary yes/no signal that is provided only after many actions have been performed.

One class of solutions to these problems use generalized dynamic programming approaches[3]. Examples are Sutton's temporal difference learning [117] and Watkins' Q-Learning [69]. These techniques can be very slow. This is because when complex behavior is required it is hard to formulate a reward measure which provides enough information about how the problem should be solved. For instance if the task is to learn to make a biped robot walk, a simple reward signal based on the distance covered before falling down will not indicate that walking is best achieved by taking one step at a time. Many RL techniques will be forced to try different actions at random until a successful sequence is generated. This becomes problematic as more complex behavior is required.

Despite this, many experimental robots use (quasi-) RL to adapt their behavior, such as [81] which

[2] Appendix C describes the most fundamental feed-forward neural network, the multi layer perceptron. [71] contains a lot of basic information about neural network architectures and learning algorithms. [49] contains revised and expanded information, including a complete derivation of backpropagation, temporal difference learning, generalization theory, radial basis function nets, dynamic and recurrent neural nets (with training algorithms). It has pseudo code for the more important learning algorithms. Also see Amari [5].

[3] For a description of dynamic programming, see Appendix D.

uses a modular feed-forward neural network to coordinate the basic reflexes of a robot so that it can adapt to an unknown indoor environment.

1.4.5 Eligibility-based RL techniques, and others

Another class of RL solution uses the "eligibility principle", which states that the controller parameters that have contributed to the current system state are the ones that need to change if the system state is incorrect. An eligibility value is associated with each parameter (the parameters are usually neural network weights), which indicates how much influence it has had on the control output, and thus its influence on the present system state. The concept of eligibility is one way to solve the classic AI "credit assignment" problem, i.e. to decide which parameters of a system get credit for responsibility for the system's actions. Eligibility values record the potential for some action-producing element to be changed by the reward signal (so that it is more likely to generate its action in the future). The eligibility model is based to some degree on biological models of classical conditioning (see section 2.3.2 and [6] for more information on classical conditioning). The FOX algorithm is a member of this class of RL techniques.

A classic example is Barto's adaptive heuristic critic [10]. In this technique an associative search element (ASE) makes associations between input sensors and the output action. An adaptive critic element (ACE) takes the binary penalty signal and makes it more useful by trying to anticipate the sensor states that lead to failure. Eligibility values are used in both elements to allow the penalty signal to affect a wider range of action-producing states. This allows states that lead to system failure to be discredited, making the corresponding incorrect control actions less likely in the future.

Other eligibility based RL methods include the drive reinforcement model [61], the constant prediction model [73], the Sutton/Barto [118] and time-derivative [119] methods, and SOP [126]. These models have varying degrees of biological realism and practical utility [72].

There are various methods which claim to be reinforcement learning because they follow Barto's ASE/ACE architecture, but which do not have an eligibility component. For example the stochastic real valued (SRV) reinforcement learning approach of [39] has an actor/critic architecture which uses neural networks with stochastic output nodes. It was trained using backpropagation on a robot peg-in-hole insertion task. Similarly Venugopal [125] uses stochastic search to train recurrent neural networks to learn the control dynamics of autonomous underwater vehicles.

Other methods claim to be reinforcement learning simply because some kind of training is performed. For example, [62] implements "reinforcement learning optimization" without any of the elements mentioned above, to learn the correct modulation of robot motor programs. [80] implements "complimentary reinforcement backpropagation" to train a neural network to control a modified toy car.

1.5 Summary

Having multiple interacting behavior-implementing modules is a useful design principle for "intelligent" robot controllers. The full potential of this paradigm can only be realized if the controller is able to learn about its environment. Many different learning methods have been explored, but this thesis will consider only one: eligibility based reinforcement learning, implemented in the FOX controller. It will be shown that FOX is an adaptive optimal controller in its own right, as well as a useful *learning component* for intelligent control systems.